Computationally Efficient Equalizer Design

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Weiwei Zhou
Master of Science
GyeongSang National University, 2007
Bachelor of Science
University of Science and Technology Liaoning, 2004

Director: Dr. Jill K. Nelson, Professor
Department of Electrical and Computer Engineering

Summer Semester 2014
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my family for their motivation, patience, and support since the beginning of my studies.

Acknowledgments

I would like to thank my advisor Dr. Jill K. Nelson for her encouragement, guidance, and patience throughout the entire process. I wish to thank Dr. Ananya Sen Gupta for her inspiration and support for my research work. I would like to thank Dr. Bernd-Peter Paris, Dr. Shih-Chun Chang, and Dr. Jie Xu for serving in my committee and providing guidance for my studies. I would like to acknowledge and give special thanks to ECE department for allowing me to conduct my research and providing any assistance requested. I thank to Dr. Hossein Roufarshbaf, Mehdi Farrokhrouz, Fatemeh Pishdadian and all my friends in Underwater Acoustic and Signal Processing, and Communication Networking lab.

Table of Contents

List of Tables

List of Figures

Abstract

COMPUTATIONALLY EFFICIENT EQUALIZER DESIGN

Weiwei Zhou, PhD

George Mason University, 2014

Dissertation Director: Dr. Jill K. Nelson

Intersymbol interference (ISI) caused by frequency selective multipath propagation is a primary source of distortion in wireless communication systems. ISI significantly degrades system performance, and hence channel equalization is typically employed at the receiver to mitigate the harmful effects of ISI. An equalizer can be designed to operate on either a symbol-by-symbol or sequential basis. Symbol-by-symbol based equalizers estimate the transmitted symbols one at a time, while sequential-detection based equalizers make an estimate of the full transmitted sequence based on the received signal over a full block of data. In this work, we propose computationally efficient methods to design both symbol-by-symbol and sequential equalizers for various communications scenarios.

In symbol-by-symbol schemes, we focus on the computationally efficient design of a maximum asymptotic efficiency (MAE) equalizer. The MAE equalizer achieves an attractive balance between performance and complexity. It minimizes bit error rate as the signal-to-noise ratio approaches infinity while retaining simple implementation by using a linear structure. However, its design requires solving quadratic programming problems and hence has high computational complexity. We propose a geometrically-inspired approach to the MAE equalizer design that dramatically reduces complexity. Additionally, we extend the MAE equalizer to applications in which the channel varies significantly with time by proposing a pre-equalization technique which enables the MAE equalizer to be designed only once despite channel variations. The combination of the two proposed methods simplifies the design of the MAE equalizer, facilitating its use for time-varying channels with longer delay spreads.

In sequential detection schemes, we focus on communication problems which involve detecting data transmitted over channels with a small number of sparsely spaced channel taps. Such sparse channels are present in applications such as underwater acoustic (UWA) communications, ultra-wide band (UWB) communications, and high-definition television (HDTV) systems. We propose a tree-search based sequential equalizer that considers only the significant channel coefficients. In addition, we consider situations in which the sparse channel is unknown and no training data is available. We develop a blind sequential detection method by incorporating a novel greedy algorithm into a tree-search based sequential detector. The proposed technique reduces complexity and yields improved performance relative to existing matching pursuit (MP) based methods.

Chapter 1: Introduction

Emerging wireless communication technologies have revolutionized the way we work, live and interact with each other, penetrating every aspect of our lives. The demand of high-speed and high-quality wireless communication services is growing rapidly, such as high-speed Internet access, high-quality video transmission and so on. In order to deliver such services to users, we can increase the channel bandwidth to increase the amount of data that can be sent over the channel. Theoretically, we can transmit as much data as we wish in the presence of a channel with infinite bandwidth. In reality, however, all types of transmission channels are of limited bandwidth. The limitations arise from the physical properties of the channel or from regulations on the bandwidth to prevent interference from other sources. Therefore, we would like to develop innovative technologies on the design of communication systems to handle the challenges brought by wireless channels.

A wireless communication system consists of three components as shown in Fig. 1.1. The transmitter obtains information from a source and transforms it into symbols that can be transmitted over a radio frequency channel. The channel possibly impairs the transmitted signal during transmission. The receiver then has the job to detect the transmitted signal. If the receiver works perfectly, the output of the receiver will be the same as the input to the transmitter.

In the system, a radio wave, propagating from the transmitter to the receiver, might

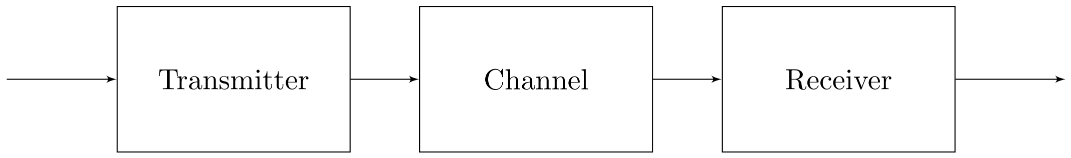


Figure 1.1: Three basic components of a communication system.

be reflected or diffracted by objects which are close to the receiver, like buildings, trees and cars. A simple example of the wireless channel is shown in Fig. 1.2. A base station sends a series of symbols to a cellphone. The signal travels along multiple paths with different lengths. Three incoming signals are received at the cell phone. In this scenario, one symbol of the transmitted signal can be corrupted by a previous symbol due to a second signal path. This kind of phenomenon is called intersymbol interference (ISI). For wireless communications, multipath propagation of the transmitted signal is a primary cause of ISI. ISI can significantly degrade the performance of a communication system, which is one of the major obstacles to reliable and high-speed wireless communications. Therefore, the transmitter and receiver should be designed to minimize the effect of ISI. There are two popular approaches employed in communication systems to reduce ISI. The first approach, which is at the transmitter end, designs bandlimited transmitting pulses using the Nyquist pulse-shaping criterion [1]. The second approach is to use equalization/detection at the receiver end to reduce the effects of the ISI introduced by dispersive channels. In this work, we will focus on the latter approach and design innovative equalization technologies achieving a balance between performance and implementation complexity for particular applications.

1.1 Motivation and Contributions

Equalization is a process of compensating for the distortion (ISI) introduced during transmission over a channel in order to improve the accuracy of transmitted signal estimation [2]. An equalizer can be designed to operate on either a symbol-by-symbol or sequential basis. Symbol-by-symbol based equalizers estimate the transmitted symbols one at a time, while sequential-detection based equalizers make an estimate of the full transmitted sequence based on the received signal. A large variety of research has been conducted to design equalizers of both types to mitigate the effect of ISI [3–11].

For symbol-by-symbol equalization, equalizers usually employ linear finite impulse response (FIR) filters, where feedback might be also used. The coefficients of equalizers are

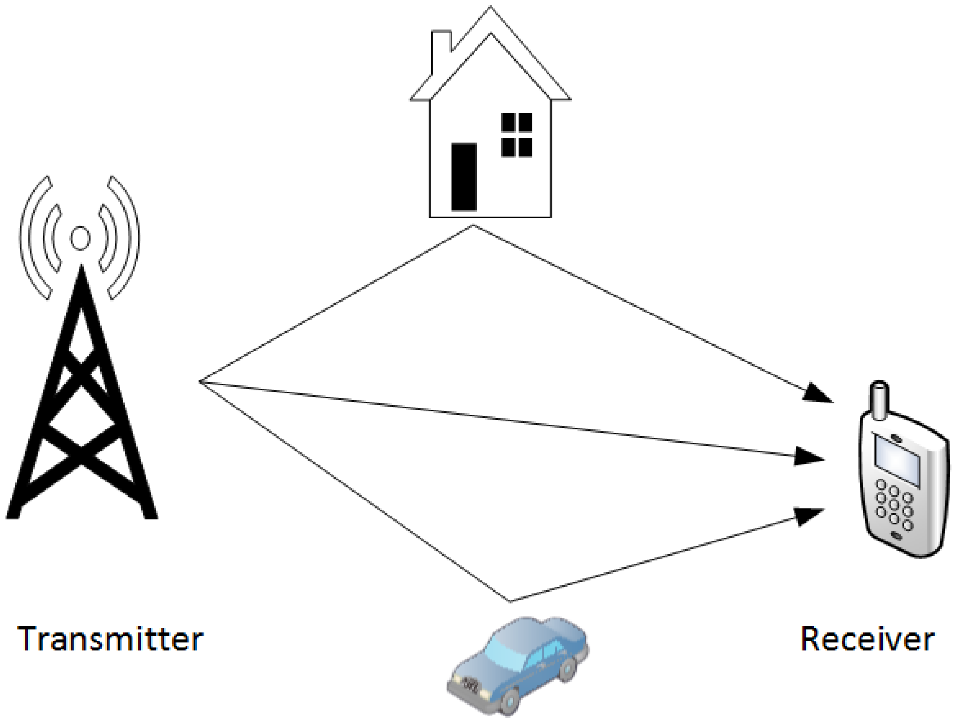


Figure 1.2: An example of a wireless channel.

designed or optimized to compensate for the negative effect introduced by the channel. Various types of methods have been proposed to obtain these coefficients. For example, the well-known zero forcing (ZF) equalizer is designed by finding the inverse of the channel impulse response, and the minimum mean square error (MMSE) equalizer is designed to minimize the mean square error (MSE) between the received and desired signal. Symbol-by-symbol based equalizers are generally simple to implement. However, they typically have limitations on their performance in an environment with severe ISI.

In this work, we study the structure of symbol-by-symbol based equalizers, and seek to design equalizers with enhanced performance while retaining simple implementation. A recently proposed equalization method called the maximum asymptotic efficiency (MAE) equalizer is such a method that can achieve this balance. We will focus on the computationally efficient design of the MAE equalizer in this work. The MAE equalizer is optimized in an asymptotic scenario and minimizes the probability of error only when the noise variance $\sigma \to 0$. It provides much better performance than ZF and MMSE equalizers when SNR is relatively high. In addition, the MAE equalizer uses the same linear structure as the MMSE equalizer to ensure its simple implementation. Hence, the MAE equalizer finds a middle point between performance and computational complexity. However, computing the MAE equalizer coefficients requires solving quadratic programming problems and hence suffers from high computational complexity. In order to further simplify the coefficient computation, we propose an alternative way to design the MAE equalizer. We develop a geometrically-inspired approach to the MAE equalizer design that dramatically reduces computational complexity. Additionally, we consider situations in which the channel varies significantly with time. We propose a pre-equalization technique in which, rather than designing the equalizer to match the changing channel, we apply a pre-filter to match a fixed one, and design the MAE equalizer only once for the fixed channel.

For sequential-detection based equalization, equalizers are designed using the sequential detection methods which are originally proposed for decoding convolutional and tree codes. Different from the symbol-by-symbol equalization, the whole transmitted sequence is estimated simultaneously by choosing the sequence with the maximum likelihood (ML) of being transmitted given the received signal. Equalizers based on maximum likelihood sequence estimation (MLSE) are considered to be optimal in in terms of minimizing sequence error rate.

Most sequential-detection based equalization methods were proposed to handle transmission over general ISI channels. For some applications such as underwater acoustic (UWA) communications, ultra-wide band (UWB) communications, and high definition television (HDTV) systems, the discrete-time channel impulse response has a very large channel memory, but only a small number of significant channel coefficients contributes to the distortion of the transmitted signal. We call such channels sparse channels. When we apply the generally designed sequential detection techniques for such channels, all channel coefficients are considered to estimate the transmitted signal, yielding high computational burden for the receiver, and decreasing the accuracy of estimates. Therefore, if we take the channel sparsity into consideration, the equalizer can be designed in a more efficient way, and its performance can be improved at the same time.

In this thesis, we are dedicated to developing computationally efficient sequential-detection based equalization methods for sparse ISI channels. We make our contribution on studying the characteristic of sparse channels, and propose a sequential-detection based equalizer that only operates on the significant channel coefficients. The situations that the sparse channel is unknown at the receiver is also considered. In order to avoid transmitting training sequences to estimate the channel, we develop methods that only uses received signal and some necessary priors about the transmitted signal. The equalization process is divided into two steps: 1)First, the channel sparsity is estimated; 2) Secondly, a sequential detection method is incorporated with the estimated sparsity. We prove that the computational complexity is reduced significantly by doing so.

1.2 Organization of the Thesis

The thesis is organized as follows: In Chapter 2, some mathematical background on equalization is provided, and existing equalization methods are discussed. In Chapter 3, the maximum asymptotic efficiency (MAE) equalizer is introduced. A computationally efficient algorithm is proposed to reduce the design complexity of the MAE equalizer, and an adaptive pre-filter is developed using constellation mapping to make the MAE equalizer practical for time-varying channels. We consider the sequential-detection based equalizer for sparse channels in Chapter 4. A novel multiple-tree algorithm is proposed to reduce the complexity of the conventional tree search algorithm when applied to sparse channels. In Chapter 5, we extend the sparse sequential equalizer to consider scenarios in which the channel is unknown at the receiver. A blind computationally efficient sequential detection method is developed using a greedy algorithm. Conclusions and avenues for future work are presented in Chapter 6.

Chapter 2: Background and Survey of Literature for Existing Methods

In communication system, when the frequency response of the channel changes significantly within the bandwidth of the transmitted signal, different frequency components of the signal will experience different gains during transmission and the channel is said to be frequency selective. In the time domain, equivalently, the effect of a symbol will spread to adjacent symbols, causing intersymbol interference (ISI). In order to better illustrate the concept of ISI, we will discuss some mathematical details to describe the communication system.

2.1 Intersymbol Interference

Let us consider a bandlimited wireless communication system shown in Fig. 2.1. A sequence of information symbols $x(t)$ convolved with a pulse-shaping filter $h_p(t)$, passes through a dispersive channel $h_c(t)$, and is then processed by a receive filter $h_r(t)$ to produce symbol estimates [12]. We assume that additive white Gaussian noise (AWGN) $w_g(t)$ is present on the channel. The output signal $y(t)$ can be expressed as

$$y(t) = x(t) * h_p(t) * h_c(t) * h_r(t) + w_g(t) * h_r(t) \tag{2.1}$$

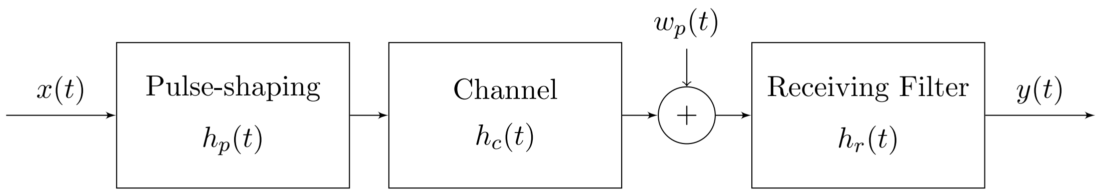


Figure 2.1: Block diagram of the discrete-time equivalent communication system.

Since the pulse-shaping filter, the dispersive channel and the receive filter are all linear time invariant (LTI) filters, they can be combined into one LTI filter, represented by $h(t) = h_p(t) * h_c(t) * h_r(t)$. Due to the receive filter, the noise is no longer white. In order to apply known equalization techniques designed using assumption of white noise, a noise-whitening filter is usually included in the receive filter to make the colored noise white. Hence, the communication system can be simplified as

$$y(t) = x(t) * h(t) + w(t), \tag{2.2}$$

where $w(t)$ is the white Gaussian noise after the noise-whitening filter. Assuming that the receiver has knowledge of the signal phase and symbol timing, the receive filter is designed to match the transmit pulse shape and the channel impulse response. Sampling the output of the receive filter, the system can be presented as an equivalent discrete-time model, which is expressed as

$$y_k = \sum_{i=0}^{L_h-1} x_{k-i} h_i + w_k, \tag{2.3}$$

where h_i, $i = 0, \ldots, L_h - 1$, are the equivalent discrete-time filter coefficients of the combined filter $h(t)$. x_k, y_k and w_k are the samples of the transmitted signal, received signal, and whitened AWGN noise, respectively, at the k-th sampling instant.

Let us redraw the equivalent discrete-time communication system with an equalizer in Fig. 2.2. y_k is the input of the equalizer, and the output of the equalizer $\hat{x}_k$ ideally only contains the desired information symbol x_k and a noise term with small variance. Assuming the symbols x_k have unit average power and the energy of the channel is normalized to be

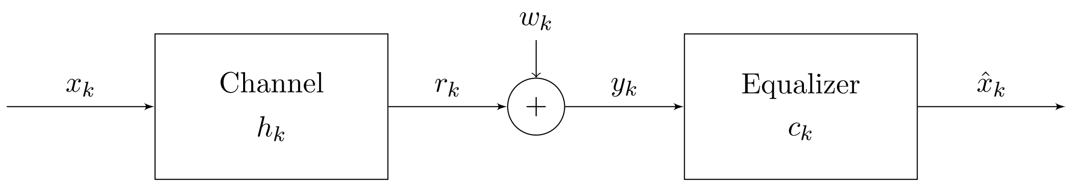


Figure 2.2: Block diagram of the discrete-time equivalent communication system.

1, the signal-to-noise ration (SNR) is

$$\begin{aligned} SNR &= \frac{\text{signal power}}{\text{noise power}} \\ &= \frac{\sum_{i=0}^{L_h-1} |h_i|^2}{\sigma^2} \\ &= \frac{1}{\sigma^2}, \end{aligned} \tag{2.4}$$

where σ^2 is the variance of the noise w_k.

We expand (2.3) as follows:

$$y_k = x_k h_0 + \sum_{i=1}^{L_h-1} x_{k-i} h_i + w_k. \tag{2.5}$$

For the desired information symbol at the k-th sampling instant, x_k, the residual term

$$\sum_{i=1}^{L_h-1} x_{k-i} h_i \tag{2.6}$$

is the intersymbol interference. The ISI smears the adjacent symbols together and introduces additional dependencies among them. We need to design the receiver to mitigate the effect of ISI using knowledge of how the signal is corrupted during transmission. An equalizer is such a mechanism to eliminate ISI.

An equalizer can be designed to operate on either a symbol-by-symbol or a sequential basis. The sequential equalizers are grounded in approaches that were originally designed for decoding convolutional and tree codes [13]. Most of them map the possible transmitted symbols onto a trellis or tree structure. They apply different strategies to navigate the structure in search of the most likely path, often incurring high computational complexity. In contrast, most symbol-by-symbol equalizers are much easier to implement, but they tend to have limitations on their performance in severe ISI environments. A simple illustration of the categories of equalizers is shown in Fig. 2.3. We will describe some popular channel equalization approaches in each category in the following sections.

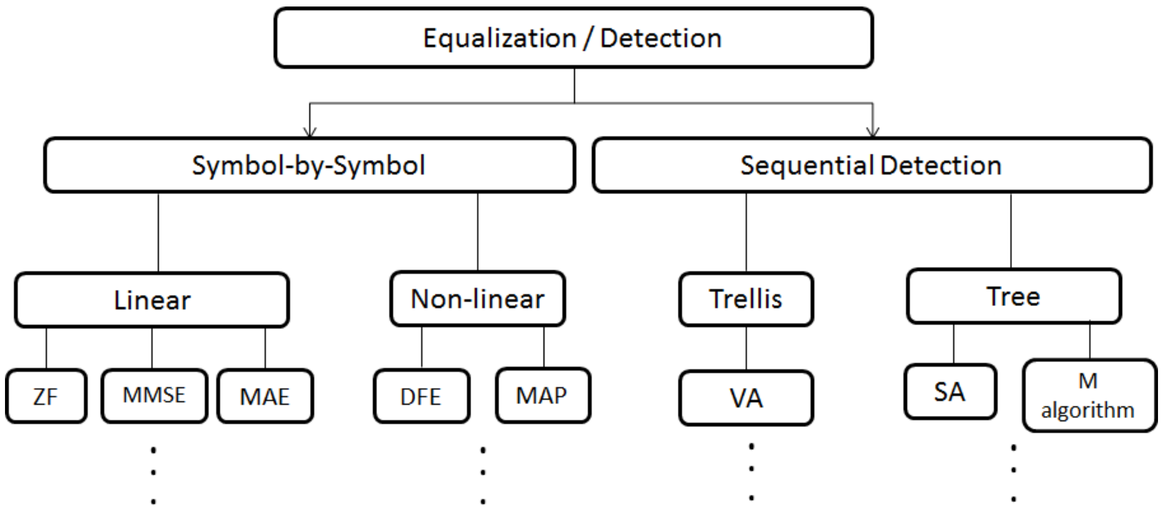


Figure 2.3: Illustration of different categories of equalizers.

2.2 Conventional Symbol-by-Symbol Equalization

Based on their structure, the symbol-by-symbol based equalizers can be divided into two categories: linear equalizers and non-linear equalizers.

2.2.1 Linear Equalizer

A linear equalizer processes the received signal by using a linear transversal filter structure, shown in Fig. 2.4. The transversal filter uses a tapped delay line structure, where the received signal y_k is multiplied by the weight coefficients of the filter c_i, $i = 0, \dots, L_c - 1$, and the results are summed together to give the output of the filter. The weight coefficients c_i are adjusted to reduce ISI. Considering a linear equalizer of length L_c with weight coefficients $\boldsymbol{c} = [c_0, c_1, \dots, c_{L_c-1}]^T$, the equalizer output at time k is

$$\tilde{x}_k = \sum_{i=0}^{L_c-1} c_i y_{k-i} = \boldsymbol{c}^T \boldsymbol{y}_k, \tag{2.7}$$

where $\boldsymbol{y}_k = [y_k, \dots, y_{k-L_c+1}]^T$ is the vector of input to the equalizer. $\tilde{x}_k$ is then quantized and the corresponding nearest information symbol is found to form the final decision $\hat{x}_k$. If the output of the linear equalizer $\hat{x}_k$ is not identical to the desired information symbol x_k, an error occurs. Considerable research has been conducted to optimize the weight coefficients of the equalizer. We introduce three criteria that are commonly used: zero-forcing (ZF), minimum mean square error (MMSE), and maximum asymptotic efficiency (MAE).

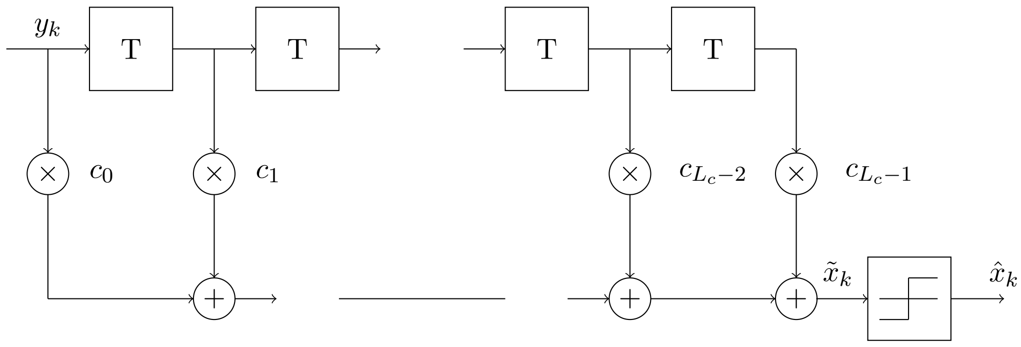


Figure 2.4: A linear equalizer with length L_c using a transversal filter structure.

Zero-forcing Equalizer

The zero forcing (ZF) criterion was first proposed by Robert Lucky to design the coefficients of a linear equalizer [1]. A ZF equalizer is based on the criterion to minimize the ISI, which forces the impulse response of the channel and the equalizer to become a unit pulse. Let us consider a linear equalizer with an infinite number of taps c_i, $-\infty < i < \infty$. We assume the impulse response of the discrete-time linear channel is h_j, $0 \leq j \leq L_h - 1$. The ZF criterion can be expressed as

$$\sum_{i=-\infty}^{\infty} c_i h_{k-i} = \begin{cases} 1, & k = 0 \\ 0, & k \neq 0 \end{cases}. \tag{2.8}$$

Thus, it is easy to see that a ZF equalizer with infinite length is simply an inverse filter to the channel,

$$C(e^{j\omega}) = \frac{1}{H(e^{j\omega})}, \tag{2.9}$$

where $C(e^{j\omega})$ and $H(e^{j\omega})$ are the frequency response of the ZF equalizer and the channel, respectively. Theoretically, the ZF equalizer with infinite length can completely eliminate ISI. Considering stability and realizability constraints [14], ZF equalizers are implemented as an FIR filter with adjustable tap coefficients in practical applications to approximate this inverse. To find the coefficients of the finite-length ZF equalizer, we write the input of the equalizer in a matrix form,

$$\begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-L_c+1} \end{bmatrix} = \begin{bmatrix} h_0 & \dots & h_{L_h-1} & 0 & \dots & 0 \\ 0 & h_0 & \dots & h_{L_h-1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & h_0 & \dots & h_{L_h-1} \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \\ \vdots \\ y_{k-L_h-L_c+1} \end{bmatrix} + \begin{bmatrix} w_k \\ w_{k-1} \\ \vdots \\ w_{k-L_h-L_c+1} \end{bmatrix}, \tag{2.10}$$

which can be simplified as

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k. \tag{2.11}$$

We should emphasize that the FIR ZF equalizer does not completely remove ISI because it has finite length. However, the length of the equalizer L_c can be chosen sufficiently large so that the equalizer spans the length of the ISI. As L_c increases, the residual ISI can be reduced. In order to determine the coefficients, we force the multiplication of the channel matrix $\mathbf{H}$ and the ZF equalizer coefficient vector $\mathbf{c}_{ZF}$ to be a unit vector,

$$\begin{bmatrix} h_0 & \dots & h_{L_h-1} & 0 & \dots & 0 \\ 0 & h_0 & \dots & h_{L_h-1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & h_0 & \dots & h_{L_h-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{L_c-1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \tag{2.12}$$

which can be expressed as

$$\mathbf{H}\mathbf{c}_{ZF} = I_d, \tag{2.13}$$

where d is the decision delay that indicates the position of the 1 element in vector I. The coefficients of the ZF equalizer can be obtained by taking the inverse of the matrix $\mathbf{H}$,

$$\mathbf{c}_{ZF} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H I_d. \tag{2.14}$$

With knowledge of the channel, the ZF equalizer is very easy to implement. However, in the process of developing a ZF equalizer, the effect of noise is neglected. If the impulse response of the channel has a deep null like the example in Fig. 2.5 for the 3-tap channel $\mathbf{h} = [0.407, 0.815, 0.407]$, the corresponding ZF equalizer will be its inverse and greatly enhance the power of noise.

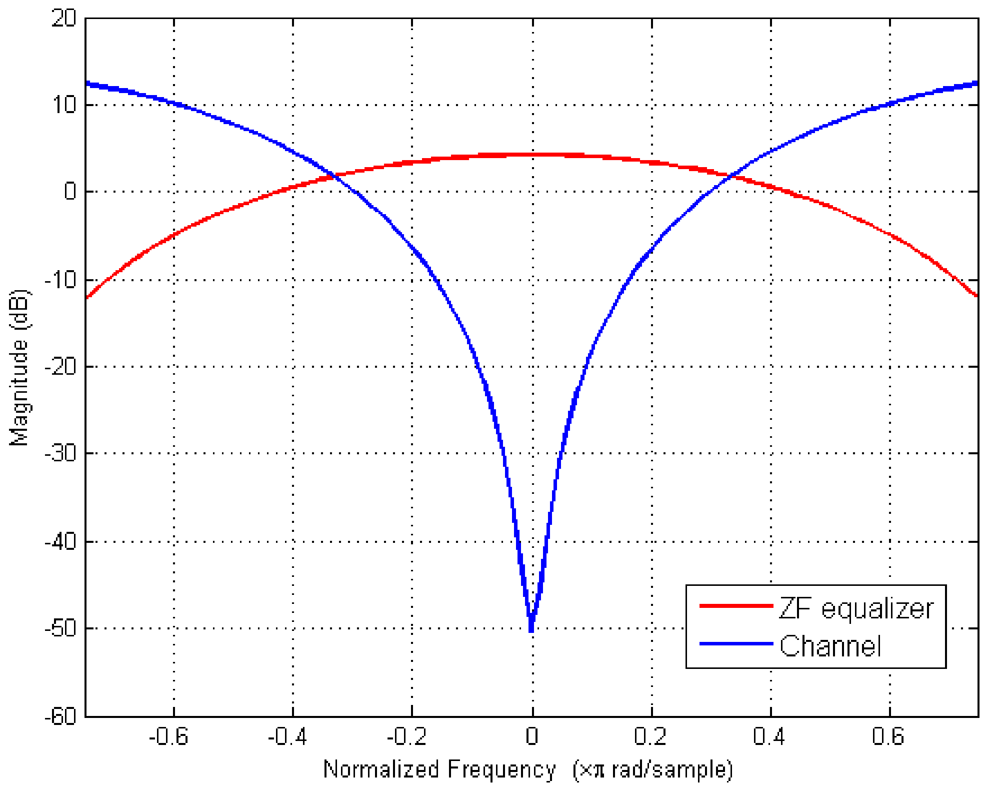


Figure 2.5: An example of the impulse response of the 3-tap channel $\mathbf{h} = [0.407, 0.815, 0.407]$ with a deep null and the corresponding 3-tap ZF equalizer.

MMSE Equalizer

To overcome the disadvantages of the ZF equalizer, an alternative method was proposed based on the criterion of minimization of the mean square error (MSE) between the equalizer output signal $\hat{x}_k$ and the desired signal x_k [15],

$$\text{MSE} \triangleq J(\mathbf{c}) = E\{|x_k - \hat{x}_k|^2\} = E\{|x_k - \mathbf{c}^T \mathbf{y}_k|^2\}, \tag{2.15}$$

The MSE is a function of $\mathbf{c}$, and the coefficients of the MMSE equalizer can be obtained by minimizing $J(\mathbf{c})$ [16],

$$\begin{aligned} \mathbf{c}_{MMSE} &= \underset{\mathbf{c}}{\operatorname{argmin}}\, J(\mathbf{c}) \\ &= (\mathbf{H}^H \mathbf{H} + \sigma^2 \boldsymbol{I})^{-1} \mathbf{H}^H I_d, \end{aligned} \tag{2.16}$$

where $\boldsymbol{I}$ is the identity matrix and σ^2 is the variance of the AWGN. Compared to the ZF equalizer coefficients computed in (2.14), the MMSE equalizer [1] takes noise into account (the noise term $\sigma^2 \boldsymbol{I}$ is included in (2.16)). It balances the effect of the noise and ISI at the equalizer output. Hence, an MMSE equalizer yields better performance than a zero-forcing equalizer when noise is presented. As SNR increases, the performance of the two equalizers converges because the effect of the noise term $\sigma^2 \boldsymbol{I}$ decreases. A performance comparison between a 3-tap ZF equalizer and a 3-tap MMSE equalizer can be seen in Fig. 2.6 for the 3-tap channel $\mathbf{h} = [0.407, 0.815, 0.407]$. The frequency response of the channel is shown in Fig. 2.5 with a deep null. The ZF equalizer inverts the channel and enhances the noise power during the equalization process. The MMSE equalizer takes noise into consideration, which makes the performance difference between the two methods significant when SNR is low.

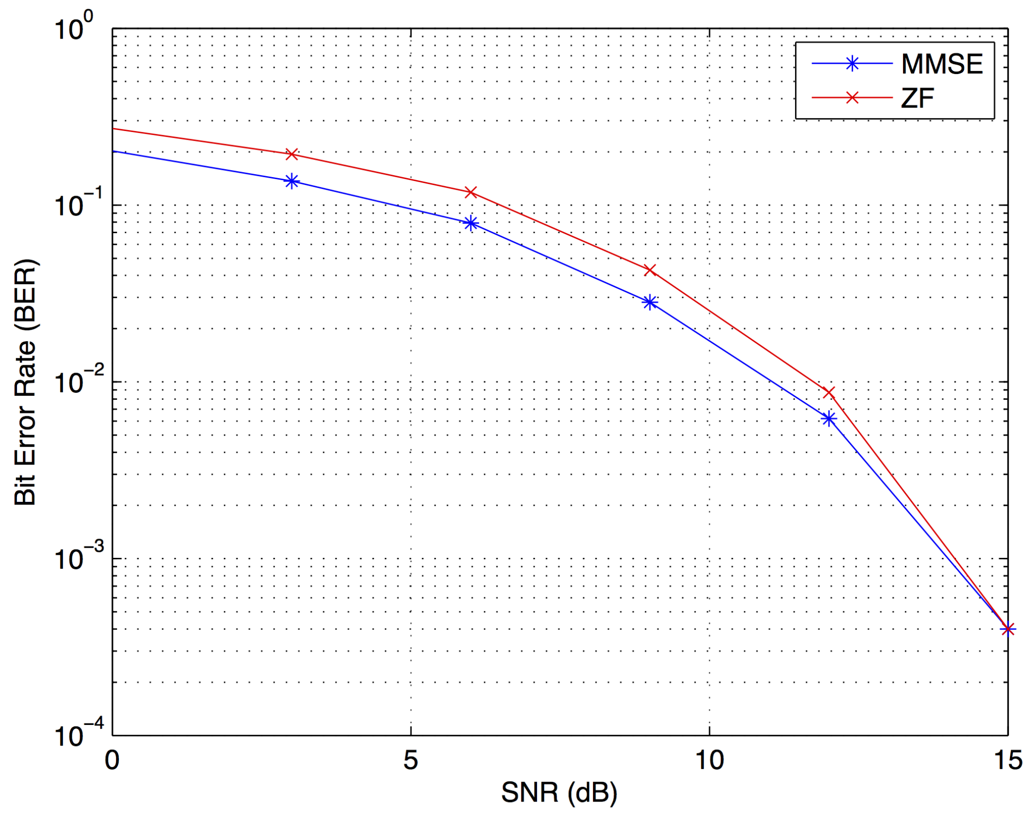


Figure 2.6: A performance comparison between a 3-tap ZF equalizer and a 3-tap MMSE equalizer for channel $\mathbf{h} = [0.407, 0.815, 0.407]$ as SNR increases.

MAE Equalizer

Another form of the linear symbol-by-symbol based equalizer, recently proposed based on the asymptotic performance of the system, is called the maximum asymptotic efficiency (MAE) equalizer. Asymptotic efficiency (AE) [17], introduced originally in the context of multi-user detection (MUD) [18], is a popular measure to evaluate the performance of multi-user detectors with respect to their ability to mitigate interference from other users. We can define asymptotic efficiency analogously for equalization of ISI channels, where the detector is mitigating interference from other symbols rather than other users. The MAE equalizer, introduced in [19], uses a geometric method to find a decision hyperplane that maximizes the asymptotic efficiency of the equalizer. The normal vector of the hyperplane is equivalent to the weight coefficients of a linear equalizer. The MAE equalizer is optimal as the signal-to-noise ratio approaches infinity with respect to minimizing the probability of error.

The MAE equalizer is not the only method to use a geometric way to analyze equalization problems. Other approaches that apply geometric methods for channel equalization include [20–24]. For example, the optimal Bayesian solution, determining and applying a nonlinear decision boundary to the received signal, is proposed in [24]. In [21], a multiple-decision-boundary method is proposed to eliminate the effect of ISI. But compared to those methods, the MAE equalizer uses only a single hyperplane which is able to provide optimal performance in high SNR with relatively low computational complexity. Due to the balance that the MAE equalizer achieves between complexity and performance, we will focus on developing techniques so that the MAE equalizer can be better utilized in communication systems. The details of the MAE equalizer will be illustrated in the next chapter.

2.2.2 Nonlinear Equalizers

The nonlinear equalizers employ a nonlinear filter or other nonlinear structure to estimate the transmitted symbols from the ISI-affected observations. We introduce the decision feedback equalizer and the maximum a posteriori equalizer in this section.

Decision Feedback Equalizer

The decision feedback equalizer (DFE) is a widely used nonlinear equalizer, using a structure that contains two transversal filters: a feedforward transversal filter and a feedback transversal filter, shown in Fig. 2.7. Similar to the linear equalizer, the previous decisions are made by the feedforward transversal filter; the feedback transversal filter constructs the ISI term contributed by the previously detected symbols. The ISI term is then subtracted from the output of the feedforward filter. Although the DFE applies nonlinear structure, the weight coefficients of the feedforward filter and the feedback filter, $\boldsymbol{c}_{ff}$ and $\boldsymbol{c}_{fb}$, can be found based on the ZF and MMSE criteria that were described in Section 2.2.1. The output of the DFE is

$$\hat{x}_k = \sum_{i=-L_{ff}+1}^{0} y_{k-i} c_i + \sum_{i=1}^{L_{fb}-1} x_{k-i}^d c_i = \mathbf{c}_{ff}^T \boldsymbol{y}_k + \mathbf{c}_{fb}^T \tilde{\mathbf{x}}_k \tag{2.17}$$

where $\mathbf{y}_k = [y_{k+L_{ff}-1}, \ldots, y_0]^T$ are the input to the feedforward filter, and $\tilde{\mathbf{x}}_k = [x_{k-1}^d, \ldots, x_{k-L_{fb}+1}^d]$ are the symbols determined by the decision device.

In general, a DFE yields a significant improvement in performance compared to a linear equalizer having the same number of taps. However, the performance of the DFE is degraded when incorrectly detected symbols are fed through the feedback filter. This phenomenon is called error propagation, which enhances ISI and causes decision errors. Many methods mitigating error propagation have been proposed and incorporated into the receiver design [25–27].

Decision feedback can also be combined with the MAE equalizer, which makes the MAE equalizer more powerful in practical communication systems. We will discuss this combination in Chapter 3.

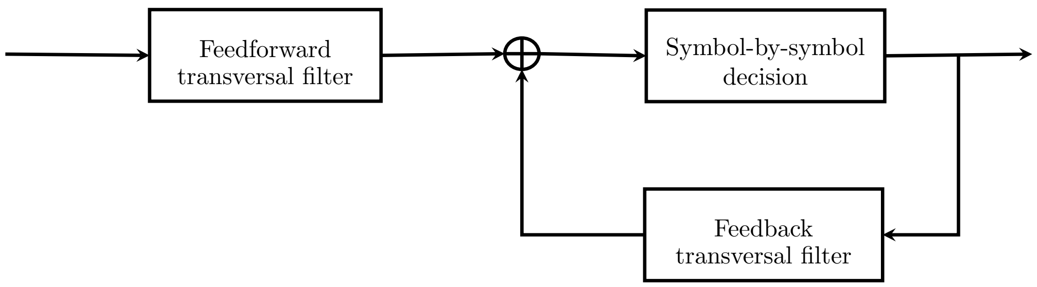

Feedforward transversal filter
Symbol-by-symbol decision
Feedback transversal filter

Figure 2.7: Structure of decision feedback equalizer.

Maximum A Posteriori Equalizer

The maximum a posteriori (MAP) criterion [28], providing an optimal solution with respect to minimizing the bit error rate (BER) of the system, is another criterion to design SS based nonlinear equalizers. The MAP equalizer finds the transmitted signal x_k which maximizes the posterior density $P(x_k|\boldsymbol{Y}_1^N)$, where $\boldsymbol{Y}_1^N = \{y_1, \ldots, y_N\}$ is the received sequence. Applying Bayes theorem, the MAP criterion can be expressed as

$$\hat{x}_k = \underset{\boldsymbol{x_k}}{\operatorname{argmax}}\, P(x_k|\boldsymbol{Y}_1^N) = \underset{\boldsymbol{x_k}}{\operatorname{argmax}}\, \frac{P(\boldsymbol{Y}_1^N|x_k)P(x_k)}{P(\boldsymbol{Y}_1^N)} = \underset{\boldsymbol{x_k}}{\operatorname{argmax}}\, P(\boldsymbol{Y}_1^N|x_k)P(x_k), \quad (2.18)$$

where $P(x_k)$ is the prior probability of the transmitted signal x_k. MAP equalization can be implemented using the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [29], which applies an iterative forward-backward algorithm that operates on a trellis. The number of states on the trellis increases exponentially with channel memory. For many practical applications, constructing and navigating a trellis with a large number of states is prohibitively complex. For example, with 8 phase-shift keying (PSK) modulation and a 5-tap baseband channel, the MAP algorithm requires $8^5 = 32768$ states at each stage of the trellis, leading to an unaffordable load for the receiver.

2.3 Conventional Sequential Detection Based Equalization

In contrast to symbol-by-symbol based equalizers, sequential-detection based equalizers produce an estimate of the whole transmitted sequence based on the received signal sequence. The sequential-detection based equalizers estimate the transmitted sequence by choosing the sequence with maximum likelihood (ML) of being transmitted given the received signal. Various methods have been proposed to attain or approximate the ML solution. Most of them are implemented by using the structure of a graph, such as a tree or a trellis. Each path in the graph corresponds to a possible realization of the transmitted sequence. Different strategies of searching are performed to find the "best" path based on the metrics assigned for paths. The mathematical expression of the process can be shown as

$$\{\hat{x}_1, \ldots, \hat{x}_n\} = \underset{\boldsymbol{l}}{\operatorname{argmax}}\, P\{x_1^{(l)}, \ldots, x_n^{(l)} | y_1, \ldots, y_n\}, \tag{2.19}$$

where l denotes the l-th path of the graph.

2.3.1 Viterbi Algorithm

The Viterbi algorithm (VA) uses the structure of a trellis [30] with a finite set of states. Each state at a given time instant represents a possible realization of a portion of the transmitted sequence. The VA computes a set of metrics associated with the observation symbols and finds the most likely path through the trellis. For all the paths that lead into the same state, it decides which of them is the most likely to occur, i.e. the path with the greatest metric. The VA then keeps the survivor path with greatest metric and discards the other paths into that state. The process can be shown in Fig. 2.8, where only one survivor is kept among the paths coming into the same state. The VA attains the ML solution, but the number of states grows exponentially with the channel memory L_h. For an L_h-tap ISI channel with M-ary modulation, the VA needs to construct M^{L_h} states. In applications with a long channel, the VA becomes prohibitively complex.

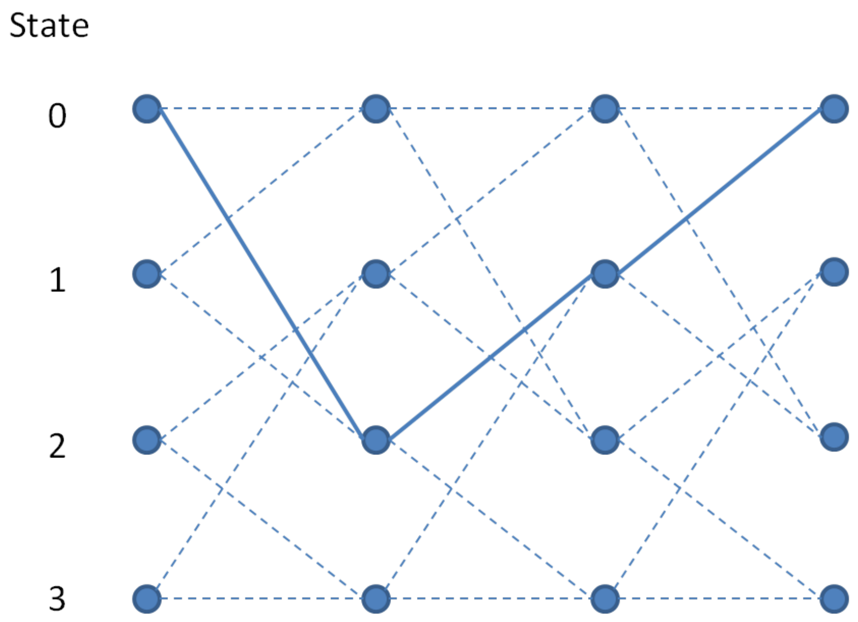


Figure 2.8: An example of the VA using trellis. The VA keeps the survivor path with greatest metric and discards the other paths coming into the state. The solid line in the figure denotes the survivor path and the dashed lines indicate discarded paths.

2.3.2 Tree Search Algorithm

An alternative optimal method is to use tree structure [13] which relies on a tree representation of the search space spanned by transmitted symbols. An exhaustive tree search generates all possible transmitted sequences through a tree and evaluates the likelihood of each sequence. Since the number of possible tree branches increases exponentially with time, the exhaustive tree search is never used, even when the number of branches is small, which has motivated the development of tree search algorithms with reduced complexity.

Numerous sub-optimal approaches that approximate the performance of the VA and exhaustive tree search have been proposed. The M algorithm [31] is a breadth-first strategy for searching the tree. At a given depth of the tree, it extends all branches for the existing nodes, then keeps the M paths with the best path metrics. In contrast, the stack algorithm (SA) [32] is a best-first method. At each time step, the algorithm extends only the path with

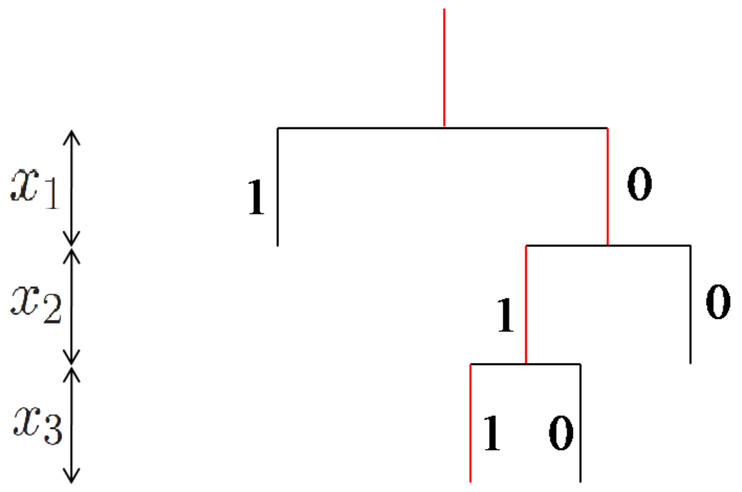


Figure 2.9: An example of the stack algorithm navigating a tree. The algorithm extends only the path with the largest metric at each stage. The red line denotes the survivor path that the stack algorithm chooses. Hence 011 is the corresponding estimate of the transmitted sequence.

the largest metric. The SA is more efficient and extends fewer paths than the M-algorithm when SNR is high. However, for relatively low SNR, the SA jumps around the tree during the search process, causing the number of required computations to be quite large. Since the number of branches extended in the tree is independent of the length of the channel, the M-algorithm and the SA, without sacrificing much performance compared to the VA, are more efficient for channels with long delay spread.

The stack algorithm navigating a tree with depth 3 is shown in Fig. 2.9. At time 1, metrics are computed for both the left and right branches, and the corresponding paths are stored in a stack. The algorithm compares the two metrics and puts the path with larger metric at the top of the stack. At time 2, the top path in the stack is extended and replaced by its two children. The path with largest metric is moved to the top of the stack. The process is repeated and terminated at time 3 when the top path reaches a leaf of the tree. The red line in Fig. 2.9 shows the path that the SA chooses as the survivor path. In this example, the estimated symbols $\{\hat{x}_1, \hat{x}_2, \hat{x}_3\}$ are $\{0, 1, 1\}$.

In our research, we will focus on the SA and consider its application on sparse ISI channels that are widely encountered in UWA, UWB and HDTV communication systems. The conventional SA designed for general communication channels is modified so that it implements equalization specifically for sparse ISI channels. Both known-channel and unknown-channel scenarios are considered. Our goal is to develop a method for detecting data transmitted over sparse channels that both performs well and is computationally efficient.

Chapter 3: Efficient Design and Implementation of the Maximum Asymptotic Efficiency (MAE) Equalizer

We have discussed conventional approaches for equalizer design based on both symbol-by-symbol and sequential detection bases. Equalizers using the MAP criterion provide optimal detection with respect to minimizing bit error rate but suffer from high design and implementation complexity. The MMSE linear equalizer provides a low-complexity approach but suffers significant performance degradation relative to the optimal solution. In this section, we will focus on the maximum asymptotic efficiency (MAE) equalizer which achieves a balance between performance and complexity by combining sophisticated design with low runtime complexity.

The MAE equalizer is implemented as a tapped delay line and hence has the same runtime complexity as the simple MMSE linear equalizer. However, design of the MAE equalizer involves finding the minimum distance between two convex hulls. Its design complexity is exponential in the length of channel and equalizer, making it impractical for long channels. We propose a method that exploits the relationship between the channel vectors and the convex hull formed by the noise-free channel outputs to design the MAE equalizer directly from the channel coefficients without requiring a search of the convex hull. The equalizer design complexity is reduced to $O(N \log N)$, where N is determined by the length of the channel and equalizer. Furthermore, when the MAE equalizer is applied for time-varying channels, frequent redesign renders the effective operational complexity of the MAE equalizer impractical for implementation. We address this issue by mapping the time-varying channel to a fixed channel for which the MAE equalizer is pre-designed via a linear pre-filter.

The maximum asymptotic efficiency and the design of the MAE equalizer are described in Section 3.1. In Section 3.2, the proposed computationally efficient design of the MAE equalizer is presented, and simulation results are provided to evaluate its computational efficiency. In Section 3.3, an adaptive pre-equalization technique is designed and combined with the MAE equalizer to equalize time-varying channels. The performance of the resulting adaptive equalizer is compared to conventional adaptive equalization methods.

3.1 Maximum Asymptotic Efficiency Equalizer

Asymptotic efficiency (AE), originally introduced in the context of multi-user detection (MUD), measures the loss of signal energy due to multiuser interference as SNR tends to infinity [18]. Maximizing asymptotic efficiency has been used as a criterion for designing MUD systems [33]. Because dispersive channel equalization and MUD share a common structure, MUD techniques may be naturally adapted to address intersymbol interference (ISI) mitigation. The maximum asymptotic efficiency (MAE) equalizer, introduced in [19], identifies the decision hyperplane in signal space that minimizes bit error rate in the high SNR regime.

While the runtime complexity of the MAE equalizer is equivalent to that of an MMSE linear equalizer, the design complexity is much higher. Identifying the MAE decision hyperplane requires quadratic programming (QP) to search for the minimum distance between convex hulls, the complexity of which is exponential in the length of channel and equalizer. In this section, we present a novel computationally efficient algorithm for finding the MAE decision hyperplane when binary phase shift keying (BPSK) signaling is used. The proposed algorithm exploits the geometric properties of the constellations generated by noise-free channel outputs. We show that the decision hyperplane of the MAE equalizer can be determined directly from the channel coefficients, avoiding any need for quadratic programming and reducing the design complexity to $O(N \log N)$, where $N = 2^{L_c+L_h-2}$ is determined by the length of the channel L_h and equalizer L_c.

3.1.1 System Model

The communication model we consider, shown in Fig.2.2, includes a length-L_h linear time-invariant ISI channel with coefficients $\mathbf{h} = [h_0, h_1, \cdots, h_{L_h-1}]$, additive white Gaussian noise w_k, and an MAE equalizer with dimension (length) L_c. The channel output is denoted by y_k. The input to the MAE equalizer, $\mathbf{y}_k = [y_k, y_{k-1}, \cdots, y_{k-L_c+1}]^T$, can be written as

$$\begin{aligned} \mathbf{y}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \qquad (3.1) \\ &= \mathbf{r}_k + \mathbf{w}_k, \end{aligned}$$

where $\mathbf{x}_k = [x_k, x_{k-1}, \cdots, x_{k-L_c-L_h+2}]^T$ denotes the transmitted symbols modulated using M-ary pulse amplitude modulation (PAM), and $\mathbf{r}_k = [r_k,\ r_{k-1},\ \cdots,\ r_{k-L_c+1}]^T$ denotes the noise-free channel output. We assume that the channel $\mathbf{h}$ is known at the receiver and the corresponding channel matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \begin{pmatrix} h_0 & h_1 & \cdots & h_{L_h-1} & 0 & \cdots & 0 \\ 0 & h_0 & h_1 & \cdots & h_{L_h-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{L_h-1} \end{pmatrix}.$$

The input vector $\mathbf{x}_k$ may take one of $N = M^{L_h+L_c-1}$ possible combinations of M-PAM data. From (3.1), we can see that $\mathbf{r}_k$ also has N possible combinations. The signal constellation is formed by considering all possible values of the noise-free output vector $\mathbf{r}_k$. Each possible $\mathbf{r}_k$ forms an L_c-dimensional constellation point for which the i-th element of $\mathbf{r}_k$ gives the point's coordinate in the i-th dimension. There are $M^{L_c+L_h-1}$ possible points in the space; these points can be grouped into M subsets according to the value of x_{k-d}, where x_{k-d} takes one of the M-ary possible transmitted symbol values, and d denotes the decision delay of the equalizer. Let us take the channel $\mathbf{h} = [0.5, 1]$ and a 2-tap equalizer

Table 3.1: All possible noise free output for channel $\mathbf{h} = [0.5, 1]$ and a 2-tap equalizer.

x_k	x_{k-1}	x_{k-2}	r_k	r_{k-1}
1	1	1	1.5	1.5
1	1	-1	1.5	-0.5
-1	1	1	0.5	1.5
-1	1	-1	0.5	-0.5
1	-1	1	-0.5	0.5
1	-1	-1	-0.5	-1.5
-1	-1	1	-1.5	0.5
-1	-1	-1	-1.5	-1.5

as an example. The decision delay d is set to be 1 and binary phase shift keying (BPSK) is used. The corresponding noise free output r_k is listed in Table 3.1. There are 8 constellation points which correspond to the $N = 8$ possible combinations of $\mathbf{r}_k = [r_k, r_{k-1}]$. We can plot them in the Fig. 3.1, where the blue constellation points in the figure correspond to $x_1 = +1$ and the red constellation points correspond to $x_1 = -1$. Letting $C_d^{(j)}$, $1 \leq j \leq M$, denote the subset of constellation points for which x_{k-d} is equal to the j-th symbol, the full constellation C can be written as

$$C = \cup\, C_d^{(j)},\ 1 \leq j \leq M. \tag{3.2}$$

Note that the set of points belonging to the full constellation C remains fixed; changing the decision delay d changes only the members of C in each subconstellation.

For ease of presentation, we consider binary phase shift keying (BPSK), or equivalently $M = 2$, in this chapter. For the binary case, notation can be simplified to

$$C_d^{(\pm)} = \{\mathbf{r}_k | x_{k-d} = \pm 1\}. \tag{3.3}$$

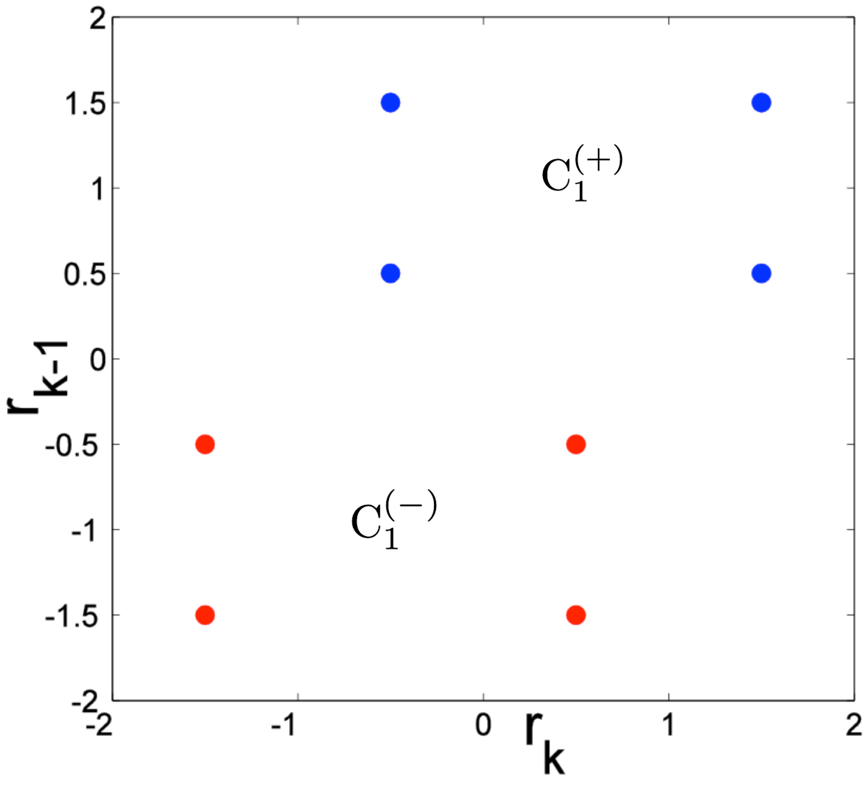


Figure 3.1: Constellation points for channel $\mathbf{h} = [0.5, 1]$ and a 2-tap equalizer. The decision delay is $d = 1$.

Since we consider linear channels, $C_d^{(+)}$ and $C_d^{(-)}$ exhibit odd symmetry with respect to the origin of the constellation space. The equalization process can be viewed as a classification problem that seeks to classify a received signal vector $\mathbf{y}_k$ as belonging to the class defined by one of the symbol values s_i, $1 \leq i \leq M$. A linear equalizer estimates each bit by forming an inner product between the received vector $\mathbf{y}_k$ and the equalizer taps $\mathbf{c} = [c_0, c_1, \cdots, c_{L_c-1}]$ and mapping the result to the closest symbol value. For binary signaling, this process can be equivalently viewed as determining where the received vector falls relative to a hyperplane (defined by the equalizer tap values) through constellation space.

3.1.2 MAE Equalizer Design

Asymptotic efficiency (AE) , introduced originally in the MUD applications, is defined as the rate of decay of the detector's probability of error as noise variance approaches 0. Due to the common structure of the MUD and equalization, we can define asymptotic efficiency analogously for equalization of ISI channels, where the detector is mitigating interference from other symbols rather than other users. Mathematically, the asymptotic error exponent of an equalizer, or equivalently the log of the asymptotic slope of the bit error rate, can be written as [34]

$$\eta = \lim_{\sigma \to 0} 2\sigma^2 \frac{1}{\log(P_e)}, \tag{3.4}$$

where P_e denotes the probability of error of the equalizer as a function of SNR, and σ denotes the standard deviation of the additive noise. The MAE equalizer is designed to maximize η which is equivalent to minimizing the BER as SNR approaches infinity.

Let $CH(C_d^{(\pm)})$ denote the convex hulls formed from the constellation points $C_d^{(\pm)}$, where the convex hull is defined as follows,

Definition 1. *The convex hull of a set of points, $C = \{p_1, \ldots, p_k\}$, is the intersection of all convex sets that contains C, or more mathematically, the set of all convex combinations of points in C:*

$$CH(C) = \{\sum_{i=1}^{k} \lambda_i p_i, \;\; p_i \in C, \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1\},$$

where $\{\lambda_i\}$ are parameters used to define the point in the convex hull. In short, the convex hull of a set C is the smallest convex set which includes C. In Fig. 3.2, an example of convex hulls for $CH(C_d^{(+1)})$ and $CH(C_d^{(-1)})$ are shown for a 3-tap equalizer operating on a 3-tap channel. We can see that the dimension of the convex hull is determined by the length of the equalizer.

By exploiting the geometric properties of the constellations, the maximum AE is found

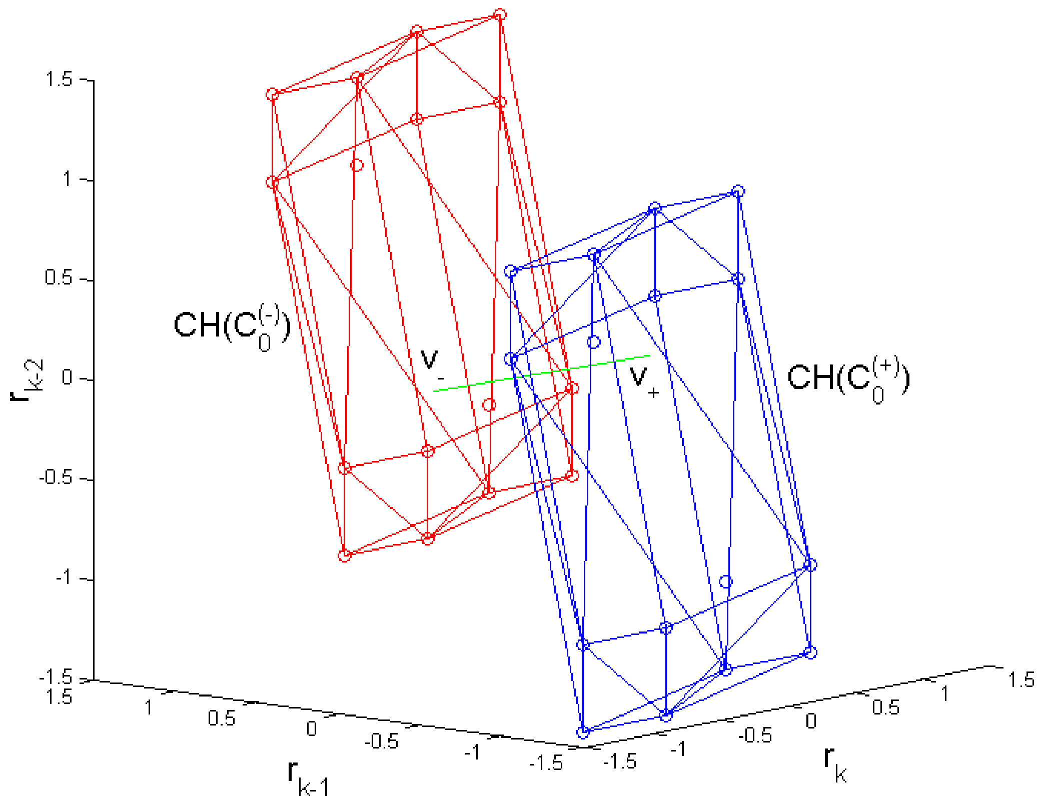


Figure 3.2: An example of convex hulls for a 3-tap equalizer operating on a 3-tap channel using decision delays $d = 0$.

to be governed by the minimum distance between the convex hulls [35].

$$\eta_{\mathbf{max}} = \frac{||v_+ - v_-||^2}{4} = ||v_+||^2, \tag{3.5}$$

where $v_\pm$ denote the vectors to the points on each convex hull which have minimum distance to the origin. The above equation uses the property $v_+ = -v_-$ for symmetric convex hulls across the origin. The asymptotically optimal decision boundary is the hyperplane passing through the origin and perpendicular to the vector v_+. Fig. 3.3 shows an example constellation for a two-tap channel $\mathbf{h} = [1, 0.5]$ with a two-dimensional MAE equalizer ($L_c = 2$) using decision delays $d = 0$ and $d = 1$.

The equalization process now can be viewed as a classification problem. The bit estimate generated by the MAE equalizer at each time instant k is computed by determining the side of the hyperplane on which the received vector $\mathbf{y}_k$ lies. This is efficiently computed via an inner product:

$$\hat{x}_{k-d} = \begin{cases} 1, & \langle v_+, \mathbf{y}_k \rangle \geq 0 \\ -1, & \langle v_+, \mathbf{y}_k \rangle < 0. \end{cases} \tag{3.6}$$

The constellation points belonging to each convex hull vary with the chosen decision delay. Consequently, minimum distance will change with decision delay, and one can define an optimal delay that yields the largest $||v_+||$. In Fig. 3.3, for example, $d = 1$ provides better performance than $d = 0$ due to larger minimum distance. The search for the optimal delay is out of the scope of this thesis, but would be a good topic for our future research.

3.1.3 Simulation Results for MAE Equalizer

In order to illustrate the performance of the MAE equalizer, we present numerical simulation results to provide a comparison between the MAE equalizer and conventional equalization methods. Fig. 3.4 compares the performance of the MAE equalizer and the conventional MMSE equalizer for a 3-tap channel and a 5-tap channel. We notice in both cases that

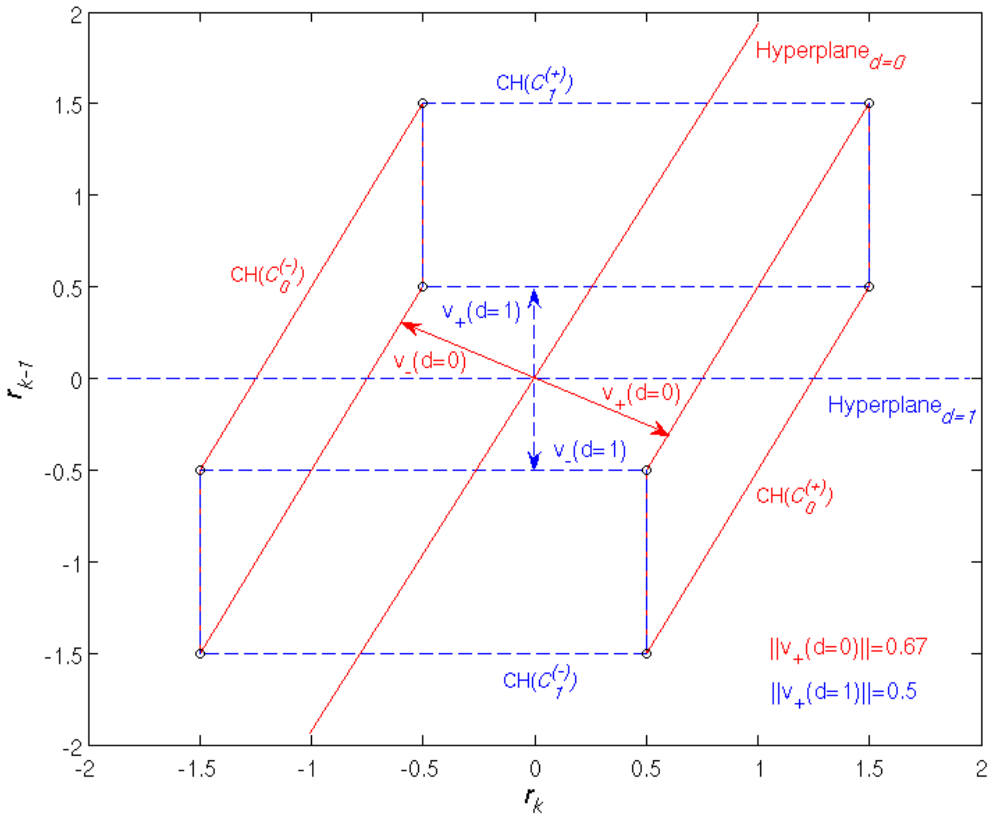


Figure 3.3: Convex hulls and optimal decision hyperplanes for a two-dimensional equalizer operating on a two-tap channel $\mathbf{h} = [1, 0.5]$ using decision delays $d = 0$ (blue lines) and $d = 1$ (red lines).

for low SNR, the performance of the MMSE equalizer is superior to that of the MAE equalizer, but as SNR increases the MAE equalizer outperforms the MMSE equalizer. This is expected, as the MAE equalizer is optimized for performance in high SNR when detection errors are primarily due to ISI rather than additive noise.

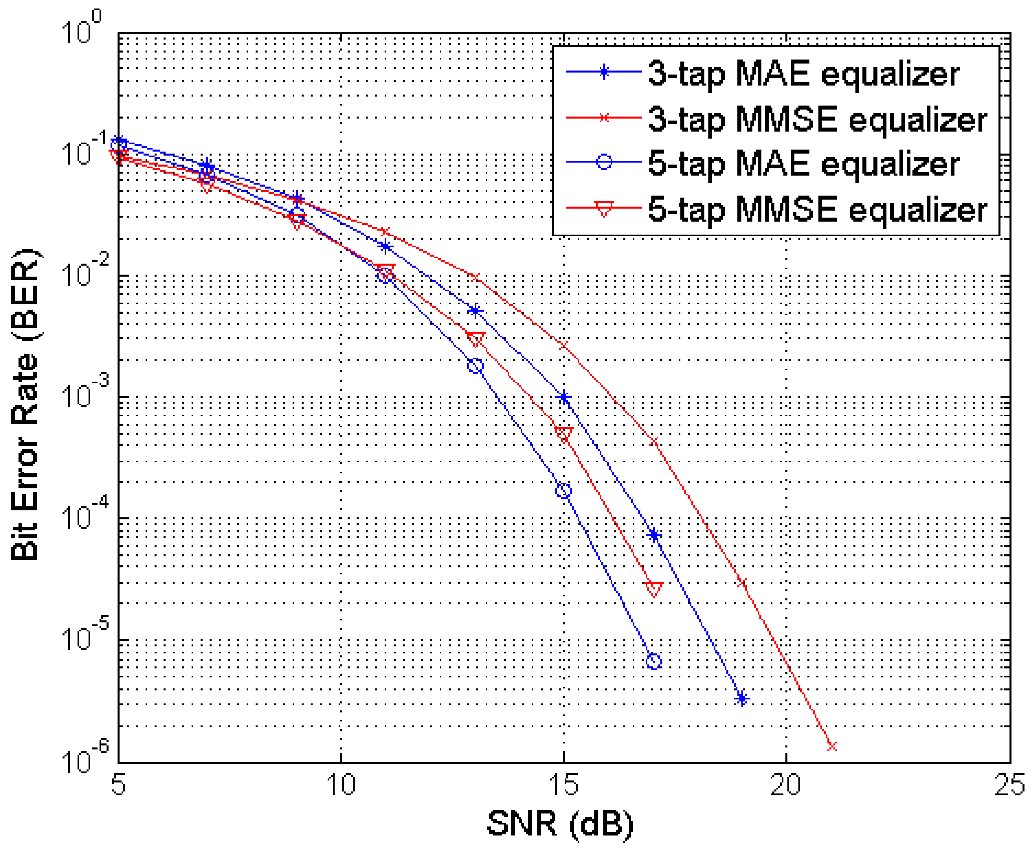


Figure 3.4: BER comparison between the MAE and MMSE equalizer for a 3-tap and a 5-tap channel.

The MAE equalizer is designed based on the assumption that perfect knowledge of the communication channel is available. In reality, however, only a channel estimate is available, and this estimate will have some error relative to the true channel. Additionally, the channel may experience small variations over time; when these variations are very small, we ignore them and treat the channel as constant. In these cases, equalizers are designed based on approximation to the channel. Therefore, it is worthwhile to examine the robustness of equalizers to channel perturbations. In order to achieve this, we conducted simulations to examine the performance of the MAE and MMSE equalizers for channels with unknown perturbations.

Fig. 3.5 shows the performance comparison between the MAE and MMSE equalizers for a 3-tap channel where the channel taps are perturbed by a bounded error of approximation proportional to α. The perturbation for each channel tap h_i, denoted by Δh_i, is modeled as $\Delta h_i = \min(B, w)$, where $w \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian perturbation error and $B = \alpha\sigma$ is the bound on the perturbation. α is used to scale the perturbation bound and is set for $\alpha = 0, 0.5, 0.8, 2$. The performance comparison of the two equalization methods shows similar trends to that seen in Fig. 3.5 for channels without perturbations. We observe that the MAE equalizer is more robust to channel perturbations than is the MMSE equalizer when SNR is high. Therefore, even when channel estimation is imperfect, the MAE equalizer provides better performance than the MMSE equalizer when SNR is relatively high.

3.2 Efficient Geometric Algorithm for MAE Equalizer Design For Binary Signaling

The MAE equalizer is very attractive in its superior performance at high SNR and in its low runtime complexity. However, the design of the MAE equalizer is computationally demanding. The design of the MAE decision hyperplane H_o requires searching for the minimum distance vector v_+, which is equivalent to finding the minimum distance between the origin and convex hull $CH(C_d^{(+)})$ [36]:

$$||v_+|| = \min_{\boldsymbol{\lambda}} || \sum_{k=1}^{2^{L_h+L_c-2}} \lambda_k \mathbf{r}_k^{(+)} || \quad \text{subject to } \sum_k \lambda_k = 1,\ 0 \leq \lambda_k \leq 1, \tag{3.7}$$

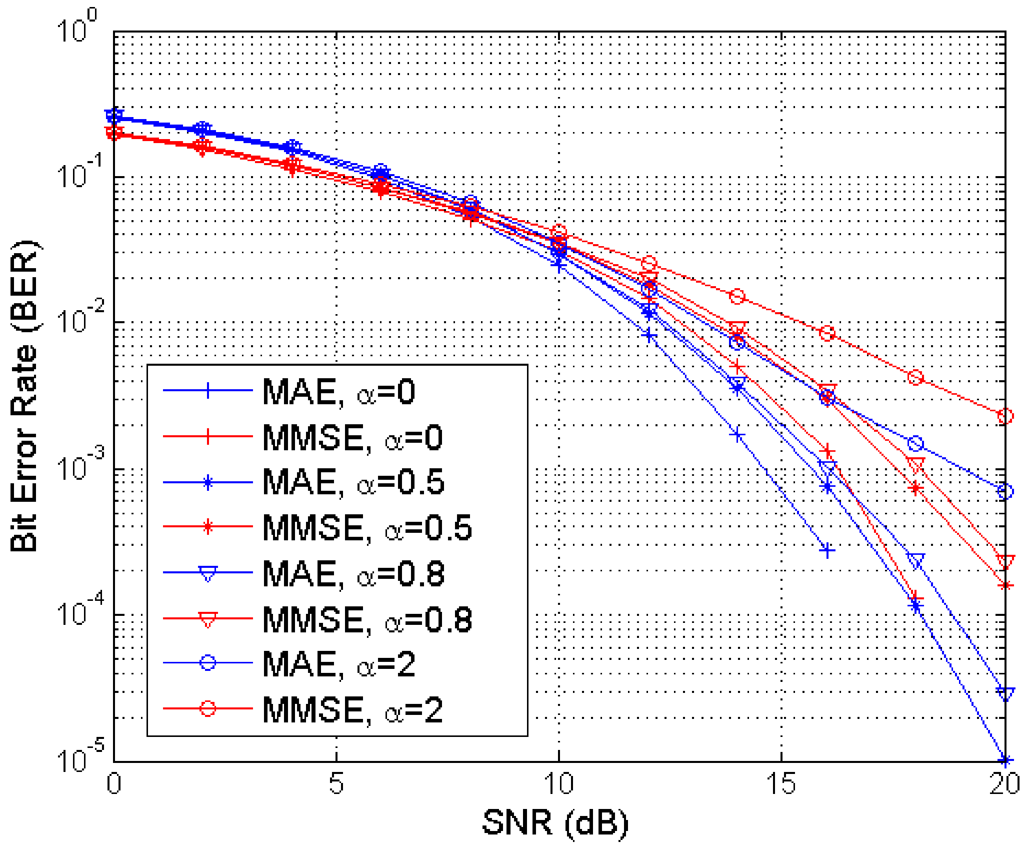


Figure 3.5: BER for the MAE and MMSE equalizer plotted against SNR for a 3-tap channel where the channel taps are perturbed by a bounded error of approximation proportional to α.

where $\mathbf{r}_k^{(+)}$ denotes the constellation points in $C_d^{(+)}$, and $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_{2^{L_h+L_c-2}}]$ are the weighting parameters used to define the points in $CH(C_d^{(+)})$. The quadratic programming required to determine $||v_+||$ is computationally intensive [37], making the MAE equalizer impractical to design for long channels and/or time-varying systems. In order to avoid solving (3.7), we present an efficient algorithm to find the decision hyperplane directly from the channel. The proposed geometric approach is applicable only for BPSK signalling. Further studies can be conducted to extend the algorithm to M-ary modulated signals but are outside the scope of this thesis.

3.2.1 Channel Vectors

To illustrate the proposed method, we define the channel vector $\boldsymbol{g}_i$, $0 \leq i \leq L_h + L_c - 2$, which denotes the i-th column of the channel matrix $\mathbf{H}$,

$$\boldsymbol{g}_0 \triangleq \begin{bmatrix} h_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \boldsymbol{g}_{L_h-1} \triangleq \begin{bmatrix} h_{L_h-1} \\ h_{L_h-2} \\ \vdots \\ h_0 \end{bmatrix}, \ldots, \boldsymbol{g}_{L_h+L_2-2} \triangleq \begin{bmatrix} 0 \\ \vdots \\ 0 \\ h_{L_h-1} \end{bmatrix}. \tag{3.8}$$

For the $L_c = 2$ scenario, the properties of 2-dimensional channel vectors $\boldsymbol{g}_i$ have been well studied in [38] to blindly identify the communication channel. For application of the MAE equalizer, we do not want to restrict L_c to 2 but instead wish to consider MAE equalizers of arbitrary length. Hence, we need to geometrically analyze the constellation sets for an arbitrary L_c-dimensional space. In the analysis below, we focus on $C_d^{(+)}$. Given the symmetry imposed by the linear channel and BPSK signaling, the results will hold in a symmetric fashion for $C_d^{(-)}$. We start our analysis by rewriting the constellation equation in vector form:

$$C_d^{(+)} : \{\boldsymbol{r}_k = \boldsymbol{g}_{d+1} + \sum_{i \neq d} \boldsymbol{g}_i x_{k-i}\}, \quad x_{k-i} \in \{\pm 1\}. \tag{3.9}$$

Given the decision delay d, the constellation points are binary (BPSK) combinations of the channel vectors.

3.2.2 Channel Vector Properties

Before describing the efficient MAE equalizer design, we present some important definitions and prerequisite properties of the channel vectors.

Definition 2. *In L_c-dimensional space, the convex hull $CH(C_d^{(+)})$ has L_c types of facets: 0-dimensional facets are called vertices, 1-dimensional facets are called edges, and m-dimensional facets are called m-facets, where $2 \leq m \leq L_c - 1$ [39].*

Definition 3. *A support hyperplane H_s of the constellation $C_d^{(+)}$ is a hyperplane that intersects $C_d^{(+)}$ and such that $C_d^{(+)}$ is contained entirely in one of the half-spaces of H_s [40].*

Assume that no taps of the linear channel h are equal to 0; this assumption guarantees that no pair of vectors $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{L_h+L_c-2}$ are parallel to each other. The theorems below are proved for the case in which $CH(C_d^{(+)})$ and $CH(C_d^{(-)})$ do not overlap. In the case of overlap, a reduced convex hulls method [41] or a decision feedback equalizer structure [20] can be applied to separate the two convex hulls.

Theorem 1. *For an L_c-dimensional convex hull $CH(C_d^{(+)})$, every edge E is parallel to one of the channel vectors $\boldsymbol{g}_i$, where $i \in \{0, 1, \ldots, d-1, d+1, \ldots, L_h + L_c - 1\}$, and E has length $2||\boldsymbol{g}_i||$.*

Proof of Theorem 1. See Appendix A.1 □

Theorem 1 shows that all the edges of the convex hull $CH(C_d^{(+)})$ is determined by the channel vectors. It also implies that there exists a one-to-many mapping between a channel vector and the convex hull edges. With this theorem, we are able to determine the 1-dimensional facets of $CH(C_d^{(+)})$ from the channel vectors, which is the fundamental for the study of high-dimensional facets.

Theorem 2. *The constellation $C_d^{(+)}$ and corresponding convex hull $CH(C_d^{(+)})$ are symmetric around the point $P(\boldsymbol{g}_d)$, where $P(\boldsymbol{g}_d)$ denotes the point corresponding to the vector $\boldsymbol{g}_d$ in the signal space and d is the decision delay. Hence the facets of $CH(C_d^{(+)})$ come in parallel pairs.*

Proof of Theorem 2. See Appendix A.2 □

Theorem 3. *For an L_c-dimensional convex hull $CH(C_d^{(+)})$, every $(L_c - 1)$-facet is parallel to a channel hyperplane constructed by $L_c - 1$ of the $L_h + L_c - 2$ channel vectors.*

Proof of Theorem 3. See Appendix A.3 □

For example, when $L_c = 3$ and $L_h = 3$, the convex hull constructed lies in a 3-dimensional space. Each 2-facet is a face of $CH(C_d^{(+)})$, which is parallel to one of the planes constructed by two channel vectors. There are $\binom{4}{2} = 6$ possible vectors connecting two points in $C_d^{(+)}$ that are parallel to this plane.

From Theorems 2 and 3, we observe that for a linear channel, the corresponding convex hull $CH(C_d^{(+)})$ has a highly symmetric structure. Its centroid is determined by the channel vector $\boldsymbol{g}_d$. All the facets of the convex hull are symmetric with respect to the centroid, and the hyperplanes on which they are located are parallel to the hyperplanes constructed by the channel vectors with a shifted distance.

Theorem 4. *For an L_c-dimensional convex hull $CH(C_d^{(+)})$ generated by a linear channel* $\mathbf{h}$*, the minimum distance point v_+ will be located on an $(L_c - 1)$-facet.*

Proof of Theorem 4. See Appendix A.4 □

Combining the four theorems above, we can conclude that

1. The support hyperplane, on which an $(L_c - 1)$-facet of $CH(C_d^{(+)})$ lies, can be obtained by shifting one of the channel hyperplanes from the origin to a point determined by channel vectors.

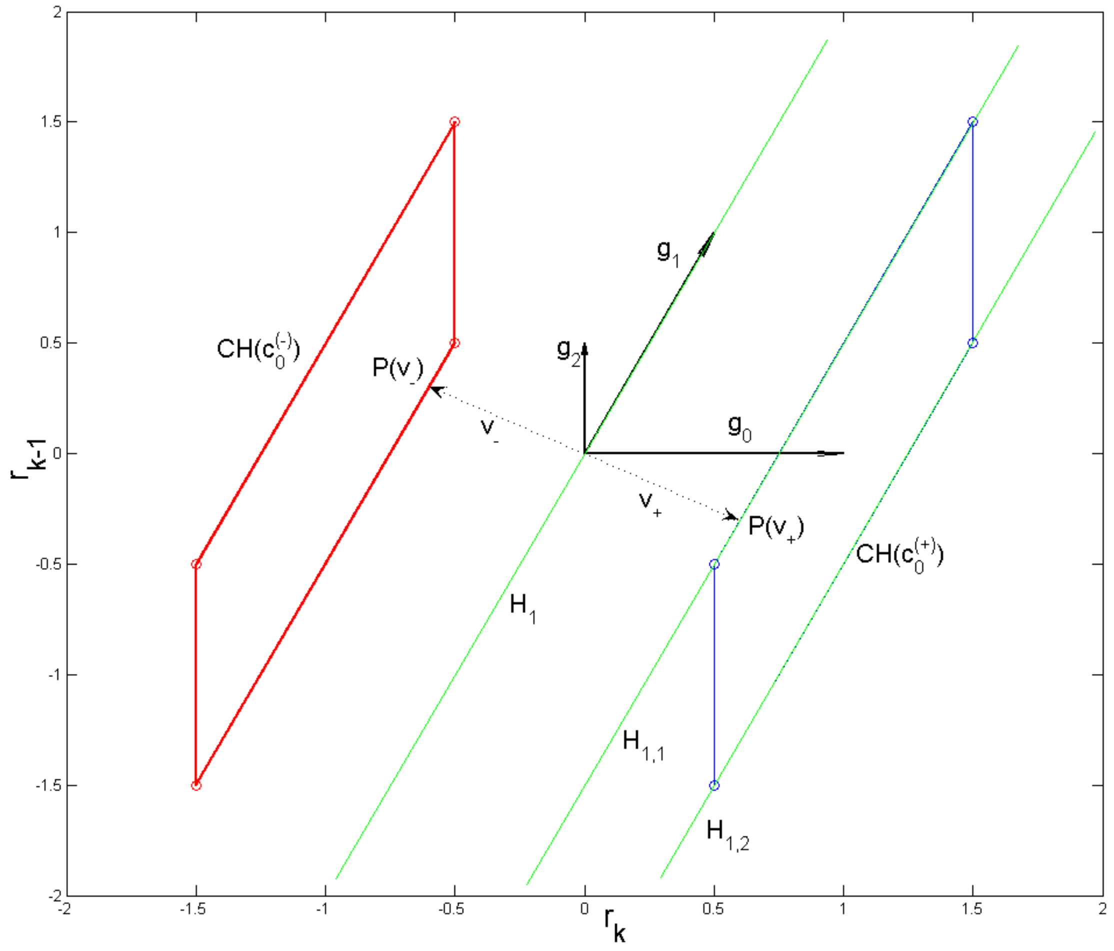


Figure 3.6: Channel vectors, optimal decision hyperplane and convex hulls with corresponding support hyperplane for a 2-dimensional equalizer operating on a 2-tap channel $h[n] = \delta[n] + 0.5\delta[n-1]$.

2. The channel hyperplane that is parallel to the facet containing v_+ will be the decision hyperplane H_0 of the MAE equalizer.

Thus, the design of the MAE equalizer is simplified to determining which of the channel hyperplanes is the decision hyperplane. A two-dimensional example is shown in Figure 3.6 with $L_c = 2$ and $L_h = 2$. The hyperplanes (lines) $H_{1,1}$ and $H_{1,2}$, on which two ($L_c - 1 = 1$)-facets (edges) of $CH(C_d^{(+)})$ are located, are the hyperplanes parallel to the line H_1 determined by the channel vector $\boldsymbol{g}_1$. In this example, v_+ is on the line $H_{1,1}$, and hence H_1

is the decision hyperplane for the MAE equalizer.

3.2.3 Geometric Algorithm for MAE Equalizer Design

Before describing the algorithm for finding the shifted channel hyperplane containing v_+, we first define the *support margin* of a hyperplane.

Definition 4. *The support margin of a support hyperplane of $CH(C_d^{(+)})$ is the smallest Euclidean distance from the support hyperplane to the decision hyperplane, or equivalently (given the constellation symmetry) the smallest Euclidean distance from the support hyperplane to the origin.*

For an L_c-dimensional constellation $C_d^{(+)}$, there are $\binom{L_h+L_c-2}{L_c-1}$ channel hyperplanes denoted by H_m, $m \in \left\{1, \dots, \binom{L_h+L_c-2}{L_c-1}\right\}$, and determined by the $L_c - 1$ channel vectors $\{\boldsymbol{g}_{i_1}, \dots, \boldsymbol{g}_{i_{L_c-1}}\}, \{i_1, \dots, i_{L_c-1}\} \in \{1, \dots, d-1, d+1, \dots, L_c - 1\}$. Each H_m can be shifted from the origin to 2^{L_h-1} different points $P(\boldsymbol{p}_{m,k} = \boldsymbol{g}_d + \sum_{j \notin \{i_1, \dots, i_{L_c-1}, d\}} \boldsymbol{g}_j x_{k-j})$. Let us denote the shifted hyperplanes by $H_{m,k}$, $k \in \{1, \dots 2^{L_h-1}\}$.

A two-dimensional example is shown in Figure 3 with $L_h = 3$ and $L_c = 2$, to illustrate the relationship between the hyperplanes. (While $L_c < L_h$ is unlikely to be chosen in practice, it facilitates explanation of the various hyperplanes using a two-dimensional figure.) There are a total of $\binom{3}{1}$ channel hyperplanes (lines) H_m, $m = 1, 2, 3$. For the hyperplane H_2 determined by the channel vector $\boldsymbol{g}_2$, there are $2^{L_h-1} = 4$ shifted hyperplanes $H_{m=2,k=1,\dots,4}$. The relationship between hyperplanes can be described as follows:

1. For each m, $m \in \left\{1, \dots, \binom{L_h+L_c-2}{L_c-1}\right\}$, $H_{m,k}$, with $k \in \{1, \dots 2^{L_h-1}\}$, are shifted from the channel hyperplane H_m. The hyperplane $H_{m,k}$ with minimum Euclidean distance $D_{m,k}$ from the origin is the marginal support hyperplane $H_m^{(+)}$ parallel to H_m. We can find an $H_m^{(+)}$ corresponding to each channel hyperplane H_m.

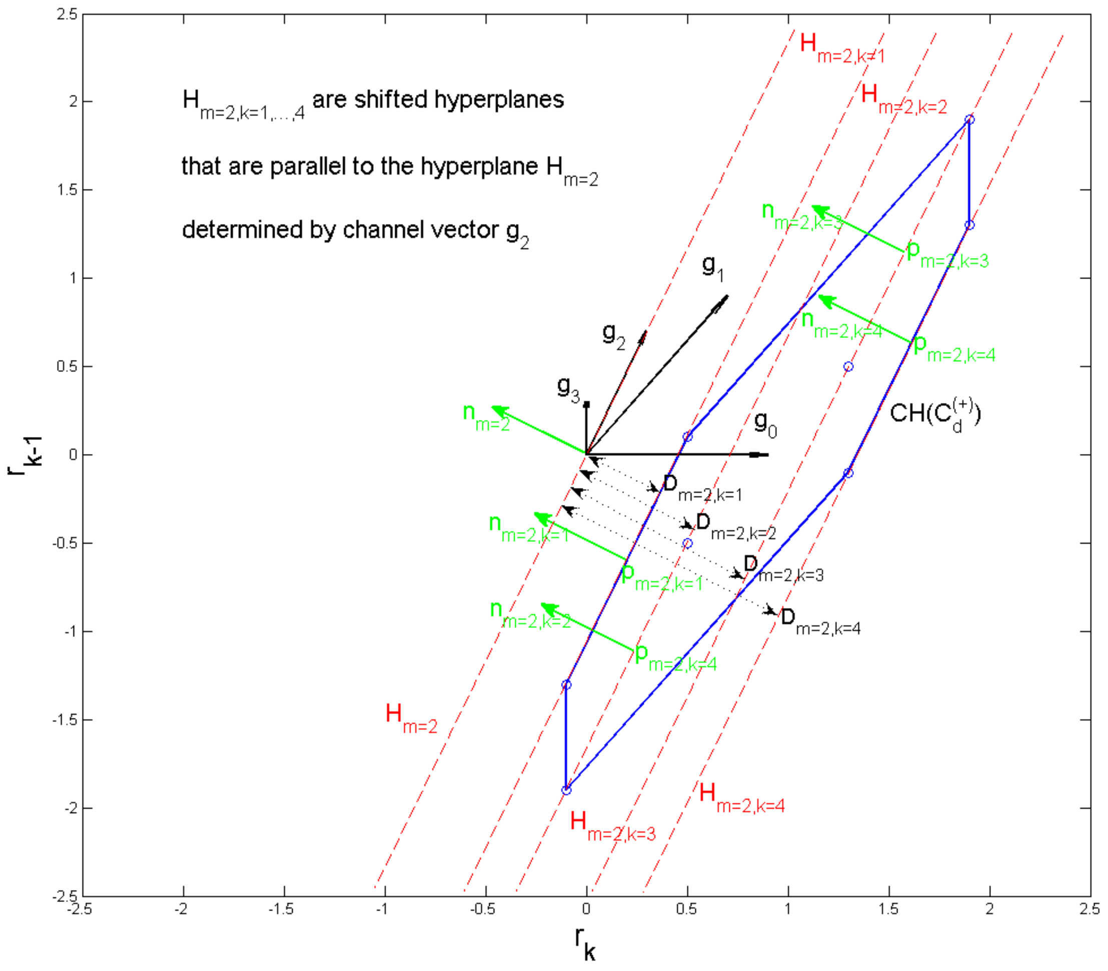


Figure 3.7: Illustration of hyperplane H_m, shifted hyperplanes $H_{m,k}$, and their corresponding normal vectors for a 3-tap channel and 2-tap MAE equalizer with $d = 0$.

2. The channel hyperplane H_m corresponding to the $H_m^{(+)}$ with the maximum support margin is the decision hyperplane H_o for the MAE equalizer.

Given these relationships, we can find the MAE decision hyperplane H_0 using the following two-step procedure:

$\boxed{\text{Step 1.}}$ Find the marginal support hyperplane $H_m^{(+)}$ parallel to each channel hyperplane H_m, and find D_m, the associated support margin.

$$H_m^{(+)} = H_{m,k} \quad \text{for} \quad k = \underset{k}{\text{argmin}}\, D_{m,k}. \tag{3.10}$$

$D_{m,k}$ is the signed distance from the origin to the shifted hyperplane $H_{m,k}$, computed as

$$D_{m,k} = \frac{< \mathbf{0} - \boldsymbol{p}_{m,k}, \boldsymbol{n}_{m,k} >}{||\boldsymbol{n}_{m,k}||}, \tag{3.11}$$

which projects the vector $\boldsymbol{p}_{m,k}$ onto the direction of the normal vectors $\boldsymbol{n}_{m,k}$. The support margin D_m for each marginal support hyperplane $H_m^{(+)}$ is defined as

$$D_m = \begin{cases} \min ||D_{m,k}||, & \text{if } D_{m,k=k_1} \cdot D_{m,k=k_2 \neq k_1} > 0 \\ 0, & \text{if } D_{m,k=k_1} \cdot D_{m,k=k_2 \neq k_1} \leq 0 \end{cases}. \tag{3.12}$$

The condition $D_{m,k=k_1} \cdot D_{m,k=k_2,k_2 \neq k_1} > 0$ ensures that the origin and the points of $C_d^{(+)}$ are not on the same side of $H_{m,k}$. If the origin and $C_d^{(+)}$ lie on the same side of one of the shifted hyperplanes $H_{m,k}$, then there is no marginal support hyperplane parallel to H_m, and we set $D_m = 0$. In Fig. 3.7, $H_{m=2,k=1}$, with minimum distance $||D_{m=2,k=1}||$, is the marginal support hyperplane parallel to $H_{m=2}$.

$\boxed{\text{Step 2.}}$ Find the marginal support hyperplane with maximum support margin. The corresponding channel hyperplane is the decision hyperplane H_0 of the MAE equalizer.

$$H_o = H_m, \text{ where } m = \underset{m}{\operatorname{argmax}}\, D_m \tag{3.13}$$

Summarizing the computations required to find H_0, we performed $2^{L_h-1}\binom{L_h+L_c-2}{L_c-1}$ computations of the distance from the origin to a hyperplane. If we denote the number of the constellation points in $C_d^{(+)}$ by $N = 2^{L_h+L_c-2}$, the complexity of the proposed algorithm is $O(N \log N)$.

3.2.4 Empirical Complexity Comparison

To illustrate the reduced complexity of the proposed method, the design of the MAE equalizer is carried out for different length channels and equalizers using both the quadratic programming design method (MAE-QP) and the proposed geometric design method (MAE-GM). Since the complexity of the quadratic programming approach cannot be expressed in closed form, MATLAB is used to compare the complexity (in terms of required CPU time) of the two approaches for 100 randomly generated length-3 channels. The MATLAB function `fmincon` is used to solve the quadratic programming problem, and the initial point is chosen to be the constellation point closest to the origin. The simulations were implemented on a computer equipped with i7-920 CPU, 6G RAM, and Microsoft Windows 7 x64. Table 3.2 shows the results for a 3-tap channel and an MAE equalizer of length $L_c = 3, 5, 7$, and 9. Table 3.3 shows the results for a 5-tap MAE equalizer and a channel of length $L_h = 2, 3, 4$, and 5. The complexity reduction achieved by the proposed geometric approach increases dramatically with equalizer length and channel length. For $L_h = 3$ and $L_c = 9$, complexity is reduced by a factor of 10^5.

Table 3.2: CPU Time of MAE-QP and MAE-GM to Finding H_0, for a 3-tap channel

CPU Time (seconds) for $L_h = 3$				
	$L_c = 3$	$L_c = 5$	$L_c = 7$	$L_c = 9$
MAE-QP	0.1	4.20	117.90	8380.70
MAE-GM	0.0071	0.013	0.019	0.0292

Table 3.3: CPU Time of MAE-QP and MAE-GM to Finding H_0, for a 5-tap MAE equalizer

CPU Time (seconds) for $L_c = 5$				
	$L_h = 2$	$L_h = 3$	$L_h = 4$	$L_h = 5$
MAE-QP	2.0320	4.20	42.4541	388.1398
MAE-GM	0.0082	0.013	0.0991	0.2870

3.3 MAE Equalizer for Time-varying Channels Using Adaptive Constellation Mapping

We have introduced a computationally efficient method that makes MAE equalizer design practical for channels with longer delay spreads. The MAE equalizer is well suited for time-invariant channels, since the design process must be performed only once. Even in the presence of mild channel estimation error, the MAE equalizer provides strong performance when SNR is relatively high. For channels whose impulse response varies significantly over time, however, the MAE equalizer presents prohibitive complexity, since the multi-dimensional search must be repeated each time the noise-free signal constellation changes significantly. Therefore, we propose adaptation of the constellation to match a fixed MAE equalizer to leverage the high performance benefits of MAE equalization without paying the high price of redesign each time the channel changes.

3.3.1 Time-varying Channel Model

Consider a model channel $\mathbf{h}$ that reflects the true channel reasonably well in terms of number of taps and their values, and suppose an MAE equalizer has been designed for the noise-free signal constellation $C(\mathbf{h})$ generated by $\mathbf{h}$. Let $\mathbf{h}'$ denote a perturbation of the model channel $\mathbf{h}$ due to, for example, changes in the environment through which a wireless signal propagates. In the proposed adaptive structure, the MAE equalizer is preceded by a linear pre-filter that is designed to map the noise-free signal constellation $C(\mathbf{h}')$ generated by the modified channel $\mathbf{h}'$ to the original constellation $C(\mathbf{h})$. The pre-filter $\mathbf{c} = [c_0, c_1, \cdots, c_{L_c-1}]$ is designed to minimize (or approximately minimize when the modified channel is unknown) the mean square error (MSE) between the model noise-free signal constellation and the constellation at the output of the pre-filter.

We employ a simple Markov model for a time-varying channel in the equalizer design and performance evaluation presented in this section. The evolution of the vector of channel taps over time is given by

$$\mathbf{h}_{k+1} = \mathbf{h}_k + \beta \boldsymbol{q}_k, \tag{3.14}$$

where $\boldsymbol{q}_k$ is a unit variance AWGN vector and β controls the extent of variation of the channel. First, we address the case in which the time-varying channel is known at each time instant. (In practice, the time-varying impulse response will not be known exactly but may be estimated by transmitting training data, for example.) Second, we address the case in which the time-varying channel is unknown, and the pre-filter is adapted in an LMS-based decision-directed mode.

3.3.2 MAE Equalizer with Adaptive Pre-filter for Known Time-varying Channel

The system model for the adaptive MAE equalizer when the time-varying channel is known is shown in Fig. 3.8. The MSE to be minimized is the Euclidean distance between points in the model constellation $C(\mathbf{h})$ and points in the constellation formed by the convolution of the modified channel and the pre-filter:

$$\begin{aligned} J_k(\mathbf{c}) &= E\{|r_k - r'_k|^2\} \\ &= E\{|\mathbf{h}_b^T \mathbf{x}_k - (\mathbf{c}^T \mathbf{H}_k \mathbf{x}_k + \mathbf{c}^T \boldsymbol{w}_k)|^2\}, \end{aligned} \tag{3.15}$$

where $J_k(\mathbf{c})$ denotes the MSE at time index k, $\mathbf{h}_b^T$ denotes the first row of $\mathbf{H}$, and $\mathbf{H}_k$ denotes the matrix of the time-varying channel at time k. Taking the derivative of the MSE with respect to the pre-filter weight vector $\mathbf{c}$ yields

$$\begin{aligned} \frac{\partial J_k(\mathbf{c})}{\partial \mathbf{c}^T} &= -\mathbf{H}_k E\{|\mathbf{x}_k|^2\}\mathbf{h}_b + \mathbf{H}_k E\{|\mathbf{x}_k|^2\}\mathbf{H}_k^T \mathbf{c} + E\{|\boldsymbol{w}_k|^2\}\mathbf{c} \\ &= -\mathbf{H}_k \mathbf{h}_b + \mathbf{H}_k \mathbf{H}_k^T \mathbf{c} + \sigma^2 I_{L_c} \mathbf{c}, \end{aligned} \tag{3.16}$$

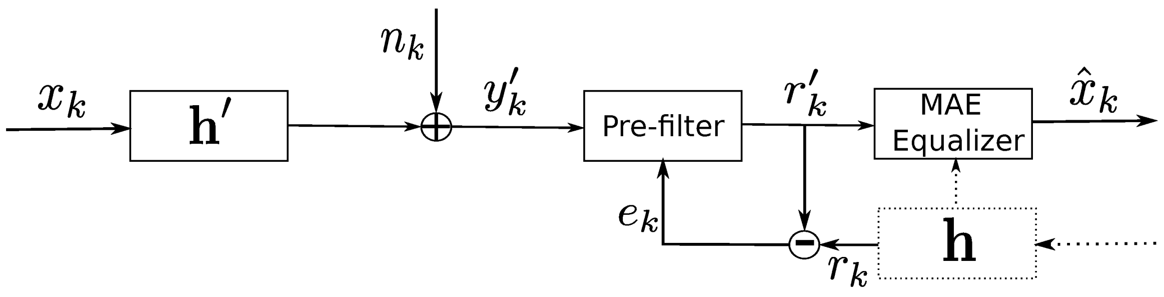


Figure 3.8: Block diagram of channel and receiver structure for adaptive MAE equalizer when the time-varying channel is assumed known.

where I_{L_c} denotes the $L_c \times L_c$ identity matrix. Setting the derivative to zero, the pre-filter weights are found to be

$$\begin{aligned} \mathbf{c} &= (\mathbf{H}_k\mathbf{H}_k^T + \sigma^2 I)^{-1}\mathbf{H}_k\mathbf{h}_b \qquad (3.17) \\ &= \Gamma^{-1}\boldsymbol{P}. \end{aligned}$$

In the second line of (3.18), the pre-filter weight vector is expressed in the form of the Wiener-Hopf equations [42], but the cross-correlation term $\boldsymbol{P}$ includes contributions from both the original and the modified channels.

Fig. 3.9 and Fig. 3.10 provide two examples of original, modified, and pre-filtered constellations occurring for the given channel model with $\beta = 0.1$. Fig. 3.9 considers a two-tap channel $\mathbf{h} = [0.86, 0.51]$ and modified channel $\mathbf{h}' = [0.96, 0.28]$. A length $L_c = 2$ pre-filter is used to map $C(\mathbf{h}')$ to $C(\mathbf{h})$. The modified constellation is quite skewed relative to the original, but the pre-filter succeeds in mapping it to nearly exactly its original form. Note that, when the pre-filter is appended, the effective length of the channel is $L_h + L_c - 1$, and hence the number of constellation points increases from $2^{L_h+L_c-1}$ to $2^{L_h+L_c+L_c-2}$. As a result, the pre-filtering maps multiple constellation points onto a single point of the original noise-free signal constellation.

Fig. 3.10 considers a three-tap channel $\mathbf{h} = [0.31, 0.91, 0.22]$ and modified channel

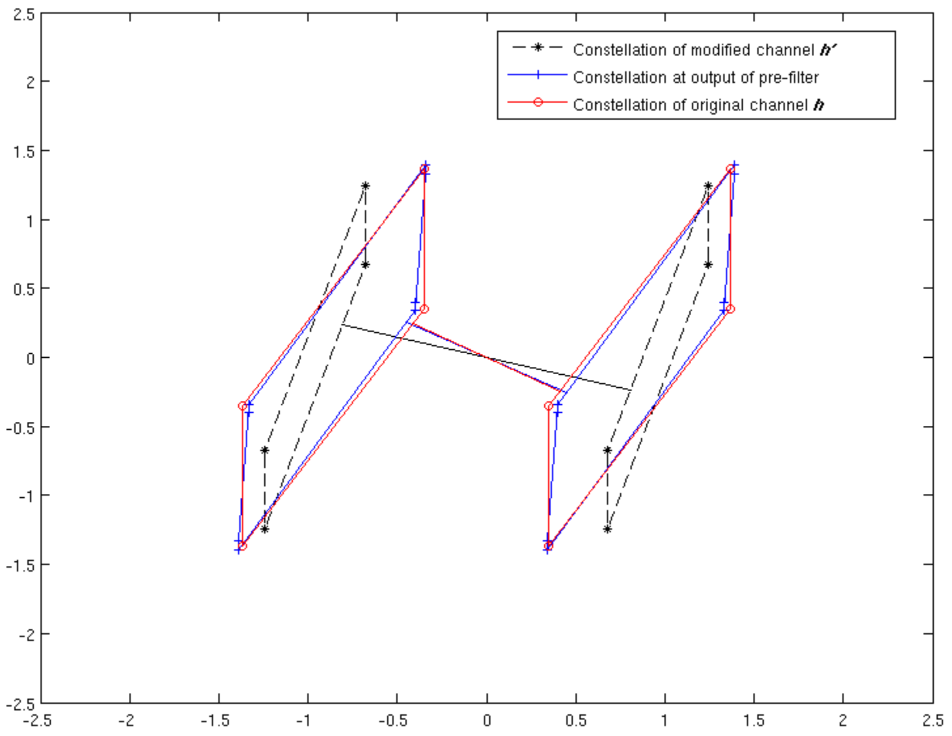


Figure 3.9: Comparison among pre-filtered, modified, and original noise-free signal constellations for $\mathbf{h} = [0.86, 0.51]$ and $\beta = 0.1$.

$\mathbf{h}' = [0.066, 0.9492, 0.3078]$ with a length $L_c = 3$ pre-filter. Again, the MSE-based pre-filter is able to map the modified constellation to one much closer to the original, though the difference between the original and pre-filtered constellations is visibly larger here than in Fig. 3.9. Though some of the loss in performance may be due to the particular channel modification that took place in this example, it is also likely due to the fact that the constellation size has grown exponentially with channel length, posing a greater challenge to the pre-filter.

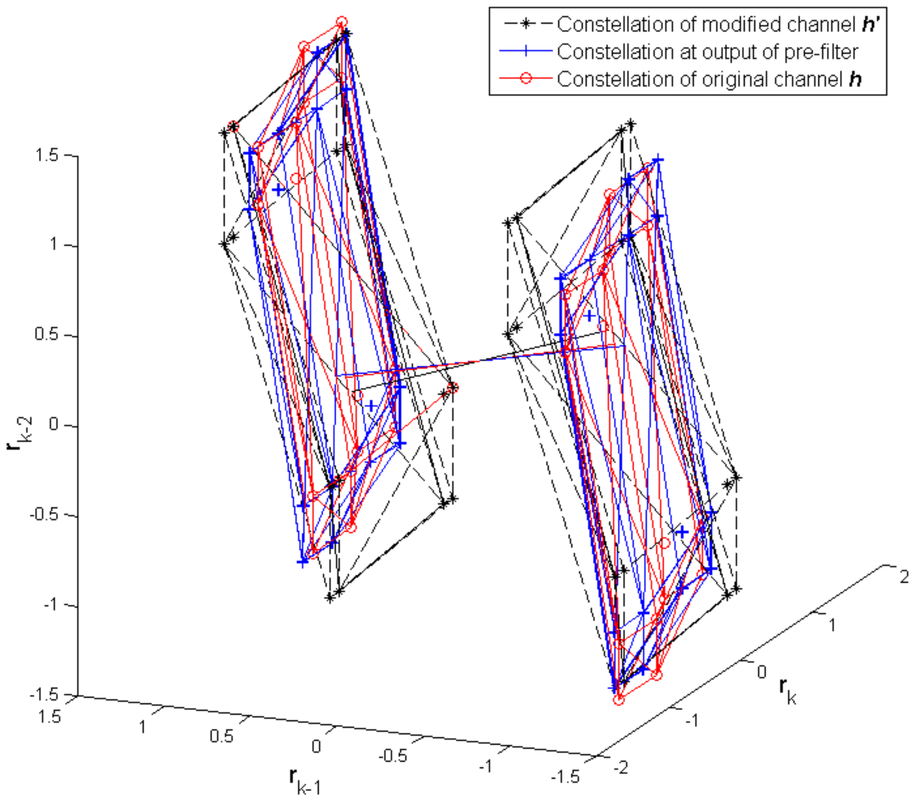


Figure 3.10: Comparison among pre-filtered, modified, and original noise-free signal constellations for $\mathbf{h} = [0.31, 0.91, 0.22]$ and $\beta = 0.1$.

3.3.3 Simulation Results for the Adaptive MAE Equalizer with Knowledge of Time-varying Channels

In order to evaluate the performance of the proposed adaptive MAE equalizer, the algorithm has been simulated and its performance compared to that of a conventional MMSE linear equalizer. A three-tap initial known channel $\mathbf{h} = [0.31, 0.91, 0.22]$ is used for the simulations; three-tap and five-tap pre-filters are applied prior to MAE equalization. For fair comparison, MMSE equalizers of lengths five and seven (the effective length of the combined pre-filter and MAE equalizer) are considered, as well. Two million symbols are transmitted to generate each performance curve. The decision delay for both receivers is set to $d = 1$.

We assume that the time-varying channel is known at each time index k. Fig. 3.11 show the bit error rate as a function of SNR achieved by the adaptive MAE equalizer and by the MMSE equalizer. Both algorithms have been simulated for β value of 0.1. The adaptive MAE equalizer consistently outperforms the MMSE equalizer in the high SNR regime, and the performance gain increases with increasing SNR. (As expected the asymptotic efficiency-based detector performs slightly worse than the MMSE equalizer for low SNR, since it is not designed to overcome the effects of additive noise.) The simulations performed show an improvement of up to approximately 3 dB for $\beta = 0.1$.

3.3.4 MAE Equalizer with Adaptive Pre-filter for Unknown Time-varying Channel

Transmitting training data to compute channel estimates reduces available bandwidth for information, particularly when the channel is changing rapidly. Reducing the frequency with which training data must be transmitted improves spectral efficiency. In order to achieve this, we assume that the training sequence is transmitted only once, and the initial channel is estimated using this training sequence at the beginning of the transmission. For such cases, we can consider an alternative approach to pre-filtering in which the pre-filter is adapted in a decision-directed manner. Specifically, a variable step size (VSS) LMS algorithm [43] is employed to update the pre-filter taps using decisions made on previously transmitted

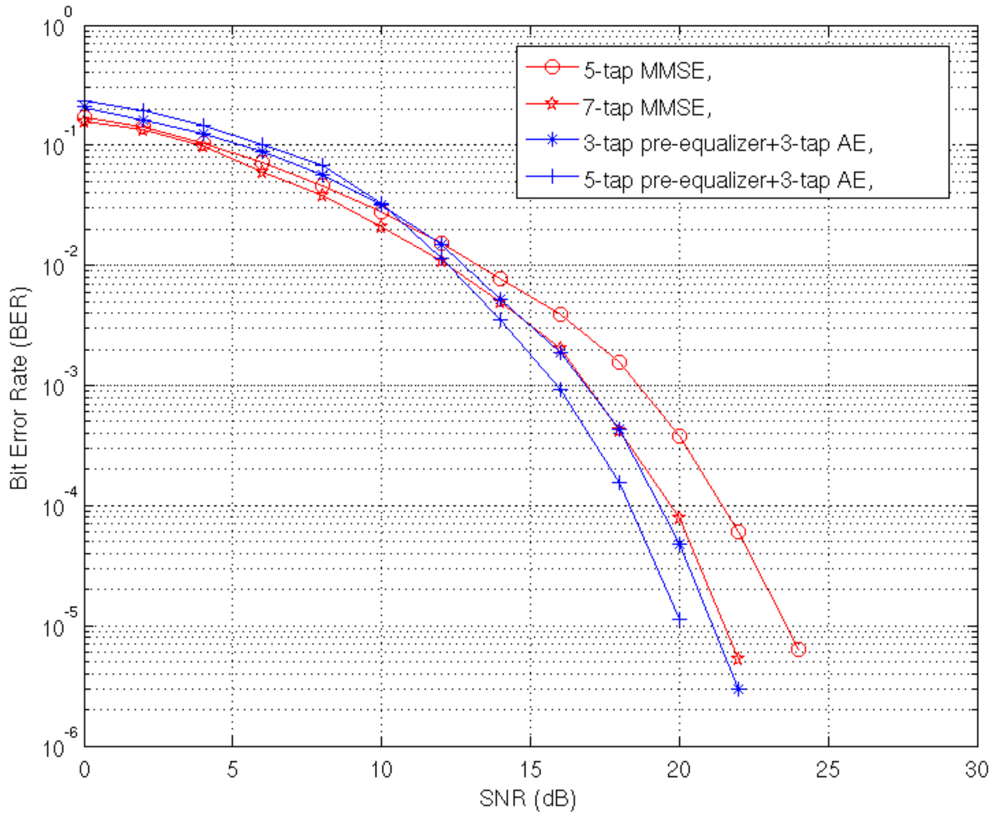


Figure 3.11: Performance of the adaptive MAE equalizer and the linear MMSE equalizer on a three-tap time-varying channel for whose impulse response is assumed known at each k. The variation of the channel is set as $\beta = 0.1$.

symbols. A block diagram of the channel and receiver structure for the decision-directed pre-filter is shown in Figure 3.12.

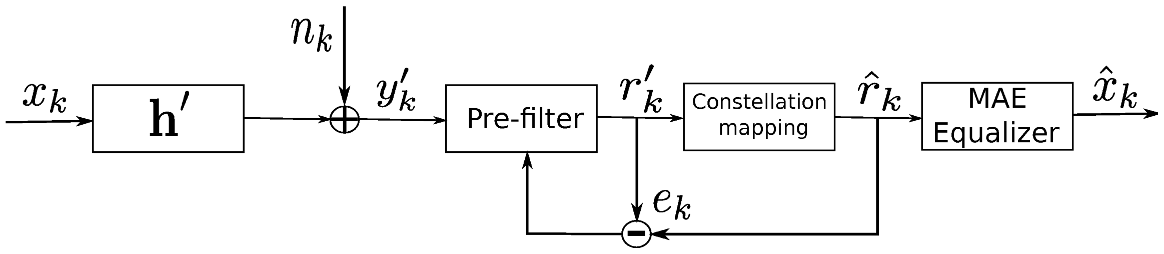


Figure 3.12: Block diagram of channel and receiver structure for adaptive MAE equalizer when pre-filter tap weights are updated via the VSS-LMS algorithm.

The motivation for employing the VSS-LMS algorithm is to provide better tracking capability of the time-varying channel. Just like the conventional LMS algorithm, our proposed VSS algorithm is a gradient search algorithm that computes the coefficients of the equalizer by minimizing the MSE. In this application, MSE is defined as the average squared Euclidean distance between constellation points in the initial channel and points in the constellation formed by the convolution of the time-varying channel and the pre-filter at time index k. The algorithm is expressed as follows [43]:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + 2\mu_k e_k r'_k, \tag{3.18}$$

where μ_k is the adaptation step size at time index k, and e_k is the predication error defined as $e_k = r'_k - \hat{r}_k$. In the conventional LMS algorithm, μ_k is a constant, while the step size is adjusted by the prediction error in the VSS-LMS algorithm. When the channel varies at each time k, a large e_k will increase the step size to provider faster tracking, and a small e_k will lead to a decrease in the step size to yield smaller misadjustment. We refer to the procedures in [43] to adjust the step size as

$$\mu_{k+1} = \begin{cases} \mu_{\max}, & \text{if} \quad \alpha\mu_k + \gamma e_k^2 > \mu_{\max}, \\ \mu_{\min}, & \text{if} \quad \alpha\mu_k + \gamma e_k^2 < \mu_{\min}, \\ \alpha\mu_k + \gamma e_k^2, & \text{otherwise,} \end{cases} \tag{3.19}$$

with

$$0 < \alpha < 1, \quad \gamma > 0, \tag{3.20}$$

where α and γ are two weighting factors used to control the change of step size. They are chosen empirically based on simulations. $\mu_{\min}$ and $\mu_{\max}$ are chosen based on the covariance matrix R of the input signal to the pre-filter $\mathbf{y}'$, where $\mathbf{y}' = \mathbf{x}_k\mathbf{h}' + \mathbf{n}_k$. The covariance matrix can be expressed as follows with the assumption that the transmitted signal and channel are independent:

$$\begin{aligned} R &= E\{\mathbf{y}'_k\mathbf{y}'^T_k\} \\ &= E\{\mathbf{h}'\mathbf{x}_k\mathbf{x}_k^T\mathbf{h}'^T\} + E\{\mathbf{n}_k\mathbf{n}_k^T\} \\ &= \sigma_s^2 E\{\mathbf{h}'\mathbf{h}'^T\} + \sigma^2 I, \end{aligned} \tag{3.21}$$

where σ_s^2 is the signal power, and σ^2 is the noise variance. $\mu_{\min}$ is chosen as the step size of the conventional LMS algorithm to provide a minimum level of tracking for time-varying channels,

$$0 < \mu_{\min} < \frac{1}{\lambda_{\max}}, \tag{3.22}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix R. $\mu_{\max}$ is chosen to ensure that the mean-square error of the algorithm is bounded. The condition is defined as

$$\mu_{\min} < \mu_{\max} \leq \frac{2}{3\mathrm{tr}(R)}, \tag{3.23}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. The initial step size μ_0 is chosen to be $\mu_{\max}$. Further details about the implementation of the VSS-LMS algorithm and its convergence properties can be found in [43].

3.3.5 Simulation Results for the Adaptive MAE Equalizer without Knowledge of Time-varying Channels

In order to evaluate the performance of the proposed adaptive MAE equalizer, we first consider the case that the channel is changed only once, and compare its performance to that of an equalizer designed using the VSS-LMS algorithm. In this comparison, we can see the convergence rate of the proposed adaptive MAE equalizer and performance of the pre-filter to map the constellations of the modified channel.

For the simulation of this case, a three-tap initial channel $\mathbf{h} = [0.9123, 0.3619, 0.1917]$ is assumed to be known. The modified channel $\mathbf{h}' = [0.9084, 0.3087, 0.2730]$ is unknown to the equalizers. A three-tap pre-filter is applied prior to MAE equalization. For fair comparison, an MMSE equalizer of length five (the effective length of the combined pre-filter and MAE equalizer) is considered. 1000 blocks of 2000 BPSK symbols are transmitted to generate each performance curve. The decision delay for both receivers is set to $d = 0$. For the VSS-LMS algorithm, we empirically set the weighting factors to $\alpha = 0.97$ and $\gamma = 4.8^{-4}$, which are shown to have good performance in [43] when time-varying channels are generated by using Gaussian random variables with unit variance as we do in the simulation. We used the output of the initial channel $\mathbf{h}$ to compute the covariance matrix and set the conditions for the step size for both methods as $\mu_{\min} = 0.0005$, $\mu_{\max} = 0.1$ according to (3.22) and (3.23). The step size is adjusted using (3.19) to track the variance of the channel by the prediction error.

The performance comparison of the proposed MAE equalizer and the MMSE equalizer is shown in Fig. 3.13. The adaptive MAE equalizer consistently outperforms the MMSE equalizer in the high SNR regime, and the performance gain increases with increasing SNR. As expected the asymptotic efficiency-based detector performs slightly worse than the MMSE equalizer for low SNR, since it is not designed to overcome the effects of additive noise. In Fig. 3.13, we also show the performance for a five-tap MAE equalizer designed for channel $\mathbf{h}$ and a five-tap MAE equalizer designed for $\mathbf{h}'$. These two BER curves provide the worst case when a fixed MAE equalizer is used for the modified channel, and the best case when the modified channel is known to design the corresponding MAE equalizer.

In order to show the performance for tracking the constellation variation, we show the MSE between constellation points after constellation mapping and constellation points for the initial channel as the algorithm converges. It is shown in Fig. 3.14 for SNR equal to 6 and 12 dB. The MSE is averaged over 1000 blocks. We observe that it takes about 200 iterations for the pre-filter to converge at both 6 dB and 12 dB. The steady-state error of the pre-filter at 6 dB is higher than that at 12 dB due to the noise level. We can also visualize the convergence in a 3-dimensional example shown in Fig. 3.15. The pre-filter succeeds in mapping the modified constellation $C(\mathbf{h}')$ to nearly exactly its original form.

In order to better show the improvement achieved by the proposed method for time-varying channels, we used the channel model given in (3.14). The original channel $\mathbf{h} = [0.9123, 0.3619, 0.1917]$ is assumed to be known, and the initial channel $\mathbf{h}_0 = \mathbf{h}$. 1000 blocks of 2000 BPSK symbols are transmitted a time-varying channel as modeled in (3.14). Since we use the same original channel, we also apply the same parameter values for the VSS-LMS algorithm. We compare the proposed method with a five-tap pre-filter and a five-tap MAE equalizer to a ten-tap MMSE equalizer. β is set to be 0.05 to maintain the variance of the channel in a reasonable range.

The performance comparison is shown in Fig. 3.16. The MMSE equalizer outperforms the proposed adaptive MAE method when SNR is low. As SNR increases, the adaptive MAE equalizer shows performance gains over the MMSE equalizer, achieving up to 2 dB

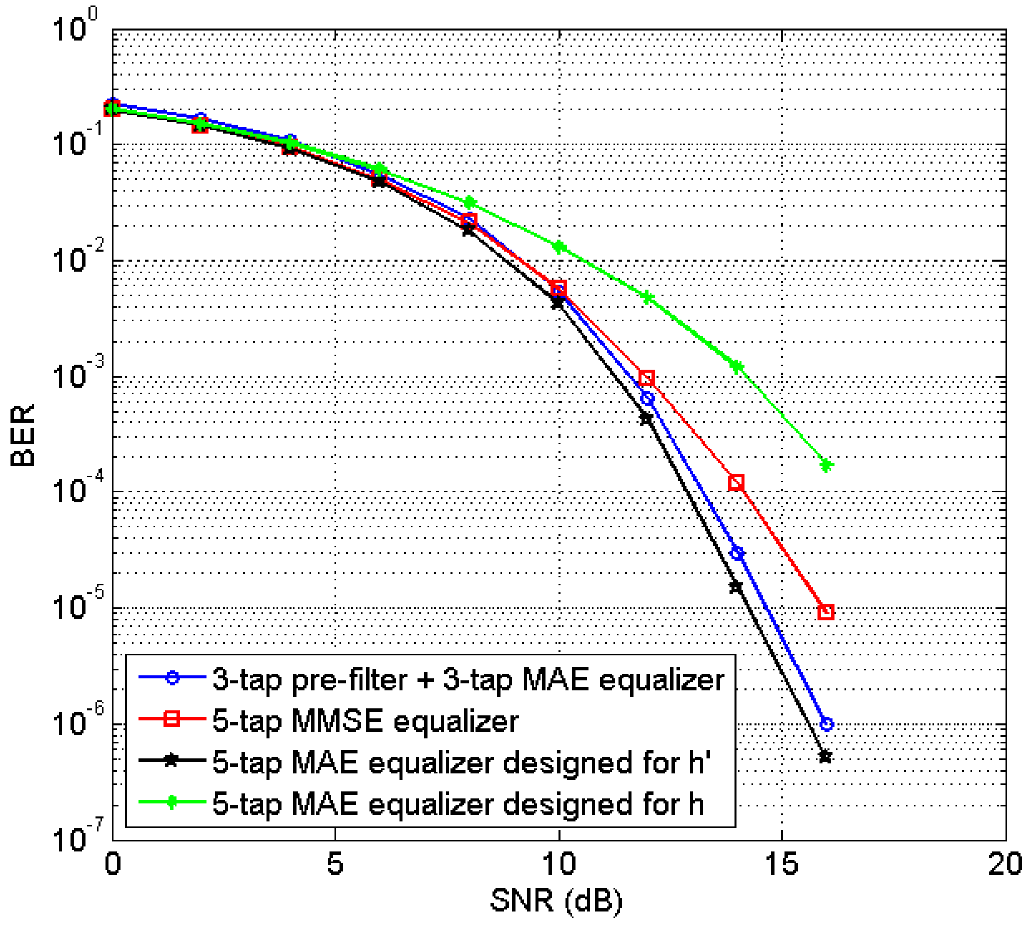


Figure 3.13: Performance comparison between the adaptive MAE equalizer and the MMSE equalizer on a three-tap channel which changes one time from $\mathbf{h} = [0.9123, 0.3619, 0.1917]$ to $\mathbf{h}' = [0.9084, 0.3087, 0.2730]$.

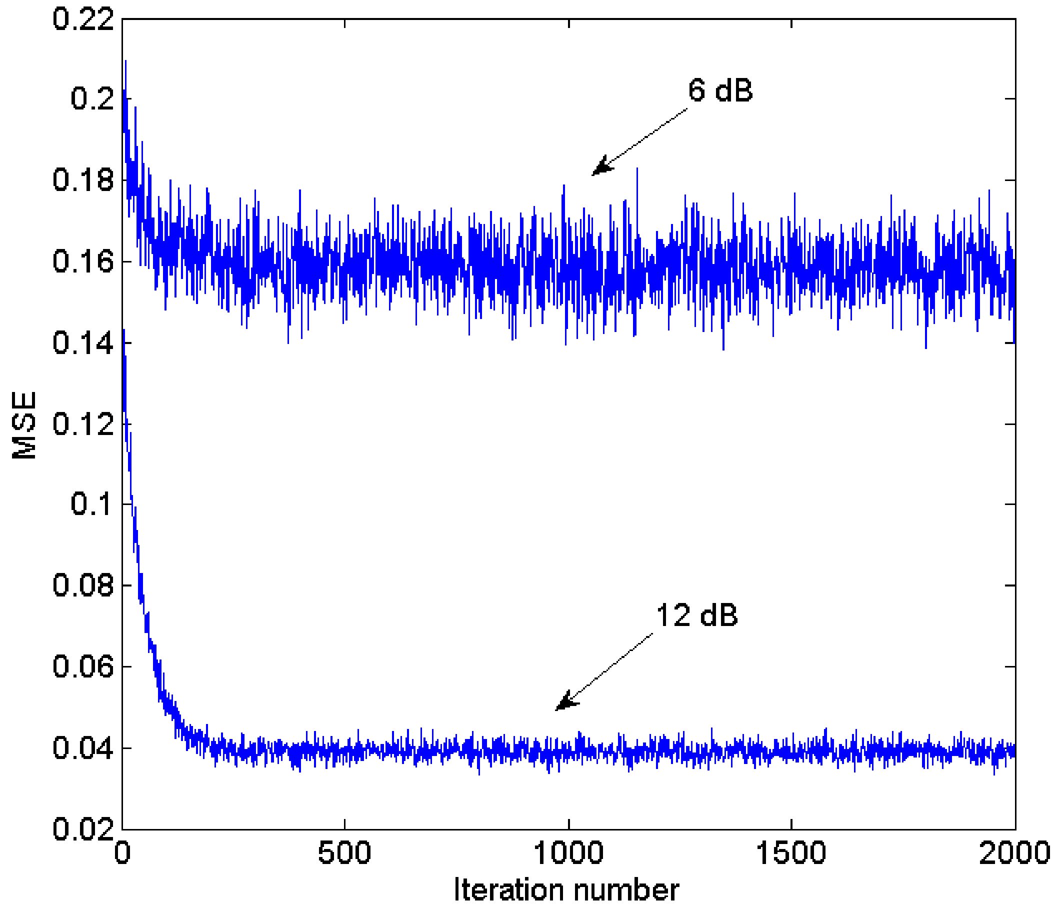


Figure 3.14: Convergence of the constellation points when the channel is changed only once for SNR = $6, 12$ dB.

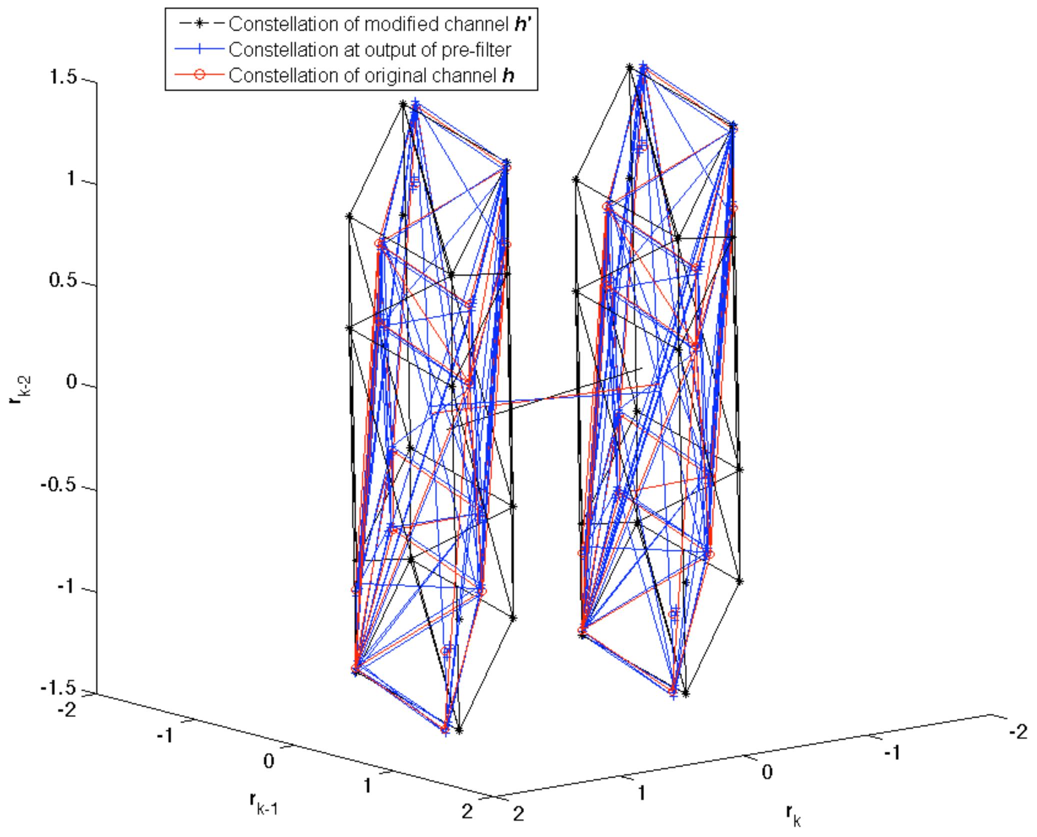


Figure 3.15: Comparison among pre-filtered, modified, and original noise-free constellations for $\mathbf{h} = [0.9123, 0.3619, 0.1917]$ and $\mathbf{h}' = [0.9084, 0.3087, 0.2730]$.

gain for $\beta = 0.05$. We show the average MSE between constellation points after constellation mapping and constellation points for the initial channel as the algorithm converges in Fig. 3.17 for SNR equal to 6 dB and 12 dB. We observe that the pre-filter takes about 400 iterations to converge at 12 dB, while at 6 dB it takes more time (about 500 iterations) to converge. At low SNR, the slow convergence rate and high steady-state error will lead to mismatch of the constellations and performance loss of the MAE equalizer.

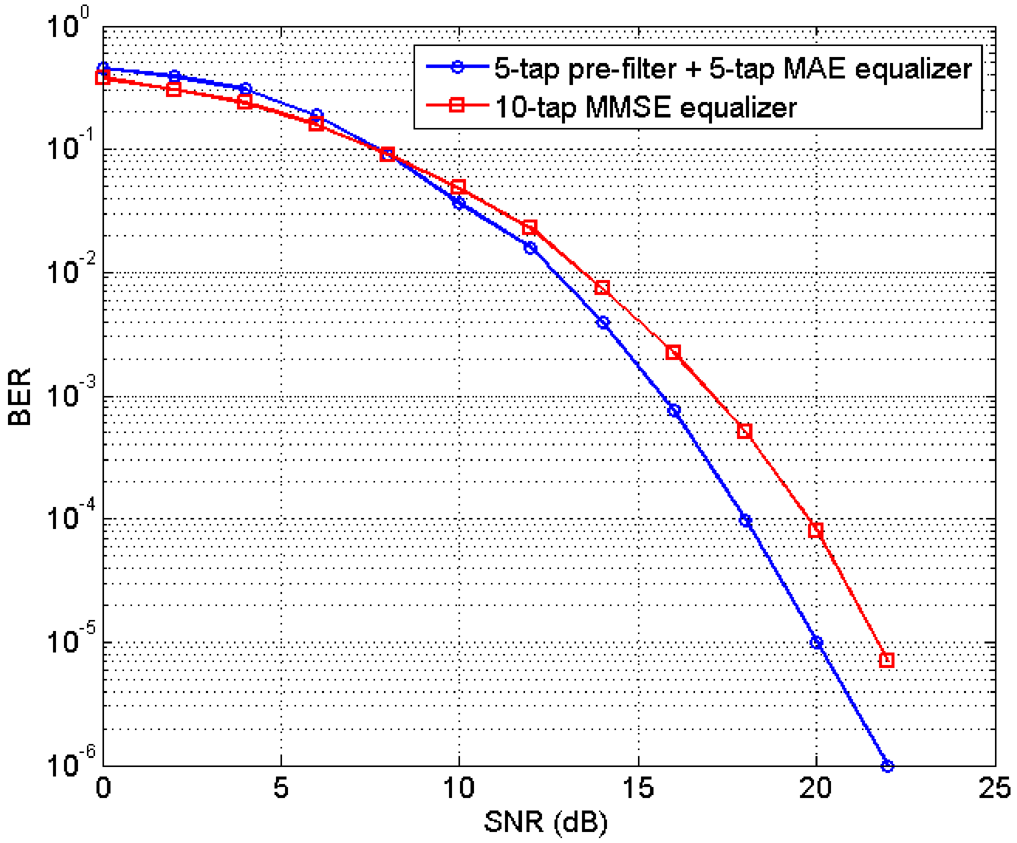


Figure 3.16: Performance comparison of the proposed adaptive MAE equalizer and the MMSE equalizer for $\beta = 0.05$

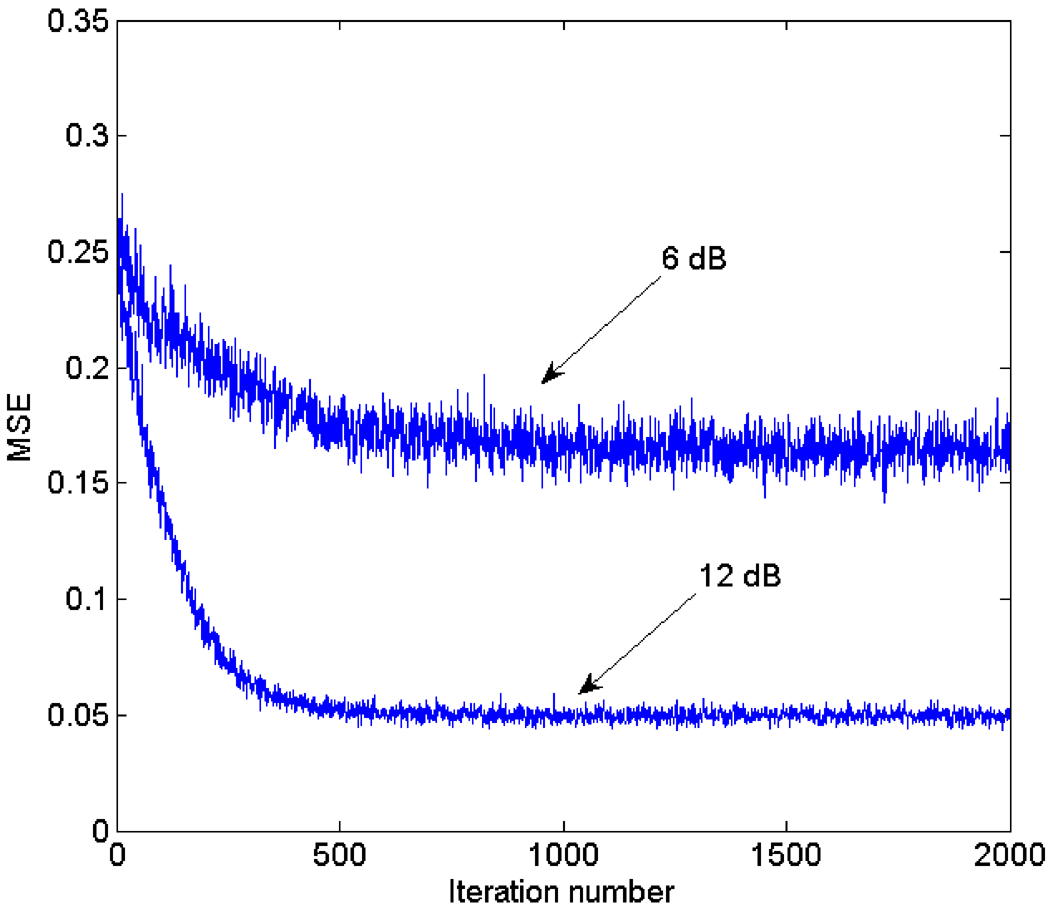


Figure 3.17: Convergence of the constellation points for time-varying channels when SNR is 6 dB and 12 dB.

3.4 Application to Channels with Overlapping Convex Hulls

We have presented a geometrically-inspired design of the MAE equalizer. With the proposed computationally efficient design and the adaptive pre-filter, the MAE equalizer can be used for time-varying channels with relatively large delay spread. However, we have not considered channels for which the convex hulls of subconstellations overlap, a common scenario. When the convex hulls overlap, they can no longer be separated by a hyperplane. In order to address this scenario, we can combine the MAE equalizer with a feedback filter. This combination has the same structure as the DFE, where the feedforward transversal filter is replaced by a linear MAE equalizer, shown in Fig. 2.7.

Let us recall the expression for constellations of the noise-free channel output $\mathbf{r}_k$, where $C_d^{(\pm)} = \{\mathbf{r}_k | x_{k-d} = \pm 1\}$. The number of constellation points in each subconstellation $C_d^{(+)}$ or $C_d^{(-)}$ is $2^{L_h+L_c-2}$ when BPSK is used. We assume that a feedback filter with length L_{fb} is used and a correctly estimated symbol vector $\hat{\mathbf{x}}_{fb}$ is fed back. Thus, a new set of constellations can be constructed as

$$C_{d,fb}^{(\pm)} = \{\mathbf{r}_k | x_{k-d} = \pm 1 \cap \mathbf{x}_{fb} = \hat{\mathbf{x}}_{fb}\}. \tag{3.24}$$

The number of constellation points in each subconstellation $C_{d,fb}^{(+)}$ or $C_{d,fb}^{(-)}$ is now reduced to $2^{L_h+L_c-L_{fb}-2}$. Without feedback, the MAE equalizer needs to find the hyperplane that best separates the two convex hulls with $2^{L_h+L_c-2}$ constellation points. As a result of feedback, the number of constellation points in each subconstellation is reduced by a factor of $2^{L_{fb}}$, shrinking the corresponding convex hulls. The number of feedback taps required to separate the convex hulls is, of course, dependent upon the channel response. In order to visualize the result of the feedback, we take the 4-tap channel $\mathbf{h} = [0.35, 0.25, 0.75, 0.55]$ and a 2-tap MAE equalizer with a feedback filter as an example. Fig. 3.18 shows the constellations and corresponding overlapped convex hulls when no feedback filter is used. Fig. 3.19 shows the constellations and corresponding convex hulls when a length-1 feedback filter is used. We

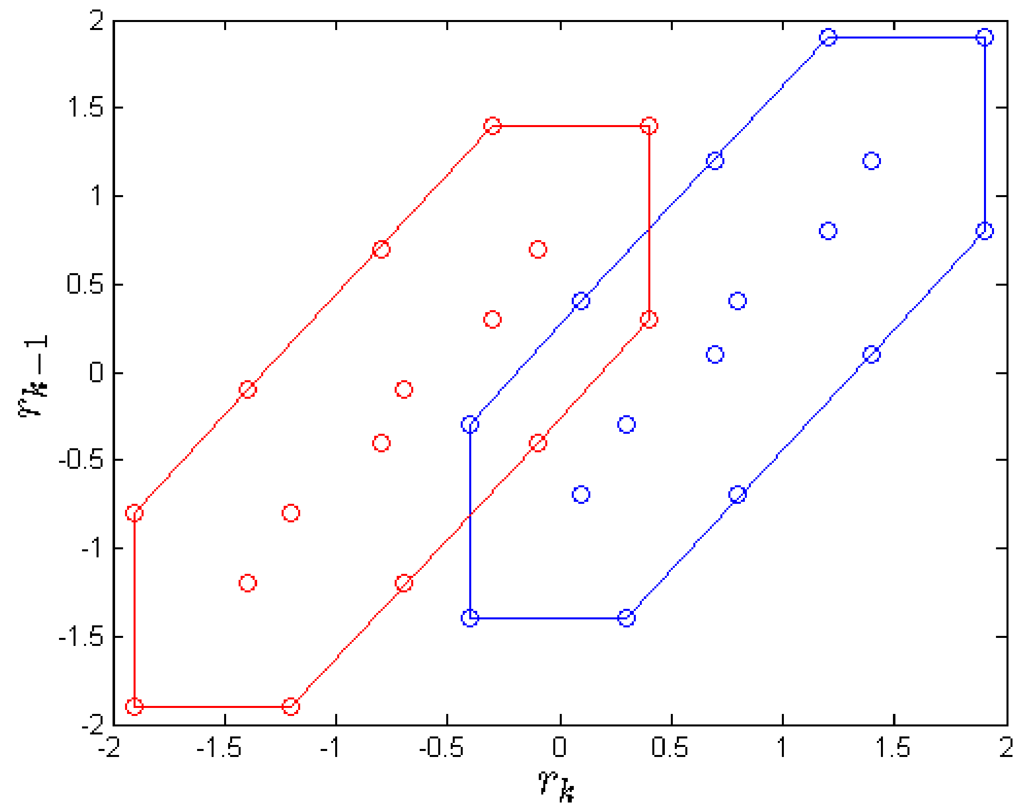


Figure 3.18: The constellations and corresponding convex hulls for the 4-tap channel $\mathbf{h} = [0.35, 0.25, 0.75, 0.55]$ and a 2-tap MAE equalizer without feedback.

can see that the number of constellation points in each subconstellation is reduced from 16 to 8, and the new convex hulls are separated. Thus, the principle of the MAE equalizer which seeks the best separating hyperplane can be applied.

From the design of the MAE equalizer, we also learned that its performance is governed by the minimum distance between the two convex hulls. With appropriate selection of the feedback, the convex hulls can be well separated, increasing minimum distance. As shown in Fig. 3.20 where a length-2 feedback filter is used, the convex hulls are further separated, increasing minimum distance. This is also a geometrical explanation for why the MAE-DFE has better performance than the MAE linear equalizer with the same feedforward filter length. With feedback, the DFE can shrink the convex hulls of the subconstellations

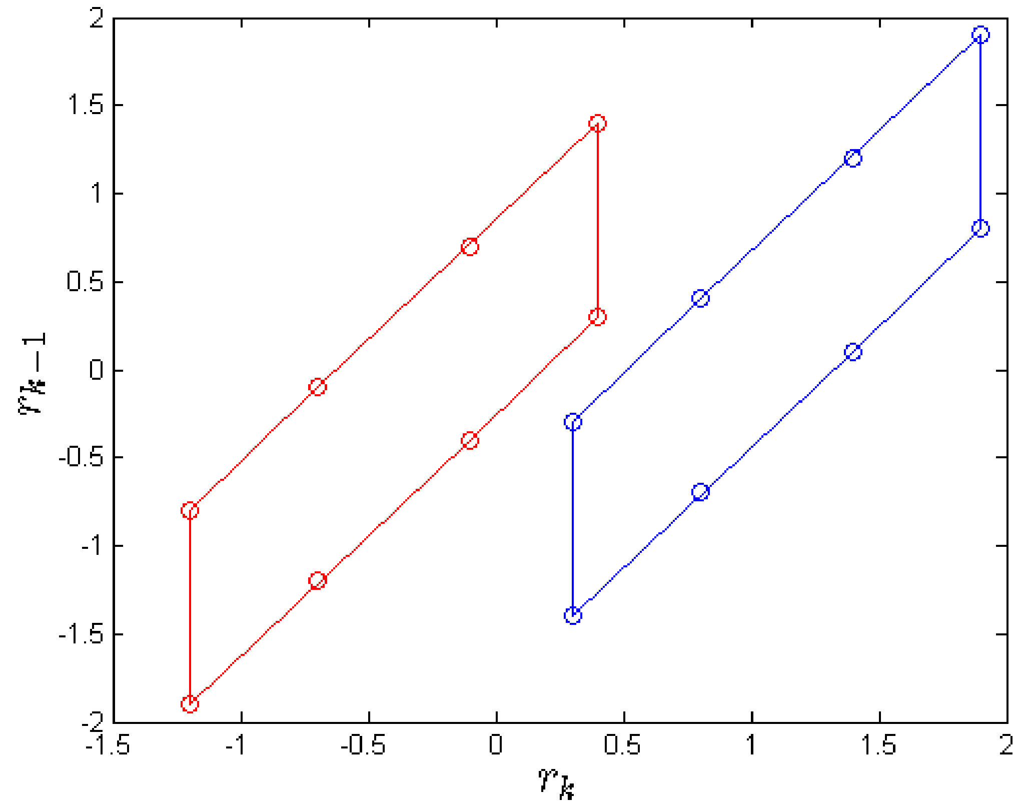


Figure 3.19: The constellations and corresponding convex hulls for the 4-tap channel $\mathbf{h} = [0.35, 0.25, 0.75, 0.55]$ and a 2-tap MAE equalizer with a length-1 feedback filter.

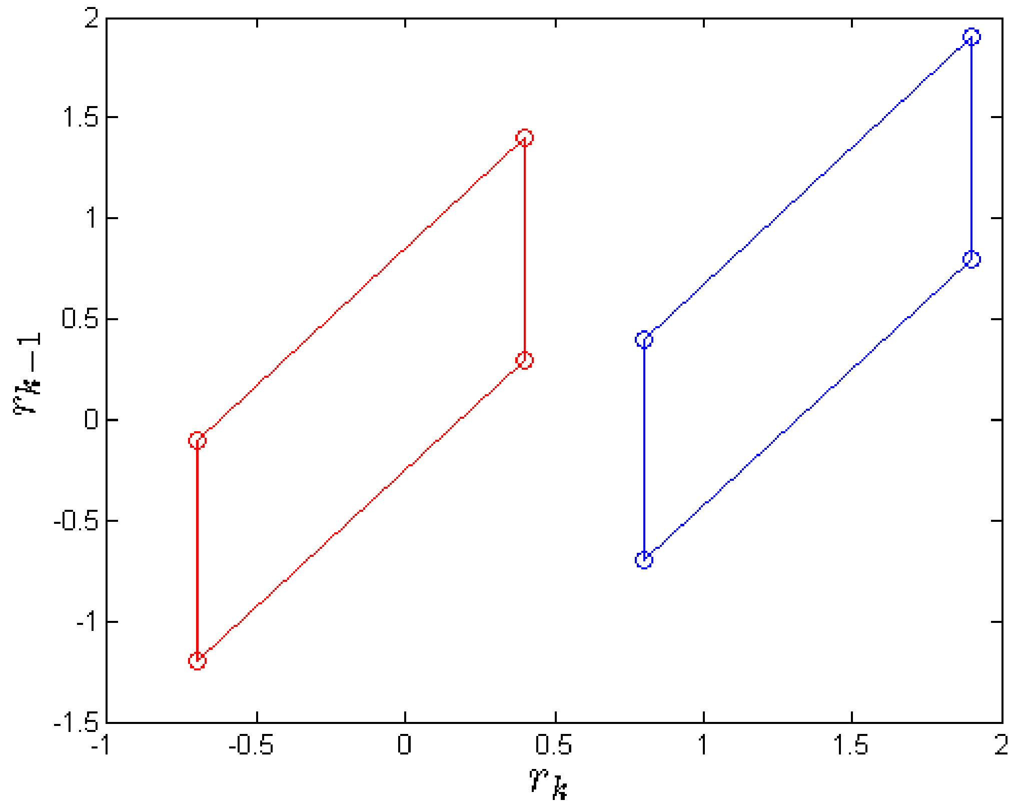


Figure 3.20: The constellations and corresponding convex hulls for the 4-tap channel $\mathbf{h} = [0.35, 0.25, 0.75, 0.55]$ and a 2-tap MAE equalizer with a length-2 feedback filter.

and separate them better than the linear equalizer. However, similar to the regular DFE [44, 45], when the MAE-DFE attains its maximum effective feedback length, the minimum distance between convex hulls is maximized, and no performance improvement is gained with increased feedback length. We show in Appendix B that if the decision delay d is given, $L_c = d + 1$ is the maximum effective feedforward filter length, and $L_b = L_h - 1$ is the maximum effective feedback filter length. These results coincide with those for the DFE designed using the MMSE criterion, as shown in [44].

In addition to improved performance, adding feedback to the MAE equalizer also reduces equalization complexity. In Section 3.2, a geometrically-inspired method is proposed to efficiently design the MAE equalizer. It is shown that the decision hyperplane can be found directly from channel vectors defined by columns of the channel matrix $\mathbf{H}$. With length-L_b feedback, the constellations defined in (3.24) can be written in the form of channel vectors:

$$C_{d,fb}^{(\pm)} : \{\mathbf{r}_k = \sum_{i=0}^{L_h+L_c-L_b-2} \boldsymbol{g}_i x_{k-i} + \mathbf{q} | x_{k-d} = \pm 1\}, \tag{3.25}$$

where $\mathbf{q} = \sum_{L_h+L_c-L_b-1}^{L_h+L_c-2} \boldsymbol{g}_i x_{k-i}$ is a deterministic vector. The decision hyperplane can be determined by the hyperplane constructed by the channel vectors, $\boldsymbol{g}_i, 0 \leq i \leq L_h+L_c-L_b-2$. With feedback $\mathbf{x}_{fb} = \hat{\mathbf{x}}_{fb}$, the corresponding channel vectors, $\boldsymbol{g}_i, L_h + L_c - L_b - 1 \leq i \leq L_h + L_c - 2$, are excluded from consideration. As a result, the computational complexity of designing the MAE equalizer is further reduced.

Chapter 4: Sequential Detection for Sparse Channels via a Multiple Tree Algorithm

We have presented several popular techniques for symbol-by-symbol equalization. The MMSE and MAE equalizers are shown to have strong performance in mitigating ISI. However, they neglect the effect of other symbols when performing estimation. In contrast, sequential detection based equalization employs methods originally proposed for decoding convolutional and tree codes. It exploits the information in the full received sequence and estimates the transmitted sequence as a whole given the observations. In most cases, sequential equalizers performed better than symbol-by-symbol equalizers with respect to minimizing probability of error. The Viterbi algorithm (VA), stack algorithm (SA) and M-algorithm are widely used methods in sequential equalization for general ISI channels. Our work focuses on applying sequential equalization to sparse channels, in which only a few elements of the impulse response have significant weight. When standard sequential equalization algorithms are used on these channels without taking advantage of the sparse channel structure, opportunities for performance improvement are lost. We aim to exploit channel structure to maximize performance of the receiver.

Sparse ISI channels are encountered in a wide range of applications, such as UWA [46] , UWB [47,48] and HDTV systems [49]. The channels in these applications have a very large memory but only a small number of non-zero channel taps. The large memory of these channels often renders conventional maximum likelihood (ML) detection approaches, such as the VA, prohibitively complex. Numerous methods have been proposed to reduce the computational burden of data detection by exploiting channel sparsity. In [50], it was shown that only a small number of filter coefficients in the decision feedback equalizer (DFE) are significant for sparse ISI channels, and a method for predicting the significant taps was proposed. Algorithms to find the locations of such coefficients were well studied in [51, 52]. It was observed in [53] and [54] that when using the VA for a sparse channel, many paths in the trellis have the same probability metric due to the large number of non-active channel taps. A parallel trellis Viterbi algorithm (PTVA) was proposed in [53] to avoid redundant metric computations, but it is applicable only to zero padding channels, i.e., channels that have an equal number of zeros between active (non-zero) channel taps. In [54], a multi-trellis Viterbi algorithm (MVA) was designed for general sparse channels using multiple irregular trellises. Both the PTVA and MVA were proposed to approximate the ML solution and reduce complexity by avoiding redundant path metric computations, but their complexity still grows exponentially with the number of active channel taps.

In this chapter, we propose a computationally efficient tree-search based sequential detection method called the multiple tree algorithm (MTA). The proposed technique makes use of the stack algorithm (SA) for tree search, which was originally developed for decoding convolutional and tree codes [32]. Similar to the PTVA and MVA, the proposed algorithm reduces complexity by eliminating repeated metric computations within the detection process. However, it also takes advantage of the efficiency of the SA at moderate-to-high SNR to further reduce the computational load for long sparse channels with a large number of active taps. In contrast to the MVA, which uses a multi-trellis structure, the proposed method employs multiple trees, each with a reduced number of branches relative to a full search tree and each producing estimates of a subset of the transmitted bits. Each subtree takes a subset of the received sequence as input and obtains soft estimates of the bits that are not available to it from other subtrees. By eliminating paths with the same metric, the MTA significantly improves sequential detection efficiency for sparse ISI channels.

The remainder of the chapter is organized as follows. The system model and background on the stack algorithm are presented in Section 4.1. In Section 4.2, the proposed MTA using hard information (HMTA) is described in detail, and a step-by-step procedure is provided. In Section 4.3, the MTA using soft information (SMTA) is illustrated. The comparison between the HMTA and SMTA is analyzed in Section 4.4. Complexity analysis and simulation results of the proposed method are presented in Section 4.4.2.

4.1 Sequential Detection Using the Stack Algorithm

The conventional stack algorithm designed for general channels is normally prohibitive to use for sparse ISI channels, which gives us motivation to derive reduced-complexity methods to deal with sparse ISI channels. The proposed MTA takes advantage of the channel sparsity and modifies the basic structure of the the conventional SA. In order to better illustrate the proposed MTA, in this section, we will first introduce sequential detection via the SA using a single tree for general ISI channels, and the proposed MTA will be presented in the next section.

4.1.1 System Model

We consider a discrete-time, baseband equivalent communication system, shown in Fig. 4.1. A length-N sequence of information symbols $\mathbf{x}_1^N = \{x_1, \ldots, x_k, \ldots, x_N\}$ is passed over a length-L_h sparse ISI channel $\mathbf{h}$ with additive white Gaussian noise (AWGN) $w_k \sim \mathcal{N}(0, \sigma^2)$. The channel output sequence $\mathbf{y}_1^N = \{y_1, \ldots, y_k, \ldots, y_N\}$ is then processed by a detector to generate symbol estimates $\hat{\mathbf{x}}_1^N = \{\hat{x}_1, \ldots, \hat{x}_k, \ldots, \hat{x}_N\}$. We assume that the receiver knows the sparse channel, $\mathbf{h}$,

$$\mathbf{h} = [h_0, 0, \ldots, 0, h_1, 0, \ldots, 0, h_{L_a-1}]^T. \tag{4.1}$$

The channel has only L_a active (non-zero) taps, $\mathbf{h}_a = [h_0, h_1, \ldots, h_{L_a-1}]^T$, where $L_a \ll L_h$. Let p_i denote the position of the i-th active channel tap. The received signal at time instant k can be expressed as

$$y_k = \sum_{i=0}^{L_a-1} h_i x_{k-p_i} + w_k = \mathbf{h}_a^T \mathbf{x}_{k-p_0}^{k-p_{L_a-1}} + w_k, \tag{4.2}$$

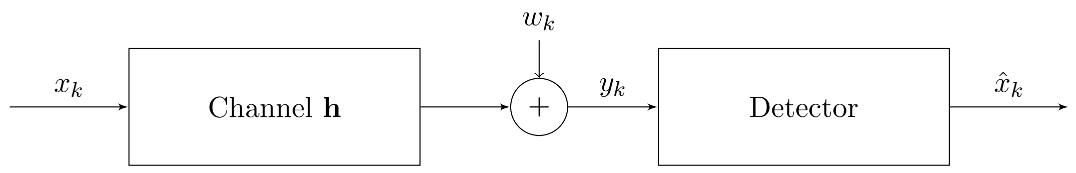


Figure 4.1: Block diagram of the discrete-time equivalent communication system. Information symbols x_k are transmitted over a sparse ISI channel $\mathbf{h}$ with AWGN w_k. The channel output y_k is then processed by a detector to generate symbol estimates $\hat{x}_k$.

where $\mathbf{x}_{k-p_0}^{k-p_{L_a-1}} = \left[x_{k-p_0}\, x_{k-p_1}\, \ldots\, x_{k-p_{L_a-1}}\right]$ are the input bits on which y_k depends. For simplicity, we assume the input sequence $\mathbf{x}_1^N$ is BPSK modulated, i.e., $x_k \in \{+1, -1\}$. It is straightforward to extend the proposed algorithm to M-ary modulation schemes.

4.1.2 Stack Algorithm

The SA is generally described in Chapter 2. In this section, we will illustrate the SA in detail, and show how it is applied in sequential equalization. The SA is a best-first tree-search technique that extends the single most likely path at each iteration [32]. When the tree represents the space spanned by a sequence of transmitted bits, the SA can be used to approximate the ML solution to sequence detection. In such applications, each path in the tree represents a possible realization of the transmitted sequence. Each path has an associated metric, which expresses the likelihood that the corresponding bit sequence was transmitted, conditioned on the observations. A set of possible paths and their metrics are stored in a stack (or list) in order of decreasing metric value. At each time step, the SA extends the path with the highest likelihood, e.g. the top path in the stack. The algorithm terminates when the top path in the stack reaches a leaf of the tree, or equivalently when the top path represents a full block of transmitted bits.

To navigate a tree using the SA, paths are extended from depth 1 to N progressively. Hence, the path metric, which is a measure of the likelihood of a path, must be determined for a partial-length sequence ($n < N$) given the full received sequence to decide which path in the stack should be extended. The probability that a particular bit sequence was transmitted conditioned on the observed sequence can be written as

$$P(\mathbf{x}_1^n|\mathbf{y}_1^N) = \frac{P(\mathbf{y}_1^N|\mathbf{x}_1^n)P(\mathbf{x}_1^n)}{P(\mathbf{y}_1^N)}. \tag{4.3}$$

Eliminating $P(\mathbf{y}_1^N)$ since it is equal for all paths, we can simplify the above expression and write the path metric for the length-n input sequence $\mathbf{x}_1^n$ as

$$m(\mathbf{x}_1^n) = P(\mathbf{y}_1^N|\mathbf{x}_1^n)P(\mathbf{x}_1^n). \tag{4.4}$$

Because of the memory of the channel, the received sequence beyond the path of interest, $\mathbf{y}_{n+1}^N$, is dependent on some elements of $\mathbf{y}_1^n$. However, in order to develop a closed-form expression for the path metric, we make an approximation and assume that $\mathbf{y}_1^n$ and $\mathbf{y}_{n+1}^N$ are independent. With this assumption, we have the following expression for the metric of a length-n input sequence:

$$m(\mathbf{x}_1^n) = P(\mathbf{x}_1^n)P(\mathbf{y}_1^n|\mathbf{x}_1^n)P(\mathbf{y}_{n+1}^N|\mathbf{x}_1^n). \tag{4.5}$$

The input bits of the sequence $\mathbf{x}_1^n$ are assumed to be independent and equally likely to be ± 1. Hence, we can easily get

$$P(\mathbf{x}_1^n) = (\frac{1}{2})^n. \tag{4.6}$$

$P(\mathbf{y}_1^n|\mathbf{x}_1^n)$ is the path metric used for the VA and is given by the cascade of the conditional probability of each $y_k \in \mathbf{y}_1^n$:

$$\begin{aligned} P(\mathbf{y}_1^n|\mathbf{x}_1^n) &= \prod_{k=1}^{n} P(y_k|\mathbf{x}_1^n) = \prod_{k=1}^{n} P(y_k|\mathbf{x}_{k-L_h+1}^k) \qquad (4.7) \\ &= \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_k - \mathbf{h}^T\mathbf{x}_{k-L_h+1}^k)^2\right). \end{aligned}$$

Equation (4.7) incorporates the property that the received symbol y_i is only dependent on the input symbols $\mathbf{x}_{k-L_h+1}^{k}$ due to the length of the channel L_h, and thus other symbols in the transmitted sequence are ignored. Furthermore, we assume that the received sequence $\mathbf{y}_{n+1}^{N}$ is independent of the first n transmitted symbols, $\mathbf{x}_1^n$. The bias term $P(\mathbf{y}_{n+1}^{N}|\mathbf{x}_1^n)$ that compensates the variant length of paths in the stack is then expressed as

$$\begin{aligned} P(\mathbf{y}_{n+1}^{N}|\mathbf{x}_1^n) &\approx P(\mathbf{y}_{n+1}^{N}) \\ &= \sum_{j=1}^{j=2^{N-n}} P(\mathbf{y}_{n+1}^{N}|\{\mathbf{x}_{n+1}^{N}\}_j)P(\{\mathbf{x}_{n+1}^{N}\}_j), \end{aligned} \tag{4.8}$$

where $\{\mathbf{x}_{n+1}^{N}\}_j$ is one of 2^{N-n} possible realizations of sequence $\mathbf{x}_{n+1}^{N}$. Such a summation over 2^{N-n} is too complex to be computed for the path metric. Thus, to make it simple, we make an assumption of independence between the future received symbols. Hence, we have

$$P(\mathbf{y}_{n+1}^{N}) \approx \prod_{k=n+1}^{N} P(y_k). \tag{4.9}$$

Given knowledge of $\mathbf{h}$ and $\mathbf{x}_{k-L_h+1}^{k}$, the received signal y_k is only dependent on the L_h input bits $\mathbf{x}_{k-L_h+1}^{k}$ and $y_k \sim \mathcal{N}(\mathbf{h}^T\mathbf{x}_{k-L_h+1}^{k}, \sigma^2)$. Thus, the probability $P(y_k)$ can be obtained by averaging over all possible length-L_h binary sequences, which is a summation of 2^{L_h} Gaussian terms. To find an expression of reasonable complexity for practical implementation, we apply the approach used in [55] and approximate the Gaussian mixture by a single Gaussian term. Then, the likelihood $P(\mathbf{y}_{n+1}^{N})$ is approximated as

$$P(\mathbf{y}_{n+1}^{N}) \approx \prod_{k=n+1}^{N} \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{y_k^2}{2\sigma_b^2}\right), \tag{4.10}$$

where $\sigma_b^2 = \sigma^2 + \mathbf{h}^T\mathbf{h}$. Substituting equations (4.7) and (4.10) into equation (4.5) and discarding the term 2^{L_h} which is constant for all paths, we obtain the following expression for the path metric:

$$\begin{aligned} m(\mathbf{x}_1^n) &= \left(\frac{1}{2\sqrt{2\pi\sigma^2}}\right)^n \left(\frac{1}{\sqrt{2\pi\sigma_b^2}}\right)^{N-n} \prod_{k=n+1}^{N} \exp\left(-\frac{y_k^2}{2\sigma_b^2}\right) \\ &\times \prod_{k=1}^{n} \exp\left(-\frac{1}{2\sigma^2}(y_k - \mathbf{h}^T\mathbf{x}_{k-L+1}^k)^2\right). \end{aligned} \tag{4.11}$$

When the SA is used to detect data transmitted over an ISI channel, the metric for a given path is governed by the symbol sequence associated with that path $\mathbf{x}_1^n$, the channel impulse response $\mathbf{h}$, and the output observations $\mathbf{y}_1^N$. The SA provides an efficient way to navigate the tree by extending only the path with largest metric. However, when we apply the SA for sparse ISI channels, due to the zero taps of the channel, multiple paths in the tree will have the same products $\mathbf{h}^T\mathbf{x}_{k-L+1}^k$ and hence the same path metric. In Section 4.2, we propose a novel sequential detection method wherein we replace the single search tree with multiple parallel trees to avoid repeated metric computations.

4.2 Multiple Tree Algorithm Using Hard Information

Rather than searching a single tree, the MTA uses multiple parallel subtrees designed to avoid redundant metric computations. For example, given a simple channel $\mathbf{h} = [h_0, 0, h_1]$, two subtrees can be constructed. One estimates the transmitted sequence $\{x_1, x_3, \ldots\}$, and the other estimates $\{x_2, x_4, \ldots\}$. The construction of the subtrees is determined by the positions of the active channel taps and is designed such that no two paths have the same metric. Moreover, the shorter depth of each subtree reduces the number of nodes to be visited.

Let J denote the number of parallel subtrees. The channel output sequence $\mathbf{y}_1^N$ is divided into J subsets $\{\mathbf{y}_j^{N_j}\}$, $j = 1, \ldots, J$, where $\mathbf{y}_j^{N_j} = \{y_j, y_{j+J}, \ldots, y_{N-J+j}\}$ is the input to the j-th subtree. $\mathbf{x}_j^{N_j} = \{x_j, x_{j+J}, \ldots, x_{N-J+j}\}$ is the subset of the transmitted sequence estimated in the j-th subtree and is referred to as the state of subtree j. When the MTA is used, the channel output y_k may be fed to a subtree whose state does not include all the symbols x_{k-p_i}, $i = 0, \ldots, L_a - 1$, on which y_k is dependent. We obtain estimates of such symbols from the subtrees in which they are contained and incorporate them into the path metric as needed. Those estimates could be hard estimates or soft estimates. The hard estimate of x_{k-p_m} is either $+1$ or -1, while the soft estimate provides the probability that x_{k-p_m} is equal to each possible value. We will explore the difference between exchanging these two estimates among subtrees in Section 4.4. In this section, we will only consider MTA using hard estimates (HMTA). The path metric for the HMTA is derived in Section 4.2.1, and the selection of the number of subtrees J is described in Section 4.2.2. A step-by-step description of the HMTA is given in Section 4.2.3.

4.2.1 Derivation of the Path Metric

The SA governs the order in which the paths of each subtree are explored. The sequence represented by each path is a non-contiguous subset of the transmitted sequence. As a result, the conventional sequential detection path metric must be modified to incorporate hard estimates of the symbols not contained in each subtree state. Let $\mathbf{x}_j^{n_j}$ denote the first $1+(n_j-j)/J$ symbols estimated by the j-th subtree. In subtree j, the path metric expresses the likelihood that the information sequence $\mathbf{x}_j^{n_j}$ is transmitted given knowledge of the full received sequence $\mathbf{y}_j^{N_j}$,

$$P(\mathbf{x}_j^{n_j}|\mathbf{y}_j^{N_j}) = \frac{P(\mathbf{y}_j^{N_j}|\mathbf{x}_j^{n_j})P(\mathbf{x}_j^{n_j})}{P(\mathbf{y}_j^{N_j})}. \tag{4.12}$$

In order to develop a closed-form expression for the path metric that can be computed with moderate complexity, we make two assumptions similar to the SA:

1) The received sequence beyond the path of interest, $\mathbf{y}_{n_j+J}^{N_j}$, when conditioned on $\mathbf{x}_j^{n_j}$, is independent of the previous received sequence $\mathbf{y}_j^{n_j}$.

2) The sequence $\mathbf{y}_{n_j+J}^{N_j}$ is independent of the first $1 + (n_j - j)/J$ symbols $\mathbf{x}_j^{n_j}$ that subtree j estimates.

Eliminating $P(\mathbf{y}_j^{N_j})$ since it is equal for all paths, we can simplify the expression in (3) and write the path metric for the input sequence $\mathbf{x}_j^{n_j}$ as

$$\begin{aligned} m(\mathbf{x}_j^{n_j}) &= P(\mathbf{y}_j^{N_j}|\mathbf{x}_j^{n_j})P(\mathbf{x}_j^{n_j}) \\ &\approx P(\mathbf{x}_j^{n_j})P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})P(\mathbf{y}_{n_j+J}^{N_j}|\mathbf{x}_j^{n_j}) \\ &\approx P(\mathbf{x}_j^{n_j})P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})P(\mathbf{y}_{n_j+J}^{N_j}), \end{aligned} \tag{4.13}$$

where the second and third lines in (4.13) incorporate assumptions 1) and 2), respectively. Assuming the information symbols are independent and equally likely to be ± 1, the prior probability of the transmitted symbols is given by

$$P(\mathbf{x}_j^{n_j}) = (\frac{1}{2})^{1+(n_j-j)/J}. \tag{4.14}$$

Since the SA extends only the path with the largest metric at each stage, the paths stored in the stack will generally be of varying lengths. $P(\mathbf{y}_{n_j+J}^{N_j})$, which serves as a bias term that compensates for varying path length. Instead of averaging over all possible sequences $\mathbf{x}_{n_j+J}^{N_j}$, we assume the future received symbols y_k, $k = n_j + J, \ldots, N_j$, are independent of each other, and approximate the bias term by a product of Gaussian terms which is similar to that of the SA,

$$P(\mathbf{y}_{n_j+J}^{N_j}) \approx \prod_{k=n_j+J}^{N_j} P(y_k) \tag{4.15}$$

$$\approx \prod_{k=n_j+J}^{N_j} \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{y_k^2}{2\sigma_b^2}\right),$$

where $\sigma_b^2 = \sigma^2 + \mathbf{h}^T\mathbf{h} = \sigma^2 + \mathbf{h}_a^T\mathbf{h}_a$.

The likelihood term $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$ is given by the cascade of the conditional probabilities of each $y_k \in Y_j^{n_j}$,

$$P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j}) = \prod_{k=j:J:n_j} P(y_k|\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}). \tag{4.16}$$

As mentioned, the channel output y_k may be fed to a subtree whose state does not include all the bits on which y_k is dependent. Let us denote by $\mathbf{x}_{(a)}^j = \{x_{k-p_i} \in \mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}$ and $\mathbf{x}_{(u)}^j = \{x_{k-p_i} \notin \mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}$, $i \in \{0, \ldots, L_a - 1\}$, the F bits available and the D bits unavailable, respectively, in the j-th subtree state. The conditional likelihood of y_k can be written as

$$P(y_k|\mathbf{x}_j^{n_j}) = P\left(y_k|\mathbf{x}_{(a)}^j\right) \tag{4.17}$$

$$= P\left(y_k|\mathbf{x}_{(a)}^j, \mathbf{x}_{(u)}^j = \mathbf{q}\right),$$

where the vector $\mathbf{q} = \{\mathbf{q}(d)\}$, $1 \leq d \leq D$, denotes a length-D BPSK sequence. Given the sequence $\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}$, the received element y_k is Gaussian, $y_k \sim \mathcal{N}(\mathbf{h}_a^T \mathbf{x}_{k-p_0}^{k-p_{L_a-1}}, \sigma^2)$. Therefore,

$$P\left(y_k | \mathbf{x}_{(a)}^j, \mathbf{x}_{(u)}^j = \mathbf{q}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T \{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}_{(u)}^j = \mathbf{q}})^2}{2\sigma^2}\right), \tag{4.18}$$

where $\{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}_{(u)}^j = \mathbf{q}}$ denotes the sequence $\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}$ with $\mathbf{x}_{(u)}^j = \mathbf{q}$.

$\mathbf{x}_{(u)}^j(d) = \mathbf{q}(d)$ is the hard information that must be obtained from another subtree. We determine the hard estimate based on which bit value has larger probability. Suppose, for example, that symbol $\mathbf{x}_{(u)}^j(d)$ is available in subtree t, $t \neq j$. If all paths in subtree t are explored to depth N_t, there will be $2^{(N_t - t + J)/J}$ paths, each of which corresponds to a realization of sequence $\mathbf{x}_t^{N_t}$. Therefore, the probability of $\mathbf{x}_{(u)}^j(d) = \mathbf{q}(d)$ can be exactly computed based on the likelihoods of the paths which have $\mathbf{x}_{(u)}^j(d) = \mathbf{q}(d)$, , mathematically expressed as

$$P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}(d)) = \sum_{\mathbf{x}_t^{N_t}:\, \mathbf{x}_j^{(u)}(d) = \mathbf{q}(d)} P(\mathbf{x}_t^{N_t} | \mathbf{y}_t^{N_t}). \tag{4.19}$$

When the SA is used, however, not all paths in a subtree are explored. Thus, $P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}(d))$ can be obtained only approximately based on the paths present in the stack for subtree t,

$$P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}(d)) \approx \sum_{\mathbf{x}_t^{n_t} \in S_t:\, \mathbf{x}_{(u)}^j(d) = \mathbf{q}(d)} P(\mathbf{x}_t^{n_t} | \mathbf{y}_t^{N_t}), \tag{4.20}$$

where S_t denotes the stack for subtree t.

Since $\mathbf{q}(d)$ only takes value of $+1$ and -1 by using BPSK, the hard estimate of $\mathbf{x}_{(u)}^j(d)$

from t-th subtree is made by

$$\hat{\mathbf{x}}^j_{(u)}(d) = \underset{\mathbf{x}^j_{(u)}(d)\in\{1,-1\}}{\text{argmax}} \left(P(\mathbf{x}^j_{(u)}(d)=+1),\ P(\mathbf{x}^j_{(u)}(d)=-1)\right). \tag{4.21}$$

The observation likelihood term, $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$, can be obtained by substituting (4.18), which incorporates (4.20) and (4.21), into (4.16).

$$\begin{aligned} P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j}) &= \prod_{k=j:J:n_j} P(y_k|\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}) \quad (4.22)\\ &= \prod_{k=j:J:n_j} P\left(y_k|\mathbf{x}^j_{(a)}, \mathbf{x}^j_{(u)}=\mathbf{q}\right)\\ &= \prod_{k=j:J:n_j} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T\{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}^j_{(u)}=\hat{\mathbf{x}}^j_{(u)}})^2}{2\sigma^2}\right). \end{aligned}$$

Substituting (4.16) and (4.22) into (4.13) produces a closed-form expression for the path metric that is used in each subtree of the HMTA,

$$\begin{aligned} m(\mathbf{x}_j^{n_j}) &\approx P(\mathbf{x}_j^{n_j})P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})P(\mathbf{y}_{n_j+J}^{N_j}) \quad (4.23)\\ &\approx (\frac{1}{2})^{1+(n_j-j)/J} \prod_{k=n_j+J}^{N_j} \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{y_k^2}{2\sigma_b^2}\right)\\ &\quad \times \prod_{k=j:J:n_j} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T\{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}^j_{(u)}=\hat{\mathbf{x}}^j_{(u)}})^2}{2\sigma^2}\right). \end{aligned}$$

4.2.2 Determination of the number of subtrees

In order to determine the number of subtrees J, we use the criterion of minimizing the number of soft estimates exchanged among subtrees. Recall the expression of the sparse channel output given in (4.2). The first $p_1 - p_0$ channel outputs y_k, $k = 1, \dots, p_1 - p_0$, are dependent only on single symbols x_k, $k = 1, \dots, p_1 - p_0$, respectively. Outputs y_k, $k = p_1 - p_0 + 1, \dots, N$, are dependent on multiple input symbols. In order to minimize the number of soft estimates exchanged, we need to maximize the number of symbols x_{k-p_i}, $i = 1, \dots, L_a - 1$, available in subtree j. Due to the varying number of zeros between active channel taps, we can only guarantee that two symbols, x_k and x_{k-p_i}, are in subtree j. The first $p_1 - p_0$ symbols can be arranged to form the first bit for each subtree state, and x_k, $p_1 - p_0 < k \leq p3$, can then be estimated without obtaining information from other subtrees; this is accomplished by setting $J = p_1 - p_0$. To estimate x_k, $k > p_3$, subtree j must obtain hard estimates from other trees.

Let us consider a 6-tap sparse channel with 3 active taps as an example, $\mathbf{h} = [h_0, 0, 0, h_1, 0, h_2]$. The received signal at instant index k, y_k, can be expressed as,

$$y_k = h_0 x_k + h_1 x_{k-3} + h_2 x_{k-5} + w_k. \tag{4.24}$$

The number of subtrees J can be determined to be $p_1 - p_0 = 3$. The subtrees are constructed as shown in Fig. 4.2. Considering each state of the subtrees, the bits x_k, $k = 1, \dots, 3$, can be estimated using only y_k, $k = 1, \dots, 3$. x_4 is estimated using $\hat{x}_1$ from subtree 1 and $\mathbf{y}_1^4$. x_5 is estimated using $\hat{x}_2$ from subtree 2 and $\mathbf{y}_2^5$. The information exchange is started from x_6 ($p_3 + 1 = 6$). x_k, $k = 6, \dots, N$, is estimated using $\hat{x}_{k-3}$ obtained in its own subtree and $\hat{x}_{k-5}$ from other subtrees. Only one estimate, $\hat{x}_{k-5}$, is exchanged among subtrees. By setting $J = 3$, x_k, $k = 1, \dots, 5$, can be estimated without obtaining information from other subtrees.

If we set the number of subtrees J to a value other than 3, for example $J = 4$, the subtrees can be constructed as in Fig 4.3. The information exchange is started from x_4. Furthermore, in order to estimate x_k, $k = 6, \dots, N$, two hard estimates need to be obtained from other subtrees. Similar results can be shown for other values $J \neq 3$. Therefore, the minimization of the information exchange is achieved when the number of subtrees is

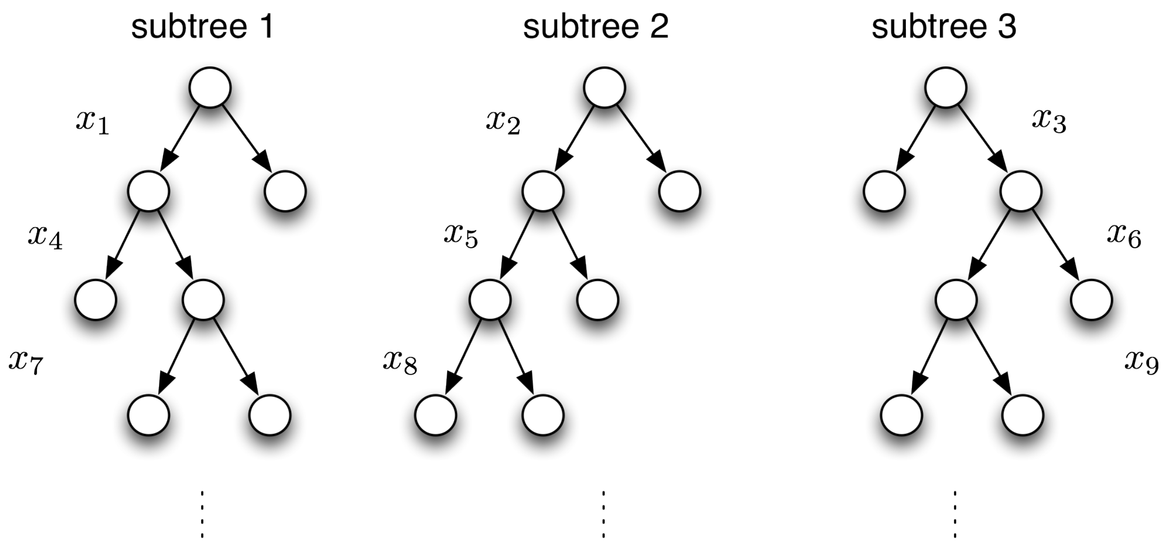


Figure 4.2: The construction of the subtrees for channel $\mathbf{h} = [h_0, 0, 0, h_1, 0, h_2]$ and $J = 3$.

$J = p_1 - p_0$.

4.2.3 Summary of the MTA Using Hard Information

To better illustrate the HMTA algorithm and the passing of estimates among subtrees, consider a 6-tap sparse channel with 3 active taps as an example, $\mathbf{h} = [h_0, 0, 0, h_1, 0, h_2]$. The channel output at time k is

$$y_k = h_0 x_k + h_1 x_{k-3} + h_2 x_{k-5} + w_k. \tag{4.25}$$

Since there are two zero taps between the first two active taps, we use $J = 3$ subtrees, and the output sequence is divided into 3 subsets, $\mathbf{y}_1^{N-2} = \{y_{3l+1}\}$, $\mathbf{y}_2^{N-1} = \{y_{3l+2}\}$ and $\mathbf{y}_3^N = \{y_{3l+3}\}$, $l = 0, 1, \ldots, (N-3)/3$, shown in Fig. 4.4. The output y_k is dependent on the inputs x_k, x_{k-3} and x_{k-5}. Subtree 1 takes $\mathbf{y}_1^{N-2}$ as its input and incorporates the hard estimates of x_{3l-4}, $l = 3, \ldots, (N-3)/3$, from subtree 2 into the path metrics. Subtree 2 takes $\mathbf{y}_1^{N-1}$ as its input and obtains the hard estimates of x_{3l-3}, $l = 3, \ldots, (N-3)/3$, from subtree 3. Subtree 3 takes $\mathbf{y}_1^N$ as its input and obtains the hard estimates of x_{3l-2},

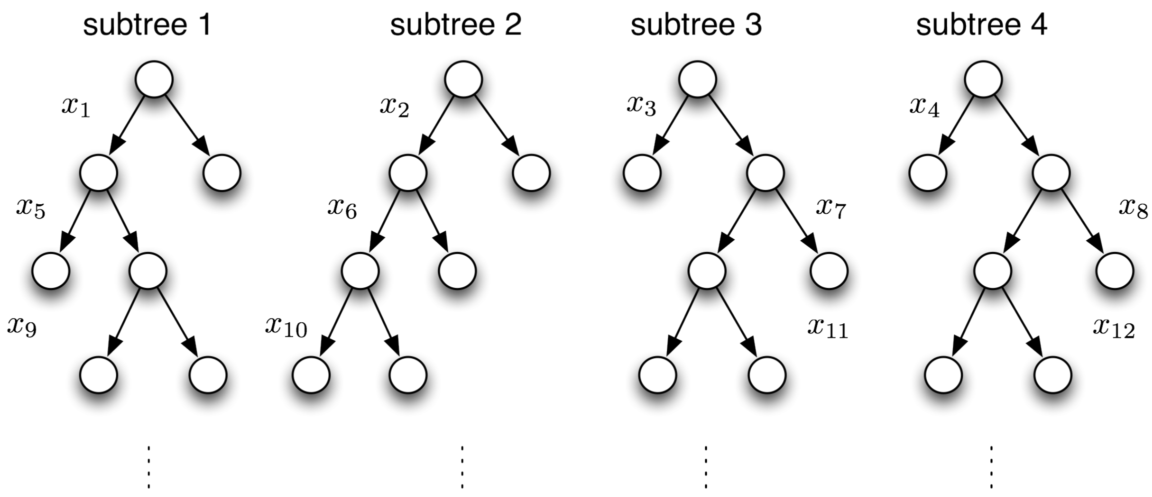


Figure 4.3: The construction of the subtrees for channel $\mathbf{h} = [h_0, 0, 0, h_1, 0, h_2]$ and $J = 4$.

$l = 3, \ldots, (N-3)/3$, from subtree 1.

The HMTA procedure is summarized below. Without loss of generality, we assume the sparse channel has $L_a = 3$ active taps. A flowchart representation of the HMTA is shown in Fig.4.5.

$\boxed{1}$ Initialization: Construct $J = p_1 - p_0$ parallel subtree modules, each of which represents the search space spanned by transmitted symbols $\mathbf{x}_j^{N_j} = \{x_{Jl+j}\}$, where $j = 1, \ldots, J$ and $l = 0, 1, \ldots, (N/J - 1)$. Divide the received signal into J subsets, $\mathbf{x}_j^{N_j} = \{y_{Jl+j}\}$. Insert a root node with metric 0 into a stack for each subtree. Set $j = 1$ and $l = 1$.

$\boxed{2}$ Extend the path at the top of S_j, the stack for subtree j, to all of its children and compute the new path metrics. The hard estimates generated by subtree $(J(l-1) + j - (p_2 - p_1)) \oplus J$ are taken as input, where $\oplus$ denotes the modulo operation. For a branch at depth l, the metric is computed using the estimate $\hat{x}_{J(l-1)+j-(p_2-p_1)}$ from subtree $(J(l-1)+j-(p_2-p_1)) \oplus J$ if $Jl+j \geq p_2$. Otherwise, no hard estimate needs to be obtained.

$\boxed{3}$ Store all explored paths and their metrics in S_j in order of decreasing metric value.

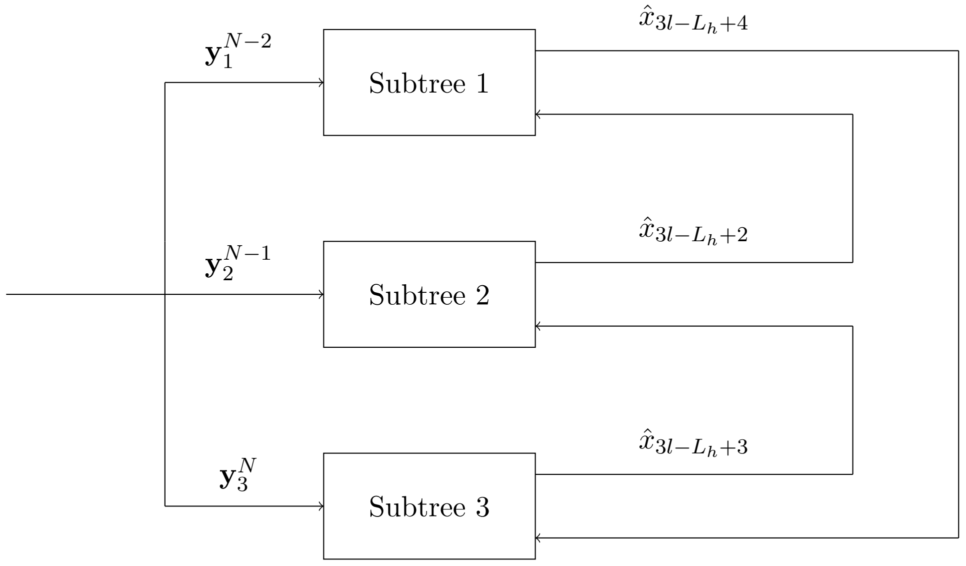


Figure 4.4: Block diagram of an equalizer using the MTA with 3 subtrees for channel $\mathbf{h} = [h_0, 0, 0, h_1, 0, h_2]$. $\hat{x}_{3l-L_h+a}$, $a = 2, 3, 4$, is the hard decision of corresponding symbol exchanged between subtrees.

$\boxed{\mathbf{4}}$ If the depth of the top path in S_j is l, go to $\boxed{\mathbf{5}}$. Otherwise, return to $\boxed{\mathbf{2}}$.

$\boxed{\mathbf{5}}$ Within subtree j, generate hard estimate $\hat{x}_{J(l-1)+j}$ using (4.21); pass the hard estimate to subtree $(J(l-1) + j - (p_2 - p_1) - 1) \oplus J$ if $Jl + j \geq p_2$.

$\boxed{\mathbf{6}}$ If $j < J$, set $j = j + 1$ and return to $\boxed{\mathbf{2}}$. Otherwise, go to $\boxed{\mathbf{7}}$.

$\boxed{\mathbf{7}}$ If the top paths of all J stacks have reached a leaf of the corresponding subtree, i.e., $l = (N - J)/J$, stop and output the sequences from all J subtrees in order. Otherwise, set $l = l + 1$ and $j = 1$, and return to $\boxed{\mathbf{2}}$.

4.2.4 Simulation Results for the HMTA

To illustrate the computational advantage of the proposed HMTA algorithm, we compare it to the conventional VA, SA and MVA for a sparse channel with length L_h and L_a active channel taps. We take the number of computations of a path metric as the complexity measure. For the conventional VA, for each time instant, there will be M^{L_h} evaluations of

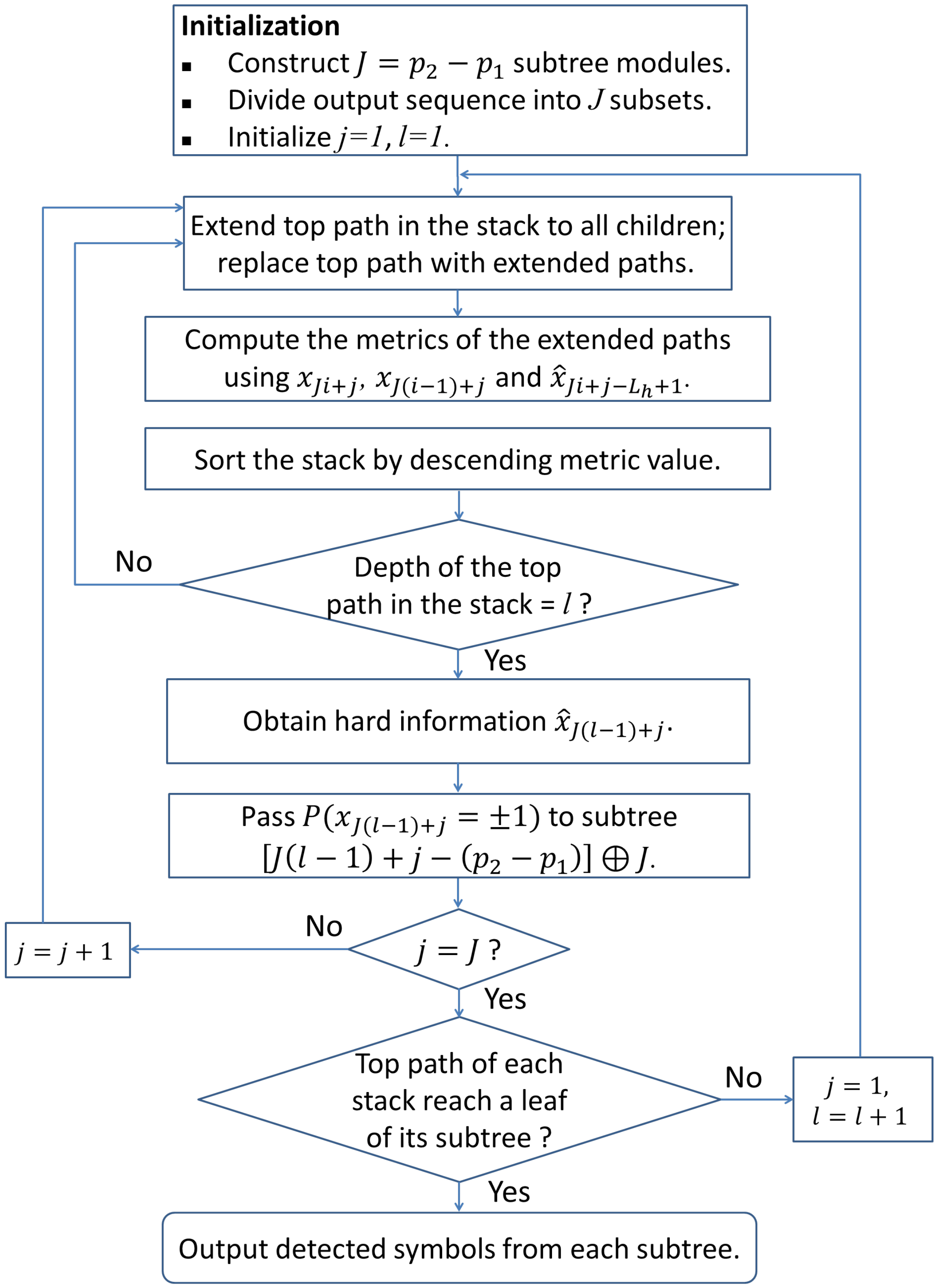

Initialization
Construct $J = p_2 - p_1$ subtree modules.
Divide output sequence into J subsets.
Initialize j=1, l=1.
Extend top path in the stack to all children; replace top path with extended paths.
Compute the metrics of the extended paths using x_{Ji+j}, $x_{J(i-1)+j}$ and $\hat{x}_{Ji+j-L_h+1}$.
Sort the stack by descending metric value.
Depth of the top path in the stack = l ?
No
Yes
Obtain hard information $\hat{x}_{J(l-1)+j}$.
Pass $P(x_{J(l-1)+j} = \pm 1)$ to subtree $[J(l-1)+j-(p_2-p_1)] \oplus J$.
$j = J$?
No
$j = j + 1$
Yes
Top path of each stack reach a leaf of its subtree ?
No
$j = 1$, $l = l + 1$
Yes
Output detected symbols from each subtree.

Figure 4.5: Flowchart representation of the MTA method using hard information

a path metric. For an input sequence with length N, the computational complexity of the conventional VA is NM^{L_h}. The HMVA divides the whole trellis into multiple sub-trellises, each of which has a reduced number of states. The typical number of sub-trellises is $3L_h$ [54]. To estimate each bit x_k, the number of metric evaluations is M^{L_a}. The computational complexity of the MVA is NM^{L_a} for a length-N input sequence. The number of path extensions for the SA varies with the channel and with the realizations of the data sequence and channel noise [32]. Since the complexity of the SA is a random variable, we evaluate the complexity of the proposed HMTA algorithm empirically. Note that, when the HMTA is performed, erasures may occur when a leaf of the tree is not reached within the maximum number of branch explorations, which leads to a loss of performance for the detector [56].

Simulations have been conducted for a length $L_h = 6$ sparse channel with $L_a = 3$ active taps. 1000 blocks of 1200 BPSK symbols are passed over the sparse channel. For each data block, the active channel taps $\mathbf{h}_a = [h_0, h_1, h_2]$ are drawn from a Gaussian distribution with mean 0 and variance 1, and the channel energy is normalized to 1. The stack size of the SA is set to 10^5 to make erasures rare. The number of computations performed per transmitted block as a function of SNR is shown in Table 4.1 for the four detection methods considered. As SNR increases, the complexity of the VA and MVA remain constant, while the complexity of the HMTA decreases dramatically, as would be expected of a stack-based algorithm. For SNR from 5 dB to 7 dB, the HMTA achieves significant computational complexity reduction compared to the other three methods. For SNR of 9 dB or greater, the SA and HMTA are likely to follow a single path and explore very little of the tree, causing their similar complexity. We have conducted simulations for various L_h and L_a. The results show that the advantage of the HMTA on computational complexity becomes more obvious as L_h and L_a increase.

A performance comparison of the four methods is shown in Fig. 4.6. The HMTA suffers some performance loss compared to the other three methods, but when SNR is over 9 dB, all four methods have similar performance. In order to understand the source of the

Table 4.1: Computations performed for the VA, MVA, SA and HMTA using a length-1200 sequence for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps.

Number of Computations				
	VA	**MVA**	**SA**	**HMTA**
4dB	76800	9600	154518	122047
5dB	76800	9600	14429	8539
6dB	76800	9600	7551	5809
7dB	76800	9600	5059	4205
8dB	76800	9600	3286	2524
9dB	76800	9600	2726	2441
10dB	76800	9600	2512	2407
11dB	76800	9600	2408	2402

performance loss suffered by HMTA, we have conducted the same simulations for a zero-padding channel. In this case, there is no information exchange between subtrees, and the performance of the HMTA is identical to that of the SA. Therefore, for a general sparse channel, the increase in BER over the conventional SA is likely due to the quality of the information exchanged. In the next section, we will explore techniques for exchanging soft information which provides a better performance over HMTA. However, in applications that consider low complexity as a top priority, the proposed HMTA method using hard estimates can provide relatively good performance (about 2 dB loss of performance compared to the VA).

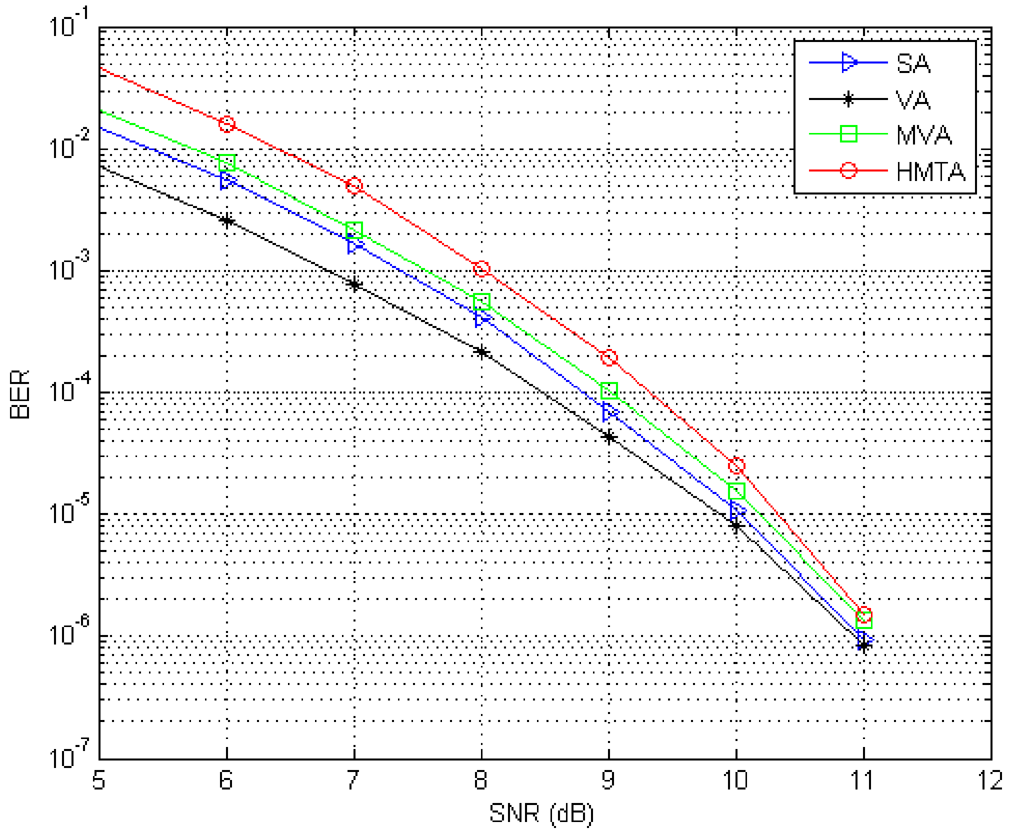


Figure 4.6: Performance comparison of the VA, SA, MVA and HMTA for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps.

4.3 Multiple Tree Algorithm Using Soft Information

The HMTA dramatically reduces the computational complexity for estimating a data sequence transmitted over a sparse ISI channel in high SNR. However, simulation results have shown that the HMTA does not perform as well as competing methods, particularly at low SNR. Our simulation results show that the loss of performance is due to the hard estimates made and exchanged among subtrees. The hard estimates are made based on the branches that have been extended in the corresponding subtree, where information is lost when we quantize the bit value to be $+1$ or -1. Especially when SNR is low, the algorithm is more likely to make wrong hard estimates on certain symbols, and the loss of a large amount of information results in decreased performance of the HMTA. In order to enhance the reliability of the information exchanged in the MTA, instead of making hard estimates, we compute soft estimates, which are defined as the probability that each symbol is equal to $+1$ or -1, and incorporate them into the path metric. We denote the MTA using soft estimates as SMTA.

4.3.1 Derivation of the Path Metric

The path metric of SMTA for the input sequence $\mathbf{x}_j^{n_j}$ is similar to that of the HMTA shown in (4.13),

$$m(\mathbf{x}_j^{n_j}) \approx P(\mathbf{x}_j^{n_j})P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})P(\mathbf{y}_{n_j+J}^{N_j}). \tag{4.26}$$

The prior term $P(\mathbf{x}_j^{n_j}) = (\frac{1}{2})^{1+(n_j-j)/J}$ and the bias term $P(\mathbf{y}_{n_j+J}^{N_j})$ is the same as what is defined in (4.16) for HMTA. We follow the same procedures as followed for HMTA to derive the path metric of the SMTA. The only difference is the conditional likelihood of y_k, which can be written as

$$\begin{aligned} P(y_k|\mathbf{x}_j^{n_j}) &= P\left(y_k|\mathbf{x}_{(a)}^j\right) \qquad (4.27) \\ &= \sum_l P\left(y_k|\mathbf{x}_{(a)}^j, \mathbf{x}_{(u)}^j = \mathbf{q}_l\right) P(\mathbf{x}_{(u)}^j = \mathbf{q}_l), \end{aligned}$$

where the vector $\mathbf{q}_l = \{\mathbf{q}_l(d)\}$, $1 \le d \le D$, denotes the l-th realization of a length-D BPSK sequence. Given the sequence $\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}$, the received signal y_k is Gaussian, $y_k \sim$

$\mathcal{N}(\mathbf{h}_a^T \mathbf{x}_{k-p_0}^{k-p_{L_a-1}}, \sigma^2)$. Therefore,

$$P\left(y_k | \mathbf{x}_{(a)}^j, \mathbf{x}_{(u)}^j = \mathbf{q}_l\right) = \tag{4.28}$$
$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T \{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}_{(u)}^j = \mathbf{q}_l})^2}{2\sigma^2}\right),$$

where $\{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}_{(u)}^j = \mathbf{q}_l}$ denotes the sequence $\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}$ with $\mathbf{x}_{(u)}^j = \mathbf{q}_l$. Noting that the elements of $\mathbf{x}_{(u)}^j$ are independent, we have

$$P(\mathbf{x}_{(u)}^j = \mathbf{q}_l) = \prod_{d=1}^{D} P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}_l(d)), \tag{4.29}$$

where $P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}_l(d))$ is the soft estimate that must be obtained from another subtree. Compared to the hard estimates, the soft estimates keep the probabilities of the possible values that $\mathbf{x}_{(u)}^j(d)$ can take. All the information of $\mathbf{x}_{(u)}^j(d)$ are maintained and exchanged among subtrees, which provides better quality of the exchanged information than that of the HMTA.

Suppose, for example, that symbol $\mathbf{x}_{(u)}^j(d)$ is available in subtree t, $t \neq j$. The soft estimate $P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}_l(d))$ is obtained similar to the HMTA,

$$P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}_l(d)) \approx \sum_{\mathbf{x}_t^{n_t} \in S_t:\, \mathbf{x}_{(u)}^j(d) = \mathbf{q}_l(d)} P(\mathbf{x}_t^{n_t} | \mathbf{y}_t^{N_t}), \tag{4.30}$$

where S_t denotes the stack for subtree t.

Substituting (4.28) and (4.29) into (4.26) produces a closed-form expression for the path metric of the SMTA.

$$\begin{aligned} m(\mathbf{x}_j^{n_j}) &\approx P(\mathbf{x}_j^{n_j})P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})P(\mathbf{y}_{n_j+J}^{N_j}) \qquad (4.31) \\ &\approx (\frac{1}{2})^{1+(n_j-j)/J} \prod_{k=n_j+J}^{N_j} \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{y_k^2}{2\sigma_b^2}\right) \\ &\quad \times \prod_{k=j:J:n_j} \left\{ \sum_l \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T \{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}_{(u)}^j=\mathbf{q}_l})^2}{2\sigma^2} \right) \prod_{d=1}^{D} P(\mathbf{x}_{(u)}^j(d) = \mathbf{q}_l(d)) \right] \right\}. \end{aligned}$$

A flowchart of the SMTA algorithm is given in Fig. 4.7 for a sparse channel with $L_a = 3$ active taps.

4.3.2 Simulation Results for the SMTA

To illustrate the complexity and performance of the proposed SMTA, we also compare it to the VA, SA and MVA. The same simulation conditions used for the HMTA are used here. The average number of computations performed per transmitted block as a function of SNR is shown in Table 4.2 for the four detection methods considered. For SNR from 5 dB to 10 dB, the SMTA achieves significant computational complexity reduction compared to the other three methods. For SNR greater than 10 dB, the SA and SMTA are likely to follow a single path and explore very little of the tree, causing their similar complexity.

A performance comparison of the four methods is shown in Fig. 4.8. The SMTA suffers some performance loss compared to the VA, but its performance is very close to that of the SA and MVA. The performance difference among the methods decreases as SNR increases. Therefore, with only slight performance degradation, the proposed SMTA provides a computationally efficient approach to data detection for sparse ISI channels at moderate SNRs.

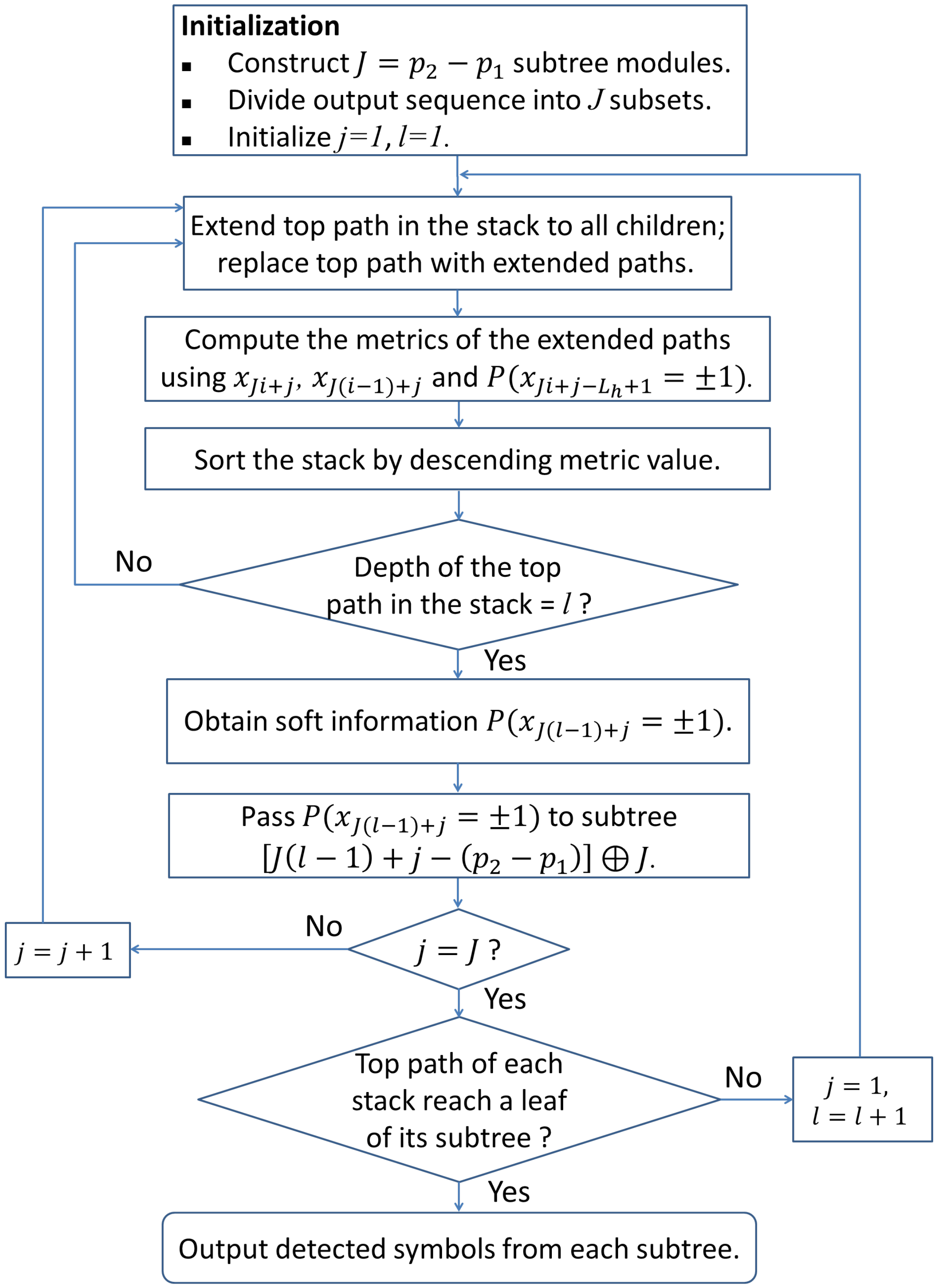

Initialization
Construct $J = p_2 - p_1$ subtree modules.
Divide output sequence into J subsets.
Initialize $j=1$, $l=1$.
Extend top path in the stack to all children; replace top path with extended paths.
Compute the metrics of the extended paths using x_{Ji+j}, $x_{J(i-1)+j}$ and $P(x_{Ji+j-L_h+1} = \pm 1)$.
Sort the stack by descending metric value.
Depth of the top path in the stack = l ?
No
Yes
Obtain soft information $P(x_{J(l-1)+j} = \pm 1)$.
Pass $P(x_{J(l-1)+j} = \pm 1)$ to subtree $[J(l-1) + j - (p_2 - p_1)] \oplus J$.
$j = J$?
No
$j = j + 1$
Yes
Top path of each stack reach a leaf of its subtree ?
No
$j = 1$, $l = l + 1$
Yes
Output detected symbols from each subtree.

Figure 4.7: Flowchart representation of the MTA method using soft information

Table 4.2: Computations performed for the VA, MVA, SA and SMTA using a length-1200 sequence for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps.

Number of Computations				
	VA	**MVA**	**SA**	**SMTA**
4dB	76800	9600	154518	74698
5dB	76800	9600	14429	5687
6dB	76800	9600	7551	4758
7dB	76800	9600	5059	3207
8dB	76800	9600	3286	2426
9dB	76800	9600	2726	2411
10dB	76800	9600	2512	2405
11dB	76800	9600	2408	2401

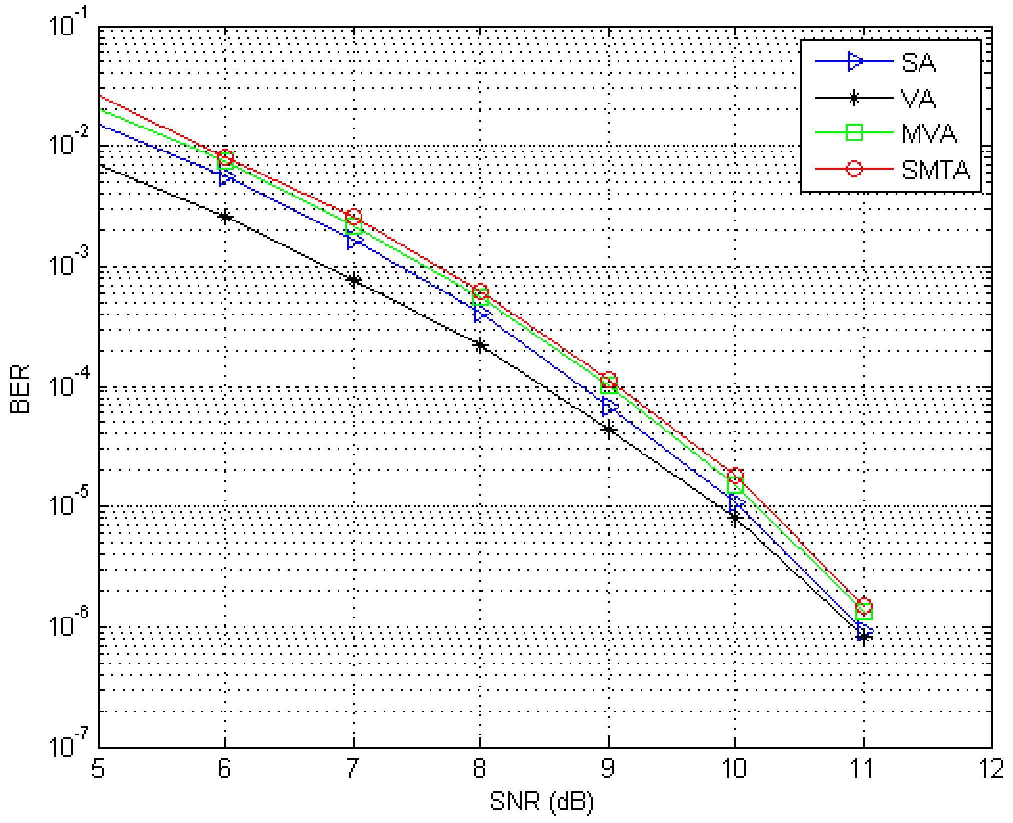


Figure 4.8: Performance comparison of the VA, SA, MVA, and SMTA for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps.

4.4 Comparison between the HMTA and SMTA

The simulation results show that the SMTA exhibits performance very similar to that of the SA and notably better than the HMTA. The only difference between the SMTA and HMTA is the information exchanged among the subtrees. We can see that the shared soft information has higher reliability than the hard information. In order to illustrate the difference between the hard and soft information, we analyze the quality of these two kinds of estimates and study their reliability as the tree search proceeds.

4.4.1 Soft Information vs. Hard Information

In order to compare the quality of the hard and soft estimates exchanged among subtrees, we assume that the knowledge of the transmitted sequence $\mathbf{x}_1^N$ is available, which means for subtree j, we can locate the correct path associated with the sequence $\mathbf{x}_j^{N_j}$. The corresponding likelihood term of the path metric for a partial length sequence, $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$, can be obtained using (4.16) as

$$\begin{aligned} P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j}) &= \prod_{k=j:J:n_j} P(y_k|\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}) \qquad (4.32) \\ &= \prod_{k=j:J:n_j} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T \mathbf{x}_{k-p_0}^{k-p_{L_a-1}})^2}{2\sigma^2}\right). \end{aligned}$$

Let us define two likelihood ratios, γ_h^l and γ_s^l, as

$$\gamma_h^l = \frac{P(\mathbf{y}_j^{n_j}|\mathbf{x}_{j(h)}^{n_j})}{P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})}, \qquad (4.33)$$

$$\gamma_s^l = \frac{P(\mathbf{y}_j^{n_j}|\mathbf{x}_{j(s)}^{n_j})}{P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})}, \qquad (4.34)$$

where $P(\mathbf{y}_j^{n_j}|\mathbf{x}_{j(h)}^{n_j})$ and $P(\mathbf{y}_j^{n_j}|\mathbf{x}_{j(s)}^{n_j})$ are the likelihood terms in the path metric of HMTA and SMTA, respectively. $\mathbf{x}_{j(h)}^{n_j}$ is the sequence associated with l-th path in subtree j for HMTA which contains the hard estimates obtained from other subtrees. $\mathbf{x}_{j(s)}^{n_j}$ is the sequence associated with l-th path in subtree j for SMTA that contains the soft estimates obtained from other subtrees. The likelihood term $P(\mathbf{y}_j^{n_j}|\mathbf{x}_{j(h)}^{n_j})$ indicates the probability that the sequence $\mathbf{y}_j^{n_j}$ is received given the possible transmitted sequence $\mathbf{x}_{j(h)}^{n_j}$ which includes the hard estimates, while $P(\mathbf{y}_j^{n_j}|\mathbf{x}_{j(s)}^{n_j})$ represents the same probability, given $\mathbf{x}_{j(s)}^{n_j}$ which includes the soft estimates. Therefore, the ratios γ_h^l and γ_s^l show how likely it is that $\mathbf{x}_{j(h)}^{n_j}$ and $\mathbf{x}_{j(s)}^{n_j}$ are following the correct path $\mathbf{x}_j^{n_j}$. Since the likelihood $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$ is maximized, we will have $0 < \gamma_h^l, \gamma_s^l \leq 1$. As the tree search proceeds, paths with different lengths are extended. For a fair comparison between the hard and soft information, we average the ratios we obtain for each path by the total number of extended paths,

$$\gamma_h = \frac{1}{L^h} \sum_l \gamma_h^l, \tag{4.35}$$

$$\gamma_s = \frac{1}{L^s} \sum_l \gamma_s^l, \tag{4.36}$$

where L^h and L^s denote the numbers of extended paths for HMTA and SMTA, respectively. When comparing the two ratios γ_h and γ_s, the larger ratio indicates that the exchanged information is more likely to be the correct estimate.

In order to compare the hard and soft information, we conducted simulations for a 6-tap sparse channel with 3 active taps. A sequence with 1200 BPSK symbols is passed over the sparse channel. The active channel taps are drawn from a Gaussian distribution, $\mathcal{N}(0, 1)$. The HMTA and SMTA are used to detect the transmitted sequence. At each path of the two methods, γ_h^l and γ_s^l are computed. When the tree search reaches a leaf, the total number of extended paths are counted for all the subtrees to obtain the average ratios. We plot the resulting average ratios γ_h and γ_s against different values of SNR in Fig. 4.9. From the simulation results, we can see that when SNR is low, the soft estimates are more likely to be correct than the hard estimates. As the SNR increase, the two ratios converge which indicates that the hard and soft estimates have the similar likelihood to follow the correct path in the tree. We also plot the performance comparison between the HMTA and SMTA for the simulations conducted in the previous two sections in Fig. 4.10. The tendency of the two ratios coincides with the BER performance comparison for the HMTA and SMTA.

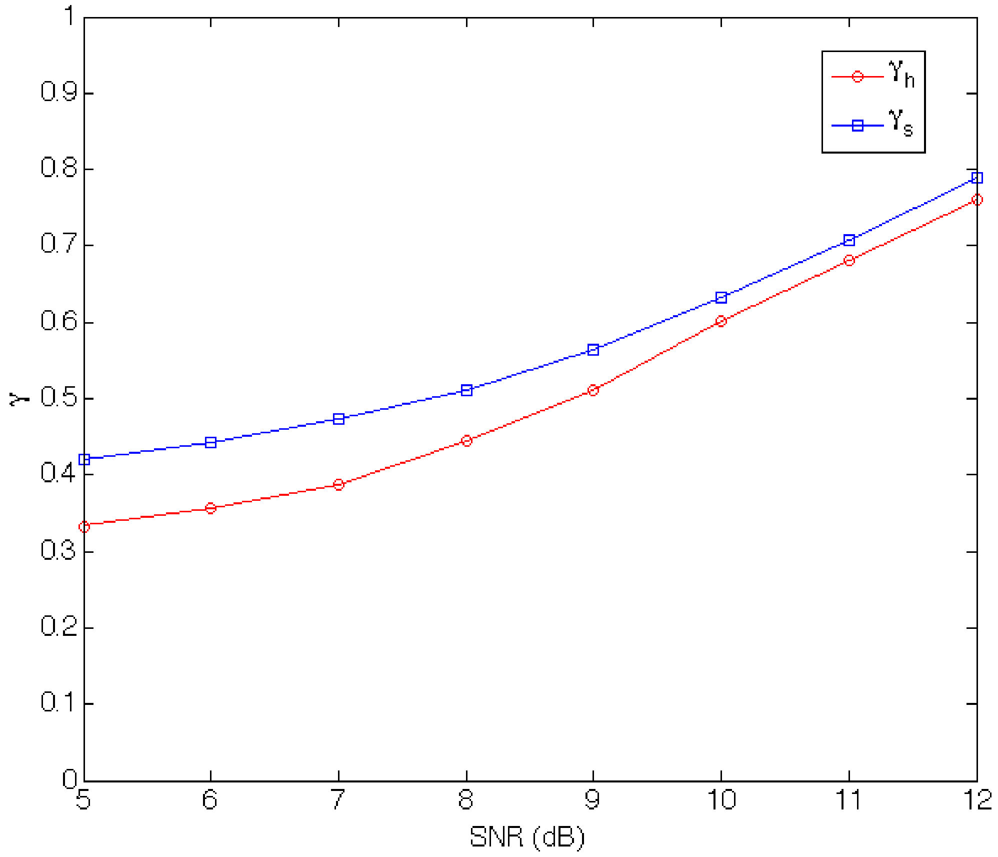


Figure 4.9: Comparison between the two ratios γ_h and γ_s for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps.

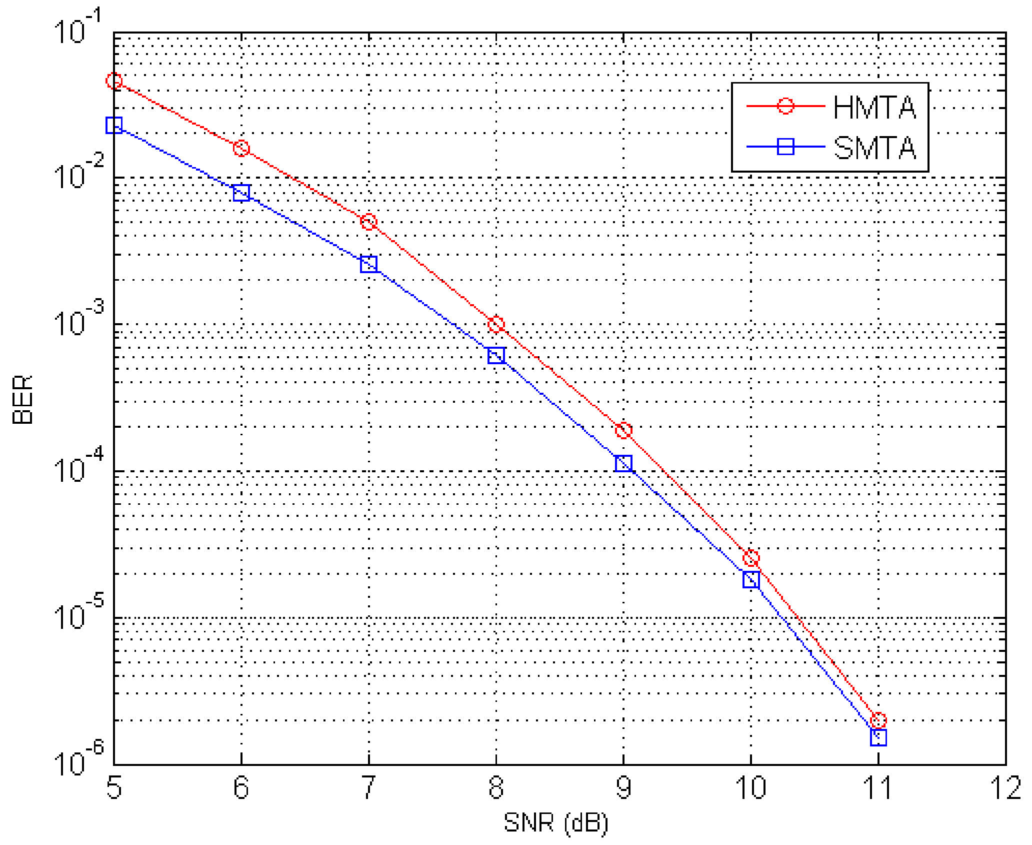


Figure 4.10: Performance comparison of the HMTA and SMTA for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps.

4.4.2 Complexity Comparison

Soft information is shown to be more reliable, and hence the SMTA outperforms the HMTA. However, calculating the path metric when soft information is used requires additional computational effort. The soft and hard estimates, both obtained by computing the probability of all possible values that the unavailable bits can take, have very similar computational complexity. Therefore, we take the complexity of computing the path metric as the criterion to compare the two methods. We assume that the hard estimates $\hat{\mathbf{x}}^j_{(u)}$ and soft estimates

$P(\mathbf{x}_{(u)}^{j} = \mathbf{q}_l)$ have already been obtained.

The path metric of the HMTA and SMTA has three components: the prior probability of the transmitted symbols $P(\mathbf{x}_j^{n_j})$, conditional likelihood term $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$, and bias term $P(\mathbf{y}_{n_j+J})^{N_j}$. The computation of $P(\mathbf{x}_j^{n_j})$ and $P(\mathbf{y}_{n_j+J})^{N_j}$ are same for HMTA and SMTA. Thus, we only need to consider the conditional likelihood term $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$. For the HMTA, the conditional likelihood of y_k in the path metric is defined as

$$P(y_k|\mathbf{x}_j^{n_j}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T \{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}_{(u)}^{j} = \hat{\mathbf{x}}_{(u)}^{j}})^2}{2\sigma^2} \right). \tag{4.37}$$

With hard estimates, the corresponding unavailable bits are set to the value of $\hat{\mathbf{x}}_{(u)}^{j}$, and $P(y_k|\mathbf{x}_j^{n_j})$ is computed by performing the $\exp(\cdot)$ operation only once. Hence, to obtain the conditional likelihood term $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$, the $\exp(\cdot)$ operation is implemented $1 + (n_j - j)/J$ times.

For the SMTA, we consider all the possibilities of the unavailable bits values, and the conditional likelihood of y_k is given by

$$\sum_l \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - \mathbf{h}_a^T \{\mathbf{x}_{k-p_0}^{k-p_{L_a-1}}\}_{\mathbf{x}_{(u)}^{j} = \mathbf{q}_l})^2}{2\sigma^2} \right) \prod_{d=1}^{D} P(\mathbf{x}_{(u)}^{j}(d) = \mathbf{q}_l(d)) \right], \tag{4.38}$$

where $\mathbf{q}_l$ is the l-th realization of a length-D binary vector. D is the number of unavailable bits. It is easy to see that there are 2^D combinations of the binary vector, and l takes the values $l = 1, \ldots, 2^D$. To compute the conditional likelihood term $P(\mathbf{y}_j^{n_j}|\mathbf{x}_j^{n_j})$, the unavailable bits are set to the l-th binary sequence, and the $\exp(\cdot)$ operation needs to be performed $[1 + (n_j - j)/J]2^D$ times.

The complexity comparison between the two methods is provided in Table 4.3. We can see that the metric computation of the SMTA is more complex than that of the HMTA by a factor of 2^D. As D increases, the complexity gap between the two methods becomes more obvious. The better performance of the SMTA is obtained by increasing computational complexity. However, since the SMTA uses soft estimates in the tree search process, the SMTA is more likely to follow the correct path. Hence, the SMTA extends fewer branches of the tree and fewer path metric computations must be performed, especially when the SNR is low. To verify this, we run simulations for the two methods using 1000 blocks of a length-1200 sequence for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps. The number of metric computations is averaged over the 1000 blocks for the two methods and is shown in Table 4.4. From the simulation results, we can see that the SMTA has higher metric computational complexity but requires a smaller number of metric computations. For this simulation, the sparse channel has 3 active taps, and therefore we have $J = 3$ and $D = 1$. At 4 dB, the average number of the $\exp(\cdot)$ operations performed is $122047 \times 400 = 48818800$ for the HMTA and $74698 \times 400 \times 2^1 = 59758400$ for the SMTA. The SMTA has about 20% higher complexity than the HMTA for this case. When D is not very large, the computational complexity of the SMTA is only slightly increased compared to the HMTA. The extra effort may be deemed worthwhile when improvement in detection performance is desired.

Table 4.3: Computational complexity between the HMTA and SMTA in terms of the number of $\exp(\cdot)$ operation.

Path Metric Computations	
	Number of $\exp(\cdot)$ **operation**
HMTA	$1 + (n_j - j)/J$
SMTA	$[1 + (n_j - j)/J]2^D$

Table 4.4: Number of metric computations for the HMTA and SMTA using a length-1200 sequence for a length-$L_h = 6$ sparse channel with $L_a = 3$ active taps.

Number of Path Metric Computations		
	HMTA	**SMTA**
4dB	122047	74698
5dB	8539	5687
6dB	5809	4758
7dB	4205	3207
8dB	2524	2426
9dB	2441	2411
10dB	2407	2405
11dB	2402	2401

Chapter 5: Computationally Efficient Sequential Detection for Unknown ISI Sparse Channels

In the previous chapters, we introduced several equalization techniques in both symbol-by-symbol and sequential detection schemes. We note that all of them are proposed on the assumption that the channel is known at the receiver. Therefore, channel estimation needs to be performed using a training sequence before the equalizers can operate. The training sequence is a sequence of bits sent prior to the information data and is known at the receiver. The communication channel is estimated based on knowledge of the training sequence and its observations. Transmitting a training sequence consumes bandwidth that could otherwise be used for transmission of information, and training sequences must be transmitted at regular intervals when the channel is time varying. In order to improve spectral efficiency, blind or semi-blind methods for channel identification and equalization have been proposed [57–63]. The term "blind" means that the receiver has no information about either the transmitted sequence or the channel. Only the received signal and any available priors on the transmitted signal are used to design the receiver. In [55], a framework is given using a Bayesian maximum likelihood sequence detector for ISI channels. The channel taps are considered as stochastic variables drawn from a known probability distribution. A tree search stack based algorithm is used to estimate the transmitted sequence based on the Bayesian probability metric, which includes learning the channel implicitly. The method achieves promising performance results. However, it only provides a fundamental structure for general channels. When the channel is sparse, the method is not able to describe the channel sparsity, resulting in lost opportunity to enhance the performance of the detector.

In this chapter, we will introduce several blind sequential detection techniques using the stack algorithm for sparse ISI channels. In Section 5.1, two conventional sparse channel estimation methods will be introduced. In Section 5.2, we combine sparse channel estimation with the stack algorithm to achieve blind sequential detection. In Section 5.3, a novel computationally efficient blind sequential detection method that does not explicitly estimate the channel is proposed. The proposed method reduces the computational complexity by approximately 75% and yields better performance than the conventional methods introduced in Section 5.2.

5.1 Sparse Channel Estimation

The wireless channel poses a great challenge for reliable high speed communications. The transmitted signal experiences various types of distortion, as described in Chapter 2. If the channel can be accurately estimated, we can efficiently recover the transmitted signal at the receiver. All of the equalization techniques we have introduced in previous chapters are proposed based on knowledge of the communication channel. The symbol-by-symbol ZF and MMSE equalizers are designed directly from channel parameters, shown in (2.14) and (2.16). The sequential stack algorithm incorporates knowledge of the channel in computing the path metric to determine the most likely transmitted sequence, shown in (4.11). However, in a realistic communication system, the channel is unknown and must be estimated if we want to apply these methods. Furthermore, in applications such as UWA, UWB and HDTV systems, the channels are sparse. Conventional methods (like least-squares estimation) used to estimate general channels will have poor performance. In the next three subsections, we will introduce methods that effectively estimate sparse channels, including matching pursuit (MP) and orthogonal matching pursuit (OMP). These two methods estimate the channel by using a training sequence. We will show that they can be combined with the stack algorithm to achieve blind sequential detection in Section 5.2.

5.1.1 System Model

We first introduce the notation for describing a communication system with a sparse channel. A sequence x_k, $k = 1, 2, \ldots, M$, is transmitted over a length-L_h sparse ISI channel $\mathbf{h}$

with additive white Gaussian noise (AWGN) $w_k \sim \mathcal{N}(0, \sigma^2)$. The received signal at time instant k, y_k, $k = 1, 2, \ldots, M$, can be expressed as

$$y_k = \sum_{i=0}^{L_h-1} h_i x_{k-i} + w_k. \tag{5.1}$$

We assume that the channel has only L_a active (non-zero) taps, $\mathbf{h}_a = [h_0, h_1, \ldots, h_{L_a-1}]^T$, where $L_a \ll L_h$.

$$\mathbf{h} = [h_0, 0, \ldots, 0, h_1, 0, \ldots, 0, h_{L_a-1}]^T. \tag{5.2}$$

The length-M received sequence can be expressed in a matrix form as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} x_1 & 0 & 0 & 0 & \ldots & 0 & \\ x_2 & x_1 & 0 & 0 & \ldots & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_M & x_{M-1} & x_{M-2} & \ldots & x_{M-L_h+2} & x_{M-L_h+1} & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{L_h} & x_{L_h-1} & x_{L_h-2} & \ldots & x_2 & x_1 & \end{bmatrix} \times \begin{bmatrix} h_0 \\ 0 \\ \vdots \\ h_1 \\ \vdots \\ 0 \\ h_{L_h-1} \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}, \tag{5.3}$$

where we assume that $M > L_h$ to ensure accurate estimation of the channel. The equation above in matrix form can be simplified as

$$\mathbf{y} = \mathbf{X}\mathbf{h} + \mathbf{w}, \tag{5.4}$$

where the signal matrix $\mathbf{X} \in R^{M \times L_h}$. Let us denote the columns of the matrix $\mathbf{X}$ as $\mathbf{a}_i$, $i = 1, 2, \ldots, L_h$. The received sequence, $\mathbf{y}$, can be considered as a linear sparse combination of the columns $\mathbf{a}_i$. In order to estimate the channel $\mathbf{h}$, the detector explores the properties of the received sequence $\mathbf{y}$ and the matrix $\mathbf{X}$. To illustrate the sparse channel estimation methods, we will assume that the matrix $\mathbf{X}$ is known. In the blind sequential detection that we will introduce in following sections, $\mathbf{X}$ is no longer available and we will replace it with other information. We also assume that the receiver has knowledge of the length of the channel L_h. Most sparse channel estimation methods follow a two-step procedure: the sparsity of the channel is estimated first and then the active channel taps are estimated based on the estimated sparsity. MP and OMP are two methods that are widely used due to their easy implementation.

5.1.2 Matching Pursuit

Matching pursuit (MP), originally proposed for sparse recovery problems, is a type of numerical technique which finds the best "matching" sparse solution to the associated linear system from an over-complete collection of elementary signals [64]. It can be used in a wide variety of applications such as image coding [65–67], direction of arrival estimation [68], and sparse signal representation [69]. Due to the common structure shared by the applications, MP received much attention in the area of sparse channel estimation [70, 71]. MP is a greedy type algorithm which iteratively generates the sparse solution for any linear underdetermined system. For sparse channel estimation, the system equations are defined in (5.3). MP uses sequential forward selection to determine the sparse representation of the channel $\mathbf{h}$ from the signal matrix $\mathbf{X}$ and received signal $\mathbf{y}$. At each iteration, the algorithm chooses the column of matrix $\mathbf{X} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{L_h}]$ that best matches the residual signal vector. The index of the column is stored, and the channel tap value at the corresponding index is computed. The process is terminated when the number of active channel taps is reached or a stop criterion is met. The procedures of MP can be summarized as:

- Initialization: Denote the residual signal vector as $\mathbf{b}_0 = \mathbf{y}$ and a collection used to store active indices as $I_p = \{\}$.
- Step 1. At the p-th iteration, selection is performed by finding the largest projection

of the residual signal vector $\mathbf{b}_{p-1}$ along the direction of each column of the matrix $\mathbf{X}$,

$$k_p = \underset{i}{\text{argmax}} \frac{|\mathbf{a}_i^H \mathbf{b}_{p-1}|^2}{||\mathbf{a}_i||^2}, \; i = 1, 2, \ldots, L_h. \tag{5.5}$$

The index collection is updated as follows:

$$I_p = \begin{cases} I_{p-1} \cup k_p, & \text{if } k_p \notin I_{p-1}, \\ I_{p-1}, & \text{otherwise.} \end{cases} \tag{5.6}$$

- Step 2. The new residual signal vector is computed by removing the projection of $\mathbf{b}_{p-1}$ along the direction of $\mathbf{a}_{k_p}$,

$$\mathbf{b}_p = \mathbf{b}_{p-1} - \frac{(\mathbf{a}_{k_p}^H \mathbf{b}_{p-1})\mathbf{a}_{k_p}}{||\mathbf{a}_{k_p}||^2}. \tag{5.7}$$

The estimated active channel tap value at position k_p is then computed as

$$\hat{h}_{k_p} = \frac{\mathbf{a}_{k_p}^H \mathbf{b}_{p-1}}{||\mathbf{a}_{k_p}||^2}. \tag{5.8}$$

The stop criterion is checked to determine if the iteration needs to be terminated. If the length of the active channel taps L_a is known and $p = L_a$, the iteration is terminated; otherwise we set $p = p + 1$ and go back to Step 1. If L_a is an unknown parameter, the iteration can be terminated when $||\mathbf{b}_p|| \leq \epsilon$, where ϵ is a small pre-defined constant [64].

Convergence of MP is guaranteed for $p \to \infty$, due to the energy conservation property of the algorithm [72]. Compared to the sparse least-squares (SpLS) algorithm [73] which requires solutions of two LS problems, MP provides a more efficient way to find the sparse representation of the channel. However, it is also obvious that at each iteration, the projection of the residual signal vector is computed over all the vectors of the dictionary matrix $\mathbf{X}$. A previously selected vector might be re-selected, which will slow down convergence and increase the computational burden. In order to avoid re-selection, an alternative MP algorithm is proposed.

5.1.3 Orthogonal Matching Pursuit

Orthogonal matching pursuit (OMP) was developed as an extension of MP [74]. The main difference between OMP and MP is that at each iteration OMP selects the index by finding the maximum orthogonal projection of the residual signal vector on to the columns of $\mathbf{X}$ in the orthogonal complement of the active index collection. Therefore, OMP avoids the re-selection problem which accelerates convergence of the algorithm and enhances the accuracy of sparse channel estimation, but requires extra effort in computation. To better illustrate OMP, we summarize its procedures as follows,

- Initialization: Denote the residual signal vector as $\mathbf{b}_0 = \mathbf{y}$ and a collection used to store active indices as $I_p = \{\}$.
- Step 1. At the p-th iteration, selection is performed by finding the largest projection of the residual signal vector $\mathbf{b}_{p-1}$ along the direction of each column of the matrix $\mathbf{X}^t$,

$$k_p = \underset{i}{\operatorname{argmax}} \frac{|\mathbf{a}_i^H \mathbf{b}_{p-1}|^2}{||\mathbf{a}_i||^2}, \; i \in \{1, \ldots, L_h\} \notin I_{p-1}. \tag{5.9}$$

The index collection is updated as follows:

$$I_p = \begin{cases} I_{p-1} \cup k_p, & \text{if } k_p \notin I_{p-1}, \\ I_{p-1}, & \text{otherwise.} \end{cases} \tag{5.10}$$

- Step 2. The re-selection problem is avoided by using the stored dictionary. The selected vector $\mathbf{a}_{k_p}$ is orthogonalized by the Gram-Schmidt algorithm [75] as,

$$\mathbf{q}_p = \mathbf{a}_{k_p} - \sum_{j=0}^{p-1} \frac{(\mathbf{a}_{k_p}^H \mathbf{q}_j)\mathbf{q}_j}{||\mathbf{q}_j||^2}. \tag{5.11}$$

  The new residual signal vector is computed by removing the orthogonal projection of $\mathbf{b}_{p-1}$ along the direction of $\mathbf{q}_{k_p}$,

$$\mathbf{b}_p = \mathbf{b}_{p-1} - \frac{(\mathbf{q}_{k_p}^H \mathbf{b}_{p-1})\mathbf{q}_{k_p}}{||\mathbf{q}_{k_p}||^2}. \tag{5.12}$$

  The estimated active channel tap value at position k_p is then computed as

$$\hat{h}_{k_p} = \frac{\mathbf{q}_{k_p}^H \mathbf{b}_{p-1}}{||\mathbf{q}_{k_p}||^2}. \tag{5.13}$$

  The same stop criterion as for MP is applied to terminate the iterations of OMP.

OMP brings in the orthogonalization mechanism to avoid the re-selection problems of MP and improves the sparsity detection accuracy with only a small amount of extra computational complexity. In order to illustrate the performance of the two introduced methods, we compare the estimation accuracy of MP and OMP using the same sparse channel and known sequence.

In the simulation, 100 blocks of a length-12 training sequence are used. For each block, a length-10 sparse channel with 3 active taps is randomly generated. The 3 active channel taps are drawn from a Gaussian distribution with 0 mean and variance 1. AWGN is added during transmission. The number of correctly detected active channel taps for 100 blocks is evaluated as the criterion of the estimation accuracy for comparison between the two methods. The simulation results are shown in Fig. 5.1 for SNR= 5 dB, 10 dB, and 15 dB.

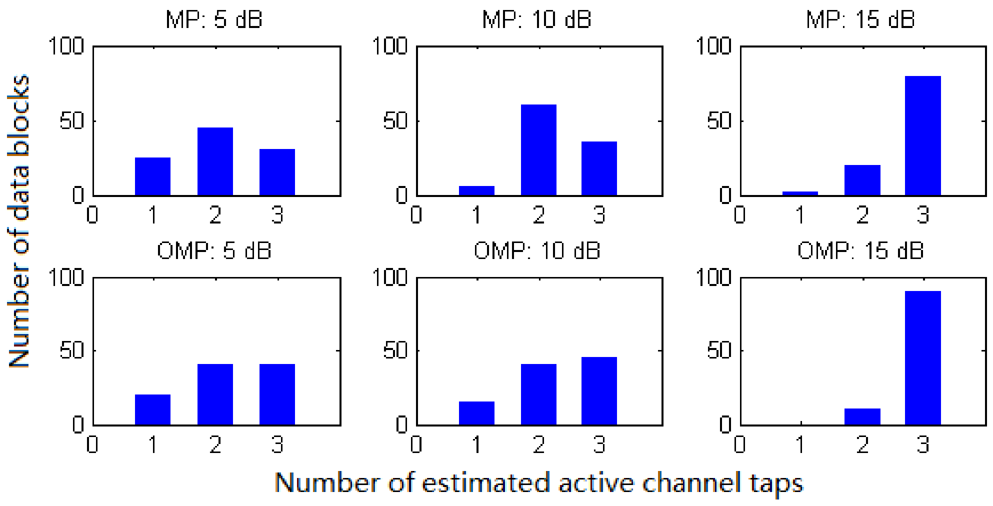


Figure 5.1: Estimation accuracy comparison between MP and OMP for 100 blocks of length-10 training sequence when SNR is 5 dB, 10 dB, and 15 dB. A length-10 sparse channel with 3 active taps is randomly generated for each block.

We can see that OMP gives more accurate channel sparsity estimation than MP for the range of the SNR values considered. Considering complexity, however, MP has its own advantage. There is a trade-off between complexity and performance for the two methods that may make one or the other more attractive for a particular application.

5.2 Blind Sequential Detection with Sparse Channel Estimation

We have introduced two popular methods for estimating sparse channels. MP and OMP are two greedy algorithms that sequentially estimate the sparse channel by using a training sequence. In this section, we will apply them in blind sequential detection problems along with the SA.

For blind sequential detection, no training sequence is available to use, and therefore the sparse channel is an unknown parameter at the receiver. In the tree structure of the SA, each path represents a possible realization of the transmitted sequence, which can be used to construct the signal matrix $\mathbf{X}$. MP/OMP can use $\mathbf{X}$ to estimate the unknown channel along each path. Using the estimated channel in combination with the observations and the symbol sequence associated with each path, the path metrics of the SA can be computed, as introduced in Chapter 4. The metrics are used to guide the tree search to find the most likely transmitted sequence. Such a mechanism establishes the foundation to combine MP/OMP and the SA.

5.2.1 System Model

In order to enhance the performance of the detector, the information bits are passed through an error control encoder with rate $R = \frac{1}{r}$. No training sequence is sent. For a block of information bits with length N, denoted by $\mathbf{b}_1^N$, a length-rN sequence of coded bits, $\mathbf{x}_1^{rN}$, is generated. The encoded sequence is transmitted over a length-L_h sparse ISI channel $\mathbf{h}$ with AWGN w_k of variance σ^2. The system model is showed in the Fig. 5.2. The received sequence $\mathbf{y}_1^{rN} = \{y_1, \ldots, y_{rN}\}$ is the input to a detector that blindly estimates the transmitted information bits $\mathbf{b}_1^N$, where the received signal at time instant k, y_k, $k = 1, 2, \ldots, rN$, is defined in (5.1). We assume that the receiver has knowledge of the code generator polynomial C, the variance of the AWGN σ^2, and the channel length L_h. The sparse ISI channel $\mathbf{h}$ is unknown. For simplicity, only BPSK encoded data is considered. Extension to transmission of data from large constellations is straightforward.

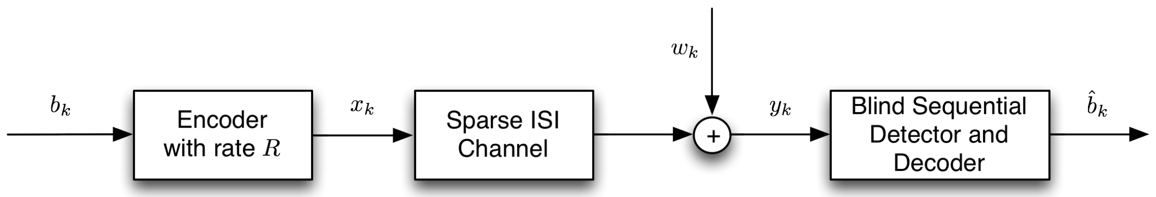


Figure 5.2: System model for blind sequential detection. The data is passed through an error control encoder with rate R. The coded sequence is transmitted over a sparse ISI channel and processed by the blind sequential detector and decoder.

In order to detect the transmitted sequence without knowledge of the channel, we can formulate a joint channel and sequence estimation problem. We will show how the solution for this problem can be found using the ML criterion first. We will then extend to using the SA to approximate the ML solution.

Letting $\tilde{\mathbf{x}}_1^{rN}$ and $\tilde{\mathbf{h}}$ denote hypotheses for the transmitted sequence and sparse ISI channel, the solution of joint channel and sequence estimation using the ML criterion in the presence of AWGN can be written as

$$\begin{aligned}\left(\hat{\mathbf{b}}_1^N, \hat{\mathbf{h}}\right) &= \underset{\tilde{\mathbf{b}}_1^N, \tilde{\mathbf{h}}}{\operatorname{argmax}}\, P(\mathbf{y}_1^{rN} | \tilde{\mathbf{b}}_1^N, C, \tilde{\mathbf{h}}) \\ &= \underset{\tilde{\mathbf{b}}_1^N, \tilde{\mathbf{h}}}{\operatorname{argmin}} \sum_{k=1}^{rN} \left\| y_k - \sum_{i=0}^{L_h-1} h_i x_{k-i} \right\|^2 \\ &= \underset{\tilde{\mathbf{b}}_1^N, \tilde{\mathbf{h}}}{\operatorname{argmin}} \left\| \mathbf{y}_1^{rN} - \tilde{\mathbf{x}}_1^{rN} \tilde{\mathbf{h}} \right\|^2 .\end{aligned} \tag{5.14}$$

We can see that the ML solution $\left(\hat{\mathbf{b}}_1^N, \hat{\mathbf{h}}\right)$ is obtained by minimizing the Euclidean distance between the received sequence $\mathbf{y}_1^{rN}$ and $\tilde{\mathbf{x}}_1^{rN}\tilde{\mathbf{h}}$. The joint minimization problem defined in (5.14) has extremely high complexity. However, the task of blind ML sequential detection can be solved in two steps [76,77]: 1) We obtain channel estimates for every possible data sequence $\tilde{\mathbf{b}}_1^N$; 2) We choose the data sequence that minimizes the Euclidean distance in (5.14) with the corresponding channel estimate. The process can be expressed mathematically as

$$\left(\hat{\mathbf{b}}_1^N, \hat{\mathbf{h}}\right) = \arg\left[\min_{\tilde{\mathbf{b}}_1^N}\left(\min_{\tilde{\mathbf{h}}} \left\| \mathbf{y}_1^{rN} - \tilde{\mathbf{x}}_1^{rN}\tilde{\mathbf{h}} \right\|^2\right)\right]. \tag{5.15}$$

The two-step procedure gives us a fundamental structure to simplify the process of finding the ML solution in joint channel and sequence detection. Furthermore, the channel estimation and sequence detection become two separated processes. We have multiple choices of different algorithms that can be applied to perform each step. In [77] for example, the first step (the inner minimization) is solved by means of LS channel estimation, and the second step (the outer minimization) is carried out using the VA. The trade-off between complexity and performance will be the criterion for us to decide which method is appropriate for specific applications.

5.2.2 Blind Sequential Detection Using MP and OMP

The two-step optimization method provides a framework for blind sequential detection problems. In [77], a blind trellis search technique is proposed using conventional LS and the VA. The results are reported to be good for short channels. However, for channels with a sparse impulse response and a very long delay spread, LS cannot describe the channel sparsity, and applying the VA on long channels requires prohibitive complexity. Therefore, we propose a computationally efficient method to combine the sparse channel estimation algorithms MP/OMP with the SA to approximate the ML solution of a joint sparse channel and data estimation problem. For ease of presentation, we denote this method as SA-MP/OMP.

In order to incorporate the SA, we need to derive a way to compute the path metric without knowledge of the sparse channel. Let $\mathbf{b}_1^n = \{b_1, \ldots, b_n\}$ denote a possible realization of the transmitted sequence that a length-n path represents. A matrix $\mathbf{X}^{(rn)}$ is constructed by using the coded sequence $\mathbf{x}_1^{rn} = \{x_1, \ldots, x_{rn}\}$ as follows:

$$\mathbf{X}^{(rn)} = \begin{cases} \begin{bmatrix} x_1 & 0 & 0 & 0 & \dots & 0 \\ x_2 & x_1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{rn} & x_{rn-1} & x_{rn-2} & \dots & \dots & 0 \end{bmatrix}, & \text{if } rn < L_h, \\ \\ \begin{bmatrix} x_1 & 0 & 0 & 0 & \dots & 0 & \\ x_2 & x_1 & 0 & 0 & \dots & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{L_h} & x_{L_h-1} & x_{L_h-2} & \dots & x_2 & x_1 & \end{bmatrix}, & \text{if } rn \geq L_h. \end{cases} \tag{5.16}$$

A residual signal vector $\mathbf{b}_0$ is initialized by the received sequence $\mathbf{y}_1^{rn}$ with length rn, $\mathbf{b}_0 = \mathbf{y}_1^{rn}$. The sparse channel estimation algorithm MP/OMP is applied using $\mathbf{X}^{(rn)}$ as the training signal matrix and $\mathbf{b}_0$ as the initial residual vector. The estimated sparse channel $\hat{\mathbf{h}}^{(rn)}$ is obtained by using the same channel estimation procedure described in (5.1.2) and (5.1.3). The path metric of the length-rn sequence $\mathbf{x}_1^{rn}$ is computed based on the estimated sparse channel $\hat{\mathbf{h}}^{(rn)}$ similar to (4.11),

$$\begin{aligned} m(\mathbf{b}_1^n) &= \left(\frac{1}{2\sqrt{2\pi\sigma^2}}\right)^{rn} \prod_{k=1}^{rn} \exp\left(-\frac{1}{2\sigma^2}(y_k - \hat{\mathbf{h}}^{(rn)^T}\mathbf{x}_{k-L_h+1}^k)^2\right) \\ &\quad \times \left(\frac{1}{\sqrt{2\pi\sigma_b^2}}\right)^{rN-rn} \prod_{k=rn+1}^{rN} \exp\left(-\frac{y_k^2}{2\sigma_b^2}\right). \end{aligned} \tag{5.17}$$

Similar to joint channel estimation and sequence detection, SA-MP/OMP employs a two-step procedure to compute the metric at each path of the tree. SA-MP/OMP first estimates the sparse channel using the realization of the coded sequence associated with a path, and then it computes the path metric using the estimated channel. The path metric is used to lead our search in the tree to find the most likely transmitted sequence. The process of tree search continues until a leaf of the tree is reached. A flowchart representation of the blind sequential detection method is shown in Fig. 5.3.

SA-MP/OMP provides an efficient way to implement blind sequential detection with good performance due to the efficiency of SA and MP/OMP (the performance of SA-MP/OMP is shown in Section 5.3.7). However, there is a notable disadvantage in the design of MP/OMP. MP/OMP is designed to find the best sparse representation of a vector $\mathbf{h}$ when the system equations can be expressed as

$$\mathbf{y} = \mathbf{X}\mathbf{h}. \tag{5.18}$$

The optimal solution is obtained in a noise-free case and under perfect knowledge of the training signal matrix $\mathbf{X}$. When MP/OMP is applied in blind sequential detection problems, noise is added when information is passed through the channel, and the signal matrix $\mathbf{X}_a$ is constructed using the possible realization of the sequence associated with a certain path, which may differ from the actual transmitted sequence $\mathbf{X}$. Therefore, the system equation becomes

$$\mathbf{y} = \mathbf{X}_a\mathbf{h} + \mathbf{w} = (\mathbf{X} + \mathbf{E})\mathbf{h} + \mathbf{w}, \tag{5.19}$$

where $\mathbf{w}$ is the noise vector and $\mathbf{E}$ denotes the difference between $\mathbf{X}_a$ and the true signal matrix $\mathbf{X}$. In [78], it is shown that MP/OMP does not perform well in estimating sparse channels when SNR is low. As a result, the inaccurate channel estimates impair the performance of SA-MP/OMP. In the next section, we propose an alternative blind sequential detection method using a novel greedy algorithm. Simulation results show that it is not only more robust, but also more efficient than the MP/OMP based methods given noisy observations.

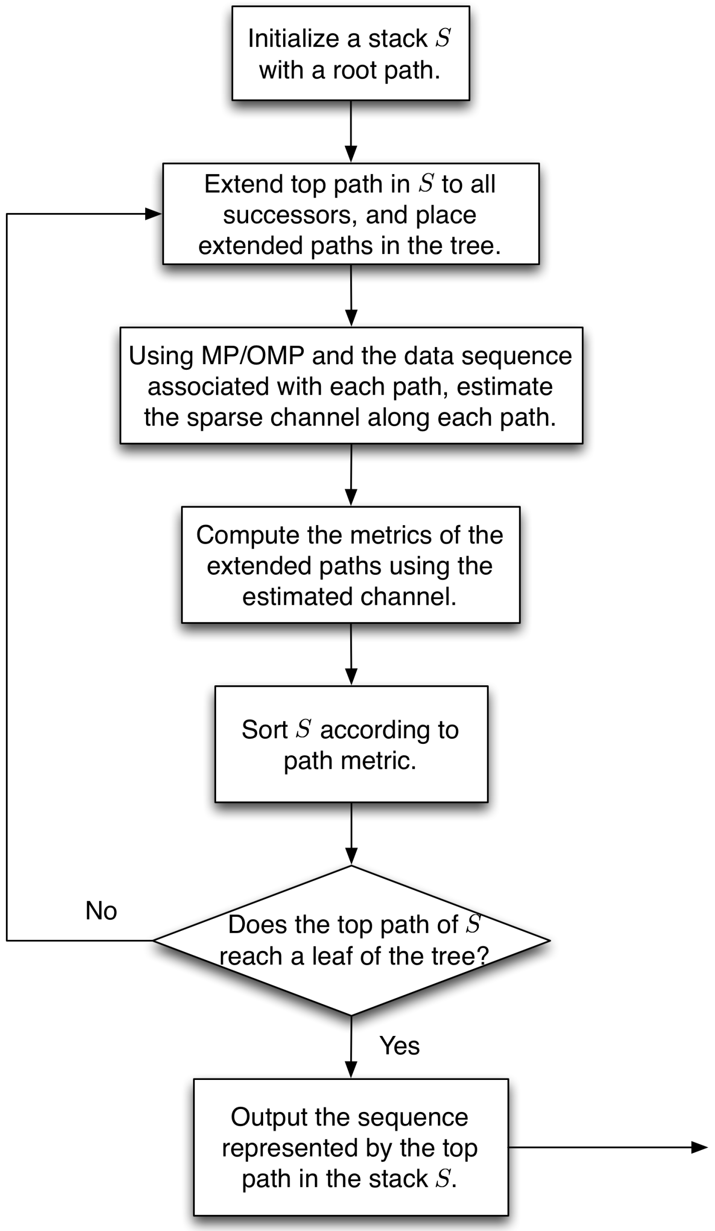

Initialize a stack S with a root path.
Extend top path in S to all successors, and place extended paths in the tree.
Using MP/OMP and the data sequence associated with each path, estimate the sparse channel along each path.
Compute the metrics of the extended paths using the estimated channel.
Sort S according to path metric.
Does the top path of S reach a leaf of the tree?
No
Yes
Output the sequence represented by the top path in the stack S.

Figure 5.3: Flowchart representation of SA-MP/OMP.

5.3 Computationally Efficient Blind Sequential Detection

The proposed SA-MP/OMP method provides an efficient way to blindly estimate the data sequence transmitted over sparse ISI channels. However, its design lacks consideration of channel noise, which will lead to performance loss, especially when SNR is low. Furthermore, due to the application of the SA, the number of extended paths could be very large, and MP/OMP must be performed along every path of a tree. Therefore, the total complexity of SA-MP/OMP could be very high. A more computationally efficient method is desirable to reduce the complexity of metric computations along each path. In this section, we propose a computationally efficient blind sequential detection method that combines the SA and a fast greedy algorithm (FGA) without explicitly estimating the sparse channel tap values. The proposed method is denoted by SA-FGA for ease of representation. Unlike MP/OMP which iteratively searches the space spanned by columns of the signal matrix, SA-FGA employs a best-first strategy to estimate the channel sparsity based on the metric of each path of the tree. Since the metric takes channel noise into consideration, SA-FGA outperforms SA-MP/OMP in a noisy environment. Additionally, the better channel estimate obtained at each path of the SA leads to a reduced number of extended paths of the tree.

5.3.1 Sparse Channel Model

Let us consider the communication system introduced in Section 5.2.1. For simplicity, only BPSK encoded data is considered. In order to describe the channel sparsity, we denote the channel by $\mathbf{h} = [h_0, \ldots, h_{L_h-1}]$ and use $d_i \in \{0, 1\}$ to denote if the i-th tap h_i is active or inactive,

$$h_i \text{ is } \begin{cases} \text{inactive}, & d_i = 0, \\ \text{active}, & d_i = 1, \end{cases} \tag{5.20}$$

$\mathbf{d} = \{d_0, \ldots, d_{L_h-1}\}$, a binary vector, denotes the sparsity of the channel. The number of active channel taps, L_a, therefore can be expressed as $L_a = ||\mathbf{d}||_0$, where the l_0-norm of vector **d** counts the number of non-zero elements in **d**.

In practical communication systems, channels cannot be exactly sparse, i.e., the inactive taps have some small values. Furthermore, the receiver has no knowledge about the exact channel. Thus, we propose a new channel model and describe channel taps h_i using two different Gaussian distributions,

$$h_i \sim \begin{cases} \mathcal{N}(0, \sigma_0^2),\ d_i = 0, \\ \mathcal{N}(0, \sigma_1^2),\ d_i = 1, \end{cases} \tag{5.21}$$

where $\sigma_0^2 \to 0$ denotes the variance of h_i in the inactive state, and $\sigma_1^2 \gg \sigma_0^2$ denotes the variance of h_i in the active state.

We further assume that the L_h channel taps are independent. The assumption is well justified, especially for wireless fading channels that are rich in scattering [79–81]. The distribution of the channel **h** is

$$\begin{aligned} P(\mathbf{h}) &= \prod P(h_i) = \prod^{||\mathbf{d}||_0} P(h_i|\text{active}) \prod^{L_h - ||\mathbf{d}||_0} P(h_i|\text{inactive}) \qquad (5.22) \\ &= \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{h_i^2}{2\sigma_1^2}\right)\right)^{||\mathbf{d}||_0} \times \left(\frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{h_i^2}{2\sigma_0^2}\right)\right)^{L_h - ||\mathbf{d}||_0} \\ &= \left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right)^{||\mathbf{d}||_0} \left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^{L_h - ||\mathbf{d}||_0} \exp\left(-\frac{\mathbf{h}^T \Gamma_{\mathbf{d}} \mathbf{h}}{2\sigma^2}\right), \end{aligned}$$

where $\Gamma_{\mathbf{d}}$ is a diagonal matrix that depends on the realization of the vector **d**.

$$\Gamma_{\mathbf{d}} = \text{Diag}\{\{\alpha_i\}\},\ \ \alpha_i = \begin{cases} \gamma_0 = \frac{\sigma_0^2}{\sigma^2},\ \text{for } d_i = 0 \\ \gamma_1 = \frac{\sigma_1^2}{\sigma^2},\ \text{for } d_i = 1 \end{cases},\ \ i = 0, \ldots, L_h - 1. \tag{5.23}$$

Under this model, the channel is treated as a stochastic random vector at the receiver. The matrix $\Gamma_{\mathbf{d}}$ describes the channel sparsity in the expression of the channel distribution. The element α_i on its diagonal shows whether a corresponding channel tap is active or inactive: $\alpha_i = \gamma_1$ indicates that the channel tap h_i is active, and $\alpha_i = \gamma_0$ indicates that h_i is inactive. We assume that the receiver only has knowledge of the variance of active and inactive taps, σ_0^2 and σ_1^2. In order to implement the SA, a new form for the path metric must be derived.

5.3.2 Stack Algorithm for Unknown ISI Channels

In Section 4.1.2, we derived the path metric of the SA when the channel is known at the receiver. We need to incorporate knowledge of the error control coding and distribution of the channel modeled in (5.22) into the derivation of a new path metric. The probability of a length-n sequence $\mathbf{b}_1^n$ given the received sequence $\mathbf{y}_1^{rN}$ and the code known at current iteration k, $C^{(k)}$, is expressed as

$$\begin{aligned} P(\mathbf{b}_1^n|\mathbf{y}_1^{rN}, C^{(k)}) &= \int_{\mathbf{h}} P(\mathbf{b}_1^n, \mathbf{h}|\mathbf{y}_1^{rN}, C^{(k)}) \, \mathrm{d}\mathbf{h} \qquad (5.24) \\ &= \frac{P(\mathbf{b}_1^n)}{P(\mathbf{y}_1^{rN}|C^{(k)})} \int_{\mathbf{h}} P(\mathbf{y}_1^{rN}|\mathbf{b}_1^n, \mathbf{h}, C^{(k)}) P(\mathbf{h}) \, \mathrm{d}\mathbf{h}. \end{aligned}$$

We can eliminate the term $P(\mathbf{y}_1^{rN}|C^{(k)})$, since it is equal for all paths, and the path metric can be written as

$$m(\mathbf{b}_1^n) = P(\mathbf{b}_1^n) \int_{\mathbf{h}} P(\mathbf{y}_1^{rN}|\mathbf{b}_1^n, \mathbf{h}, C^{(k)}) P(\mathbf{h}) \, \mathrm{d}\mathbf{h}. \qquad (5.25)$$

Similar work has been done for deriving the path metric for blind equalization of general unknown ISI channels in [55]. To extend the algorithm to sparse channels, we take the sparsity of the channel into consideration and find a closed-form result of the integral in (5.25) using the channel model given in (5.22). To compute the term $P(\mathbf{y}_1^{rN}|\mathbf{b}_1^n, \mathbf{h}, C^{(k)})$, we use the same technique used to derive the metric for known channels with the assumptions that $\mathbf{y}_1^{rn}$ is independent of $\mathbf{y}_{rn+1}^{rN}$, and $\mathbf{y}_{rn+1}^{rN}$ is independent of $\mathbf{b}_1^n$,

$$\begin{aligned} P(\mathbf{y}_1^{rN}|\mathbf{b}_1^n, \mathbf{h}, C^{(k)}) &\approx P(\mathbf{y}_1^{rn}|\mathbf{b}_1^n, \mathbf{h}, C^{(k)}) P(\mathbf{y}_{rn+1}^{rN}|\mathbf{h}) \qquad (5.26) \\ &\approx P(\mathbf{y}_1^{rn}|\mathbf{b}_1^n, \mathbf{h}, C^{(k)}) \prod_{i=rn+1}^{rN} P(y_i). \end{aligned}$$

Substituting (5.26) into (5.25) yields the following path metric expression:

$$\begin{aligned} m(\mathbf{b}_1^n) &= \left(\frac{1}{2\sqrt{2\pi\sigma^2}}\right)^{rn} \left(\frac{1}{\sqrt{2\pi(\sigma^2+1)}}\right)^{r(N-n)} \prod_{i=rn+1}^{rN} \exp\left(-\frac{y_i^2}{2(\sigma^2+1)}\right) \qquad (5.27) \\ &\quad \times \int_{\mathbf{h}} \prod_{i=1}^{rn} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{h}^T \mathbf{x}_{i-L+1}^i)^2\right) P(\mathbf{h})\, \mathrm{d}\mathbf{h} \\ &= A(y_i) \int_{\mathbf{h}} \exp\left(-\frac{1}{2\sigma^2}(R_{yy}^n[0] - 2\mathbf{h}^T \mathbf{r}_{yx}^{rn} + \mathbf{h}^T R_{xx}^{rn} \mathbf{h})\right) P(\mathbf{h})\, \mathrm{d}\mathbf{h}, \end{aligned}$$

where

$$A(y_i) = \left(\frac{1}{2\sqrt{2\pi\sigma^2}}\right)^{rn} \left(\frac{1}{\sqrt{2\pi(\sigma^2+1)}}\right)^{r(N-n)} \prod_{i=rn+1}^{rN} \exp\left(-\frac{y_i^2}{2(\sigma^2+1)}\right), \qquad (5.28)$$

$$R_{yy}^{rn}[0] = \sum_{i=1}^{rn} y_i^2, \qquad (5.29)$$

$$\mathbf{r}_{yx}^{rn} = \sum_{i=1}^{rn} y_i \mathbf{x}_{i-L_h+1}^i, \qquad (5.30)$$

and

$$R_{xx}^{rn} = \sum_{i=1}^{rn} (\mathbf{x}_{i-L_h+1}^i)(\mathbf{x}_{i-L_h+1}^i)^T. \qquad (5.31)$$

Substituting (5.22) into (5.27), we have

$$\begin{aligned} m(\mathbf{b}_1^n) &= A(y_i) \int_{\mathbf{h}} \exp\left(-\frac{1}{2\sigma^2}(R_{yy}^{rn}[0] - 2\mathbf{h}^T \mathbf{r}_{yx}^{rn} + \mathbf{h}^T R_{xx}^{rn} \mathbf{h})\right) \\ &\quad \times \sum_{\mathbf{d}} \prod_{i=0}^{L_h-1} P(h_i)\, d\mathbf{h} \\ &= A(y_i) \sum_{\mathbf{d}} \int_{\mathbf{h}} \exp\left(-\frac{1}{2\sigma^2}(R_{yy}^{rn}[0] - 2\mathbf{h}^T \mathbf{r}_{yx}^{rn} + \mathbf{h}^T R_{xx}^{rn} \mathbf{h})\right) \\ &\quad \times \prod^{||\mathbf{d}||_0} \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{h_i^2}{2\sigma_1^2}\right) \prod^{L_h-||\mathbf{d}||_0} \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{h_i^2}{2\sigma_0^2}\right) d\mathbf{h}. \end{aligned} \tag{5.32}$$

A quadratic exponential form can be obtained as follows:

$$\begin{aligned} m(\mathbf{b}_1^n) &= A(y_i) \sum_{\mathbf{d}} \prod^{||\mathbf{d}||_0} \frac{1}{\sqrt{2\pi\sigma_1^2}} \prod^{L_h-||\mathbf{d}||_0} \frac{1}{\sqrt{2\pi\sigma_0^2}} \\ &\quad \times \int_{\mathbf{h}} \exp\left(-\frac{1}{2\sigma^2}(R_{yy}^{n}[0] - 2\mathbf{h}^T \mathbf{r}_{yx}^{n} + \mathbf{h}^T (R_{xx}^{n} + \Gamma_{\mathbf{d}}) \mathbf{h}\right) d\mathbf{h}. \end{aligned} \tag{5.33}$$

The quadratic exponential term is integrable, resulting in the following expression:

$$\begin{aligned} m(\mathbf{b}_1^n) &= A(y_i) \sum_{\mathbf{d}} \prod^{||\mathbf{d}||_0} \frac{1}{\sqrt{2\pi\sigma_1^2}} \prod^{L_h-||\mathbf{d}||_0} \frac{1}{\sqrt{2\pi\sigma_0^2}} \\ &\quad \times \left|\frac{R_{xx}^{rn}}{\sigma^2} + \frac{\Gamma_{\mathbf{d}}}{\sigma_0^2}\right|^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(R_{yy}^{rn}[0] - \frac{1}{\sigma^2}(\mathbf{r}_{yx}^{rn})^T \left(\frac{R_{xx}^{rn}}{\sigma^2} + \frac{\Gamma_{\mathbf{d}}}{\sigma_0^2}\right)^{-1} \mathbf{r}_{yx}^{rn})\right). \end{aligned} \tag{5.34}$$

Using the sparse channel model we derived, we are able to incorporate the prior information of channel sparsity into the path metric of the SA. However, the path metric in (5.34) is a summation of quadratic exponential terms for all possible realizations of the vector $\mathbf{d}$. Without simplification, we need to evaluate and sum 2^{L_h} exponential terms, which is computationally impractical for channels with long delay spread. For a given sparse channel, the sparsity vector $\mathbf{d}$ is just one out of the 2^{L_h} possible binary vectors. Therefore, if the channel sparsity can be determined, the path metric can be computed only with a unique vector $\mathbf{d}$. In order to achieve this, we propose an efficient method to detect the channel sparsity based on the path metric derived in (5.34).

5.3.3 Channel Sparsity Detection Based on the Path Metric

In order to obtain the unique binary vector $\mathbf{d}$ corresponding to the sparse channel, we propose a greedy algorithm employing a best-first strategy to find the active channel tap indices. We first restructure the path metric expression for ease of representation:

$$\begin{aligned} m(\mathbf{b}_1^n) &= A(y_i)\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right)^{||\mathbf{d}||_0}\left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^{L_h-||\mathbf{d}||_0}\sigma^{rn} \qquad (5.35)\\ &\quad\times\left|R_{xx}^{rn}+\Gamma_{\mathbf{d}}\right|^{-1/2}\exp\left(-\frac{1}{2\sigma^2}R_{yy}^{rn}[0]\right)\\ &\quad\times\exp\left(\frac{1}{2\sigma^2}(\mathbf{r}_{yx}^{rn})^T\left(R_{xx}^{rn}+\Gamma_{\mathbf{d}}\right)^{-1}\mathbf{r}_{yx}^{rn}\right).\\ &= B(y_i)\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right)^{||\mathbf{d}||_0}\left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^{L_h-||\mathbf{d}||_0}\\ &\quad\times\left|R_{xx}^{rn}+\Gamma_{\mathbf{d}}\right|^{-1/2}\exp\left(\frac{1}{2\sigma^2}(\mathbf{r}_{yx}^{rn})^T\left(R_{xx}^{rn}+\Gamma_{\mathbf{d}}\right)^{-1}\mathbf{r}_{yx}^{rn}\right), \end{aligned}$$

where

$$B(y_i)=\sigma^{rn}A(y_i)\exp\left(-\frac{1}{2\sigma^2}R_{yy}^{rn}[0]\right), \qquad (5.36)$$

which is dependent only on the received signal y_i. To better illustrate the proposed algorithm, we write a step-by-step procedure as follows:

- Initialization. Let us consider a certain path of the tree which represents a length-n

sequence $\mathbf{b}_1^n$. The algorithm is initialized with the assumption that all the channel taps are inactive, e.g. $\mathbf{d}^{(0)} = [0, 0, \ldots, 0]$. The corresponding path metric can be computed as

$$\begin{aligned} m(\mathbf{b}_1^n)_{\mathbf{d}^{(0)}} &= B(y_i)\left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^{L_h} \\ &\quad \times |R_{xx}^{rn} + \Gamma_{\mathbf{d}^{(0)}}|^{-1/2} \exp\left(\frac{1}{2\sigma^2}(\mathbf{r}_{yx}^{rn})^T \left(R_{xx}^{rn} + \Gamma_{\mathbf{d}^{(0)}}\right)^{-1} \mathbf{r}_{yx}^{rn}\right) \\ &= B(y_i)\left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^{L_h} \left|\mathbf{R}(\mathbf{d}^{(0)})\right|^{-1/2} \exp\left(\frac{1}{2\sigma^2}(\mathbf{r}_{yx}^{rn})^T \left(\mathbf{R}(\mathbf{d}^{(0)})\right)^{-1} \mathbf{r}_{yx}^{rn}\right), \end{aligned} \tag{5.37}$$

where

$$\mathbf{R}(\mathbf{d}^{(0)}) = R_{xx}^{rn} + \Gamma_{\mathbf{d}^{(0)}}, \tag{5.38}$$

and

$$\Gamma_{\mathbf{d}^{(0)}} = \mathrm{Diag}\{\gamma_0\}. \tag{5.39}$$

The vector $\mathbf{d}^{(0)}$ is assigned to $\mathbf{d}_*^{(0)}$, where the vector $\mathbf{d}_*^{(p)}$ is used to store the optimal vector obtained at the p-th iteration. A vector $\mathbf{i}^*$ is defined to store optimal indices of active channel taps at each iteration. We initialize that $p = 0$ and $\mathbf{i}^* = \{\}$.

- Step 1. Changing a single 0 element of the binary vector $\mathbf{d}_*^{(p)}$ to 1 generates a set of $L_h - p$ possible binary vectors, $\mathbf{d}_j^{(p+1)}$, where $j \notin \mathbf{i}^* \in \{1, \ldots, L_h\}$ denotes that $\mathbf{d}_j^{(p+1)}$ is identical to $\mathbf{d}_*^{(p)}$ except for the j-th element.

- Step 2. The path metric $m(\mathbf{b}_1^n)_{\mathbf{d}_j^{(p+1)}}$ for each of the possible weight-$(p+1)$ vectors is computed, and the vector that corresponds to the largest metric is assigned to the vector $\mathbf{d}_*^{(p+1)}$, i.e., $\mathbf{d}_*^{(p+1)} = \mathbf{d}_{j^*}^{(p+1)}$, where

$$j^* = \underset{j}{\operatorname{argmax}}\, m(\mathbf{b}_1^n)_{\mathbf{d}_j^{(p+1)}},\ j \notin \mathbf{i}^* \in \{1, \ldots, L_h\}. \tag{5.40}$$

- Step 3. We update the optimal index vector $\mathbf{i}^*$ by including j^*, and increment p by 1:

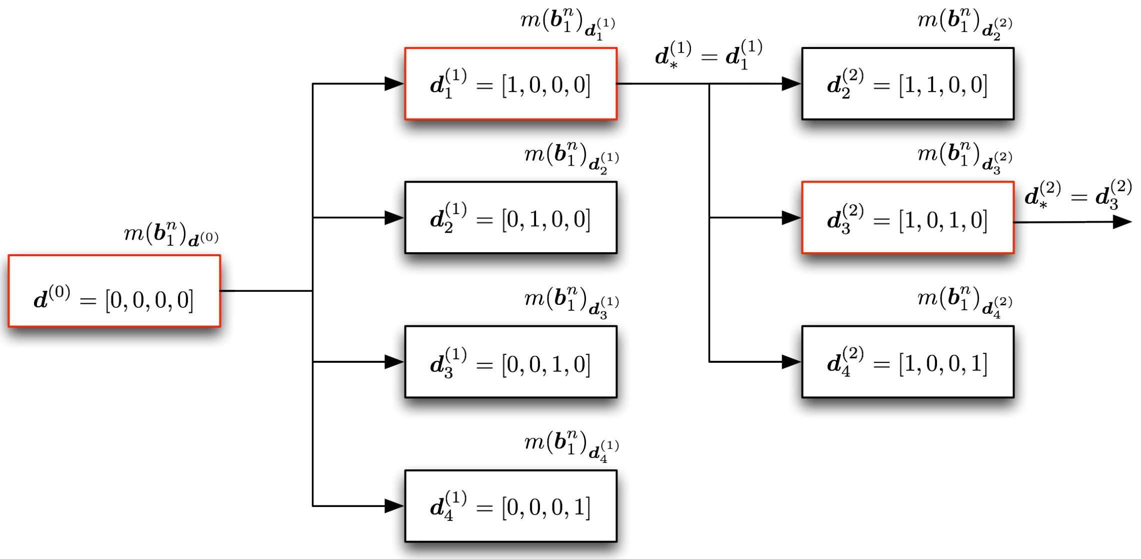
$$\mathbf{i}^* = \mathbf{i}^* \cup j^*, \qquad p = p + 1. \tag{5.41}$$

- Step 4. If all the active channel tap indices are located or a stop criterion is satisfied (Due to the involvement of metric computations, the stop criterion will be explained in details later), the iteration is terminated; otherwise, we go back to Step 1.

The final vector $\mathbf{d}_*^{(\hat{L}_a)}$ is the estimation of channel sparsity, where $\hat{L}_a$ denotes the estimate of the active channel length. $\hat{L}_a = L_a$ if L_a is known at the receiver. The corresponding metric $m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(\hat{L}_a)}}$ will be the one of this path that can be used in SA to perform the tree search. In the case that L_a is known, the total number of the metric computations is $1 + (L_a + 1)(L_h - \frac{L_a}{2})$, which is reduced dramatically compared to the original number 2^{L_h} when computing (5.34). We will show that the number of metric computations can be further reduced in the next section. An example of this process is shown in Fig. 5.4 for a length-4 channel with 2 active taps, $\mathbf{h} = [h_0, 0, h_1, 0]$. The red boxes show the optimal sparsity vectors obtained at each iteration. Notice that the FGA process performs very similarly to the SA, both relying on a tree representation. In the proposed FGA, a path of the tree represents a possible binary vector $\mathbf{d}$, and a similar best-first strategy is used to select a path based on the path metric of the SA.

5.3.4 Efficient Metric Computation

As we can see in the proposed FGA, at each iteration a new path metric is computed which involves performing the inverse of a matrix and computing its determinant. The total



$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}^{(0)}}$
$\boldsymbol{d}^{(0)} = [0,0,0,0]$
$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_1^{(1)}}$
$\boldsymbol{d}_1^{(1)} = [1,0,0,0]$
$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_2^{(1)}}$
$\boldsymbol{d}_2^{(1)} = [0,1,0,0]$
$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_3^{(1)}}$
$\boldsymbol{d}_3^{(1)} = [0,0,1,0]$
$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_4^{(1)}}$
$\boldsymbol{d}_4^{(1)} = [0,0,0,1]$
$\boldsymbol{d}_*^{(1)} = \boldsymbol{d}_1^{(1)}$
$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_2^{(2)}}$
$\boldsymbol{d}_2^{(2)} = [1,1,0,0]$
$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_3^{(2)}}$
$\boldsymbol{d}_3^{(2)} = [1,0,1,0]$
$m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_4^{(2)}}$
$\boldsymbol{d}_4^{(2)} = [1,0,0,1]$
$\boldsymbol{d}_*^{(2)} = \boldsymbol{d}_3^{(2)}$

Figure 5.4: An example of channel sparsity estimation using the FGA for a length-4 channel with 2 active taps, $\mathbf{h} = [h_0, 0, h_1, 0]$. The red boxes show the optimal sparsity vectors obtained at each iteration.

$1 + (L_a + 1)(L_h - \frac{L_a}{2})$ metric computations are still a heavy burden for the detector. By exploiting the properties of matrices, we can further reduce the computational complexity and accelerate the metric update.

In order to develop a general expression for efficient metric computation, let us consider the optimal binary vector obtained at the $(p-1)$-th iteration, $\mathbf{d}_*^{(p-1)}$. The matrix $\mathbf{R}(\mathbf{d}_*^{(p-1)})$ in the path metric is computed as

$$\mathbf{R}(\mathbf{d}_*^{(p-1)}) = R_{xx}^{rn} + \Gamma_{\mathbf{d}_*^{(p-1)}}. \tag{5.42}$$

At the p-th iteration, a set of $L_h - p$ vectors $\mathbf{d}_i^{(p)}$ is generated by turning just one 0 element of vector $\mathbf{d}_*^{(p-1)}$ to 1. The vector $\mathbf{d}_i^{(p)}$ is identical to $\mathbf{d}_*^{(p-1)}$ except that the i-th element in

$\mathbf{d}_i^{(p)}$ is 1. The matrix $\mathbf{R}(\mathbf{d}_i^{(p)})$ is

$$\mathbf{R}(\mathbf{d}_i^{(p)}) = R_{xx}^{rn} + \Gamma_{\mathbf{d}_i^{(p)}}. \tag{5.43}$$

It is obvious that the matrix $\mathbf{R}(\mathbf{d}_i^{(p)})$ has only one element at position (i, i) on the diagonal which is different from the matrix $\mathbf{R}(\mathbf{d}_*^{(p-1)})$. Thus, by using the properties of matrix inversion, the inverse and determinant of $\mathbf{R}(\mathbf{d}_i^{(p)})$ can be easily obtained from the inverse and determinant of $\mathbf{R}(\mathbf{d}_*^{(p-1)})$. We assume that we already have the inverse and determinant of $\mathbf{R}(\mathbf{d}_*^{(p-1)})$, denoted by $V = \mathbf{R}^{-1}(\mathbf{d}_*^{(p-1)})$ and $U = \left|\mathbf{R}(\mathbf{d}_*^{(p-1)})\right|$. Thus, $\mathbf{R}(\mathbf{d}_i^{(p)})$ can be expressed as

$$\mathbf{R}(\mathbf{d}_i^{(p)}) = \mathbf{R}(\mathbf{d}_*^{(p-1)}) + \beta\mathbf{u}\mathbf{u}^T, \tag{5.44}$$

where $\beta = \gamma_1 - \gamma_0$ and $\mathbf{u} = [0, \ldots, 1, \ldots, 0]^T$. The index of the 1 element in vector $\mathbf{u}$ is i. Applying the Sherman-Morrison formula [82], the inverse of $\mathbf{R}(\mathbf{d}_i^{(p)})$ can be expressed as

$$\begin{aligned} \mathbf{R}^{-1}(\mathbf{d}_i^{(p)}) &= (\mathbf{R}(\mathbf{d}_*^{(p-1)}) + \beta\mathbf{u}\mathbf{u}^T)^{-1} \\ &= \mathbf{R}^{-1}(\mathbf{d}_*^{(p-1)}) - \frac{\beta\mathbf{R}^{-1}(\mathbf{d}^{(p-1)})\mathbf{u}\mathbf{u}^T\mathbf{R}^{-1}(\mathbf{d}_*^{(p-1)})}{1 + \beta\mathbf{u}^T\mathbf{R}^{-1}(\mathbf{d}_*^{(p-1)})\mathbf{u}} \\ &= V - \frac{\beta V\mathbf{u}\mathbf{u}^T V}{1 + \beta\mathbf{u}^T V\mathbf{u}} \\ &= V - \frac{\beta}{1 + \beta V(j, j)} V(:, j)V(j, :). \end{aligned} \tag{5.45}$$

The determinant of $\mathbf{R}(\mathbf{d}_i^{(p)})$ can be computed as

$$\begin{aligned}\left|\mathbf{R}(\mathbf{d}_i^{(p)})\right| &= \left|\mathbf{R}(\mathbf{d}_*^{(p-1)}) + \beta\mathbf{u}\mathbf{u}^T\right| \qquad (5.46)\\ &= (1+\beta\mathbf{u}^T\mathbf{R}^{-1}(\mathbf{d}_*^{(p-1)})\mathbf{u})\left|\mathbf{R}(\mathbf{d}_*^{(p-1)})\right|\\ &= (1+\beta V(j,j))U.\end{aligned}$$

Combining equations (5.45) and (5.46), the path metric for vector $\mathbf{d}_i^{(p)}$ can be computed as

$$\begin{aligned}m(\mathbf{b}_1^n)_{\mathbf{d}_i^{(p)}} &= B(y_i)\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right)^{||\mathbf{d}_i^{(p)}||_0}\left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^{L_h-||\mathbf{d}_i^{(p)}||_0} \qquad (5.47)\\ &\quad\times[(1+\beta V(j,j))U]^{-1/2}\exp\left(\frac{1}{2\sigma^2}(\mathbf{r}_{yx}^{rn})^T\left(V-\frac{\beta}{1+\beta V(j,j)}V(:,j)V(j,:)\right)\mathbf{r}_{yx}^{rn}\right)\\ &= m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(p-1)}}\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right)\left(\sqrt{2\pi\sigma_0^2}\right)(1+\beta V(j,j))^{-1/2}\\ &\quad\times\exp\left(-\frac{\beta}{2(1+\beta V(j,j))\sigma^2}(\mathbf{r}_{yx}^{rn})^T V(:,j)V(j,:)\mathbf{r}_{yx}^{rn}\right)\\ &= m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(p-1)}}\frac{\sigma_0}{\sigma_1\sqrt{\theta}}\exp\left(-\frac{\beta}{2\theta\sigma^2}(\mathbf{r}_{yx}^{rn})^T V(:,j)V(j,:)\mathbf{r}_{yx}^{rn}\right),\end{aligned}$$

where $\theta = 1+\beta V(j,j)$. From the derivation above, we can see that the path metric for vector $\mathbf{d}_i^{(p)}$ can be computed via a simple update from the metric for $\mathbf{d}_*^{(p-1)}$. The computational cost for the metric update is $\frac{\sigma_0}{\sigma_1\sqrt{\theta}}\exp\left(-\frac{\beta}{2\theta\sigma^2}(\mathbf{r}_{yx}^{rn})^T V(:,j)V(j,:)\mathbf{r}_{yx}^{rn}\right)$, which is dominated by the vector multiplications. The computations of the inverse and determinant of matrices are no longer needed at every iteration, but performed only once to obtain $R^{-1}(\mathbf{d}^{(0)})$ and $\left|R(\mathbf{d}^{(0)})\right|$ when the initial metric $m(\mathbf{b}_1^n)_{\mathbf{d}^{(0)}}$ is computed.

5.3.5 Stopping Criterion

The proposed FGA described in Section 5.3.3 relies on searching for active channel indices that terminates when all the active indices have been identified. In practical communication systems, the number of active taps, L_a, is usually an unknown parameter. A stopping criterion must be implemented to terminate the iterative sparsity detection process without knowledge of L_a. In order to achieve this, let us compare the path metrics we obtained at the L_a-th and (L_a+1)-th iterations, and find out the stopping criterion from the difference between them.

At the L_a-th iteration, the optimal sparse vector with L_a non-zero elements is $\mathbf{d}_*^{(L_a)}$, and the corresponding path metric can be written as

$$\begin{aligned} m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(L_a)}} &= A(y_i)\sigma^{rn}(\frac{1}{\sqrt{2\pi\sigma_1^2}})^{L_a}(\frac{1}{\sqrt{2\pi\sigma_0^2}})^{L_h-L_a}\left|R_{xx}^{rn}+\Gamma_{\mathbf{d}_*^{(L_a)}}\right|^{-1/2} \\ &\quad\times\exp\left(-\frac{1}{2\sigma^2}(R_{yy}^{rn}[0]-(\mathbf{r}_{yx}^{rn})^T\left(R_{xx}^{rn}+\Gamma_{\mathbf{d}_*^{(L_a)}}\right)^{-1}\mathbf{r}_{yx}^{rn})\right). \end{aligned} \tag{5.48}$$

In the metric, the term $\left(R_{xx}^{rn}+\Gamma_{\mathbf{d}_*^{(L_a)}}\right)^{-1}\mathbf{r}_{yx}^{rn}$ in the exponential function takes a form that is very similar to the LS estimation of the channel, $\hat{\mathbf{h}}$. Thus, $R_{yy}^{rn}[0]-(\mathbf{r}_{yx}^{rn})^T\left(R_{xx}^{rn}+\Gamma_{\mathbf{d}_*^{(L_a)}}\right)^{-1}\mathbf{r}_{yx}^{rn}$ measures the error $\mathbf{e}$ between the received signal $\mathbf{y}_1^{rn}$ and the noise-free output predicted by the channel estimate $\hat{\mathbf{h}}$ and assumed bit sequence $\mathbf{x}_1^{rn}$.

At the (L_a+1)-th iteration, the metric $m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(L_a+1)}}$ associated with the optimal sparse vector $\mathbf{d}_*^{(L_a+1)}$ is updated from the metric $m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(L_a)}}$ according to

$$\begin{aligned} m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(L_a+1)}} &= m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(L_a)}}\frac{\sigma_0}{\sigma_1\sqrt{\theta}}\exp\left(-\frac{\beta}{2\theta\sigma^2}(\mathbf{r}_{yx}^{rn})^TV(:,j^*)V(j^*,:)\mathbf{r}_{yx}^{rn}\right) \\ &= m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(L_a)}}\frac{\sigma_0}{\sigma_1\sqrt{\theta}}\exp\left(\frac{\beta}{2\theta\sigma^2}\Delta_{L_a+1}\right), \end{aligned} \tag{5.49}$$

where j^* denotes the index of the L_a-th estimated active channel tap. From (5.49), we observe that Δ_{L_a+1} denotes the contribution of the (L_a+1)-th estimated active tap to the error $\mathbf{e}$. Assuming that the L_a active channel taps are correctly located at the L_a-th iteration, no more active taps are contributing to metric computation of $m(\mathbf{b}_1^n)_{\mathbf{d}_*^{(L_a+1)}}$. Therefore, the corresponding Δ_{L_a+1} will depend only on the additive noise. With this property, we can establish a stopping criterion by choosing a sufficiently small constant ϵ. If at the p-th iteration,

$$|\Delta_p| \leq \epsilon, \tag{5.50}$$

the iteration is terminated. The number of active channel taps can be estimated as $\hat{L}_a = p - 1$. The estimated sparse vector is $\mathbf{d}_*^{(p-1)}$. In the case that the algorithm estimates L_a correctly, we observe that one more round of iterations for $p = L_a + 1$ is implemented involving $L_h - L_a$ metric update, which will not add much computational complexity to the algorithm. When the noise level is high, the algorithm is more likely to estimate L_a incorrectly. However, since the channel sparsity is estimated using path metrics that take noise into consideration, sparsity estimation is still improved over that of the conventional MP based methods.

The proposed FGA is summarized in a flowchart shown in Fig. 5.5. Using the FGA, the path metric is computed once the sparse vector is identified and can be used to search the tree in the SA. The combination of the SA with FGA to achieve blind sequential detection is similar to SA-MP/OMP. A flowchart of the SA-FGA algorithm is shown in Fig. 5.6.

5.3.6 Computational Efficiency of Blind Sequential Detection Using the Fast Greedy Algorithm

To illustrate the computational efficiency of the proposed SA-FGA we compare it to SA-MP described in Section 5.2.2. (We consider only SA-MP because it is more efficient than SA-OMP.) Since both methods operate on a possible sequence represented by a path of the tree, we take the complexity of computing the metric for a path with length n as the

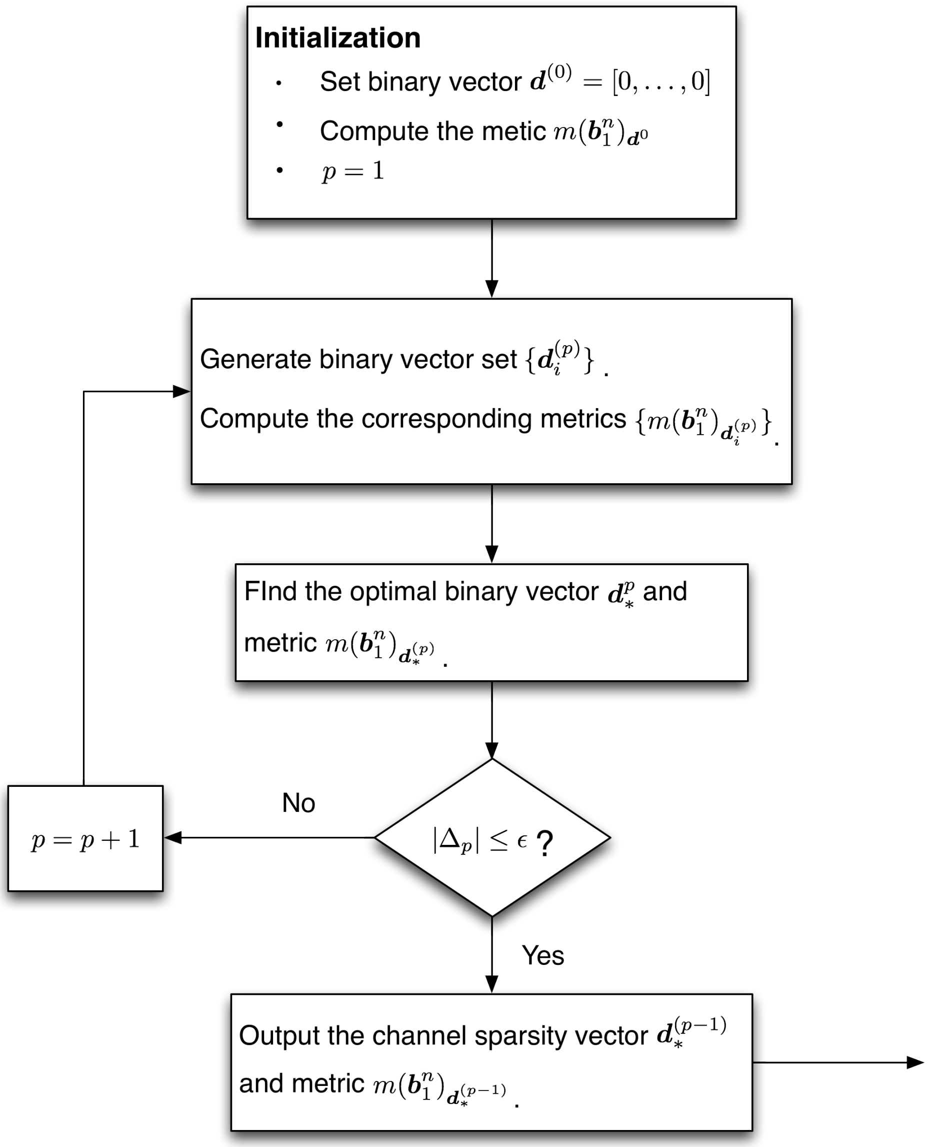

Initialization
Set binary vector $\boldsymbol{d}^{(0)} = [0, \ldots, 0]$
Compute the metic $m(\boldsymbol{b}_1^n)_{\boldsymbol{d}^0}$
$p = 1$
Generate binary vector set $\{\boldsymbol{d}_i^{(p)}\}$.
Compute the corresponding metrics $\{m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_i^{(p)}}\}$.
FInd the optimal binary vector $\boldsymbol{d}_*^p$ and metric $m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_*^{(p)}}$.
$|\Delta_p| \leq \epsilon$?
No
$p = p + 1$
Yes
Output the channel sparsity vector $\boldsymbol{d}_*^{(p-1)}$ and metric $m(\boldsymbol{b}_1^n)_{\boldsymbol{d}_*^{(p-1)}}$.

Figure 5.5: A flow chart representation of FGA to estimate the channel sparsity and update the metric.

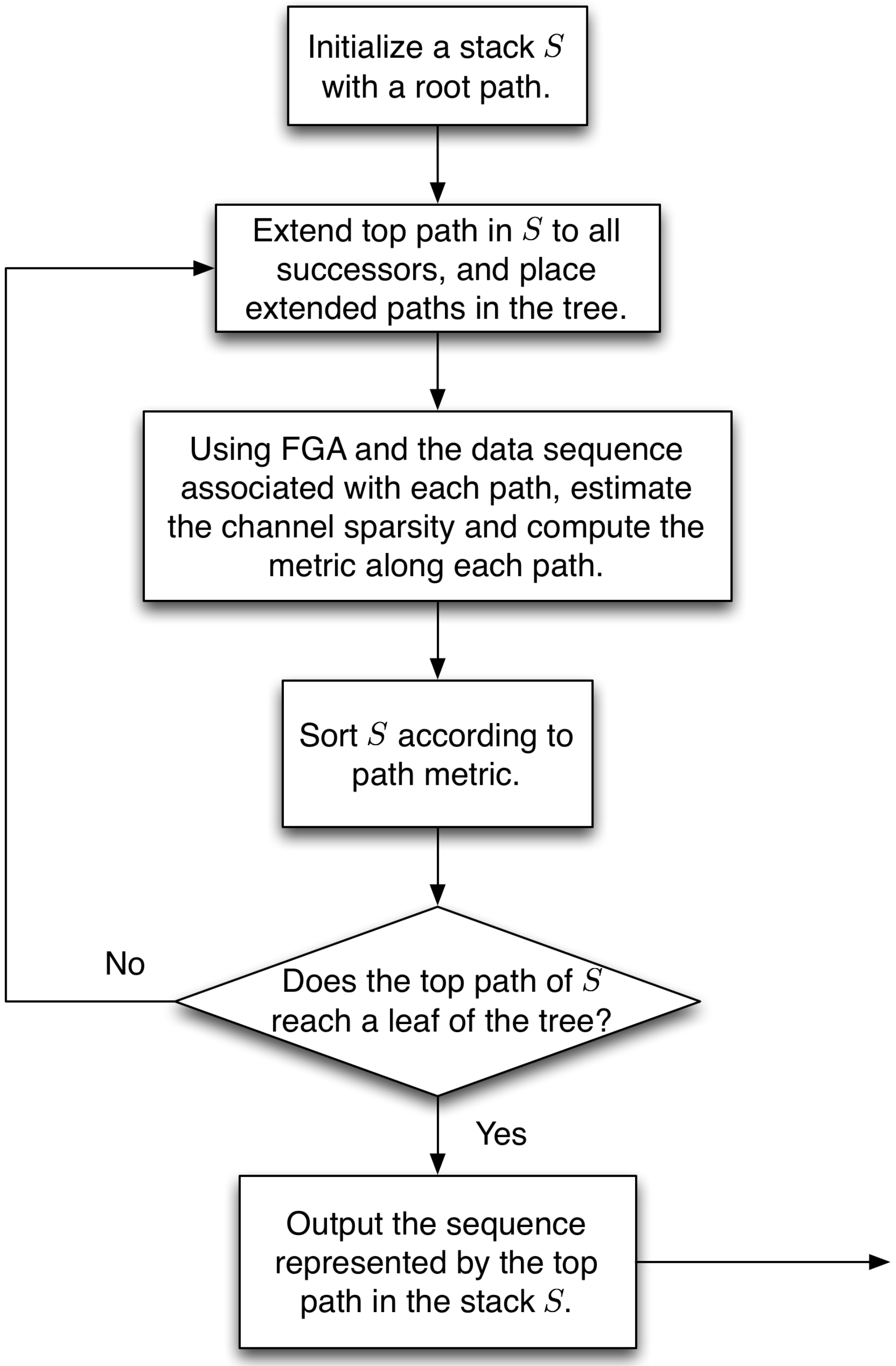

Initialize a stack S with a root path.
Extend top path in S to all successors, and place extended paths in the tree.
Using FGA and the data sequence associated with each path, estimate the channel sparsity and compute the metric along each path.
Sort S according to path metric.
Does the top path of S reach a leaf of the tree?
No
Yes
Output the sequence represented by the top path in the stack S.

Figure 5.6: A flow chart representation of the SA-FGA algorithm.

comparison criterion. In order to compute the metric for a certain path, SA-MP detects the channel sparsity first and computes the metric using the estimated channel sparsity, while SA-FGA computes the initial metric for $\mathbf{d}^{(0)} = [0, \ldots, 0]$ and updates the metric with the sparsity detection. The complexity of the metric computation for SA-MP using the estimated channel sparsity and the initial metric computation for SA-FGA are the same. Therefore, to compare the complexity of SA-MP and SA-FGA, we only need to compare their complexity in the channel sparsity detection. Their complexity depends on estimates of the active channel length, $\hat{L}_a$, which is affected by the channel itself and by the noise level.

For a path with length n, SA-MP constructs an $n \times L_h$ signal matrix $\mathbf{X}^{(n)}$ using (5.16). At the p-th iteration, a length-n residual vector $\mathbf{b}_p$ is projected onto the direction of each column of $\mathbf{X}^{(n)}$, $\mathbf{a}_i$, $i = 1, \ldots, L_h$. The projection is implemented by computing the inner product between two length-n vectors, $\mathbf{b}_p$ and $\mathbf{a}_i$. For this iteration, L_h inner product computations are performed. Furthermore, L_h norm computations of $\mathbf{a}_i$ must be performed to scale the projection shown in (5.5). The norm of $\mathbf{a}_i$ is equivalent to the inner product of $\mathbf{a}_i$ and itself. Therefore, after the estimated active channel length $\hat{L}_a^{(\text{SA-MP})}$ is obtained, $\hat{L}_a^{(\text{SA-MP})}$ iterations must be performed to estimate the sparsity. We need to perform $\hat{L}_a^{(\text{SA-MP})} L_h + L_h = (\hat{L}_a^{(\text{SA-MP})} + 1) L_h$ inner product computations between two length-n vectors.

For a length-n path, at each iteration, the SA-FGA computes the metric update $\frac{\sigma_0}{\sigma_1\sqrt{\theta}} \exp\left(-\frac{\beta}{2\theta\sigma^2}(\mathbf{r}_{yx}^n)^T V(:,j)V(j,:)\mathbf{r}_{yx}^n\right)$. We do not consider the multiplications of the constants $\frac{\sigma_0}{\sigma_1\sqrt{\theta}}$ and $\frac{\beta}{2\theta\sigma^2}$, since they are just a one-time operation. Thus, the complexity for each metric update is the inner product of two length-n vectors, $V(j,:)$ and $\mathbf{r}_{yx}^n$. To obtain the final metric, the metric update is computed $(\hat{L}_a^{(\text{SA-FGA})} + 1)(L_h - \frac{\hat{L}_a^{(\text{SA-FGA})}}{2})$ times, and we need to perform $(\hat{L}_a^{(\text{SA-FGA})} + 1)(L_h - \frac{\hat{L}_a^{(\text{SA-FGA})}}{2})$ inner product computations between two length-n vectors. The complexity comparison between SA-MP and SA-FGA is summarized

Table 5.1: Computational complexity of SA-MP and SA-FGA in terms of the inner product computations for channel sparsity detection at each path of the tree.

Complexity for channel sparsity detection at each path	
	inner product computations
SA-MP	$(\hat{L}_a^{(\text{SA-MP})} + 1)L_h$
SA-FGA	$(\hat{L}_a^{(\text{SA-FGA})} + 1)(L_h - \frac{\hat{L}_a^{(\text{SA-FGA})}}{2})$

in Table 5.1.

In the case that L_a is known at the receiver, both methods will terminate their iterations at the L_a-th loop, $\hat{L}^{(\text{SA-MP})} = \hat{L}^{(\text{SA-FGA})} = L_a$. Comparing the complexity we derived, at each path of the tree, SA-FGA performs $\frac{L_a(L_a+1)}{2}$ fewer inner product computations than SA-MP. While $\frac{L_a(L_a+1)}{2}$ may not be a large number, the computational savings occurs along each path. In the application of the SA, a large number of paths will be extended, especially when SNR is low. The total complexity reduction achieved by the SA-FGA will be significant.

5.3.7 Simulation Results

Simulation Results for Gaussian Channel Models

To illustrate the performance of the proposed SA-FGA, we compare it to SA-MP and SA-OMP. Simulations have been conducted for a length-$L_h = 10$ sparse channel with $L_a = 3$ active taps. Information bits are encoded using a rate $R = \frac{1}{2}$ convolutional code with the generator polynomials

$$\begin{aligned} \mathbf{g}_0 &= [1, 1, 1], \qquad (5.51)\\ \mathbf{g}_1 &= [1, 1, 0]. \end{aligned}$$

The initial state of the register is set to $[0, 0]$. 10000 blocks of 100 data bits are passed over the sparse channel. For each data block, the active channel taps $\mathbf{h}_a = [h_0, h_1, h_2]$ are drawn from a Gaussian distribution with zero mean and variance $\sigma_1^2 = 1$, and the inactive taps are drawn from a Gaussian distribution with zero mean and variance $\sigma_0^2 = 0.01$. The channel energy is normalized to 1. The locations of active and inactive channel taps are randomly generated for each block. For the three methods, the stack size of the SA is set to 10^5 to make erasures rare. We assume that the channel and number of active channel taps L_a are unknown.

A performance comparison of the three methods is shown in Fig. 5.7. SA-OMP has better performance than SA-MP due to the introduction of the orthogonalization process. The proposed SA-FGA outperforms both SA-MP and SA-OMP, especially when SNR is low. The performance gap occurs because MP/OMP does not take the channel noise into consideration and therefore is more likely to incorrectly estimate the sparse channel. SA-FGA, estimating the channel sparsity based on the path metric, is more robust in a noisy environment. As SNR increases, MP/OMP becomes more accurate and performs more similarly to SA-FGA.

In order to illustrate the efficiency of the SA-FGA, we evaluate the complexity of SA-MP and SA-FGA empirically by finding the average number of paths extended for each block. Since we have computed the number of inner product computations for each path in previous section, the total number of inner product computations for each sequential detection method can be obtained. The complexity comparison for various SNR values is shown in Table 5.2.

The results show that SA-FGA extends fewer paths than SA-MP, especially when SNR is low, due to the robustness of the SA-FGA in channel sparsity detection given noisy observations. SA-FGA performs $\frac{L_a(L_a+1)}{2} = \frac{3\times4}{2} = 6$ fewer inner product computations than SA-MP at each path, which results in a significant overall reduction in complexity. As we can see from Table 5.2, for 5 dB, SA-FGA performs about 3×10^5 fewer inner product computations than SA-MP, which is an approximately a 75% reduction in complexity.

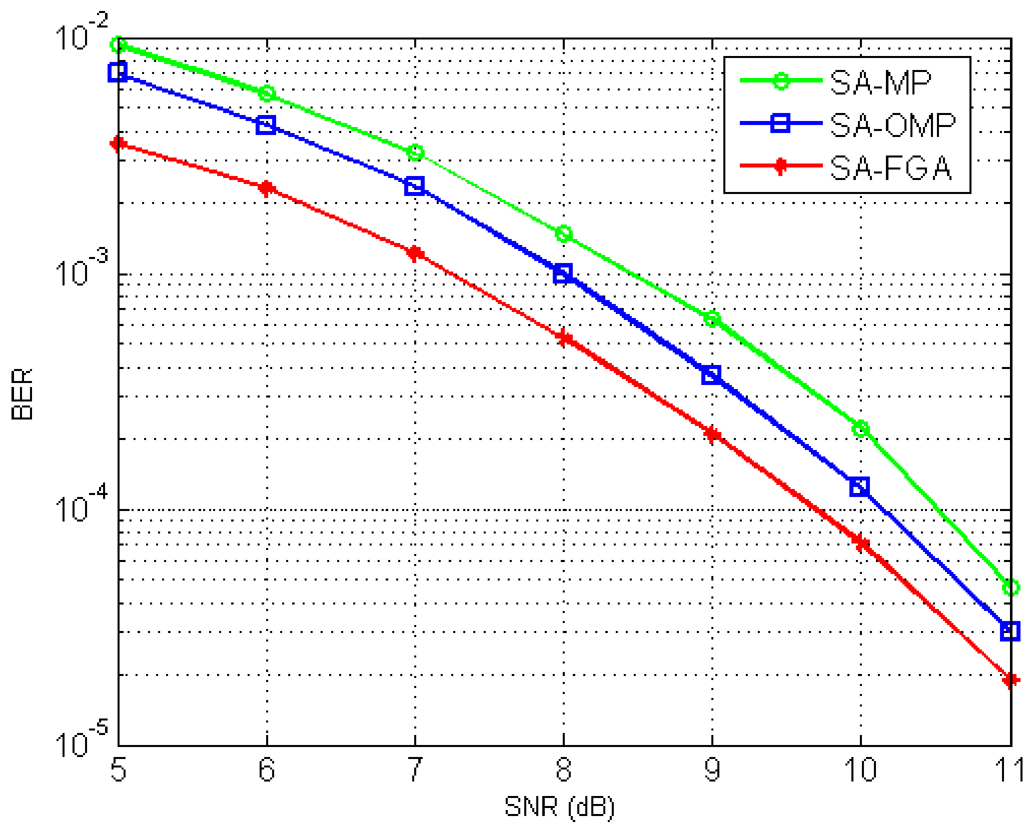


Figure 5.7: Performance comparison between SA-MP, SA-OMP, and SA-FGA using a length-100 sequence for a 10-tap channel with 3 active taps.

Table 5.2: Computational complexity comparison between SA-MP and SA-FGA for length-$L_h = 10$ sparse channels with $L_a = 3$ active taps.

Complexity Comparison				
	SA-MP		**SA-FGA**	
	Paths Extended	**Inner Product Computations**	**Paths Extended**	**Inner Product Computations**
5dB	10942	437680	4096	139264
7dB	5208	208320	3122	106148
9dB	3748	149920	2490	84660
11dB	1350	54000	864	29376

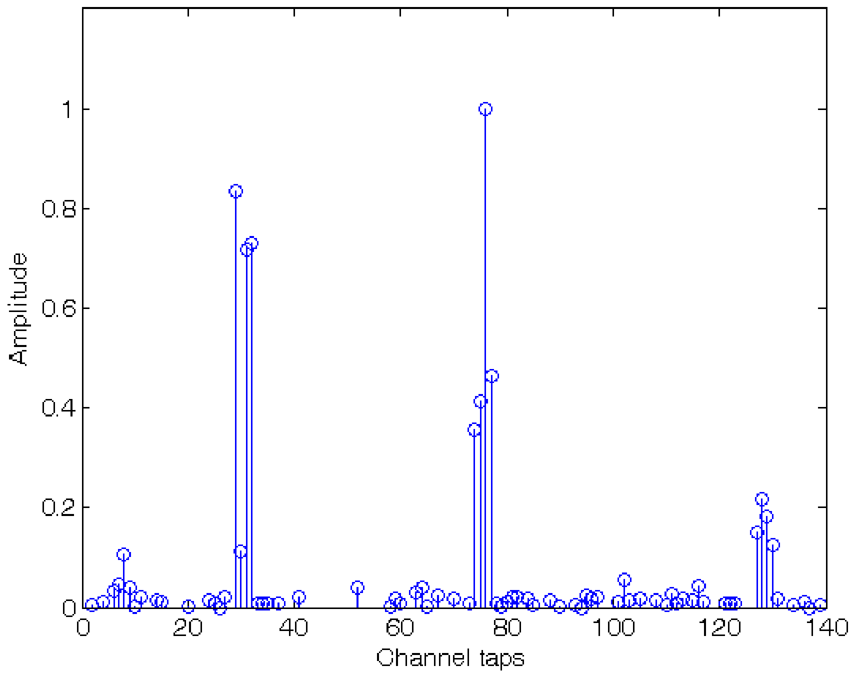
Simulation Results for Realistic Underwater Acoustic Channels

In order to evaluate the performance of the SA-FGA in a more realistic environment, we apply it to sparse channels encountered in underwater acoustic communication.

Numerous experiments have been conducted to estimate underwater acoustic channels [83–86]. The channels estimated are reported to have sparse impulse responses in all experiments. We take the channels estimated in a typical experiment, GLINT'08 [86], as an example and examine the results when the SA-FGA is applied to estimate the data sequence transmitted over those channels. The GLINT'08 experiment took place in the Mediterranean, south of the island Elba, Italy, in July 2008. The channels estimated in the experiment have a delay spread of approximately 20 ms, which is equivalent to about 140 channel taps in the discrete-time system. The active channel taps are distributed in clusters, and four major clusters can be observed in the estimated channels. Since the data of the GLINT'08 experiment is not accessible to the public, we must approximate the channel responses. Based on the description of GLINT'08 and the results shown in [86], we are able to generate similar sparse channels to the underwater acoustic channels estimated in the experiment.

In order to mimic the channels in GLINT'08, we construct a length-140 channel with 15 active taps. Information bits are encoded using a rate $R = \frac{1}{2}$ convolutional code with the same generator polynomials in as given in (5.51). The initial state of the register is set to $[0, 0]$. 5000 blocks of 200 data bits are passed over the sparse channels. For each data block, two different distributions are used to generate the active and inactive channel taps. To test the proposed method for a more realistic channel model, the 15 active taps are drawn from a Rayleigh distribution with parameter $\sigma_1 = 1$, e.g., $\mathbf{h}_a \sim \text{Rayleigh}(\sigma_1)$ (In [87], it is shown that underwater acoustic channels follow Rayleigh distributions in many measurements.) The inactive taps are drawn from a Gaussian distribution with zero mean and variance $\sigma_0^2 = 0.01$. The 15 active taps are grouped into four clusters. The locations of the clusters are randomly generated for each block. The channel energy is normalized to 1. Two examples of the generated channels are shown in Fig. 5.8. We assume that the channel and number of active channel taps L_a are unknown.

A performance comparison among SA-MP, SA-OMP, and SA-FGA is shown in Fig. 5.9. The result coincides with the one we obtained for the mixed Gaussian channel model. SA-FGA has better performance than SA-OMP and SA-MP, especially in low SNR. The performance of SA-MP and SA-OMP converges to that of the SA-FGA as SNR increases, due to the fact that MP based methods estimate the channel sparsity better as SNR increases.



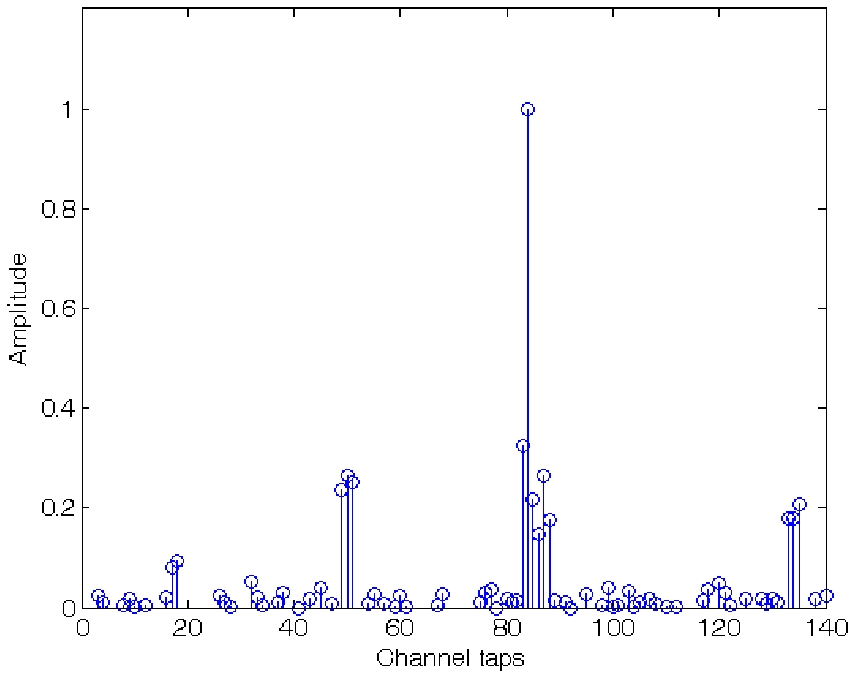


Figure 5.8: Two example channels generated similar to the GLINT'08 experiment; the length-140 channels are sparse with 15 active taps.

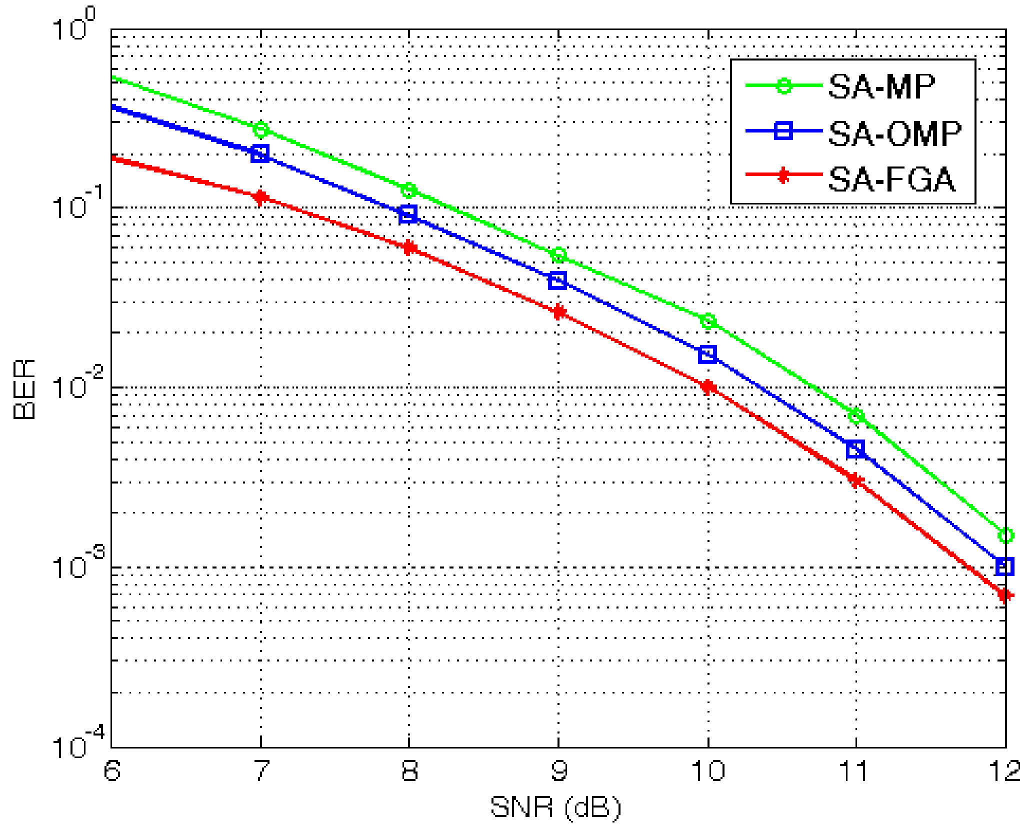


Figure 5.9: Performance comparison between SA-MP, SA-OMP, and SA-FGA for sparse channels generated according to the GLINT'08 experiment.

Chapter 6: Conclusions and Future Research

In this work, we have conducted research on developing computationally efficient equalization methods on both a symbol-by-symbol and sequential detection basis. We considered various communications applications and proposed equalization methods with a balance between performance and complexity for specific applications.

In symbol-by-symbol schemes, we concentrated on the MAE equalizer and sought a more efficient way to determine the MAE equalizer coefficients for application in practical communication systems. The MAE equalizer is very attractive due to its superior performance at high SNR, but the high design complexity limits its application for channels with large delay spreads. We proposed a geometrically-inspired algorithm for computationally efficient design of the MAE equalizer. The proposed method exploits the properties of the constellation structure of a linear channel. Rather than performing computationally intensive quadratic programming, the decision hyperplane of the equalizer is determined directly from the channel coefficients. The equalizer design complexity is reduced to $O(N \log N)$, where $N = 2^{L_h+L_w-2}$ is determined by the length of the channel and equalizer. Empirical complexity analysis reveals the significant computational savings achieved by the geometric approach. The new approach makes the MAE equalizer design practical for channels with longer delay spreads.

We also considered the scenario that no adequate training sequence is available in a time-varying communication system. The training sequence is only transmitted once, which enables us to obtain only an initial estimate of the channel. We proposed a pre-filtering based technique for extending the asymptotically optimal MAE equalizer to applications with time-varying channels. The MAE equalizer is preceded by a linear filter that is designed to track the variation of the noise-free signal constellation generated by the time-varying channel and map it to the constellation of a fixed initial channel. Since the MAE equalizer must be designed only once (for the initial channel), the runtime complexity of the adaptive MAE equalizer is governed by the lower complexity MMSE or LMS operations. Simulation results show that the adaptive MAE equalizer outperforms a standard MMSE equalizer by as much as 4 dB in high SNR with only a one-time additional complexity cost, e.g. the initial design of the MAE detector.

In sequential detection schemes, we focused on developing SA-based sequential detection methods for detecting data transmitted over sparse ISI channels. We first considered the case that the sparse channel can be determined via channel estimation techniques and is known at the receiver. By analyzing the characteristic of the sparse channel, we found that the non-active channel taps lead to repetitive computation of the same path metric when the conventional SA is used. To avoid these redundant computations, we proposed a multiple tree algorithm to replace the single-tree SA. By constructing and searching multiple trees, each of which is used to estimate a subset of the transmitted symbols, the proposed MTA reduces detection complexity by eliminating redundant paths with the same likelihood metrics. Both hard information and soft information are exchanged among subtrees. Their quality and impact on the detector performance are compared. Simulation results show that the MTA can achieve significant complexity reduction relative to competing trellis-based schemes for moderate to high SNR.

Finally, the scenario in which no training sequence is available in the system was considered. In this case, the transmitted sequence must be detected blindly. We presented a Gaussian mixture model to describe sparse ISI channels and developed a novel sequential detection method. A fast greedy algorithm was proposed to blindly detect the channel sparsity. The estimated sparsity was then used to compute the path metrics of the SA, which guide us in the search of a tree. The fast greedy algorithm and SA were combined to achieve blind sequential detection. The simulation results show that the proposed method not only reduces the computational complexity compared to the conventional methods using matching pursuit, but also provides superior performance, especially when SNR is low.

6.1 Future Research

In symbol-by-symbol schemes, based on the promising performance observed thus far, the adaptive MAE equalizer warrants further investigation. When the pre-filter technique is applied, an analysis of the range of perturbation of the time-varying channel can be conducted. When the channel changes dramatically, mapping the resulting constellation back to a fixed structure will become more difficult, and even impossible in scenarios in which the two convex hulls overlap. Hence, the pre-filter based adaptive strategy is likely best used for moderately time-varying channels. We can also explore how an adaptive MAE equalizer can be designed using the proposed geometric method. For a time-varying channel, the decision hyperplane can be determined adaptively by tracking the corresponding channel vectors, since the relationship between channel vectors and decision hyperplane remains the same. A target application for the proposed technique is intersymbol interference mitigation in shallow water acoustic communications, where time-varying reflection of the transmitted signal from the moving ocean surface and other reflectors (e.g. fish) along with platform motion results in rapidly fluctuating multipath arrivals from the transmitter to the receiver. An additional future direction is extending the algorithm from binary signaling to an M-PAM scheme. In this case, more sophisticated geometric relationships need to be derived among the multiple convex hulls.

In sequential detection schemes, the MTA can be further extended. In the current version of the MTA, the hard information and soft information exchanged among subtrees are only used to assist in the computation of the subtree metrics. We can seek a way to feed the hard and soft estimates to the subtrees to refine our estimation akin to the iterative process in turbo equalization. The refined estimation at each state will increase the likelihood that each subtree follows the correct path and hence improve the performance of the detector. For the case that no training sequence is available to estimate the channel, we can consider evaluating the complexity and performance of SA-FGA when the sparse channel is described by models other than the Gaussian model. Furthermore, with the high efficiency of the MTA, we can consider applying the multiple tree structure to the blind sequential detection method using matching pursuit and the proposed fast greedy algorithm. The sparsity detection is performed first at each state of the tree search. One difficulty is that for each path of the tree, the estimated sparsity of the channel might be different. Thus, a strategy must be found to determine to which subtree the hard and soft information is transferred, based on different sparsity estimates.

Appendix A: Appendix

For the $L_w = 2$ scenario, the properties of 2-dimensional channel vectors $\boldsymbol{g}_i$ have been well studied in [38] to blindly identify the communication channel. For the application of the MAE equalizer, we do not want to restrict L_w to 2 but instead wish to consider MAE equalizers of arbitrary length. Hence, we need to geometrically analyze the constellation sets for an arbitrary L_w-dimensional space. Assume that no taps of the linear channel h are equal to 0; this assumption guarantees that no pair of vectors $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{L_h+L_w-2}$ are parallel to each other. The theorems below are proved for the case in which $CH(C_d^{(+)})$ and $CH(C_d^{(-)})$ do not overlap. In the case of overlap, a reduced convex hulls method [41] or a decision feedback equalizer structure [20] can be applied to separate the two convex hulls.

1. Proof of Theorem 1. Consider an L_w-D $CH(C_d^{(+)})$. There is an L_w-D support hyperplane H such that the edge E falls on H and all the vertices of $CH(C_d^{(+)})$ other than the endpoints p_1, p_2 of E are on the right side of H. Thus, we have

$$\boldsymbol{u}^T p_1 + Q = 0, \tag{A.1}$$

$$\boldsymbol{u}^T p_2 + Q = 0, \tag{A.2}$$

$$\boldsymbol{u}^T p_i + Q > 0, \;\; \forall p_i \neq p_1, p_2, p_i \in C_d^{(+)}, \tag{A.3}$$

where $\boldsymbol{u}$ is the normal vector of the hyperplane H, and Q is the offset. In L_w-D space, parallel lines are defined as lines that do not intersect and are located in the same plane [88]. Let $L(g_i)$ denote the line on which the channel vector g_i lies. To prove Theorem 1, we can make a contradictory assumption that all lines of channel vectors $L(g_i)$, $i \in \{0, 1, \ldots, d-1, d+1, \ldots, L_h + L_w - 1\}$, will intersect with the line of edge E, e.g. $L(E)$. This means no $L(g_i)$ will lie on the hyperplane that is parallel to H, i.e. $\boldsymbol{u}^T \boldsymbol{g}_i \neq 0$,

$i \in \{0, 1, \ldots, d-1, d+1, \ldots, L_h + L_w - 1\}$. The proof given in appendix A of [38] shows that there is at least one g_i satisfying $\boldsymbol{u}^T \boldsymbol{g}_i = 0$ for the endpoints $p_1 = \boldsymbol{g}_d + \sum_{i \neq d} \boldsymbol{g}_i x_{k-i}^{(1)}$ and $p_2 = \boldsymbol{g}_d + \sum_{j \neq d} \boldsymbol{g}_j x_{k-j}^{(2)}$, where $x_i^{(1)} = x_j^{(2)}$ for $i \neq j$ and $x_i^{(1)} = -x_j^{(2)}$ for $i = j$. This is a contradiction to the assumption. Therefore, at least one of the channel vectors $\boldsymbol{g}_i$ is on the hyperplane parallel to H. We still need to prove that the edge E and the channel vector $\boldsymbol{g}_i$ are on the same plane. With endpoints p_1 and p_2, we have

$$\begin{aligned} E &= p_2 - p_1 \\ &= \boldsymbol{g}_d + \sum_{i \neq d} \boldsymbol{g}_i x_{k-i}^{(1)} - (\boldsymbol{g}_d + \sum_{j \neq d} \boldsymbol{g}_j x_{k-j}^{(2)}) \\ &= \sum \boldsymbol{g}_i (x_{k-i}^{(1)} - x_{k-j}^{(2)}) = 2\boldsymbol{g}_i, i \neq d. \end{aligned} \tag{A.4}$$

The edge E is an extension of the channel vector $\boldsymbol{g}_i$. The two vectors are always on the same plane. Thus we can conclude that the edge E in L_w-D space is always parallel to one of the channel vectors with length $||E|| = 2||g_i||$, $i \neq d$. $\square$

2. Proof of Theorem 2. The constellation point $p_i \in C_d^{(+)}$ is a binary combination of the channel vectors $\boldsymbol{g}_i, i \neq d$, i.e., $p_i = \boldsymbol{g}_d + \sum_{j \neq d} \boldsymbol{g}_j x_{k-j}^{(i)}$. Since all possible values of $x_k^{(i)}$ are included, the point $\bar{p}_i = \boldsymbol{g}_d - \sum_{j \neq d} \boldsymbol{g}_j x_{k-j}^{(i)}$, the symmetric point of p_i with respect to point $P(\boldsymbol{g}_d)$, also belongs to $C_d^{(+)}$. Thus, the constellation points in $C_d^{(+)}$ are symmetric with respect to the point $P(\boldsymbol{g}_d)$. To prove the symmetry of the convex hull, let us consider a facet of $CH(C_d^{(+)})$ defined by m endpoints $p_i, i = 1, \ldots, m$, located on a L_w-D hyperplane H. Thus, we have

$$\boldsymbol{u}^T p_i + Q = 0, \;\; i \in \{1, \ldots, m\}, p_i \in CH(C_d^{(+)}) \tag{A.5}$$

The points other than the endpoints in $C_d^{(+)}$ are on one side of the hyperplane H,

$$\boldsymbol{u}^T p_i + Q > 0, \ \ \forall i \notin \{1, \dots, m\}, p_i \in CH(C_d^{(+)}) \tag{A.6}$$

Then defining the normal vector $\bar{\boldsymbol{u}} = -\boldsymbol{u}$ and the offset $\bar{Q} = Q + 2\boldsymbol{g}_d$, we can write

$$\bar{\boldsymbol{u}}^T \bar{p}_i + \bar{Q} = 0, \ \ i \in \{1, \dots, m\}, \bar{p}_i \in CH(C_d^{(+)}) \tag{A.7}$$

$$\bar{\boldsymbol{u}}^T \bar{p}_i + \bar{Q} > 0, \ \ \forall i \notin \{1, \dots, m\}, \bar{p}_i \in CH(C_d^{(+)}) \tag{A.8}$$

From the equation above, we can see that the hyperplane on which all $\bar{p}_i = \boldsymbol{g}_d - p_i$ lie is a support hyperplane and is parallel to H. Thus, the convex hull $CH(C_d^{(+)})$ is symmetric with respect to the point $P(\boldsymbol{g}_d)$, and the facets of the convex hull occur in parallel pairs. □

3. Proof of Theorem 3. Denote one of the $(L_w - 1)$-facets of $CH(C_d^{(+)})$ by F, and let F be determined by L_w vertices of the convex hull. These L_w vertices define L_w edges of the facet F. The normal vector $\mathbf{u}$ of F can be obtained by performing the cross product of $L_w - 1$ of the L_w edges [89].

$$\mathbf{u} = \begin{vmatrix} e_1 & e_2 & \dots & e_{L_w-1} \\ E_1^{(1)} & E_2^{(1)} & \dots & E_{L_w-1}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ E_1^{(L_w-1)} & E_2^{(L_w-1)} & \dots & E_{L_w-1}^{(L_w-1)} \end{vmatrix}, \tag{A.9}$$

where e_i is the standard basis vector and $E_j^{(i)}$ denotes the j-th dimension of the i-th edge vector, $i, j = 1, \dots, L_w - 1$. The normal vector $\mathbf{u}$ is perpendicular to the hyperplane F,

$$\mathbf{u} \perp F. \tag{A.10}$$

From Theorem 1, there will be $L_w - 1$ channel vectors parallel to the L_w edges of the facet. Thus, the normal vector $\boldsymbol{u}$ is also perpendicular to the hyperplane T constructed by these $L_w - 1$ of $L_h + L_w - 2$ channel vectors.

$$\mathbf{u} \perp F, \;\; \mathbf{u} \perp T. \tag{A.11}$$

As F and T are perpendicular to a same vector, F and T are parallel to each other,

$$F \parallel T. \tag{A.12}$$

$\square$

4. Proof of Theorem 4. Let us make a contradictory assumption of Theorem 4 that the minimum distance point v_+ is located on an m-facet, where $0 \le m \le L_w - 2$. In the case of separate sub-constellations, for a linear channel and an L_w-tap MAE equalizer, the ith dimension of the constellation $C_d^{(+)}$ and $C_d^{(-)}$ denoted by $r_{k-i}^{(+)}$ and $r_{k-i}^{(-)}$ respectively, will be located on opposite sides of the axis for $m < i < L_w - 1$. In other words, if $r_{k-i}^{(+)} > 0$, then $r_{k-i}^{(-)} < 0$, or if $r_{k-i}^{(+)} < 0$, then $r_{k-i}^{(-)} > 0$. Consider $C_d^{(+)}$. We can write $r_{k-i}^{(+)}$ as binary combinations of channel taps,

$$r_{k-i}^{(+)} = \sum_{j=0}^{L_h} h_j x_{k-j-i} \begin{cases} = \boldsymbol{h}^T \boldsymbol{x}_{k-i} > 0, & \text{if } r_{k-i}^{(-)} < 0 \\ = \boldsymbol{h}^T \boldsymbol{x}_{k-i} < 0, & \text{if } r_{k-i}^{(-)} > 0 \end{cases}. \tag{A.13}$$

Because the constellations include all the combinations of x_k, in the case of the ith and nth dimension of the constellations, $r_{k-i}^{(+)} > 0$ and $r_{k-n}^{(+)} > 0$, $i \ne n$, we can find a contradiction that $r_{k-i}^{(+)} = h_d - \sum_{j \ne d} h_j > 0$ and $r_{k-n}^{(+)} = -h_d + \sum_{j \ne d} h_j > 0$ need to be satisfied at same time when $\boldsymbol{x}_{k-i} = [-1, \ldots, -1, 1, -1, \ldots, -1]$ and $\boldsymbol{x}_{k-n} = [1, \ldots, 1, -1, 1, \ldots, 1]$, where d

is decision delay. The similar contradictions can be found for the case of $r_{k-i}^{(+)} > 0$ and $r_{k-n}^{(+)} < 0$, $r_{k-i}^{(+)} < 0$ and $r_{k-n}^{(+)} > 0$, and $r_{k-i}^{(+)} < 0$ and $r_{k-n}^{(+)} < 0$. The contradictions indicate that the assumption we made at the beginning can not be established. Therefore, v_+ is always located on $(L_w - 1)$-facet of $C_d^{(+)}$. $\square$

Appendix B: Appendix

The structure of an MAE-DFE is specified by the FFF length L_c, FBF length L_b, and decision delay d. L_c determines the dimensionality of the constellation space. L_b reduces the number of constellation points used to compute the decision hyperplane of the MAE-DFE. The constellation points belonging to each convex hull vary with the chosen d. Consequently, these three parameters determine the minimum distance between convex hulls, which measures the performance of the MAE-DFE. In this appendix, we present practical rules to determine maximum effective lengths of the FFF and FBF in the design of the MAE-DFE given the channel characteristics.

It has been shown in [45] that each feedback filter tap eliminates one postcursor ISI term. Given L_c, L_h and d, the maximum effective FBF length is defined as

$$L_b = L_c + L_h - d - 2. \tag{B.1}$$

For $L_b > L_c + L_h - d - 2$, the coefficients of FBF will be 0, implying that no performance enhancement is achieved.

We will show that $L_c = d+1$ is the maximum effective FFF length if the decision delay d is given. To prove this, we assume that $L_c > d+1$. A FBF with length $L_b = L_c + L_h - d - 2$ is used, and the feedback vector $\hat{\mathbf{x}}_{k-d}^{fb} = [x_{k-d-1}, \ldots, x_{k-d-L_b}]$ is assumed to be correct. There are $2^{L_h+L_c-2}$ constellation points in the subconstellation $C_{d,fb}^{(+)}$:

$$\begin{aligned} \mathbf{r}_k^{(i)} &= [r_k^{(i)}, r_{k-1}^{(i)}, \ldots, r_{k-L_c+1}^{(i)}] \\ &= [\mathbf{r}_{k,d+1}^{(i)}, \mathbf{r}_{k,L_c-d+1}^{(i)}], \quad 1 \le i \le 2^{L_h+L_c-2} \end{aligned} \tag{B.2}$$

where $\mathbf{r}_k^{(i)}$ denotes the ith point of $C_{d,fb}^{(+)}$. The vector $\mathbf{r}_{k,d+1}^{(i)} = [r_k^{(i)}, \ldots, r_{k-d}^{(i)}]$ and $\mathbf{r}_{k,L_c-d+1}^{(i)} =$

$[r^{(i)}_{k-d-1}, \ldots, r^{(i)}_{k-L_c-1}]$. For $\mathbf{r}^{(i)}_{k,L_c-d+1}$, each coordinate is given as

$$r^{(i)}_{k-d-l} = \sum_{j=0}^{L_h-1} h_i x_{k-d-l-j}, 1 \leq l \leq L_c - 1. \tag{B.3}$$

Thus, the $\mathbf{r}^{(i)}_{k,L_c-d+1}$ is a deterministic vector given the feedback $\hat{\mathbf{x}}^{fb}_{k-d}$, and it becomes a constant vector for all constellation points $\mathbf{r}^{(i)}_k, 1 \leq i \leq 2^{L_h+L_c-2}$. Let us denote this vector as $\mathbf{r}_{k,L_c-d+1}$. In order to obtain the decision hyperplane of the MAE-DFE, a search for the minimum distance vector $\mathbf{v}_+$ on the convex hull $CH(C^{(+)}_{d,fb})$ is required:

$$\begin{aligned}
\boldsymbol{\lambda}_{\text{opt}} &= \underset{\boldsymbol{\lambda}}{\text{argmin}} \, || \sum_{i=1}^{2^{L_h+L_c-2}} \lambda_i \mathbf{r}^{(i)}_k || \qquad \text{(B.4)} \\
& \quad \text{s.t.} \sum_i \lambda_i = 1, \; 0 \leq \lambda_i \leq 1 \\
&= \underset{\boldsymbol{\lambda}}{\text{argmin}} \Big(|| \sum_{i=1}^{2^{L_h+L_c-2}} \lambda_i \mathbf{r}^{(i)}_{k,d+1} || + || \sum_{i=1}^{2^{L_h+L_c-2}} \lambda_i \mathbf{r}^{(i)}_{k,L_c-d+1} || \Big) \\
& \quad \text{s.t.} \sum_i \lambda_i = 1, \; 0 \leq \lambda_i \leq 1 \\
&= \underset{\boldsymbol{\lambda}}{\text{argmin}} \left(|| \sum_{i=1}^{2^{L_h+L_c-2}} \lambda_i \mathbf{r}^{(i)}_{k,d+1} || + || \mathbf{r}_{k,L_c-d+1} || \right) \\
& \quad \text{s.t.} \sum_i \lambda_i = 1, \; 0 \leq \lambda_i \leq 1 \\
&= \underset{\boldsymbol{\lambda}}{\text{argmin}} \, || \sum_{i=1}^{2^{L_h+L_c-2}} \lambda_i \mathbf{r}^{(i)}_{k,d+1} || \\
& \quad \text{s.t.} \sum_i \lambda_i = 1, \; 0 \leq \lambda_i \leq 1,
\end{aligned}$$

where the vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{2^{L_h+L_c-2}}]$ is the weighting parameter vector used to define points on $CH(C^{(+)}_{d,fb})$. The derivation of (B.4) shows that $\boldsymbol{\lambda}_{\text{opt}}$ depends only on the first $d+1$ dimensions of the constellation points, $\mathbf{r}^{(i)}_{k,d+1}, 1 \leq i \leq 2^{L_h+L_c-2}$. Therefore, $\boldsymbol{\lambda}_{\text{opt}}$ for an MAE-DFE with a length-$L_c = d+1$ FFF is same as for an MAE-DFE with $L_c > d+1$. Given the decision delay d, it is sufficient to employ a FFF with length $d+1$ in the design of the MAE-DFE.

Substituting the result of $L_c = d+1$ into (B.1), we can obtain the corresponding FBF length:

$$L_b = L_h - 1. \tag{B.5}$$

Hence, the maximum effective FBF length is equal to the memory of the given channel. It is straightforward to see that when $L_b = L_h - 1$, the FBF cancels all the postcursor ISI introduced by the previously detected symbols. All potential performance improvement gained by using decision feedback is achieved. From the constellation space, we can see that when $L_b = L_h - 1$ and $L_c = d+1$, the FBF feeds back all symbols prior to x_{k-d}, e.g., $\hat{\mathbf{x}}^{fb}_{k-d} = [x_{k-d-1}, \ldots, x_{k-d-L_b}]$. The number of constellation points in each subconstellation of the MAE-DFE is 2^{d+1}, which cannot be further reduced.

Bibliography

Bibliography

[1] J. G. Proakis and M. Salehi, *Digital Communications.* McGraw-Hill, 2007.

[2] S. Haykin and M. Moher, *Fundamentals of Wireless Communication.* Hoboken, NJ: John Wiley and Sons, Inc, March 2009.

[3] M. Tüchler, A. C. Singer, and R. Koetter, "Minimum mean squared error equalization using a priori information," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 673–683, March 2002.

[4] A. G. Klein and P. Duhamel, "Decision-feedback equalization for pulse-position modulation," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5361–5369, November 2007.

[5] J. M. Cioffi, G. P. Dudevoir, M. V. Eyuboglu, and G. D. Forney, "MMSE decision-feedback equalizers and coding. II. coding results," *IEEE Transactions on Communications*, vol. 43, no. 10, pp. 2595–2604, October 1995.

[6] P. Savazzi, L. Favalli, E. Costamagna, and A. Mecocci, "A suboptimal approach to channel equalization based on the nearest neighbor rule," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 9, pp. 1640–1648, December 1998.

[7] I. Barhumi and M. Moonen, "MLSE and MAP equalization for transmission over doubly selective channels," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 8, pp. 4120–4128, October 2009.

[8] L. Cao, C. Chen, P. Orlik, J. Zhang, and D. Gu, "A trellis-based technique for blind channel estimation and equalization," *Journal of Communications and Networks*, vol. 6, no. 1, pp. 19–25, March 2004.

[9] L. Yiin and G. Stuber, "MLSE and soft-output equalization for trellis-coded continuous phase modulation," *IEEE Transactions on Communications*, vol. 45, no. 6, pp. 651–659, June 1997.

[10] B. L. Yeap, T. Liew, J. Hamorsky, and L. Hanzo, "Comparative study of turbo equalization schemes using convolutional, convolutional turbo, and block-turbo codes," *IEEE Transactions on Wireless Communications*, vol. 1, no. 2, pp. 266–273, April 2002.

[11] M. Tüchler and A. C. Singer, "Turbo equalization: An overview," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 920–952, February 2011.

[12] A. Goldsmith, *Wireless Communications.* New York, NY: Cambridge University Press, August 2005.

[13] R. Johannesson and K. Zigangirov, *Fundamentals of convolutional coding.* IEEE Press, 1999.

[14] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Upper Saddle River, NJ: Person Prentice Hall, August 2009.

[15] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications.* John Wiley and Sons, Inc, June 2013.

[16] S. Haykin, *Adaptive Filter Theory.* UpperSaddle River, NJ: Prentice Hall, 2002.

[17] R. Lupas and S. Verdú, "Linear multi-user detectors for synchronous code-division multiple-access channels," *IEEE Transactions on Information Theory*, vol. 35, no. 1, January 1989.

[18] S. Verdú, *Multiuser Detection.* Cambridge University Press, 1998.

[19] A. S. Gupta, J. K. Nelson, W. Zhou, A. C. Singer, and J. Preisig, "A geometric approach to improve interference mitigation in multi-user detection and equalization," *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1694–1705, April 2011.

[20] C. Y. Chen, C. Heneghan, and J. M. Cioffi, "A novel decision feedback equalizer design based on generalized space translation," *The 38th Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 727–731 Vol.1, November 2004.

[21] R. Chen and W. Wu, "Adaptive asymptotic Bayesian equalization using a signal space partitioning technique," *IEEE Transactions on Signal Processing*, vol. 52, no. 5, pp. 1376 – 1386, May 2004.

[22] S. Chen, L. Hanzo, and B. Mulgrew, "Decision-feedback equalization using multiple-hyperplane for detecting ISI-corrupted M-ary PAM signals," *IEEE Transactions on Communications*, vol. 49, no. 9, 2001.

[23] S. Chen and H. C. J., "Design of the optimal separating hyperplane for the decision feedback equalizer using support vector machines," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, 2000.

[24] S. Chen and C. J. Harris, "Design of the optimal separating hyperplane for the decision feedback equalizer using support vector machines," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, 2000.

[25] M. Reuter, J. C. Allen, J. R. Zeidler, and R. C. North, "Mitigating error propagation effects in a decision feedback equalizer," *IEEE Transactions on Communications*, vol. 49, no. 11, pp. 2028–2041, November 2001.

[26] M. Y. Lin, V. Y. Krachkovsky, and G. Mathew, "Conditional ML and MAP techniques for error propagation suppression in multi-path DFE detectors," *IEEE Transactions on Magnetics*, vol. 36, no. 5, pp. 2160–2163, September 2000.

[27] R. L. Kosut, W. Chung, C. R. Johnson, and S. P. Boyd, "On achieving reduced error propagation sensitivity in DFE design via convex optimization," *The 39th IEEE Conference on Decision and Control*, vol. 5, pp. 4320–4323 vol.5, 2000.

[28] M. DeGroot, *Optimal Statistical Decisions.* McGraw-Hill, 1970.

[29] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Statistical inference for probabilistic functions of finite state Markov chains,” *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, March 1974.

[30] G. D. Forney, Jr., “The Viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[31] J. Anderson, “Limited search trellis decoding of convolutional codes,” *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 944–955, September 1989.

[32] R. E. Blahut, *Algebraic Codes for Data Transmission.* New York, NY: Cambridge University Press, 2003.

[33] A. S. Gupta and A. C. Singer, “Interference suppression for memoryless nonlinear multiuser systems using constellation structure,” *IEEE Transactions on Signal Processing*, vol. 5, no. 11, 2008.

[34] J. K. Nelson and A. C. Singer, “Asymptotic efficiency of a blind maximum likelihood sequence detector,” *The 37th Asilomar Conference on Signals, Systems and Computers*, vol. 2, 2003.

[35] A. S. Gupta and A. C. Singer, “Successive interference cancellation using constellation structure,” *IEEE Transactions on Signal Processing*, vol. 55, no. 12, pp. 5716–5730, December 2007.

[36] W. Zhou, J. K. Nelson, and A. S. Gupta, “Computationally efficient design of the MAE equalizer for binary signaling,” *The 45th Asilomar Conference on Signals, Systems and Computers*, pp. 939–943, 2011.

[37] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms.* Hoboken, NJ: John Wiley and Sons, Inc, May 2006.

[38] K. Diamantaras, “Blind channel identification based on the geometry of the received signal constellation,” *IEEE Transactions on Signal Processing*, vol. 50, no. 5, pp. 1133–1143, May 2002.

[39] M. Berg, O. Cheong, M. V. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications.* Springer, April 2008.

[40] H. G. Eggleston, *Convexity.* Cambridge University Press, 1958.

[41] M. Mavroforakis and S. Theodoridis, “A geometric approach to support vector machine (SVM) classification,” *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 671–682, May 2006.

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

[43] R. Kwong and E. Johnston, “A variable step size lms algorithm,” *IEEE Transactions on Signal Processing*, vol. 40, no. 7, pp. 1633–1642, Jul 1992.

[44] Y. Gong and C. F. N. Cowan, “Optimum decision delay of the finite-length DFE,” *IEEE Signal Processing Letters*, vol. 11, no. 11, pp. 858–861, November 2004.

[45] J. E. Smee and N. C. Beaulieu, “On the equivalence of the simultaneous and separate MMSE optimizations of a DFE FFF and FBF,” *IEEE Transactions on Communications*, vol. 45, no. 2, pp. 156–158, February 1997.

[46] C. R. Berger, Z. Wang, J. Huang, and S. Zhou, “Application of compressive sensing to sparse channel estimation,” *IEEE Communications Magazine*, vol. 48, no. 11, pp. 164–174, November 2010.

[47] N. Michelusi, U. Mitra, A. F. Molisch, and M. Zorzi, “UWB sparse/diffuse channels, part I: Channel models and Bayesian estimators,” *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5307–5319, 2012.

[48] ——, “UWB sparse/diffuse channels, part II: Estimator analysis and practical channels,” *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5320–5333, 2012.

[49] L. Fan, C. He, D. Wang, and L. Jiang, “Efficient robust adaptive decision feedback equalizer for large delay sparse channel,” *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 449–456, May 2005.

[50] S. Chen, J. Xiang, and J. Shi, “Modified decision feedback equalizer (DFE) for sparse channels in underwater acoustic communications,” *International Conference on Wireless, Mobile and Multimedia Networks*, pp. 1–4, November 2006.

[51] M. J. Lopez and A. C. Singer, “A DFE coefficient placement algorithm for sparse reverberant channels,” *IEEE Transactions on Communications*, vol. 49, no. 8, pp. 1334–1338, Auguest 2001.

[52] A. A. Rontogiannis and K. Berberidis, “Efficient decision feedback equalization for sparse wireless channels,” *IEEE Transactions on Wireless Communications*, vol. 2, no. 3, pp. 570–581, May 2003.

[53] N. C. McGinty, R. A. Kennedy, and P. Hocher, “Parallel trellis Viterbi algorithm for sparse channels,” *IEEE Communications Letters*, vol. 2, no. 5, pp. 143–145, May 1998.

[54] N. Benvenuto and R. Marchesani, “The Viterbi algorithm for sparse channels,” *IEEE Transactions on Communications*, vol. 44, no. 3, pp. 287 –289, March 1996.

[55] J. Nelson and A. Singer, “Bayesian ML sequence detection for ISI channels,” *The 40th Annual Conference on Information Sciences and Systems*, pp. 693–698, 2006.

[56] Y. S. Han and P. N. Chen, *Sequential Decoding of Convolutional Codes.* New York: Wiley, 2002.

[57] M. Chen, J. Tuqan, and D. Zhi, “A quadratic programming approach to blind equalization and signal separation,” *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2232–2244, 2009.

[58] R. Liu and Y. Inouye, "Blind equalization of MIMO-FIR channels driven by white but higher order colored source signals," *IEEE Transactions on Information Theory*, vol. 48, no. 5, pp. 1206–1214, 2002.

[59] I. Santamaria, C. Pantaleon, L. Vielva, and J. Ibanez, "Blind equalization of constant modulus signals using support vector machines," *IEEE Transactions on Signal Processing*, vol. 52, no. 6, pp. 1773–1782, 2004.

[60] V. Zarzoso and P. Comon, "Blind and semi-blind equalization based on the constant power criterion," *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4363–4375, 2005.

[61] L. Tong, G. Xu, and T. Kailath, "Blind identification and equalization based on second-order statistics: A time domain approach," *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 340–349, 1994.

[62] S. Chen, W. Yao, and L. Hanzo, "Semi-blind adaptive spatial equalization for MIMO systems with high-order QAM signalling," *IEEE Transactions on Wireless Communications*, vol. 7, no. 11, pp. 4486–4491, 2008.

[63] H. Nguyen and B. C. Levy, "The expectation-maximization Viterbi algorithm for blind adaptive channel equalization," *IEEE Transactions on Communications*, vol. 53, no. 10, pp. 1671–1678, 2005.

[64] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* New York: Springer, 2010.

[65] A. Shoa and S. Shirani, "Optimized atom position and coefficient coding for matching pursuit-based image compression," *IEEE Transactions on Image Processing*, vol. 18, no. 12, pp. 2686–2694, 2009.

[66] ——, "A block based encoding algorithm for matching pursuit image coding," *The 16th IEEE International Conference on Image Processing (ICIP)*, pp. 1913–1916, 2009.

[67] M. J. Horowitz and D. L. Neuhoff, "Image coding by matching pursuit and perceptual pruning," *International Conference on Image Processing*, vol. 3, pp. 654–657, 1997.

[68] S. Zhang, Q. Wan, and H. Wang, "DOA estimation in mechanical scanning radar systems using sparse signal reconstruction methods," *The 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM),*, pp. 1–4, 2011.

[69] T. T. Cai and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4680–4688, 2011.

[70] T. Kang and R. Iltis, "Matching pursuits channel estimation for an underwater acoustic OFDM modem," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5296–5299, 2008.

[71] S. Cotter and B. Rao, "The adaptive matching pursuit algorithm for estimation and equalization of sparse time-varying channels," *The 34th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1772–1776, 2000.

[72] S. G. Mallat and Z. Zhang, “Matching pursuits with time-frequency dictionaries,” *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[73] S. F. Cotter and B. D. Rao, “Sparse channel estimation via matching pursuit with application to equalization,” *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 374–377, 2002.

[74] Z. Hussain, J. Shawe-Taylor, D. Hardoon, and C. Dhanjal, “Design and generalization analysis of orthogonal matching pursuit algorithms,” *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5326–5341, 2011.

[75] J. E. Gentle, *Numerical Linear Algebra with Applications in Statistics.* Springer, January 1998.

[76] S. Chen, X. C. Yang, and L. Hanzo, “Blind joint maximum likelihood channel estimation and data detection for single-input multiple-output systems,” *The 6th IEE International Conference on 3G and Beyond*, pp. 1–5, November 2005.

[77] N. Seshadri, “Joint data and channel estimation using blind trellis search techniques,” *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 1000–1011, February 1994.

[78] J. Ding, L. Chen, and Y. Gu, “Perturbation analysis of orthogonal matching pursuit,” *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 398–410, 2013.

[79] G. G. Raleigh and J. M. Cioffi, “Spatio-temporal coding for wireless communications,” *IEEE Global Telecommunications Conference*, vol. 3, pp. 1809–1814 vol.3, 1996.

[80] I. E. Telatar, “Capacity of multi-antenna Gaussian channels,” *European Transactions on Telecommunications*, vol. 10, pp. 585–595, 1999.

[81] G. J. Foschini and M. J. Gans, “On limits of wireless communications in a fading environment when using multiple antennas,” *Wireless Personal Communications*, vol. 6, pp. 311–335, 1998.

[82] J. E. Gentle, *Matrix Algebra: Theory, Computations, and Applications in Statistics.* Springer, July 2007.

[83] W. Li and J. C. Preisig, “Estimation of rapidly time-varying sparse channels,” *IEEE Journal of Oceanic Engineering*, vol. 32, no. 4, pp. 927–939, 2007.

[84] C. R. Berger, J. Gomes, and J. M. F. Moura, “Study of pilot designs for cyclic-prefix OFDM on time-varying and sparse underwater acoustic channels,” *IEEE OCEANS*, pp. 1–8, 2011.

[85] S. H. Byun, W. Seong, and S. M. Kim, “Sparse underwater acoustic channel parameter estimation using a wideband receiver array,” *IEEE Journal of Oceanic Engineering*, vol. 38, no. 4, pp. 718–729, 2013.

[86] C. Berger, S. Zhou, J. C. Preisig, and P. Willett, “Sparse channel estimation for multicarrier underwater acoustic communication: From subspace methods to compressed sensing,” *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1708–1721, 2010.

[87] N. P. Chotiros, “Non-Rayleigh distributions in underwater acoustic reverberation in a patchy environment,” *IEEE Journal of Oceanic Engineering*, vol. 35, no. 2, pp. 236–241, April 2010.

[88] P. K. Suetin, A. I. Kostrikin, and Y. I. Manin, *Linear Algebra and Geometry.* CRC Press, July 1989.

[89] W. S. Massey, “Cross products of vectors in higher dimensional Euclidean spaces,” *The American Mathematical Monthly*, vol. 90, no. 10, 1983.

Biography

Weiwei Zhou was born in Hubei Province, China, in January 1982. He received the B.S. degree in electrical engineering from University of Science and Technology Liaoning, China, in 2000, and the M.S. degree in electrical engineering from GyeongSang National University, Korea, in 2007. Since 2007, he has been with the Department of Electrical and Computer Engineering in George Mason University. His research interests include adaptive array processing, statistical modeling in wireless communication system, convex optimization in signal processing, and adaptive filter design.