Direct-Form Adaptive Equalization for Underwater Acoustic Communication

by

Atulya Yellepeddi

B. Tech., Indian Institute of Technology, Roorkee (2010)

Submitted in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

and the

WOODS HOLE OCEANOGRAPHIC INSTITUTION

June 2012

Author .....
Electrical Engineering/Applied Ocean Science and Engineering
May 16, 2012

Certified by .....
James C. Preisig
Associate Scientist, Woods Hole Oceanographic Institution
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair, Committee on Graduate Students

Accepted by .....
Henrik Schmidt
Chair, Joint Committee for Applied Ocean Science and Engineering

Direct-Form Adaptive Equalization for Underwater Acoustic Communication

by

Atulya Yellepeddi

Submitted to the
Department of Electrical Engineering and Computer Science
on May 16, 2012
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering
and Applied Ocean Science and Engineering

Abstract

Adaptive equalization is an important aspect of communication systems in various environments. It is particularly important in underwater acoustic communication systems, as the channel has a long delay spread and is subject to the effects of time-varying multipath fading and Doppler spreading.

The design of the adaptation algorithm has a profound influence on the performance of the system. In this thesis, we explore this aspect of the system. The emphasis of the work presented is on applying concepts from inference and decision theory and information theory to provide an approach to deriving and analyzing adaptation algorithms. Limited work has been done so far on rigorously devising adaptation algorithms to suit a particular situation, and the aim of this thesis is to concretize such efforts and possibly to provide a mathematical basis for expanding it to other applications.

We derive an algorithm for the adaptation of the coefficients of an equalizer when the receiver has limited or no information about the transmitted symbols, which we term the Soft-Decision Directed Recursive Least Squares algorithm. We will demonstrate connections between the Expectation-Maximization (EM) algorithm and the Recursive Least Squares algorithm, and show how to derive a computationally efficient, purely recursive algorithm from the optimal EM algorithm.

Then, we use our understanding of Markov processes to analyze the performance of the RLS algorithm in hard-decision directed mode, as well as of the Soft-Decision Directed RLS algorithm. We demonstrate scenarios in which the adaptation procedures fail catastrophically, and discuss why this happens. The lessons from the analysis guide us on the choice of models for the adaptation procedure. We then demonstrate how to use the algorithm derived in a practical system for underwater communication using turbo equalization. As the algorithm naturally incorporates soft information into the adaptation process, it becomes easy to fit it into a turbo equalization framework. We thus provide an instance of how to use the information

of a turbo equalizer in an adaptation procedure, which has not been very well explored in the past. Experimental data is used to prove the value of the algorithm in a practical context.

Thesis Supervisor: James C. Preisig
Title: Associate Scientist, Woods Hole Oceanographic Institution

Acknowledgments

First, thanks to my research advisor, Dr. James Preisig for his endless patience and excellent advice. His guidance, knowledge, committment to quality reseach and penchance for putting learning first have made it a pleasure to work with him. I look forward to continuing my research with him.

My thanks to Prof. Gregory Wornell for the valuable advice and discussions and always helping with a push in the right direction when needed, not to mention finding a place for me in his group at MIT. Thanks also to Prof. Andrew Singer for having me visit UIUC for a few days and the invaluable ideas I had- both from him and members of his group there.

A great group of people that I've worked with and who have always been ready to bounce around even the most silly of ideas- Milutin Pajovic, Da Wang, Dr. Uri Erez, Dr. Yuval Kochman, Erica Daly and Thomas Reidl at UIUC, and my officemates, Gauri Joshi and Qing He- thank you for the discussions, so many of which went into the thesis, and for the hours of fun that we have had.

Speaking of fun, to all the amazing friends I have made over the last couple of years- thank you for keeping me sane and for all the times I've used your respective labs and homes as tertiary office space.

This thesis would not have been possible without the support from the agencies that funded this research- the Academic Programs Office at WHOI and the Office of Naval Research (through ONR Grant #N00014-07-10738 and #N00014-10-10259).

And finally, thanks to my incredible family and to Divya. For everything.

Contents

List of Figures

Chapter 1

Introduction

Blue, green, grey, white, or black; smooth, ruffled, or mountainous; that ocean is not silent.

-H.P. Lovecroft

Oceanography is a science born out of human desire to understand that important part of our world- the oceans. The oceans are where life on earth began, and they hold secrets that continue to influence the course of human history.

In today's world, communication in any environment is an integral part of being able to understand and exploit it. The oceans are no exception. From submarines to undersea robotics, from fisheries to oil exploration- many problems in oceanography are dependent on the ability to perform underwater wireless communications over a range of distances.

This thesis explores one major aspect of wireless underwater communication systems: adaptation algorithms for channel equalization. Adaptation algorithms are an important part of modern signal processing, and find applications in various fields, particularly in systems designed to handle time-variability. The primary objective of this thesis is to develop methods for the design and analysis of adaptation algorithms for practical communication systems, with the emphasis being laid on channel equalization for wireless underwater communication.

In this chapter, we introduce the motivation for the problems that we seek to

address with this work. We then present a bird's eye view of the traditional approaches to the design of systems for underwater communication. The chapter concludes with an overview of the organization of the thesis.

1.1 The Challenges of Underwater Communication

The objective of underwater communication is to transmit data wirelessly over ranges of a few hundred meters to a few tens of kilometers. The range of electromagnetic waves underwater, however, is only a few meters. Therefore, RF communication is infeasible in the ocean environment. Optical communications are similarly only feasible up to a few hundred meters in clear water, and the ranges diminishes in turbid water.

The typical medium of communication underwater is acoustic waves, which propagate over the ranges of interest with reasonable signal levels. The underwater acoustic channel, however, is fraught with difficulties [53]. The channel impulse response is typically long and has multipath. As a result, it is subject to time varying Inter-Symbol Interference (ISI) and Doppler effects [38].

A sample channel impulse is shown in Figure 1-1. This was collected on Julian Date 178, 2011 as part of the KAM11 Experiment off the coast of Kauai Island, Hawaii. This impulse response shows a long, time-varying channel. However, the specific environmental conditions have a profound impact on the channel characteristics, making it hard to explicitly model such a channel.

1.1.1 The Need for Adaptive Equalization

Evidently, with such a long impulse response channel, we expect ISI to be a challenge at the receiver. We hence require an *equalizer* to be part of our receiver set up. An equalizer is a system of filters which is used to invert the response of a noisy channel in order to recover the symbols transmitted through the channel without blowing up

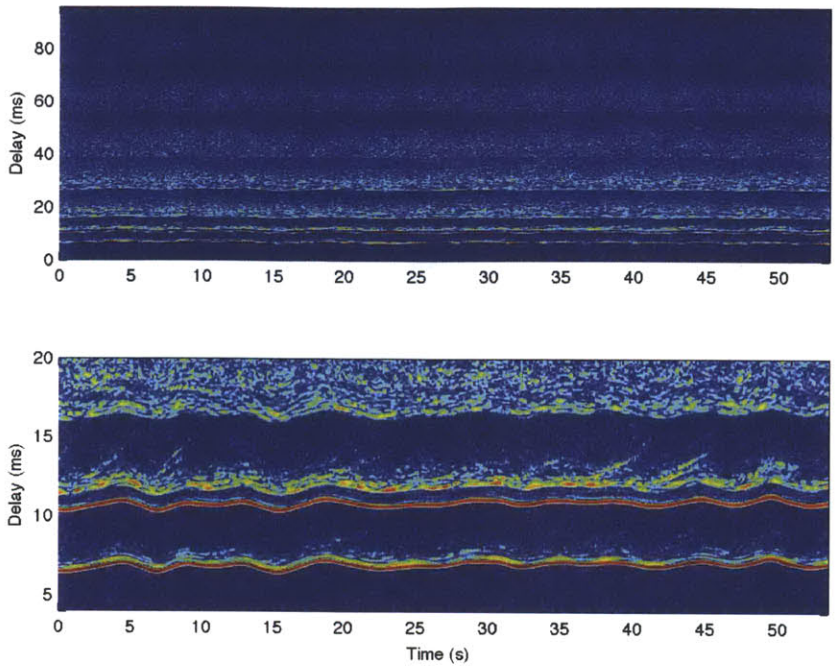


Figure 1-1: Channel delay spread- KAM11, Julian Date 178 at 1855

the noise (this is not a strict inversion, of course- it is implemented as an MMSE or Least Squares processor.

In a time-varying environment, the coefficients of the filter need to be varied with time. This is true also of a variety of environments including underwater communication, mobile radio [40] and magnetic disk recording technology [9], for instance. The adaptation of these filters in practical environments and understanding the behaviour of the adaptation algorithms is the subject of this thesis. We begin with a study of history- by looking at the prior appoaches to underwater communication systems and how they tie in with the work presented here.

1.2 An Overview of Underwater Communication Systems

A modern history of underwater communication systems probably begins with the 2nd World War. Advances in signal processing techniques have made it possible to

significantly improve the data rates, by implementing sophisticated signal processing algorithms at the receiver of the system.

We look at the various kinds of communication and signal processing algorithms and techniques that have been employed. This is necessarily a very brief and informal review. A much more complete review may be found in [50]. We focus on the specific receiver structures that are relevant to the work presented here.

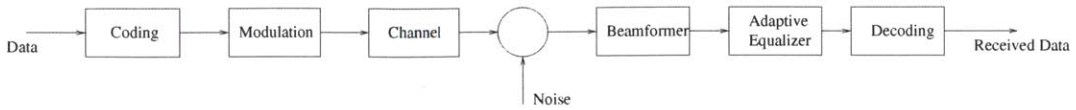
The major issue has been how we should design the system to combat the effects of time-varying multipath dispersion. Phase coherant techniques are required for reasonable data rates, but they are even further subject to these effects. So an array of sensors is used to allow for diversity to combat the multipath. The starting point, therefore is an array processor to mitigate the multipath dispersion. The resulting signal is treated as though it comes from a single ISI channel and is equalized. The optimal algorithms for such a structure are described in [52].

The beamformer is typically followed by an adaptive equalizer. Due to its low complexity, reasonable convergence and tracking properties, the Recursive Least Squares (RLS) algorithm is known to be a good choice of adaptation algorithm [23]. The adaptation algorithm is usually trained with some known data, and then used in decision directed mode.

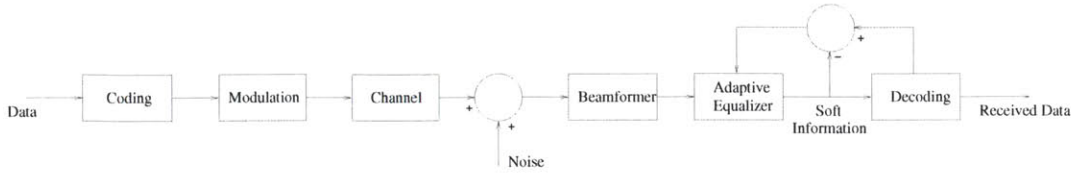
We distinguish between 2 kinds of equalization - *channel estimation* based equalization and the so called *direct-form* equalization. In channel estimation based adaptive equalization, the channel impulse response is explicitly estimated. Using the channel impulse response, an optimal equalizer can be formulated based on any one of a number of criteria [20].

On the other hand, in direct-form equalization, the equalizer coefficients are directly computed from the output of the channel. The emphasis of this thesis is on this form of equalization. This form of filter is preferred due to ease of implementation and low complexity. However, the algorithms for adaptation and methods of analysis that we will present can be used for channel estimation type approaches as well, with some modifications.

A third approach that is often employed to combat ISI is that of *Orthogonal Fre-*



(a) Basic Block Diagram



(b) Turbo-Equalizer Block Diagram

Figure 1-2: Block Diagrams of Communication System

quency Division Multiplexing (OFDM) [51]. While this has a lower cost of complexity of equalization, OFDM receivers are subject to severe Doppler effects. We will not focus on this technology in this thesis, but they have been the subject of research in the underwater acoustics community in recent times. (See, for instance, [29] and [25]).

In general, forward error correction coding is applied to the data before transmission. This adds redundancy and enables us to further improve the performance of the communication link. The standard block diagram in Figure 1-2a of the communication system, which shows how these components fit together.

The advances in the subject of near capacity achieving codes have led to the development of sophisticated turbo codes [19]. It was realized that a similar concept could be applied to equalization, where the equalizer was treated as a kind of decoder. This led to the principle of turbo equalization [27], depicted in Figure 1-2b. Here, the equalizer and decoder iterate and exchange the so-called *soft information*. This information is used by the other block in the next iteration. For a more detailed exposition of turbo equalization, see [26]. Turbo equalization has been applied to the underwater communication in [48], [8], among others. We will show that the algorithm we derive has application as an adaptive turbo equalizer, and will very naturally fit into the turbo framework.

We conclude this section with a note on Faster-than-Nyquist (FTN) signaling, which was originally introduced in [32]. This technique has recently found application to the channels of our interest [13]. It can increase the data rate without increasing

the bandwidth. However, this is at the expense of lengthening the channel impulse response, increasing the amount of ISI in the channel. Therefore, we require a good adaptive equalizer to take advantage of this method. This is another use of the improved equalizer that we present here.

1.3 Thesis Organization

This thesis is organized as follows: in Chapter 2, we provide a detailed introduction to the system layout and the RLS Algorithm. We formally describe the problem that we will attempt to solve in the thesis.

Chapter 3 contains the derivation of the soft-decision directed RLS adaptation algorithm from the EM Algorithm. In this chapter, we also build connections to blind equalization techniques which are prevalent in literature and show how the algorithm bridges the gap between the maximum likelihood based EM Algorithm and these practical approaches.

In Chapter 4, we analyze the performance of adaptive equalizers under decision directed adaptation. We also analyze the performance of the algorithm that we derive and show why we expect this to do better. This chapter also explains what insights the analysis holds for further development of the algorithm.

Chapter 5 considers a practical implementation of a turbo-equalizer system based on the soft-decision directed RLS algorithm. We demonstrate how this compares to existing turbo equalization approaches, and the choices of what kinds of codes to use in the turbo equalizer.

We then demonstrate the practical applicability of the system in Chapter 6 by testing it with actual data from the KAM11 Acoustic Experiments. The experimental setup and transmitted signals are presented in detail, and a comparison of the algorithm against the existing algorithms in the field is carried out.

Finally, Chapter 7 summarizes the results presented in the thesis, and suggests possible directions for future work.

Chapter 2

Preliminaries

2.1 Notation

Table 2.1 contains the notations that are used in the rest of the thesis without redefinition.

2.2 System Model

The block diagram of the system of interest is shown in Figure 2-1. This is the diagram of the channel and an adaptive decision feedback equalizer. In this section we describe how the different blocks operate and the mathematical models that we use for the system. The thin solid lines represent the flow of data in the equalizer and the dotted lines represent the flow of information to and from the adaptation algorithm used to adapt the coefficients of the filter. We will modify the system model as required in later chapters. A few points are worth mentioning at this stage. This diagram is incomplete in the sense that it does not include a turbo loop (a feedback loop from the decoder to the equalizer). This is done for clarity's sake. We will describe the turbo structure more in Chapter 5, and will introduce the feedback loop at that juncture.

Also with reference to Figure 1-2a, we note that we have not included a beamformer or array processor explicitly in Figure 2-1. This is because practically, we

Symbol Type	Meaning
x (non-boldface math symbol)	Scalar constant or variable
$\mathbf{u}$ (boldface, lowercase symbol)	Column vector (size inferred from context)
$\mathbf{A}$ (boldface, uppercase symbol)	Matrix
$\mathbf{I}_N$	$N \times N$ identity matrix (without subscript, size inferred from context)
$\mathrm{p}(\cdot;\cdot)$	probability distribution function, semicolon indicates parametrization by a *deterministic*, but *unknown* quantity following the semicolon.
$\mathrm{p}(\cdot\|\cdot)$	probability distribution function, pipe operator | indicates conditioning.
$\mathcal{N}(\mu, \mathbf{\Sigma})$	P.D.F. of Multivariate Normal Distribution with mean μ and covariance matrix $\mathbf{\Sigma}$.
$\sim$	Indicates distribution of random variable, eg., $x \sim \mathcal{N}(0,1)$ means random variable x is distributed as a standard normal variable.
$\mathbb{E}_{\mathrm{p}(\cdot)}[\cdot]$	Expectation of the quantity in the square brackets with respect to the p.d.f. specified in the subscript
$\mathbb{P}(\cdot)$	Probability of the event in brackets
*	Complex Conjugate
^{T}	Matrix Transpose
^{H}	Matrix Hermitian

Table 2.1: Notations used in the thesis

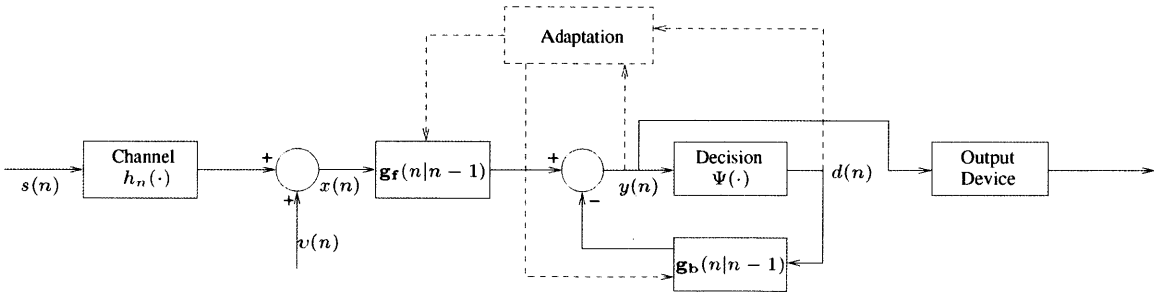


Figure 2-1: Block Diagram of Direct-Form Adaptation Equalizer

can implement the multichannel processor by stacking up the inputs from different channels into an equivalent feedforward input vector. The beamforming then happens implicitly within the equalizer.

2.2.1 The Channel

At the transmitter end, we define $s(n)$ as the symbol transmitted at time n. The symbol passes through a channel, which introduces ISI. The channel also introduces random noise, represented by $v(n)$ at time n. Thus, corresponding to the symbol transmitted at time n, we have a received signal at the output of the channel, which we denote by $x(n)$. M is the length of the channel response. This means that the output of the channel at time n depends on symbols $s(n), s(n-1), \ldots, s(n-\overline{M-1})$. If $h_n(\cdot)$ is the channel response at time n, then we write

$$x(n) = h_n(s(n), s(n-1), \ldots, s(n-\overline{M-1})) + v(n) \tag{2.1}$$

Note that we assume that the channel response is causal. We may do this with no loss of generality, because all we would need to do is introduce a suitable delay into the receiver if it is not. We will assume a linear (but time-varying) channel, so we can write the channel filter coefficients $h_k(n)$, $k = 0, 1, \ldots, M-1$ as *channel coefficients* at time n. The channel output is then a convolution, given by

$$x(n) = \sum_{k=0}^{M-1} h_k^*(n)s(n-k) + v(n) \tag{2.2}$$

We define the channel vector, $\mathbf{h}(n)$, as:

$$\mathbf{h}(n) = \begin{bmatrix} h_0(n) & \cdots & h_{M-1}(n) \end{bmatrix}^T \tag{2.3}$$

2.2.2 The Equalizer Filters

The receiver end is a Decision Feedback Filter (DFE), which has 2 filters: a feedforward filter $\mathbf{g_f}$ of length N_f and a feedback filter $\mathbf{g_b}$ of length N_b. The purpose

of the feedforward filter is to invert a portion of the channel impulse response while ensuring that it does not blow up the noise that is present in the received signal. The feedback filter cancels out residual ISI by subtracting out past symbols (or decisions) from the output of the feedforward section. Intuitively, the feedforward filter inverts the channel, and the feedback filter subtracts out the remaining ISI. We define the total equalizer dimension as $N = N_f + N_b$.

The input to the feedforward filter at any time n is the past N_f symbols at the output of the channel, which we denote by $\mathbf{x}(n)$

$$\mathbf{x}(n) = \begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-\overline{N_f-1}) \end{bmatrix}^T \tag{2.4}$$

Similarly, we denote by $d(n)$ the decision made by the system at time n. This is an estimate, in some sense, of the transmitted symbol. We shall discuss what we mean by estimate in this context more in detail later. The input to the feedback filter at time n can thus be denoted by $\mathbf{z}(n)$, where

$$\mathbf{z}(n) = \begin{bmatrix} z(n-1) & z(n-2) & \cdots & z(n-N_b) \end{bmatrix}^T \tag{2.5}$$

As these filters are adaptively varied, we specify the time at which we are looking at the filter coefficients. We denote by $\mathbf{g_f}(n|n-1)$ and $\mathbf{g_b}(n|n-1)$ the respective feedforward and feedback filters used to filter the data at time n, computed given all the data up to (and including) time $n-1$. Then, the output of the equalizer at time n, which is denoted by $y(n)$ is given by

$$y(n) = \mathbf{g_f}^H(n|n-1)\mathbf{x}(n) - \mathbf{g_b}^H(n|n-1)\mathbf{z}(n) \tag{2.6}$$

We can significantly simplify notation by defining a consolidated "overall" filter coefficient vector, denoted simply as $\mathbf{g}(n|n-1)$ at time n, by

$$\mathbf{g}(n|n-1) = \begin{bmatrix} \mathbf{g_f}(n|n-1) \\ -\mathbf{g_b}(n|n-1) \end{bmatrix} \tag{2.7}$$

Correspondingly, define the overall input vector to the equalizer $\mathbf{u}(n)$ as

$$\mathbf{u}(n) = \begin{bmatrix} \mathbf{x}(n) \\ \mathbf{z}(n) \end{bmatrix} \tag{2.8}$$

Then Equation 2.6 can be written more simply as

$$y(n) = \mathbf{g}^H(n|n-1)\mathbf{u}(n) \tag{2.9}$$

The form of the equalizer operation represented by Equations 2.7-2.9 allows for a great flexibility in choosing the form of equalizer. For instance, we need not treat linear feedforward equalizers any differently in the mathematical framework, as long as we make the right assumptions on the input vector.

2.2.3 The Channel Transfer Matrix

We also define a channel transfer matrix $\mathbf{H}(n)$ that relates the transmitted symbols to the overall input vector $\mathbf{u}(n)$. We note that $x(n)$ depends on symbols $s(n), s(n-1), \ldots, s(n-\overline{M-1})$, and $x(n-\overline{N_f-1})$ depends on symbols $s(n-\overline{N_f-1}, \ldots, s(n-\overline{N_f-1}-\overline{M-1})$. Also, assuming no error propagation, the feedback filter has symbols $s(n-1), \ldots, s(n-N_b)$. Thus, the input depends on $s(n), s(n-1), \ldots, s(n-P)$, where $P = \max(N_f + M - 2, N_b)$. The $P \times N$ channel transfer matrix (with no error propagation in the feedback filter)

$$\mathbf{H}(n) = \left[\begin{array}{cccc|c} h_0(n) & 0 & & 0 & 0 \\ h_1(n) & h_0(n-1) & & 0 & \mathbf{I}_{N_b} \\ \vdots & \vdots & \ddots & h_0(n-(N_f-1)) & \\ h_{M-1}(n) & h_{M-2}(n-1) & & \vdots & \vdots \\ 0 & h_{M-1}(n-1) & & & \\ 0 & 0 & \ddots & h_{M-1}(n-(N_f-1)) & 0 \\ \vdots & \vdots & & 0 & 0 \end{array}\right]_{P \times N} \tag{2.10}$$

and the input to the equalizer can be written as

$$\mathbf{u}(n) = \mathbf{H}^H(n) \begin{bmatrix} s(n) \\ s(n-1) \\ \vdots \\ s(n-P) \end{bmatrix} + \begin{bmatrix} v(n) \\ v(n-1) \\ \vdots \\ v(n-\overline{N_f-1}) \\ \mathbf{0}_{N_b \times 1} \end{bmatrix} \tag{2.11}$$

Note that if we assumed there was error propagation, i.e., the symbols in the feedback filter were not perfect, there would be a complex, non-linear relationship between the input to the feedback filter and the transmitted symbols, and the simple matrix relations of Equations 2.10 and 2.11 would not be sufficient to describe it.

2.2.4 Outputs and Decisions

$y(n)$ is mapped by the decision device into an estimate of the symbol used in the adaptation and feedback filters, which we denote as $d(n)$. It is important to note that the estimate used for adaptation may not be the same as the one used for the feedback filter in general. By default, we will assume this, but it is not important to the derivations presented, as will become clear. In cases where it is important, it will explicitly be stated what assumptions are being made. Finally, $y(n)$ is also mapped into an output decision based on what we need at the decoder or the next component of the system (or hard decisions if this is the final stage).

Table 2.2 summarizes the most important of the quantities which we have defined in this section. We now move to describing the purpose of the "Adaptation" block and the algorithms used for it. This will lead us to describing the problem of interest.

2.3 The Adaptation Algorithm

Since the channel is time-variant, the equalizer coefficients must be varied with time. This process is called *adaptation.* The algorithm that performs the adaptation process

Symbol	Quantity Represented
$s(n)$	Symbol transmitted at time n
M	Channel Impulse Response length
N_f	Feedforward filter length
N_b	Feedback filter length
N	Total filter length, $N_f + N_b$
P	Number of symbols on which equalizer input depends, given by $P = \max(N_f + M - 2, N_b)$
$\mathbf{u}(n)$	Overall input to the equalizer filters corresponding to $s(n)$, given by $\left[x(n) \ldots x(n - \overline{N_f - 1}), z(n-1) \ldots z(n - N_b)\right]^T$
$\mathbf{g}(n\|n-1)$	Overall coefficient vector to filter input at time n, computed given data up to (and including) time $n-1$
$y(n)$	Output of equalizer corresponding to $s(n)$
$d(n)$	Estimate of $s(n)$ used in feedback filter and adaptation
$\Psi(\cdot)$	Any general decision rule that is used to map $y(n)$ into decisions used in the feedback filter and/or in adaptation, such that $d(n) = \Psi(y(n))$

Table 2.2: Summary of Important Symbols Defined for the System

is what we will consider for the rest of the thesis.

Mathematically stated, the purpose of the adaptation algorithm is to adaptively choose a set of coefficients $\mathbf{g}(n|n-1)$ to filter the input signal $\mathbf{u}(n)$, based on a knowledge of the inputs $\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n-1)$, and a set of desired symbols $d(1), d(2), \ldots d(n-1)$. The desired symbols are the desired outputs of the adaptive filter. In our case, therefore, they should theoritically be the transmitted symbols $s(1), s(2), \ldots, s(n-1)$. We may, however, only have probabilistic information about $s(n)$. Traditionally, the symbols are assumed known at the receiver, and this is the context in which algorithms like RLS are usually implemented.

While a detailed account of the recursive algorithms used in many applications (including equalization) may be found in [20], [31] and [42], among many others , we consider one very important adaptation algorithm, called the Recursive Least Squares (RLS) Algorithm.

2.3.1 Recursive Least Squares

One common approach to implementing equalizers is using the coefficients $\mathbf{g}(n|n-1)$ that are optimal under the Least Squares criterion. This is because it is the Maximum-Likelihood estimator under a known transmitted signal and a Gaussian observation noise assumption for time-invariant channels. Possibly more importantly, it works very well in practice, and an efficient recursive solution exists [20].

The least squares approach is to estimate a system $\mathbf{g}(n+1|n)$, given observations $\mathbf{u}(1), \ldots, \mathbf{u}(n)$ and desired signals $s(1), \ldots, s(n)$ so that we minimize a cost criterion given by:

$$J_n(\mathbf{g}) = \sum_{m=1}^{n} \lambda^{n-m} |s(m) - \mathbf{g}^H \mathbf{u}(m)|^2 \tag{2.12}$$

where $\lambda \leq 1$ is an exponential weighting factor, designed to give us tracking ability. More specifically, it limits the "averaging window" and thus allows the estimator to remain responsive to additional observations as they are received. λ, therefore accounts for the time-variance of the system. $\lambda = 1$ would imply infinite memory, and would be optimal for a time-invariant system. Any $\lambda < 1$ lets us track a dynamic

system.

It may be shown that the coefficient vector that minimizes this is given by

$$\mathbf{g}(n+1|n) = \mathbf{R}_{\mathbf{u}}^{-1}(n)\left(\sum_{m=1}^{n}\lambda^{n-m}\mathbf{u}(m)s^{*}(n)\right) \tag{2.13}$$

where $\mathbf{R}_{\mathbf{u}}(n)$ is the sample autocovariance matrix of the process, given by

$$\mathbf{R}_{\mathbf{u}}(n) = \sum_{m=1}^{n}\lambda^{n-m}\mathbf{u}(m)\mathbf{u}^{H}(m) \tag{2.14}$$

The RLS algorithm is a computationally efficient way to obtain the solution of (2.13). At each time n the algorithm computes the new coefficient vector $\hat{\mathbf{g}}(n)$ from the previous coefficient vector as

$$\mathbf{g}(n+1|n) = \mathbf{g}(n|n-1) + \mathbf{k}(n)(s(n) - \mathbf{g}^{H}(n|n-1)\mathbf{u}(n))^{*} \tag{2.15}$$

$\mathbf{k}(n)$ is the Kalman Gain vector, defined by

$$\mathbf{k}(n) = \mathbf{R}_{\mathbf{u}}^{-1}(n)\mathbf{u}(n) \tag{2.16}$$

We define the term $e(n) = s(n) - \mathbf{g}^{H}(n|n-1)\mathbf{u}(n) = s(n) - y(n)$ as the *innovation* at time n.

2.3.2 Hard-Decision Directed Adaptation

It is evident from Equations 2.13 and 2.15 that the RLS adaptation procedure depends on the transmitted symbols $s(n)$. Most conventional adaptation algorithnms have this requirement. In communication systems, though, we cannot have access to this information.

Thus the usual procedure in such cases is to first transmit a sequence of symbols which is known at the receiver end. Such a sequence is called the *training* or *pilot* sequence. This is used to train the equalizer. Following this, the equalizer is switched

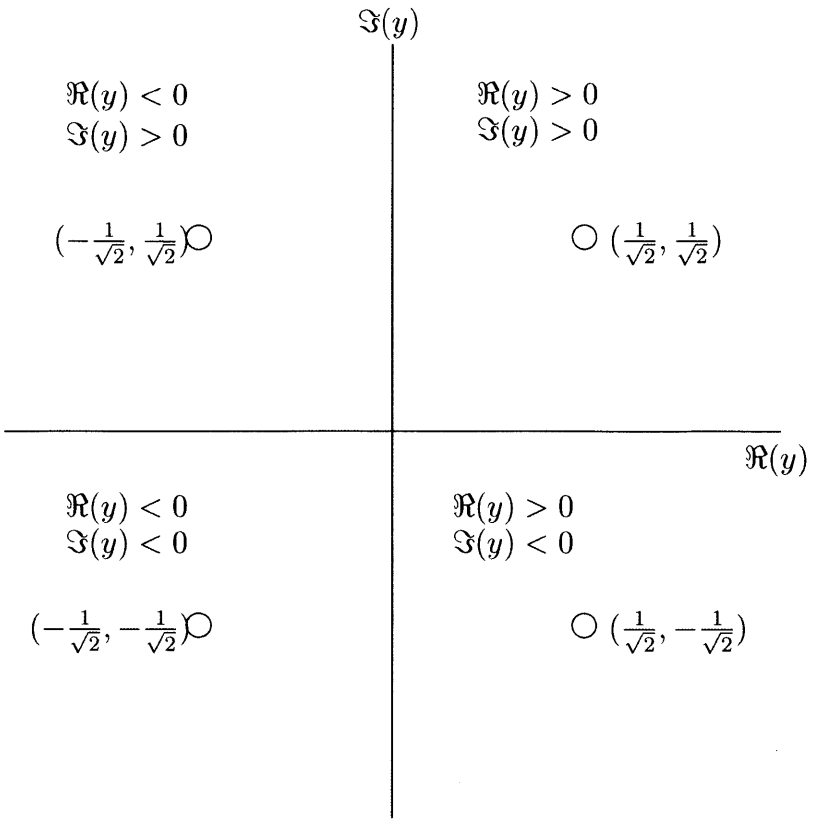


Figure 2-2: Q.P.S.K. Constellation and Output Decision Regions

into the so-called hard-decision directed mode.

In hard-decision directed mode, the equalizer maps the filter output $y(n)$ to the nearest constellation point. For example, Figure 2-2 shows how this is done with a Q.P.S.K. constellation and the corresponding decision regions on $y(n)$. This decision, which is denoted $d_h(n)$, is used in adaptation instead of the true transmitted symbol $s(n)$. That is to say, Equation 2.15 now becomes

$$\begin{aligned}\mathbf{g}(n+1|n) &= \mathbf{g}(n|n-1) + \mathbf{k}(n)(\Psi_h(y(n)) - \mathbf{g}^H(n|n-1)\mathbf{u}(n))^* \\ &= \mathbf{g}(n|n-1) + \mathbf{k}(n)(d_h(n) - \mathbf{g}^H(n|n-1)\mathbf{u}(n))^* \end{aligned} \tag{2.17}$$

where $\Psi_h(\cdot)$ represents the operation of the hard mapper. This is termed *hard-decision directed adaptation*, as the decisions made to be used in the adaptation process are hard decisions on the symbols. However, this procedure is suboptimal. We will intuitively see this in the next section, which will lead us to the statement of

the problem we seek to address in this thesis.

2.3.3 Soft Information in Adaptation

Consider, with reference to the Q.P.S.K. constellation of Figure 2-2, the points $y(n) = (0.1, 0.1)$ and $y(n) = (0.7, 0.7)$. Both of these in the hard-decision directed scheme would be mapped to the constellation point $(1/\sqrt{2}, 1/\sqrt{2})$. In fact, the innovation is larger in the first case and the algorithm makes a bigger correction to the coefficients.

However, there is larger likelihood that $y(n) = (0.1, 0.1)$ is a result of an error made in the filter output due to noise than the second case of $(0.7, 0.7)$. This would mean that we would be adapting to the wrong symbol, and as we make a bigger correction, we would be introducing a larger error into the coefficients computed for the next time instant. These errors are not correctible in the future, as we assume that the algorithm is purely recursive (i.e., we do not "go back in time" and correct errors). So in hard decision directed adaptation, we hope to make sufficiently small numbers of errors so that the coefficients are still reasonably correct.

It is evident, though, that we are losing information, so the question becomes whether there is a computationally efficient way of using that information in adaptation as well. We define as *soft information* any probabilistic information about the transmitted symbols that we can take advantage of in the adaptation process. The information about how likely it is that a particular filter output is due to an error is considered soft information. We will also interpret the definition to include priors or other information on the symbols that we may have available as a result of other components of the system.

2.4 The Problem Statement

We state the problems of interest in this thesis as follows:

- What *adaptation process* can take advantage of the soft information available to it? This question can be posed in 2 cases

  - When the equalizer is being run with no prior information about the symbols, i.e., the soft information consists only of quantities that can be computed from the filter output.
  - When some prior information is available to the equalizer about the transmitted symbols

- How much performance do we lose when we adapt in hard-decision directed mode, as opposed to adapting with known symbols? And how much can we improve this by a different algorithm?

- How do we design a practical turbo-type system from this algorithm using the priors that are always available in such a system?

In the following chapters, we address each of these issues. First we derive an adaptation algorithm that can utilize soft information, and show it to be optimal in an Expectation Maximization sense. We then analyse the performance of equalizers under hard-decision directed adaptation and the algorithm that we derive and develop understanding into their operation. Finally, we design and demostrate practical systems using this algorithm. In particular we design a turbo equalizer, which we have not introduced in this chapter. We will do so in Chapter 5.

Note that we thus focus on the effect of the adaptation algorithm on the performance of practical systems in which adaptation is required. While we attempt to keep the discussions as general as possible, we will specifically refer to the equalization setup during the derivations. We also do not focus on the effect of errors in other components of the system (specifically, the feedback filter. Error propagation in the feedback filter has been an emphasis of prior research- see, for instance, [14], [44] and [46]).

Chapter 3

The Recursive Expected Least Squares Algorithm

We introduced the problem of reliable adaptation when we do not know the transmitted symbols in section 2.4. In this chapter, we derive a recursive procedure, related to the RLS algorithm, which is specifically designed to perform adaptation under this constraint. This procedure will be designed to take advantage of whatever information is available at the receiver output, and is thus inherently more reliable than hard-decision directed equalization. The derivation will take the following steps [59]: first we look at how the problem of adaptation without knowledge of the transmitted signals can be viewed as an Expectation-Maximization type problem. This, along with analogies to the least-squares approach, will let us define a suitable cost criterion that does not depend directly on the tranmitted signals. Then we can approximate the solution to this cost criterion by a purely recursive algorithm, which will let us implement this algorithm practically with a small price to pay in terms of computational efficiency over conventional RLS.

3.1 The First Step: the Expectation-Maximization Algorithm

The Expectation Maximization (EM) Algorithm [10] is a powerful algorithm for performing Maximum-Likelihood (ML) Estimation of the parameters of a system when all the data about the system is not available. We will start from an ML framework, as this is easiest in the context of the EM algorithm, and relate this to a more computationally efficient least squares framework later.

What, however, does parameter estimation have to do with this system? At any time n, we could treat $\mathbf{g}(n+1|n)$ as a parameter of our system which maps $\mathbf{u}(n)$ into $y(n)$, an estimate of $s(n)$. The key insight here is that we can treat $s(n)$ as a random variable. This is a common approach in Bayesian estimation [28]. Thus, when we don't have access to $s(n)$, we have an *incomplete* set of observations, and wish to perform parameter estimation! This leads us to the intuition that we could somehow set up this problem in an EM framework.

We start by assuming that $\mathbf{g}(n+1|n)$ can be treated as a time-varying, deterministic (but unknown) parameter of a system characterized by a full set of observations $\mathbf{y}(n) = \begin{bmatrix} y(n) & y(n-1) & \cdots & y(1) \end{bmatrix}$ and $\mathbf{s}(n) = \begin{bmatrix} s(n) & s(n-1) & \cdots & s(1) \end{bmatrix}$. We note at this stage that the actual set of observations is $\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(1)$ and $\mathbf{s}(n)$, and using $\mathbf{y}(n)$ in the place of the $\mathbf{u}(n)$ vectors may lead to a loss of information, as the mapping between $\mathbf{u}(n)$ and $y(n)$ is not necessarily one-to-one. However, the structure of the direct adaptation equalizer is such that $\mathbf{u}(n)$ depends linearly on the channel coefficients, as opposed to the equalizer coefficients. The relationship between the equalizer coefficients and the channel coefficients is non-linear, even when the equalizer is optimal [38].

Hence, characterizing $\mathbf{u}(n)$ in terms of the parameter of interest $\mathbf{g}(n+1|n)$ is hard, making $\mathbf{u}(n)$ infeasible to work with. But why is working with $y(n)$ easier? This is an idea that falls out of our understanding of the EM procedure. With a knowledge of how the observed data depends on the parameter which we are trying to estimate, and a guess of the parameter, EM generates guesses about the data which is hidden.

The output of the equalizer itself is related to the transmitted signal $s(n)$ in a probabilistic manner, where the probabilities depend on the "parameter"- the equalizer coefficients. Using the current estimate of the parameter, $\mathbf{g}(n|n-1)$, we can generate the "observable data", $y(n)$, and use that in some probabilistic sense to generate estimates of the "hidden data", $s(n)$. This can then be used to compute a new estimate for the parameter, $\mathbf{g}(n+1|n)$.

To recap, therefore, the reason for using $y(n)$ rather than the true observed data $\mathbf{u}(n)$ is that we have a simple relationship between $y(n)$ and the parameter that we are trying to estimate. A similar relationship is not easy to find between $\mathbf{u}(n)$ and the equalizer coefficients. Thus, we allow the many-to-one transformation which depends on the coefficient vector, and use its output as the observed data. Note, however, that we do still have access to the equalizer input.

The procedure that we discussed sounds almost exactly like conventional EM, except that now the data and the parameters are all time-varying and we want to keep complexity low. We thus start the derivation by following a procedure similar to the derivation of the EM algorithm in [24]. The maximum likelihood estimate of the parameter $\mathbf{g}(n+1|n)$ at time n, given the complete data ($\mathbf{y}(n)$ and $\mathbf{s}(n)$) up to time n, would be given by

$$\begin{aligned} \mathbf{g}_{\text{ML, complete}}(n+1|n) &= \arg\max_{\mathbf{g}} \left[\log \mathrm{p}(\mathbf{s}(n), \mathbf{y}(n); \mathbf{g})\right] \\ &\triangleq \arg\max_{\mathbf{g}} l_n(\mathbf{g}) \end{aligned} \tag{3.1}$$

When we do not observe any of the symbols $s(n)$, a reasonable objective is to maximize the log-likelihood of the data we have, which is simply $\mathbf{y}(n)$. So we want to determine the maximum likelihood estimate of the parameters given the (incomplete) data, i.e.,

$$\begin{aligned} \mathbf{g}_{\text{ML, incomplete}}(n+1|n) &= \arg\max_{\mathbf{g}} \left[\log \mathrm{p}(\mathbf{y}(n); \mathbf{g})\right] \\ &\triangleq \arg\max_{\mathbf{g}} \bar{l}_n(\mathbf{g}) \end{aligned} \tag{3.2}$$

While directly computing this is hard, we observe that:

$$\begin{aligned}
\bar{l}_n(\mathbf{g}) &= \log \mathrm{p}(\mathbf{y}(n); \mathbf{g}) \\
&= \log \sum_{\mathbf{s}(n)} \mathrm{p}(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g}) \\
&= \log \sum_{\mathbf{s}(n)} \frac{\mathrm{p}(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g}) \mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n))}{\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n))} \\
&\geq \sum_{\mathbf{s}(n)} \mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n)) \log \frac{\mathrm{p}(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g})}{\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n))} \\
&\triangleq \mathcal{L}_n(\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n)), \mathbf{g}) \qquad (3.3)
\end{aligned}$$

where $\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n))$ is any valid probability distribution function , which we use for averaging. The function $\mathcal{L}_n(\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n)), \mathbf{g})$ is called the *auxilliary function* for the data up to time n. This is a standard definition in the derivation of the EM Algorithm (see, specifically, [24]).

The inequality in Equation 3.3 follows directly from the measure-theoritic form of Jensen's Inequality for concave functions, and it shows that the auxilliary function lower bounds the function we want to maximize. Now, like we do with the iterative EM algorithm, we can define a 2 stage process everytime we receive new data, as follows:

$$\hat{q}(\mathbf{s}(n)|\mathbf{y}(n)) = \arg\max_{\mathrm{q}(\cdot|\cdot)} \mathcal{L}_n(\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n)), \mathbf{g}(n|n-1)) \qquad \text{E-Step} \qquad (3.4)$$

$$\mathbf{g}(n+1|n) = \arg\max_{\mathbf{g}} \mathcal{L}_n(\hat{q}(\mathbf{s}(n)|\mathbf{y}(n), \mathbf{g}) \qquad \text{M-Step} \qquad (3.5)$$

At this stage we note the difference between the conventional EM algorithm and the procedure defined in Equations 3.4 and 3.5, which we call the *Recursive EM* procedure. In conventional EM, we have a single dataset. We repeatedly perform the E-step and M-step on the same dataset and it can be shown that the procedure converges to a local maximum of the auxilliary function [58].

However, the procedure we are defining as recursive EM operates only *once* on each dataset. The new dataset at time $n+1$ consists of the dataset at time n, plus

new data received at time $n+1$. That is, at each time n, we run the E-step and M-step once each, and call that our new estimate. Put another way, our time index is also our iteration index. The conventional EM algorithm, on the other hand only has an iteration index.

Note that there is nothing to stop us from performing multiple iterations at each time until convergence occurs. We choose not to do so for computational reasons, but we do not guarantee that the algorithm converges at each time.

Back to the derivation. To solve Equation 3.4, substitute

$$\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n)) = \mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n); \mathbf{g}(n|n-1)) \tag{3.6}$$

into Equation 3.3 . Then,

$$\begin{aligned}
&\mathcal{L}_n(\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n)), \mathbf{g}(n|n-1)) \\
&= \sum_{\mathbf{s}(n)} \mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n); \mathbf{g}(n|n-1)) \cdot \log \frac{\mathrm{p}(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g}(n|n-1))}{\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n); \mathbf{g}(n|n-1))} \\
&= \sum_{\mathbf{s}(n)} \mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n); \mathbf{g}(n|n-1)) \cdot \log \mathrm{p}(\mathbf{y}(n); \mathbf{g}(n|n-1)) \\
&= \log \mathrm{p}(\mathbf{y}(n); \mathbf{g}(n|n-1)) \cdot \sum_{\mathbf{s}(n)} \mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n); \mathbf{g}(n|n-1)) \\
&= \log \mathrm{p}(\mathbf{y}(n); \mathbf{g}(n|n-1)) \\
&= \bar{l}_n(\mathbf{g}(n|n-1))
\end{aligned} \tag{3.7}$$

But Equation 3.3 shows that for any $(\mathrm{q}(\cdot|\cdot), \mathbf{g})$ pair, $\mathcal{L}_n(\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n)), \mathbf{g}) \leq \bar{l}_n(\mathbf{g})$. The probability distribution used in Equation 3.7 causes equality to hold, implying that for $\mathbf{g} = \mathbf{g}(n|n-1)$, the probability distribution that maximizes the auxilliary function and hence solves Equation 3.4 is given by:

$$\begin{aligned}
\hat{q}(\mathbf{s}(n)|\mathbf{y}(n)) &= \arg\max_{\mathrm{q}(\cdot|\cdot)} \mathcal{L}_n(\mathrm{q}(\mathbf{s}(n)|\mathbf{y}(n)), \mathbf{g}(n|n-1)) \\
&= \mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n); \mathbf{g}(n|n-1))
\end{aligned} \tag{3.8}$$

Finally, Equation 3.5 can be replaced by a maximization of an expectation with respect to $q(\cdot|\cdot)$ of the complete log-likelihood instead of $\mathcal{L}(\cdot,\cdot)$ for any $q(\cdot|\cdot)$, as shown below,

$$\begin{aligned}
&\arg\max_{\mathbf{g}} \mathcal{L}_n(q(\mathbf{s}(n)|\mathbf{y}(n)), \mathbf{g}) \\
&= \arg\max_{\mathbf{g}} \sum_{\mathbf{s}(n)} q(\mathbf{s}(n)|\mathbf{y}(n)) \log \frac{p(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g})}{q(\mathbf{s}(n)|\mathbf{y}(n))} \\
&= \arg\max_{\mathbf{g}} \sum_{\mathbf{s}(n)} q(\mathbf{s}(n)|\mathbf{y}(n)) \log p(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g}) \\
&\qquad\qquad - \sum_{\mathbf{s}(n)} q(\mathbf{s}(n)|\mathbf{y}(n)) \log q(\mathbf{s}(n)|\mathbf{y}(n)) \\
&= \arg\max_{\mathbf{g}} \sum_{\mathbf{s}(n)} q(\mathbf{s}(n)|\mathbf{y}(n)) \log p(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g}) \\
&= \arg\max_{\mathbf{g}} \mathbb{E}_{q(\mathbf{s}(n)|\mathbf{y}(n))}[\log p(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g})] \qquad (3.9)
\end{aligned}$$

Combining Equations 3.8 and 3.9, we have a single equation that captures the results of the derivation so far:

$$\mathbf{g}(n+1|n) = \arg\max_{\mathbf{g}} \mathbb{E}_{p(\mathbf{s}(n)|\mathbf{y}(n);\mathbf{g}(n|n-1))}[\log p(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g})] \qquad (3.10)$$

To summarize, at each time n when a new input vector $\mathbf{u}(n)$ is received, we map it into a new observed data point $y(n)$ for the recursive EM algorithm using the current estimate of the parameter. Then, using the probability density function of the hidden data $\mathbf{s}(n)$ conditioned on $\mathbf{y}(n)$ and parametrized by the current coefficient vector, we find the expected log-likelihood of the complete data. Then, we perform *one* maximization of this over the parameter space to form the new estimate $\mathbf{g}(n+1|n)$, and then stop and wait for more data. Observe that this in itself is a partly recursive solution, because we update the estimate only when we get new data.

However, while we have defined what to maximize (and why) we have said nothing about how. Maximizing the expected log-likelihood in Equation 3.10 could be quite computationally intensive. Thus, we first change the log-likelihood based criterion

above to a least-squares type criterion. We can then derive a solution for it, which can be simplified to be recursive and causal.

3.2 The Expected Least Squares Cost Criterion

One possible way to think about the Least Squares Cost Criterion defined by Equation 2.12 is that it is Maximum Likelihood Estimation for Gaussian variables [28,31]. Put another way, the least squares sum is a sufficient statistic for parameter estimation with Gaussian variables in a fully observed system.

To see this, consider a time-invariant fully observed system (i.e. we observe both $\mathbf{u}(n)$ and $s(n)$. Assume $y(n) = s(n) + \eta(n)$, where $\eta(n)$ represents all the residual interference at the output of the equalizer, and $\eta(n)$ is Gaussian with mean 0 and variance σ_η^2. Then,

$$\begin{aligned}
\mathbf{g}_{\text{ML,Gaussian}} &= \arg\max_{\mathbf{g}} \left[\exp\left(-\sum_{m=1}^{n} \frac{|s(m) - \mathbf{g}^H \mathbf{u}(m)|^2}{2\sigma_\eta^2} \right) \right] \\
&= \arg\min_{\mathbf{g}} \left[\sum_{m=1}^{n} |s(m) - \mathbf{g}^H \mathbf{u}(m)|^2 \right] \\
&= \mathbf{g}_{\text{Least Squares}}
\end{aligned} \tag{3.11}$$

Now suppose that $s(n)$ does not depend on $\mathbf{g}(n+1|n)$ (which is reasonable), and that $y(n)$ is Gaussian given $s(n)$ (which, as we will see later is not a perfect assumption). Then it is evident that we can write Equation 3.10 as

$$\begin{aligned}
\mathbf{g}(n+1|n) &= \arg\max_{\mathbf{g}} \left[\mathbb{E}_{\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n);\mathbf{g}(n|n-1))} [\log \mathrm{p}(\mathbf{y}(n), \mathbf{s}(n); \mathbf{g})] \right] \\
&= \arg\min_{\mathbf{g}} \left[\mathbb{E}_{\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n);\mathbf{g}(n|n-1))} \left[\sum_{m=1}^{n} |s(m) - \mathbf{g}^H \mathbf{u}(m)|^2 \right] \right]
\end{aligned} \tag{3.12}$$

We now choose this as our cost criterion, i.e., rather than using the expected log-likelihood, we shall henceforth use the expected least squares criterion. This is our cost criterion *whether or not* the distributions are Gaussians. We merely use

Gaussianity to justify the least-squares approximation.

However, we make one modification to the cost criterion in Equation 3.12. As is usually done with least-squares for time-variant systems (see [20]) we introduce an exponential weighting factor $\lambda \leq 1$ to allow tracking of the time-variant features of the system (section 2.3.1). So the cost criterion we finally work with, which we term the *Expected Least Squares* cost criterion, is defined by

$$\bar{J}_n(\mathbf{g}) = \mathbb{E}_{\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n);\mathbf{g}(n|n-1))}\left[\sum_{m=1}^{n} \lambda^{n-m}|s(m) - \mathbf{g}^H\mathbf{u}(m)|^2\right] \tag{3.13}$$

We assume henceforth that the expectation is with respect to the distribution $\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n);\mathbf{g}(n|n-1))$ unless otherwise mentioned, and drop the subscript.

3.2.1 Solving the Expected Least Squares Criterion

A closed form solution exists for the minimum of $\bar{J}_n(\mathbf{g})$. To start with,

$$\begin{aligned}
\bar{J}_n(\mathbf{g}) &= \mathbb{E}\left[\sum_{m=1}^{n} \lambda^{n-m}|s(m) - \mathbf{g}^H\mathbf{u}(m)|^2\right] \\
&= \sum_{m=1}^{n} \lambda^{n-m}\mathbb{E}[|s(m) - \mathbf{g}^H\mathbf{u}(m)|^2] \\
&= \sum_{m=1}^{n} \lambda^{n-m}\mathbb{E}[(s(m) - \mathbf{g}^H\mathbf{u}(m))(s^*(m) - \mathbf{u}^H(m)\mathbf{g})]
\end{aligned} \tag{3.14}$$

To determine the minimum point of this function, one method we can use is to treat $\mathbf{g}$ and its Hermitian, $\mathbf{g^H}$, as 2 different variables. Then, we can write $\bar{J}_n(\mathbf{g}) \equiv \bar{J}_n(\mathbf{g}, \mathbf{g^H})$, and it can be shown that the minimum of this function with respect to $\mathbf{g}$ occurs at the point such that its partial differential with respect to $\mathbf{g^H}$ is 0 (details of this may be found in Appendix B of [20]). As $\mathbf{u}(m)$ is known for all m, we can simplify Equation 3.14 as follows:

$$\frac{\partial \bar{J}_n(\mathbf{g}, \mathbf{g^H})}{\partial \mathbf{g}^H} = \sum_{m=1}^{n} \lambda^{n-m}\mathbb{E}[\mathbf{u}(m)(s^*(m) - \mathbf{u}^H(m)\mathbf{g})] = \mathbf{0} \tag{3.15}$$

Now, we pretended that the data for the recursive EM procedure was $\mathbf{y}(n)$ rather than $\mathbf{u}(n)$ vectors, in order to be able to parametrize easily. However, we know the value of $\mathbf{u}(m)$ for all m- it is just the input to the equalizer. So there's no need to take an expectation on this- we're essentially conditioning on it here- so that the above becomes:

$$\sum_{m=1}^{n} \lambda^{n-m}(\mathbf{u}(m)d^*(m|n) - \mathbf{u}(m)\mathbf{u}^H(m)\mathbf{g}) = \mathbf{0} \tag{3.16}$$

where

$$\begin{aligned} d(m|n) &= \mathbb{E}_{\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n);\mathbf{g}(n|n-1))}[s(m)] \\ &= \sum_{\mathbf{s}(n)} s(m)\mathrm{p}(\mathbf{s}(n)|\mathbf{y}(n);\mathbf{g}(n|n-1)) \\ &= \sum_{s(m)} s(m) \sum_{\mathbf{s}(n)\backslash\{s(m)\}} \mathrm{p}(s(m), \mathbf{s}(n)\backslash\{s(m)\}|\mathbf{y}(n);\mathbf{g}(n|n-1) \\ &= \sum_{s(m)} s(m)\mathrm{p}(s(m)|\mathbf{y}(n);\mathbf{g}(n|n-1)) \\ &= \mathbb{E}[s(m)|\mathbf{y}(n);\mathbf{g}(n|n-1)] \end{aligned} \tag{3.17}$$

Rearranging Equation 3.16, we can write the closed form minimization to the ELS cost function as

$$\mathbf{g}(n+1|n) = \mathbf{R}_{\mathbf{u}}^{-1}(n)\left(\sum_{m=1}^{n} \lambda^{n-m}\mathbf{u}(m)d^*(m|n)\right) \tag{3.18}$$

where $\mathbf{R}_{\mathbf{u}}(n)$ is defined in Equation 2.14.

3.3 A Purely Recursive Approximation

The solution which is defined in Equation 3.18 cannot directly be written in a recursive form. This is because at each time n we need to compute the expectation of each of the past symbols given all the data upto and including n. When we get new data, therefore, we need to go back and update all the past "desired" values, $d(m|n), m \leq n$.

However, if we now constrain the system to be causal, so that we do not go back

and update the expectations of the past symbols, we have $\mathbf{g}(n+1|n)$ given by:

$$\begin{aligned}\mathbf{g}(n+1|n) &= \mathbf{R}_{\mathbf{u}}^{-1}(n)\sum_{m=1}^{n}\lambda^{n-m}\mathbf{u}(m)\mathbb{E}[s(m)|\mathbf{y}(m);\mathbf{g}(m|m-1)]^* \\ &= \mathbf{R}_{\mathbf{u}}^{-1}(n)\left(\sum_{m=1}^{n}\lambda^{n-m}\mathbf{u}(m)d^*(m)\right) \end{aligned} \tag{3.19}$$

where $d(m)$ is now defined by

$$d(m) = \mathbb{E}[s(m)|\mathbf{y}(m);\mathbf{g}(m|m-1)] \tag{3.20}$$

This is a sub-optimal approximation to Equation 3.18. However, the sequence $d(n)$ is causal. From a comparison of Equation 3.19 with Equations 2.13 and 2.15, it is then clear we can define a recursive solution like Equation 2.15 by replacing $s(n)$ with $d(n)$, and treating this as the new "desired sequence" of symbols. Hence, a recursive algorithm, for which we don't need to go back and update or perform matrix inversions is obtained, which we call the *Recursive Expected Least Squares* (RELS) Algorithm.

$$\mathbf{g}(n+1|n) = \mathbf{g}(n|n-1) + \mathbf{k}(n)(d(n) - \mathbf{g}^H(n|n-1)\mathbf{u}(n))^* \tag{3.21}$$

From a system point of view, this amounts to changing the decision device $\Psi(\cdot)$ of Figure 2-1 from a hard slicer to $d(n)$. But notice that $d(n)$ is simply the Bayes Least Squares estimate of $s(n)$ given the data $\mathbf{y}(n)$, parametrized by the current estimate of the underlying parameter of the distribution relating the symbol to the data $\mathbf{g}(n|n-1)$. This is an intuitively pleasing result, because it means replacing the unknown symbol with the statistically optimum estimate of $s(n)$ given the data and the current parameter estimate.

While the decision device has this satisfying result, we emphasize that it is not the output of the decision device itself that is of primary interest. We are concerned rather with what effect this has on the adaptation process. Our derivation has given us one possible way to perform this adaptation in the absence of $s(n)$.

3.4 Modelling the Probability Density Function

At the very outset, we note that computation of $d(n)$ may not be trivial. In the most general case, we would have to implement some form of MAP estimator to compute the value of $d(n)$ from the data, which itself might be computationally intensive. So let us look at the computation in more detail and attempt to simplify the computation. Define a dummy variable $s'(n)$ which is distributed as $s(n)$.

$$\begin{aligned} d(n) &= \mathbb{E}[s(n)|\mathbf{y}(n);\mathbf{g}(n|n-1)] \\ &= \sum_{s(n)} s(n)\mathrm{p}(s(n)|\mathbf{y}(n);\mathbf{g}(n|n-1)) \\ &= \sum_{s(n)} s(n)\frac{\mathrm{p}(\mathbf{y}(n)|s(n);\mathbf{g}(n|n-1))\mathrm{p}(s(n))}{\sum_{s'(n)}\mathrm{p}(\mathbf{y}(n)|s'(n);\mathbf{g}(n|n-1))\mathrm{p}(s'(n))} \\ &= \frac{\sum_{s(n)} s(n)\mathrm{p}(\mathbf{y}(n)|s(n);\mathbf{g}(n|n-1))\mathrm{p}(s(n))}{\sum_{s(n)}\mathrm{p}(\mathbf{y}(n)|s(n);\mathbf{g}(n|n-1))\mathrm{p}(s(n))} \end{aligned} \tag{3.22}$$

We assumed in Equation 3.22 that the symbol probabilities don't depend on the system parameters. Further, we assumed in section 2.2 that the system is causal, so the data at times $m < n$ is independent of $s(n)$. Then,

$$\begin{aligned} \mathrm{p}(\mathbf{y}(n)|s(n);\mathbf{g}(n|n-1)) &= \mathrm{p}(y(n),\mathbf{y}(n-1)|s(n);\mathbf{g}(n|n-1)) \\ &= \mathrm{p}(y(n)|\mathbf{y}(n-1),s(n);\mathbf{g}(n|n-1))\times \\ &\quad \times \mathrm{p}(\mathbf{y}(n-1)|s(n);\mathbf{g}(n|n-1)) \\ &= \mathrm{p}(y(n)|\mathbf{y}(n-1),s(n);\mathbf{g}(n|n-1))\mathrm{p}(\mathbf{y}(n-1);\mathbf{g}(n|n-1)) \end{aligned} \tag{3.23}$$

Substituting equation 3.23 in equation 3.22, and observing that the second term does not depend on $s(n)$, we have

$$d(n) = \frac{\sum_{s(n)} s(n)\mathrm{p}(y(n)|\mathbf{y}(n-1),s(n);\mathbf{g}(n|n-1))\mathrm{p}(s(n))}{\sum_{s(n)}\mathrm{p}(y(n)|\mathbf{y}(n-1),s(n);\mathbf{g}(n|n-1))\mathrm{p}(s(n))} \tag{3.24}$$

What we need is a suitable model for $\mathrm{p}(y(n)|\mathbf{y}(n-1),s(n);\mathbf{g}(n|n-1))$. This

can be computed offline, so there is minimal increase in the amount of computation required. However, it is not yet clear how to model the probability density function. Our analysis in Chapter 4 will start providing some insight into what the right ways are of doing so. However, for now, we choose a simple but rich model that can provide insight into the equalizer operation and also provide reasonable practical performance.

3.4.1 A Simple Case: Gaussian Residual Noise

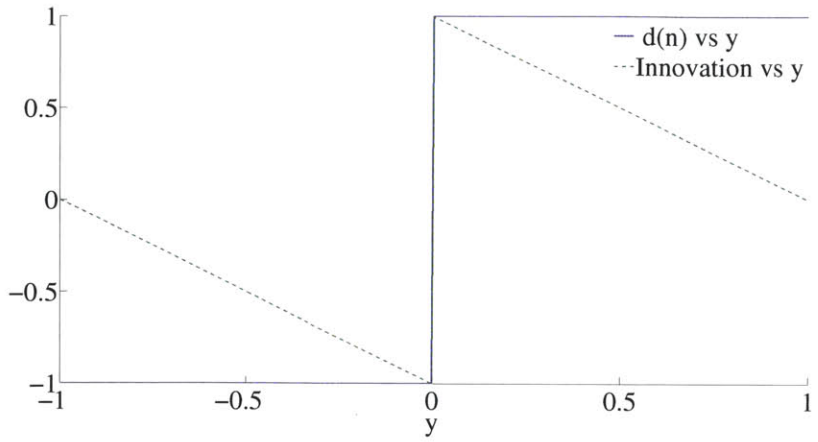
Consider a simple but illustrative example in a BPSK system. Assume that $y(n)$ is independent of $\mathbf{y}(n-1)$ given $s(n)$, Let $y(n) = s(n) + \eta(n)$, where $s(n) \in \{-1, 1\}$ and $\eta(n)$ in general depends on $\mathbf{g}(n|n-1)$. Physically, $\eta(n)$ corresponds to residual ISI, plus the residual noise after passing the input through the system $\mathbf{g}(n|n-1)$.

If the residual ISI has a large number of small random terms, and these are nearly independent of each other, we can, by an application of the central limit theorum, assume that it is Gaussian with zero mean. We also assume that the residual noise is Gaussian with zero mean, so that $\eta(n)$ is Gaussian with zero mean.

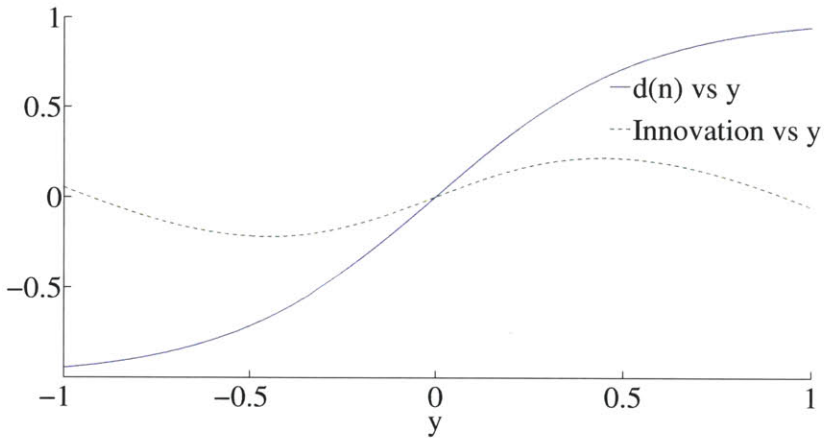
Then our model becomes $\mathrm{p}(y(n)|\mathbf{y}(n-1), s(n); \mathbf{g}(n|n-1)) \approx \mathcal{N}(s(n), \sigma_\eta^2)$. Under these circumstances, and letting $p = \mathbb{P}(s(n) = 1)$,

$$\begin{aligned} d(n) &= \frac{p \exp\left(-\frac{(y(n)-1)^2}{2\sigma_\eta^2}\right) - (1-p)\exp\left(-\frac{(y(n)+1)^2}{2\sigma_\eta^2}\right)}{p \exp - \left(\frac{(y(n)-1)^2}{2\sigma_\eta^2}\right) + (1-p)\exp\left(-\frac{(y(n)+1)^2}{2\sigma_\eta^2}\right)} \\ &= \frac{1 - \frac{1-p}{p}\exp\left(-\frac{2y(n)}{\sigma_\eta^2}\right)}{1 + \frac{1-p}{p}\exp\left(-\frac{2y(n)}{\sigma_\eta^2}\right)} \\ &= -1 + \frac{2}{1 + \frac{1-p}{p}\exp\left(-\frac{2y(n)}{\sigma_\eta^2}\right)} \end{aligned} \tag{3.25}$$

The variance σ_η^2, of course, should theoritically be time-variant and depend on the parameter $\mathbf{g}(n|n-1)$. However, if we assume that the equalizer tracks the channel reasonably well, then the statistics of the output of the equalizer can be assumed to be stationary (more will be said about this in Chapter 4). Then, the variance is a constant, and can be estimated during the training period using an estimator similar



(a) Hard Decision-Directed Adaptation



(b) Soft Decision-Directed Adaptation with Gaussian Model for PDF

Figure 3-1: Plot of the "desired symbols" $d(n)$ and innovation as a function of equalizer filter output y for BPSK system

to the one proposed in [49].

Plots of the decision functions and innovations, which, as we recall is defined as $e(n) = \Psi(y(n)) - y(n)$ are shown in Figure 3-1 for hard decisions and the decision function specified by Equation 3.25 in a particular BPSK system, where the symbols are assumed to be equiprobable. Here, the variance is found to be $\sigma_\eta^2 = 0.55$.

Interestingly, for this particular case, a decision function similar to the one used in [3], [15] and [33] in the context of blind equalization is obtained. The decision function for the symbol used in adaptation becomes a sigmoid curve. This is unsurprising given the assumptions made, but it is intuitively satisfying that our general solution reduces

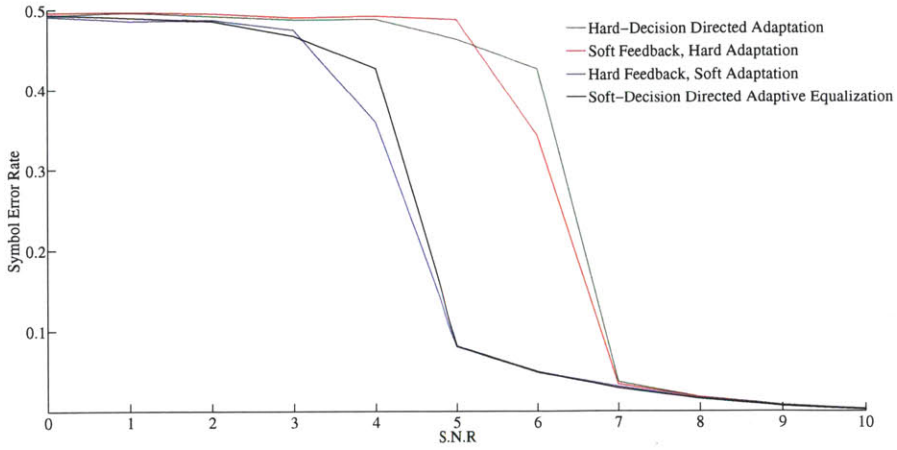


Figure 3-2: Results of Implementation of the Equalizer with Simulated Channel

to this solution when similar assumptions are made.

For QPSK and other multidimensional signalling schemes, a simple extension of this concept is used. We assume that the random variable $y(n)|s(n)$ is circularly symmetric complex normal about $s(n)$, with the additional constraint that the covariance between the real and imaginary parts is 0.

The equalizer which uses these soft decisions (with the Gaussianity assumption) $d(n)$ in both the adaptation process and the feedback filter is what we term the *Soft-Decision Directed Adaptive Equalizer*, as it essentially replaces hard decisions on the filter output by soft decisions based on the sigmoid curve of Figure 3-1. This is what we will use for practical implementation in the following chapters.

3.4.2 Simulation Results

As a verification of the performance of the soft-decision directed adaptive equalizer presented in this section, we consider Figure 3-2. This is a random walk channel of length 5 with AWGN, and all symbols are assumed equiprobable. The results presented are for BPSK signals, and the metric is Symbol Error Rate, as a function of channel SNR.

Looking at the hard decision directed equalizer and the soft-decision directed adaptive equalizer. It is evident that the soft-adaptive equalizer gives us about 2dB improvement for this very simple channel. Evidently, using soft decisions in the

adaptation process, even with the simple Gaussian model, provides an improvement in performance.

However, note that in the soft-decision adaptive equalizer, both the feedback filter and the adaptation algorithm use soft symbols. This raises the question of whether the improvement observed is due to the prevention of error propagation in the feedback filter or in the adaptation process.

Figure 3-2 also answers this question. It is clear that when we allow error propagation in the feedback filter (labelled "Hard Feedback" in the figure) but not in the adaptation process ("Soft Adaptation") , we end up with close to the same performance as the soft-decision directed adaptive equalizer, which uses soft decisions in both. Similarly, using soft-decisions in only the feedback filter but not in adaptation does not improve performance significantly from using hard-decision directed adaptation.

This indicates that the adaptation procedure is indeed important in the functioning of the adaptive equalizer, and that the soft-decision directed adaptation algorithm that we have derived does indeed provide an improvement even with simple probabilistic models.

3.4.3 SPACE08 Data and Implementations

As a test in a more realistic scenario, the hard-decision directed DFE and the soft-decision directed equalizer were tested with uncoded data transmitted during the SPACE08 Acoustic Communication Experiments. The data consists of 89 repetitions of a length-4095 pseudorandom binary sequence, which was BPSK encoded. For the test, we used a stream of data corresponding to the first 11000 symbols received. At the receiver end equalizers with fractional sampling rates of 1 and 2 samples/symbol were employed. A training period of 1000 symbols was employed to initialize the algorithms.

The soft adaptive equalizer was run with worst-case priors (all symbols are equiprobable). We also tested the algorithm with "pseudorandom" priors. This was in order to provide a sense of how it might perform if it were given priors (which themselves

have some error) from a different component of the system.

The results of the test are shown in Figure 3-3. The results indicate that as a standalone equalizer in the decision-directed mode (when the symbols are unknown), the proposed algorithm improves on using hard-decision directed mode, even with the simple Gaussian PDF assumption that we made. Furthermore, there is a failure mode in the RLS algorithm which causes large error rates at low SNR, which exists for both the hard and soft decision-directed modes. However, the threshold SNR at which this occurs is lower for the soft decision-directed mode. We will analyze this failure mode further in Chapter 4.

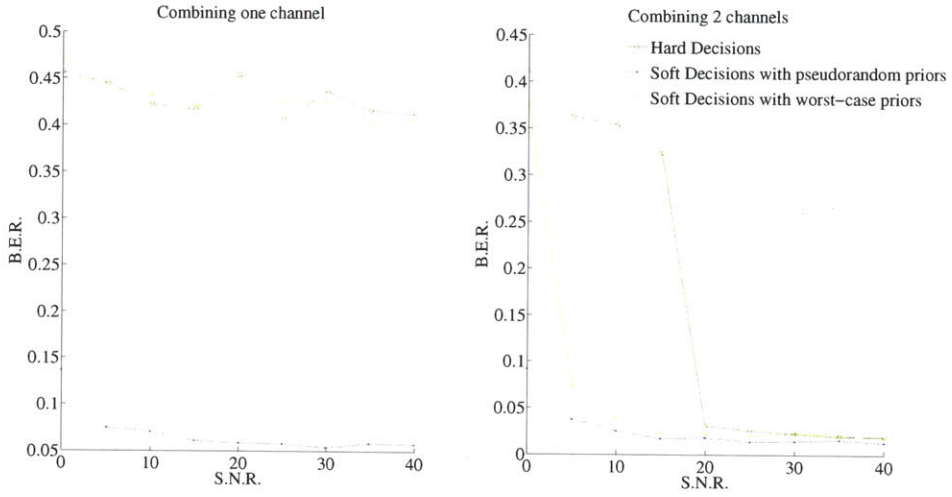
Finally, as we would expect, the soft-adaptation equalizer performs even better when we use the pseudorandomly generated priors, indicating that it may have utility in turbo equalizer systems (see Chapter 5). However, it should be noted in this case that the priors on the symbols tend to be better than in typical practical systems, so the performance estimated may be overly optimistic.

Recap and Looking Ahead

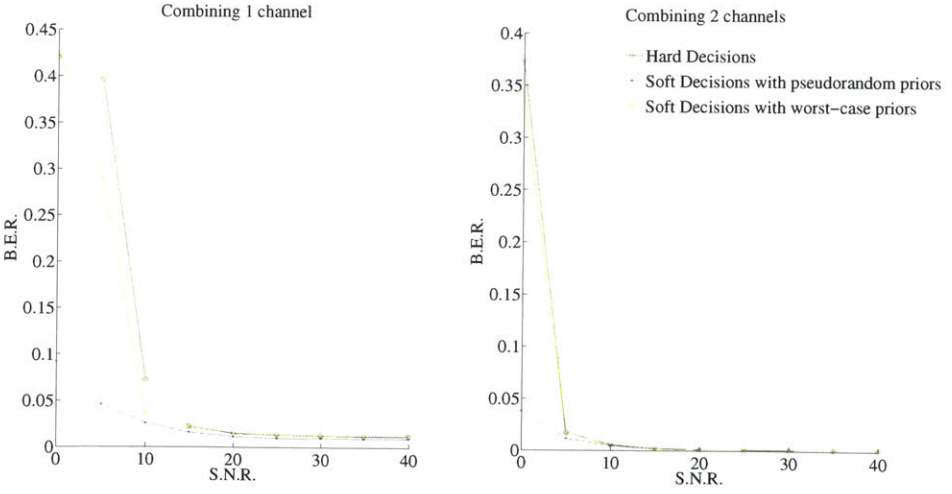
To recap what we did in this chapter, we started by asking the question of how to adapt the coefficients of an equalizer when the desired symbols are unknown. We started with the Maximum-Likelihood approach with the EM algorithm, jointly estimating at each time instant the symbols upto that time, and the equalizer coefficient vector. We specialized this by performing one iteration of the EM approach at each time step, which we called the "recursive EM" procedure.

We then changed the cost criterion of optimization from the log-likelihood, which we may not have access to and is hard to work with, to the least squares criterion. As justification, we used the fact that the two are identical for Gaussian variables, and that least squares approaches work well in practice. This is the "Expected Least Squares" cost criterion.

We then found a closed form solution for the expected least squares cost criterion. We imposed causality on the solution, as an additional way to reduce complexity, and



(a) 1 Sample/Symbol



(b) 2 Samples/Symbol

Figure 3-3: Equalizers Tested with SPACE08 Data

showed that the resulting adaptation procedure is a simple algorithm for adaptation.

Finally, we used the further approximations that the output of the equalizer filter is a stationary Gaussian process when conditioned on the transmitted symbols to design a particular decision device that is of practical use. We termed the standalone equalizer designed based on this practical assumption the "Soft-Decision Directed Adaptive Equalizer". We looked at the error rates of this equalizer in simulation with a random walk channel to convince ourselves that there was indeed some improvement in performance and that we were seeing this improvement due to adaptation. Finally, the results of implementation using the SPACE08 data were presented.

However, how much improvement can we expect by using this cost criterion over Hard-Decision Directed Adapation? And how much worse is Hard-Decision Directed Adaptation than training mode? Does it always converge to a reasonable solution or are there situations in which it fails, and if so when and why does it occur? These are the problem which we look to address in the next chapter.

Chapter 4

Performance Analysis

All models in this chapter are purely fictional. Any resemblence to any real system is purely coincidental

-Sergio Verdu

In this chapter we analyze the performance of the direct form adaptive equalizer using the RLS algorithm under decision directed adaptation and of the soft-decision directed equalizer derived in Chapter 3. The analysis is fairly complicated, so we consider the simplest possible channel models with only a few equalizer taps and with BPSK signaling. So, we start by revisiting the assumptions made in Section 2.2.

4.1 Assumptions for Analysis

For the purpose of this analysis, we update the system model represented in Figure 2-1. The first assumption we make is that the channel is time-invariant and *known for the purposes of analysis.* We also assume that the feedback filter receives the true transmitted signals, i.e., the feedback filter input is not a function of the filter outputs and the errors made. All the channels and equalizer coefficients are at symbol spacing. This leads us to the modified system model of Figure 4-1. Only the relevant parts of the system are shown, so we have taken away the output stage. We assume that the system is BPSK and all the channel and equalizer coefficients are real for simplicity.

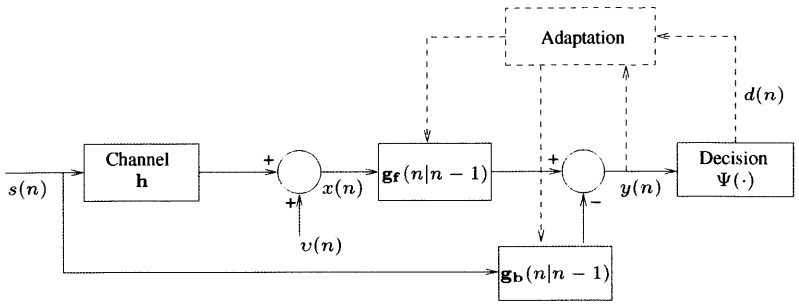


Figure 4-1: Simplified System Model for Analysis

So, we can replace all Matrix Hermitians with Transposes in this chapter, although we continue to use Hermitians for the sake of generality where it is not critical to differentiate the two. We recall at this time that various quantities have been defined in Table 2.2, and we reuse these without redefinition.

The channel is a fixed and known vector, denoted by $\mathbf{h} \in \mathbb{R}^M$. Then, given the number of feedforward and feedback taps, the channel matrix (Equation 2.10) can easily be defined. For example if we have a 2-tap channel given by, say $\mathbf{h} = \begin{bmatrix} 1 & 0.5 \end{bmatrix}^T$, and we use a 2-tap feedforward filter and 1-tap feedback filter, then the channel matrix can be written as:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 1 \\ 0 & 0.5 & 0 \end{bmatrix} \tag{4.1}$$

To repeat Equation 2.11, we can write the input to the equalizer as

$$\mathbf{u}(n) = \mathbf{H}^H(n)\mathbf{s}_P(n) + \boldsymbol{\upsilon}(n) \tag{4.2}$$

where we define

$$\boldsymbol{\upsilon}(n) = \begin{bmatrix} \upsilon(n) \\ \upsilon(n-1) \\ \vdots \\ \upsilon(n - \overline{N_f - 1}) \\ \mathbf{0}_{N_b \times 1} \end{bmatrix} \tag{4.3}$$

Here $v(n)$ is defined as the channel noise at time n. $v(n)$ is assumed to be stationary, zero-mean, Gaussian noise, which is not necessarily white. Thus, $\boldsymbol{v}(n)$ is also stationary, zero-mean and Gaussian, and we denote its covariance matrix by $\mathbf{S} = \mathbb{E}[\boldsymbol{v}(n)\boldsymbol{v}^H(n)]$. Once again, it would be sufficient to use Transposes in all the above as we have assumed that the channel and signals are all real. We have just used Hermitians for generality.

Also, $\mathbf{s}_P(n)$ is given by

$$\mathbf{s}_P(n) = \begin{bmatrix} s(n) \\ s(n-1) \\ \vdots \\ s(n-\overline{P-1}) \end{bmatrix} \tag{4.4}$$

Since the channel is time-invariant, we have a Minimum Mean Square Error solution for the equalizer coefficients [20]. We denote this solution by $\mathbf{g}_0$, and it is given by:

$$\mathbf{g}_0 = \mathbf{R}_\mathbf{u}^{-1}\mathbf{r}_{\mathbf{u}s*} \tag{4.5}$$

where $\mathbf{R}_\mathbf{u}$ is the covariance matrix of the input $\mathbf{u}(n)$, which is given, in the above case of a time-invariant known channel with known noise statistics, by

$$\begin{aligned} \mathbf{R}_\mathbf{u} &= \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)] \\ &= \mathbf{H}\mathbf{H}^H + \mathbf{S} \end{aligned} \tag{4.6}$$

and $\mathbf{r}_{\mathbf{u}s*}$ is the cross-correlation between the between the input to the equalizer and the transmitted symbols, given by

$$\begin{aligned} \mathbf{r}_{\mathbf{u}s*} &= \mathbb{E}[\mathbf{u}(n)s(n)] \\ &= \mathbf{H}^H \begin{bmatrix} 1 \\ \mathbf{0}_{(P-1)\times 1} \end{bmatrix} \end{aligned} \tag{4.7}$$

Recall from Table 2.2 that $P = \max(N_f + M - 2, N_b)$. Also, we are going to assume that $\mathbf{R_u}$ and $\mathbf{R_u}(n)$ are non-singular.

In [38], the performance of the LMMSE, Channel Estimate-DFE and passive time-reversal equalizers was compared, with the soft-decision error in the channel estimate being taken into account. However, we now attempt to define a framework in which the performance of the direct-form adaptive equalizer can be analysed when the "desired symbols" for adaptation are not exactly what the true symbols are in the underlying system. So, we are interested in the effect of bad decisions on the adaptation process.

4.2 The RLS Update Equation, Revisited

Consider some "desired" symbol sequence, $d(n)$. In training mode, $d(n) = s(n)$. For decision-directed mode in BPSK, $d(n) = \text{sign}(y(n))$, where $\text{sign}(x)$ represents the signum function. As we have shown in Chapter 3, in the RELS algorithm, we use $d(n) = \mathbb{E}[s(n)|\mathbf{y}(n); \mathbf{g}(n|n-1)]$ which, when the equalizer output is taken to be Gaussian, can be written (Section 3.4.1) as a sigmoid function of $y(n)$. Thus, in general we write $d(n) \equiv d(n, \mathbf{y}(n), \mathbf{g}(n|n-1), s(n))$

Now, the RLS update equation from Equation 2.15 (note that RELS can be substituted below everywhere and there's no difference, as the only change is the decision function), can be written for this more general desired symbol sequence, as:

$$\begin{aligned}
\mathbf{g}(n+1|n) &= \mathbf{g}(n|n-1) + \mathbf{R_u}^{-1}(n)\mathbf{u}(n)(d(n) - \mathbf{g}^H(n|n-1)\mathbf{u}(n))^* \\
&= \mathbf{g}(n|n-1) + \mathbf{R_u}^{-1}(n)\mathbf{u}(n)(d^*(n) - \mathbf{u}^H(n)\mathbf{g}(n|n-1)) \\
&= (\mathbf{I} - \mathbf{R_u}^{-1}(n)\mathbf{u}(n)\mathbf{u}^H(n))\mathbf{g}(n|n-1) + \mathbf{R_u}^{-1}(n)\mathbf{u}(n)d^*(n) \qquad (4.8)
\end{aligned}$$

Now, we make the following approximations for analysis. For a large enough n, we can apply the Law of Large Numbers [17] to $\mathbf{R_u}(n)$ and say that it is close to its

own expectation. So,

$$\begin{aligned}
\mathbf{R_u}(n) &\approx \mathbb{E}[\mathbf{R_u}(n)] \\
&= \mathbb{E}\left[\sum_{m=1}^{n} \lambda^{n-m}\mathbf{u}(m)\mathbf{u}^H(m)\right] \\
&= \sum_{m=1}^{n} \lambda^{n-m}\mathbb{E}\left[\mathbf{u}(m)\mathbf{u}^H(m)\right] \\
&= \mathbf{R_u}\sum_{m=1}^{n} \lambda^{n-m} \\
&\approx \frac{1}{1-\lambda}\mathbf{R_u} \quad \text{(for large } n\text{)}
\end{aligned} \tag{4.9}$$

Also, we make the following " direct-averaging" type assumption [20] on the matrix $\mathbf{u}(n)\mathbf{u}^H(n)$, and assume that the variance of the matrix is not too large. Thus we can replace this matrix by its expectation, and expect that over time the variances balance themselves out. So we use

$$\mathbf{u}(n)\mathbf{u}^H(n) \approx \mathbf{R_u} \tag{4.10}$$

Plugging Equations 4.9 and 4.10 into 4.8, we get

$$\mathbf{g}(n+1|n) \approx \lambda\mathbf{g}(n|n-1) + (1-\lambda)\mathbf{R_u}^{-1}\mathbf{u}(n)d^*(n) \tag{4.11}$$

We'll analyze the performance based on this equation.

4.3 The Distribution of the Coefficient Vector

We start by attempting to find the steady-state distribution of the coefficient vector, i.e., the probability distribution function of $\mathbf{g}(n+1|n)$ as $n \to \infty$. We denote this steady state PDF by $\mathrm{p}_{\mathbf{g}}^{(\infty)}$.

Consider the joint distribution of $(\mathbf{g}(n+1|n), \mathbf{g}(n|n-1))$. From Bayes' theorem

it is evident that

$$\mathrm{p}_{\mathbf{g}(n+1|n)}(\mathbf{g}) = \int_{\mathbb{R}^N} \mathrm{p}_{\mathbf{g}(n+1|n)|\mathbf{g}(n|n-1)}(\mathbf{g}|\mathbf{g}')\mathrm{p}_{\mathbf{g}(n|n-1)}(\mathbf{g}')\, d\mathbf{g}' \tag{4.12}$$

The integral is over $\mathbb{R}^N$ as we have assumed the system is real. At the steady state, the distributions of $\mathbf{g}(n+1|n)$ and $\mathbf{g}(n|n-1)$ become the same, and we can write the equation relating the 2 quantities as

$$\mathrm{p}_{\mathbf{g}}^{(\infty)}(\mathbf{g}) = \int_{\mathbb{R}^N} \mathrm{p}_{\mathbf{g}(n+1|n)|\mathbf{g}(n|n-1)}(\mathbf{g}|\mathbf{g}')\mathrm{p}_{\mathbf{g}}^{(\infty)}(\mathbf{g}')\, d\mathbf{g}' \tag{4.13}$$

Thus, $\mathrm{p}_{\mathbf{g}}^{(\infty)}(\cdot)$ is the solution to a homogenous Fredholm integral equation of the second kind [36], whose Kernal function is the transition distribution of the RLS algorithm with the update equation given by Equation 4.11. In other words, from Equation 4.11, we can form a probabilistic map of the transition probabilities of the coefficient vector from time n to $n+1$. This process is Markov in a continuous N-dimensional state space, so the steady state is given by the solution to the Fredholm equation in Equation 4.13.

Discretizing this into a matrix notation provides more insight into the operation of the system. We will explain this further in the next section, but we start by attempting to solve the equation directly in a simple case.

4.3.1 The Training Mode Probability Kernal and Approximating It

The solution to the Fredholm equation (and in fact, even whether a closed form solution exists), depends on the form of the transition probability kernal function, $\mathrm{p}_{\mathbf{g}(n+1|n)|\mathbf{g}(n|n-1)}(\mathbf{g}|\mathbf{g}')$. We will start by determining the functional form of the probability kernal for an equalizer when $d(n) = s(n)$, i.e., in training mode. This will provide us hints on how to set up the system for more complicated systems.

In the BPSK training case, Equation 4.11 can be written as

$$\mathbf{g}(n+1|n) \approx \lambda \mathbf{g}(n|n-1) + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{u}(n)s(n) \tag{4.14}$$

Since the noise $\boldsymbol{\upsilon}(n) \sim \mathcal{N}(\mathbf{0}, \mathbf{S})$, $\mathbf{u}(n)$ conditioned on $\mathbf{s}_P(n)$ is also normally distributed with a mean $\mathbf{H}^T\mathbf{s}_P(n)$ and covariance matrix $\mathbf{S}$. So the random variable $\mathbf{v}(n) = \mathbf{u}(n)s(n)$ has a normal distribution as well, conditioned on $\mathbf{s}_P(n)$.

$$\mathbf{v}(n)|\mathbf{s}_P(n) \sim \mathcal{N}(\mathbf{H}^T\mathbf{s}_P(n), \mathbf{S}) \tag{4.15}$$

Note that we have used the fact that the statistics of the noise do not change whether the noise term is multiplied by 1 or by -1, i.e., the PDF of $\boldsymbol{\upsilon}(n)s(n)$ is the same as the PDF of $\boldsymbol{\upsilon}(n)$ conditioned on all $s(n)$.

Assuming $1, -1$ are equiprobable at all times, we can thus write the PDF of $\mathbf{v}(n)$ as

$$\mathbf{v}(n) \sim \frac{1}{2^P} \sum_{\mathbf{s}\in\{-1,1\}^P} \mathcal{N}(\mathbf{H}^T\mathbf{s}, \mathbf{S}) \tag{4.16}$$

Thus, the random variable $\mathbf{w}(n) = (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{u}(n)s(n)$ has a distribution given by

$$\mathbf{w}(n) \sim \frac{1}{2^P} \sum_{\mathbf{s}\in\{-1,1\}^P} \mathcal{N}((1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{H}^T\mathbf{s}, (1-\lambda)^2\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{S}\mathbf{R}_{\mathbf{u}}^{-1}) \tag{4.17}$$

as linear combinations of Gaussian mixtures are Gaussian mixtures.

This gives us the PDF of the conditional kernal in this case, which is given by

$$\mathrm{p}_{\mathbf{g}(n+1|n)|\mathbf{g}(n|n-1)}(\mathbf{g}|\mathbf{g}') = \frac{1}{2^P} \sum_{\mathbf{s}\in\{-1,1\}^P} \mathcal{N}(\mathbf{g}; \lambda\mathbf{g}' + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{H}^T\mathbf{s}, (1-\lambda)^2\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{S}\mathbf{R}_{\mathbf{u}}^{-1}) \tag{4.18}$$

This immediately presents a problem, as the kernal is non-separable. The closed form of the Fredholm integral equation exists for separable kernals, and is hard to compute for the non-separable form of the kernal that we have obtained. Thus, even in the simplest possible case when the transmitted symbols are known, we cannot solve exactly for the steady-state PDF of the coefficient vector.

4.4 Predicting the Mean Behaviour

It is possibly to simplify the problem significantly if we choose only to look at the behaviour of the mean of the coefficients. We expect the coefficients to be clustered about the mean, so if we knew how the mean behaved, we could develop some insight into the behaviour of the system.

We define $\boldsymbol{\mu}_{\mathbf{g}}(n+1|n) = \mathbb{E}[\mathbf{g}(n+1|n)]$ and look at the PDF of the mean $\mathrm{p}_{\boldsymbol{\mu}_{\mathbf{g}}(n+1|n)}(\boldsymbol{\mu})$, just as we did in the previous section for $\mathbf{g}$. However, the PDF of evolution of the mean is significantly simpler than that for the coefficient vector.

For example, consider Equation 4.14 in training mode. Taking the expectations on both sides,

$$\begin{aligned}
\mathbf{g}(n+1|n) &\approx \lambda \mathbf{g}(n|n-1) + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{u}(n)s(n) \\
\Rightarrow \mathbb{E}[\mathbf{g}(n+1|n)] &= \lambda \mathbb{E}[\mathbf{g}(n|n-1)] + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbb{E}[\mathbf{u}(n)s(n)] \\
&= \lambda \mathbb{E}[\mathbf{g}(n|n-1)] + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{r}_{\mathbf{u}d^*} \\
\Rightarrow \boldsymbol{\mu}_{\mathbf{g}}(n+1|n) &= \lambda \boldsymbol{\mu}_{\mathbf{g}}(n|n-1) + (1-\lambda)\mathbf{g_0} \qquad (4.19)
\end{aligned}$$

Where $\mathbf{g_0}$ was defined in Equation 4.5 as the MMSE equalizer coefficients. With $\delta_x(y)$ defined as the Dirac impulse function of y at x, we have,

$$\mathrm{p}_{\boldsymbol{\mu}_{\mathbf{g}}(n+1|n)|\boldsymbol{\mu}_{\mathbf{g}}(n|n-1)}(\boldsymbol{\mu}|\boldsymbol{\mu}') = \delta_{\boldsymbol{\mu}}(\lambda\boldsymbol{\mu}' + (1-\lambda)\mathbf{g_0}) \qquad (4.20)$$

We now apply compute the steady state probability density function of $\boldsymbol{\mu}_{\mathbf{g}}(n+1|n)$, denoted by $\mathrm{p}^{(\infty)}_{\boldsymbol{\mu}_{\mathbf{g}}}(\cdot)$ by applying Equation 4.13, as follows:

$$\begin{aligned}
\mathrm{p}^{(\infty)}_{\boldsymbol{\mu}_{\mathbf{g}}}(\boldsymbol{\mu}) &= \int_{\mathbb{R}^N} \mathrm{p}_{\boldsymbol{\mu}_{\mathbf{g}}(n+1|n)|\boldsymbol{\mu}_{\mathbf{g}}(n|n-1)}(\boldsymbol{\mu}|\boldsymbol{\mu}')\mathrm{p}^{(\infty)}_{\boldsymbol{\mu}_{\mathbf{g}}}(\boldsymbol{\mu}')\, d\boldsymbol{\mu}' \qquad (4.21) \\
&= \int_{\mathbb{R}^N} \delta_{\boldsymbol{\mu}}(\lambda\boldsymbol{\mu}' + (1-\lambda)\mathbf{g_0})\mathrm{p}^{(\infty)}_{\boldsymbol{\mu}_{\mathbf{g}}}(\boldsymbol{\mu}')\, d\boldsymbol{\mu}' \qquad (4.22)
\end{aligned}$$

It is easy to verify that $\mathrm{p}^{(\infty)}_{\boldsymbol{\mu}_{\mathbf{g}}}(\boldsymbol{\mu}) = \delta_{\boldsymbol{\mu}}(\mathbf{g_0})$ is a solution to Equation 4.22. This result is not in the least surprising. It shows that in training mode, the steady state

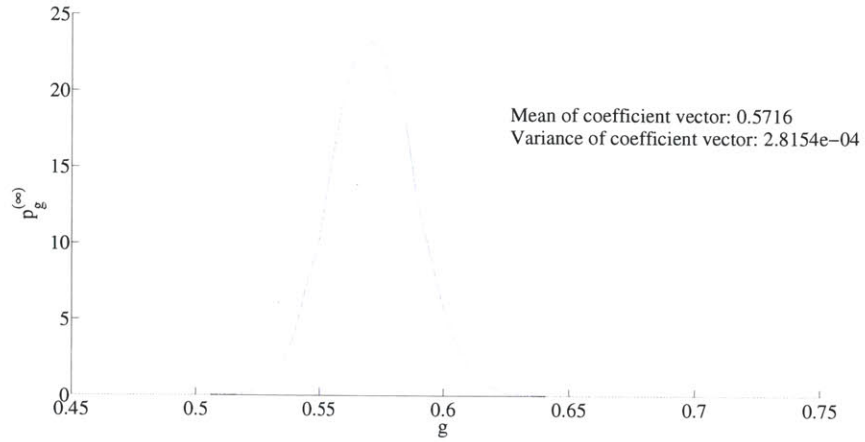


Figure 4-2: Distribution of the coefficient of a one-dimensional feedforward equalizer equalizing a 2-tap channel, with A.W.G.N.

probability density function of the mean of the coefficient vector is a delta function at the LMMSE solution. Without errors in the adaptation this is what we would expect it to be.

To verify this, we start with a simple channel. Suppose that we have a 2-tap channel, where the taps are $\mathbf{h} = \begin{bmatrix} 1 & 0.5 \end{bmatrix}^T$ Let the channel noise $v(n)$ be zero-mean, white and Gaussian, with a variance of $\sigma_v^2 = 10^{-0.3}$ (3dB SNR). Then, the (scalar) input to the equalizer is given by $u(n) = s(n)+0.5s(n-1)+v(n)$. Let $N_f = 1, N_b = 0$, so that we are simply multiplying the input by a scalar constant. Then, the channel matrix is $\mathbf{H} = \begin{bmatrix} 1 & 0.5 \end{bmatrix}^T$, and the MMSE equalizer coefficient (given by Equation 4.5) is $g_0 = 1/(1 + 0.25 + 10^{-0.3}) = 0.5710$.

Figure 4-2 shows the distribution of the coefficient vector (obtained by histogramming the coefficients obtained by multiple simulations of an equalizer in training mode).

Evidently, the mean is quite close to the LMMSE solution. Further, the variance of the coefficient about the mean is quite small. This means that the coefficient is tightly clustered about the mean. It has been shown in [20] (particularly, Section 14.5) that the variance of the coefficient vector in the training mode varies as $(1 - \lambda)$ when $\lambda \neq 1$. Typically, λ is close to 1, so that the variance is small. This fact allows us to approximate the coefficient with its mean. We will use this fact in the

development that follows.

4.4.1 The Matrix Formulation

Before proceeding to determining the Kernal and Steady State PDFs for hard-decision and soft-decision directed adaptive equalizers, we first show how to set up the problem as a matrix problem, which will be useful in computing the steady state PDFs in harder cases. This is a common way of numerically dealing with Fredholm equations of the second kind [1].

We note that the integral of Equation 4.21 can be approximately written as a matrix equation. Suppose we define a set $\boldsymbol{\Omega} \subset \mathbb{R}^N$ of some values of the mean vectors, i.e., each $\boldsymbol{\mu} \in \boldsymbol{\Omega}$ is an N-vector at which we would like to evaluate the steady state probability of the mean of the coefficients (note that $\boldsymbol{\Omega}$ could have infinitely many elements).

Assuming that the set $\boldsymbol{\Omega}$ has almost the entire steady state probability, we can approximately write Equation 4.21 as:

$$\mathrm{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)}(\boldsymbol{\mu}) \approx \sum_{\boldsymbol{\mu}' \in \boldsymbol{\Omega}} \left[\mathrm{p}_{\boldsymbol{\mu}_\mathbf{g}(n+1|n)|\boldsymbol{\mu}_\mathbf{g}(n|n-1)}(\boldsymbol{\mu}|\boldsymbol{\mu}')\, d\boldsymbol{\mu}' \right] \mathrm{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)}(\boldsymbol{\mu}') \tag{4.23}$$

So, define the vector $\mathbf{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)}$ as the vector of steady state probabilities of μ evaluated at the elements of $\boldsymbol{\Omega}$, i.e.,

$$\mathbf{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)}(k) = \mathrm{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)}(\boldsymbol{\mu}_k), \quad \boldsymbol{\mu}_k \in \boldsymbol{\Omega},\ k \in \mathbb{N} \tag{4.24}$$

Similarly define the matrix $\mathbf{K}$ as the conditional probability matrix, given by

$$\begin{aligned} \mathbf{K}(k,m) = \mathrm{p}_{\boldsymbol{\mu}_\mathbf{g}(n+1|n)|\boldsymbol{\mu}_\mathbf{g}(n|n-1)}(\boldsymbol{\mu}_k|\boldsymbol{\mu}_m)\, d\boldsymbol{\mu}, \\ (\boldsymbol{\mu}_k, \boldsymbol{\mu}_m) \in \boldsymbol{\Omega} \times \boldsymbol{\Omega},\ (k,m) \in \mathbb{N} \times \mathbb{N} \end{aligned} \tag{4.25}$$

Then it's evident that Equation 4.23 can be written as a set of linear equations,

given in the simple matrix form as

$$\mathbf{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)} = \mathbf{K}\mathbf{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)} \tag{4.26}$$

But this is the eigenvector problem! That is, $\mathbf{p}_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)}$ is the eigenvector of the conditional distribution kernal matrix $\mathbf{K}$ corresponding to the eigenvalue 1. This is logical, as all we have done is approximate the integral equation by a matrix equation.

The existence of a non-negative eigenvector corresponding to the eigenvalue 1 of the $\mathbf{K}$ matrix is guaranteed by the Perron-Frobenius theorem [35]. While this theorem has various aspects, the special case of the theorem that is relevant can be stated as follows:

Theorem 4.1. *Let* $\mathbf{A}$ *be any non-negative, irreducible matrix, with spectral radius* $\rho(\mathbf{A}) = r$. *Then the following hold:*

- r *is positive, real and the largest eigenvalue of* $\mathbf{A}$ *is equal to* r. *This is termed the Perron-Frobenius eigenvalue.*
- $\mathbf{A}$ *has a right eigenvector corresponding to the eigenvalue* r, *all of whose elements are non-negative, and,*
- $\min_i \sum_j a_{ij} \leq r \leq \max_i \sum_j a_{ij}$, *where* a *are the elements of* $\mathbf{A}$

A stochastic matrix is a square matrix such that the sum of all rows (or columns) is unity. By the theorem above, $r = 1$ and there exists an eigenvector with positive real entries, which when normalized is the steady state distribution of the Markov chain whose transition probability distribution is the stochastic matrix.

In this case, technically, we have a stochastic matrix "normalized" by $d\boldsymbol{\mu}$, the intervals into which we quantize the $\mathbb{R}^N$ space to approximate the integral by a sum. This is easily found numerically. So the numerical procedure to compute the steady-state distribution is as follows

- Select a set of values of $\boldsymbol{\Omega}$ at which we would like to evaluate the steady state PDF of $\mu_\mathbf{g}(n+1|n)$.

- For each pair of values $(\boldsymbol{\mu}_k, \boldsymbol{\mu}_m) \in \boldsymbol{\Omega} \times \boldsymbol{\Omega}$, compute $\mathbf{K}(k, m)$, the probability of the mean changing from $\boldsymbol{\mu}_m$ to $\boldsymbol{\mu}_k$ in one step. Note that we are quantizing the $\boldsymbol{\mu}$ space, and also restricting its size (from $\mathbb{R}^N$ to the size of the set spanned by elements of $\boldsymbol{\Omega}$).

- Determine the eigenvector of $\mathbf{K}$ corresponding to the largest eigenvalue (which is guaranteed theoritically to be 1). This is the steady state PDF of the coefficient vector.

4.5 Steady-State Distribution of Mean of Coefficient for Hard and Soft-Decision Adaptation

We now move to determining the distribution of the coefficient vector for the RLS algorithm in hard-decision directed adaptation and the soft-decision directed adaptative equalizer (Section 3.4.1).

From our discussion above, it becomes clear that what we are looking for are the probabilities of the mean of the coefficient vector at the next time step given the mean at the current time. Consider the update equation of the RLS algorithm, Equation 4.11, as applied to hard-decision adaptation.

In BPSK hard-decision directed adaptation, the output of the equalizer, $y(n) = \mathbf{g}^T(n|n-1)\mathbf{u}(n)$ either has the same sign as $s(n)$, in which cases as error is not made, or has the opposite sign (and then an error is made). Define E as the event that $d(n) \neq s(n)$ given $\mathbf{g}(n|n-1)$ is the coefficient vector used for filtering and $P_E(\mathbf{g}(n|n-1))$ as the probability of E. That is E is the event that an error is made, and correspondingly, define E^c as the event that an error is not made.

Then it is clear that $d(n)$ for hard-decision directed adaptation with BPSK is given by

$$d(n) \equiv d(n, s(n), \mathbf{g}(n|n-1)) = \begin{cases} s(n) & \text{w.p. } 1 - P_E(\mathbf{g}(n|n-1)) \\ -s(n) & \text{w.p. } P_E(\mathbf{g}(n|n-1)) \end{cases} \tag{4.27}$$

We will show in Section 4.6 that we can compute the probability that an error is made given $\mathbf{g}(n|n-1)$ readily in a closed form. However, it is also easy to compute this using Monte-Carlo simulation, and this is the approach we take.

Now, going back to Equation 4.11, we have

$$\begin{aligned}
&\mathbf{g}(n+1|n) \approx \lambda \mathbf{g}(n|n-1) + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbf{u}(n)d(n) \\
\Rightarrow\ &\mathbb{E}[\mathbf{g}(n+1|n)] = \lambda \mathbb{E}[\mathbf{g}(n|n-1)] + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbb{E}[\mathbf{u}(n)d(n)] \\
\Rightarrow\ &\boldsymbol{\mu}_{\mathbf{g}}(n+1|n) = \lambda \boldsymbol{\mu}_{\mathbf{g}}(n|n-1) + (1-\lambda)\mathbf{R}_{\mathbf{u}}^{-1}\mathbb{E}[\mathbf{u}(n)d(n)] \qquad (4.28)
\end{aligned}$$

Using $d(n)$ from Equation 4.27 in Equation 4.28, we have for hard-decision directed adaptation,

$$\boldsymbol{\mu}_{\mathbf{g}}(n+1|n) \approx \begin{cases} \lambda \boldsymbol{\mu}_{\mathbf{g}}(n|n-1) + (1-\lambda)\mathbb{E}[\mathbf{u}(n)s(n)|E^c] \\ \qquad\qquad \text{w.p. } 1 - P_E(\boldsymbol{\mu}_{\mathbf{g}}(n|n-1)) \\ \lambda \boldsymbol{\mu}_{\mathbf{g}}(n|n-1) - (1-\lambda)\mathbb{E}[\mathbf{u}(n)s(n)|E] \\ \qquad\qquad \text{w.p. } P_E(\boldsymbol{\mu}_{\mathbf{g}}(n|n-1) \end{cases} \qquad (4.29)$$

We have assumed that the error when the coefficient vector is distributed about a mean $\boldsymbol{\mu}_{\mathbf{g}}(n|n-1)$ is the same as the error made when the coefficient vector is equal to $\boldsymbol{\mu}_{\mathbf{g}}(n|n-1)$. This is an acceptable approximation as long as the coefficient vector is fairly tightly clustered about the mean (i.e., its second and higher order moments are quite small).

Thus, we have, written in a different form,

$$\mathrm{p}_{\boldsymbol{\mu}_{\mathbf{g}}(n+1|n)|\boldsymbol{\mu}_{\mathbf{g}}(n|n-1)}(\boldsymbol{\mu}|\boldsymbol{\mu}') \approx \begin{cases} P_E(\boldsymbol{\mu}'), \text{ when} \\ \qquad \boldsymbol{\mu} = \lambda\boldsymbol{\mu}' - (1-\lambda)\mathbb{E}[\mathbf{u}(n)s(n)|{\boldsymbol{\mu}'}^H\mathbf{u}(n) \neq s(n)] \\ 1 - P_E(\boldsymbol{\mu}'), \text{ when} \\ \qquad \boldsymbol{\mu} = \lambda\boldsymbol{\mu}' + (1-\lambda)\mathbb{E}[\mathbf{u}(n)s(n)|{\boldsymbol{\mu}'}^H\mathbf{u}(n) = s(n)] \end{cases} \qquad (4.30)$$

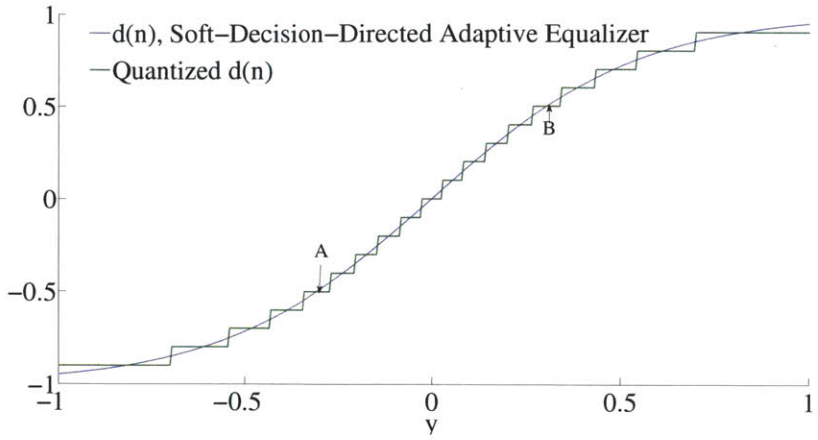


Figure 4-3: Plot of decision function and its quantized version for the soft-decision directed adaptive equalizer

Evidently, for each value $\boldsymbol{\mu}'$ there are 2 values of $\boldsymbol{\mu}$ which have non-zero probabilities. Similar to the way we did in Section 4.4, we can define a set $\boldsymbol{\Omega}$ of $N-$vectors at which we want to find the steady state probabilities. Then, we compute the 2 values of $\boldsymbol{\mu}$ for each $\boldsymbol{\mu}' \in \boldsymbol{\Omega}$, and quantize these 2 values to the nearest point in $\boldsymbol{\Omega}$. Thus, we can populate the $\mathbf{K}$ matrix and find the eigenvector corresponding to eigenvalue 1. The conditional expectations and probabilities in Equation 4.30 can computed fairly quickly with a Monte-Carlo approach, which we describe soon.

Firts, though, we generalize this to the soft-decision directed adaptive equalizer. We know that in this case the form of the decision function $d(n)$ as a function of $y(n)$ is a sigmoid curve, such as the one in Figure 4-3. For the purpose of analysis, we use a quantized form as also shown in the figure. We define $\mathcal{Y}$ as the set of partitions of y, i.e., the set of intervals of y such that all values within an interval map to a particular quantized value.

Beginning from Equation 4.28 and in a similar manner to the hard-decision directed adaptation analysis, we now look at regions conditioned on which the expectation of the quantity $\mathbf{u}(n)d(n)$ is the same. Note that in the hard-decision directed $d(n)$ was either $s(n)$ or $-s(n)$, so we could express Equation 4.30 in terms of $s(n)$ rather than $d(n)$. Here, however, $d(n)$ takes, in theory, infinitely many values, and we look at a quantized subset of these for analysis. So, in the quantized framework,

we must look at the partitions such that $\mathbb{E}[u(n)d(n)]$ are the same for different $s(n)$, as the mean will "propagate" in the same direction in each of those cases.

For example, region A in Figure 4-3 has the same probability under $s(n) = +1$ as region B (which is symmetric with A about 0) has under $s(n) = -1$ and vice-versa. So we look at the filter inputs that give us a particular range of $y(n)$ for a particular filter coefficient. Then compute the mean of $\mathbf{u}(n)d(n)$, where here $d(n)$ refers to the quantized value in that region. Thus, the update equation can be written as

$$\begin{aligned} \mathrm{p}_{\boldsymbol{\mu}_{\mathbf{g}}(n+1|n)|\boldsymbol{\mu}_{\mathbf{g}}(n|n-1)}(\boldsymbol{\mu}|\boldsymbol{\mu}') = \mathbb{P}(\boldsymbol{\mu}'^{T}\mathbf{u}(n) \in A), \text{ when} \\ \boldsymbol{\mu} = \lambda\boldsymbol{\mu}' + (1-\lambda)\mathbb{E}[\mathbf{u}(n)d(n)|\boldsymbol{\mu}'^{T}\mathbf{u}(n) \in A], A \in \mathcal{Y} \end{aligned} \tag{4.31}$$

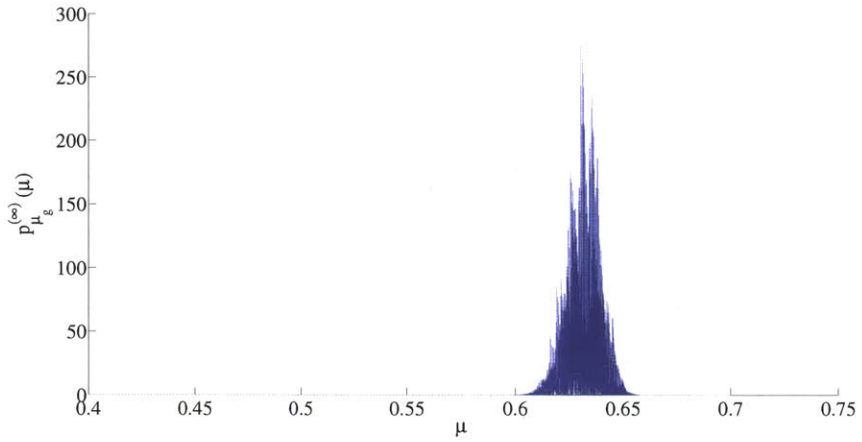
Of course, the regions can be made as small as desired. Note that hard-decision directed adaptation performance can be computed as a special case of this with 2 quantization regions ($y(n) \geq 0$ and $y(n) < 0$) and $d(n) = 1, -1$ respectively in these regions. Thus Equation 4.30 is a special case of Equation 4.31.

We still need to discuss how to compute the expectations and probabilities in the different regions in Equation 4.31 (and hence, of course, Equation 4.30). As mentioned already, the easiest way to do this is Monte-Carlo Simulation. Given a channel matrix, $\mathbf{H}$, we generate a number of i.i.d. uniform symbol sequences, $\mathbf{s}_P(n)$, and $N-$dimensional Gaussian noise sequences $\boldsymbol{\upsilon}(n)$ with covariance matrix $\mathbf{S}$, which is easy to do. Thus, we generate instances of the input vector $\mathbf{u}(n)$. Then for each $\boldsymbol{\mu}' \in \boldsymbol{\Omega}$, we compute the equalizer output $y(n)$ corresponding to each input. The fraction of the $y(n)$ values in a particular interval A is an estimate of $\mathbb{P}(\boldsymbol{\mu}'^{T}\mathbf{u}(n) \in A)$, and the expectation is approximately the mean of $\mathbf{u}(n)d(n)$ for that interval A.

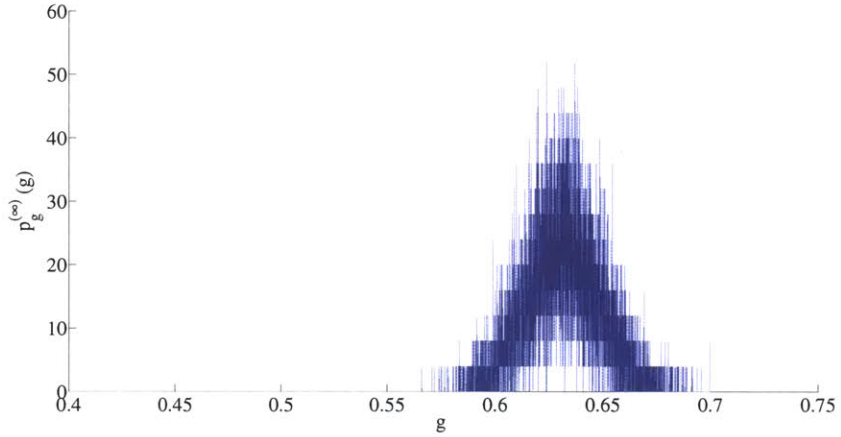
4.5.1 Performance Predictions

We now present the steady state PDFs predicted by the procedure above, compare them against simulation and highlight some important characteristics of these.

We start with the one-dimensional equalizer that we used for Figure 4-2. To



(a) Predicted Steady State Distribution of the Mean of the Coefficient Vector



(b) Simulated Coefficient Vector Distribution

Figure 4-4: 1 tap equalizer for 2-tap channel under hard-decision directed adaptation

recap, the channel transfer matrix is $\mathbf{H} = \begin{bmatrix} 1 & 0.5 \end{bmatrix}^T$, and the noise is white, zero-mean and Gaussian with variance $10^{-3/10}$. The equalizer forgetting factor λ is set to 0.99. We plot the predicted steady state distribution of the mean $p_{\boldsymbol{\mu}_\mathbf{g}}^{(\infty)}$ and the simulated distribution of the coefficient vector (*not* the mean!) in Figure 4-4.

First, note that the prediction of the distribution of the mean of the coefficient vector is fairly good. The region in which the mean is clustered is what the simulation predicts as the center of the region in which the coefficient vector is clustered as we would expect.

The cluster is centered at around $g = 0.65$, as opposed to the training mode

mean of 0.57. Hard decision directed adaptation *raises* the mean of the coefficient vector cluster. This happens because the adaptation procedure assumes some wrong decisions are correct. Thus, it biases the equalizer coefficient to be more positive. The equalizer thus "thinks" it is correct and that the SNR is higher than it actually is, so it raises the coefficient corresponding to the signal that is being demodulated.

What happens if we perform soft-decision directed adaptation? The predicted distribution of the mean of the coefficient and simulated distribution of the coefficient are shown in Figure 4-5. Evidently the clusters in this case are centered closer to the MMSE solution. This is due to the conservative nature of the adaptation process that prevents errors from biasing the coefficients too strongly. This is a trend we will continue to see.

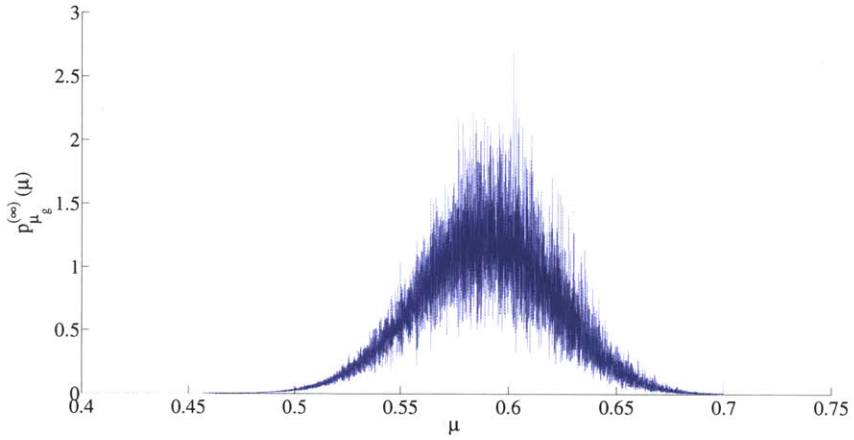
To move to a more complicated case, we consider the same channel $\mathbf{h} = \begin{bmatrix} 1 & 0.5 \end{bmatrix}^T$, with the same noise model (zero-mean A.W.G.N. with a variance $10^{-0.3}$. However, now we assume that there are 2 feedforward taps, rather than just one. The channel matrix is now

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \\ 0 & 0.5 \end{bmatrix} \tag{4.32}$$

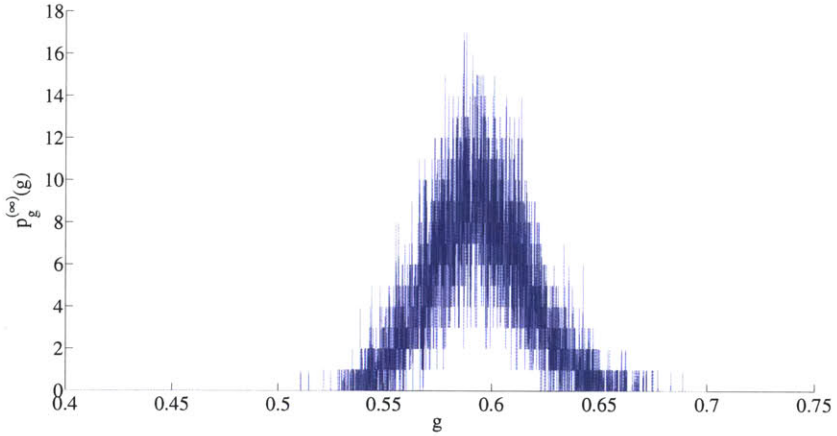
and $\mathbf{S} = 10^{-0.3}\mathbf{I}_2$. The optimal coefficient vector is given by $\mathbf{g_0} = \begin{bmatrix} 0.622 & -0.178 \end{bmatrix}^T$. Once again, we set the forgetting factor $\lambda = 0.99$. We consider the hard-decision directed algorithm first. Figure 4-6 has the predicted distribution of the mean of the coefficient vector. Figure 4-7 has the simulated distribution.

Once again, we see the same trend. The hard-decision directed adaptive equalizer is "overconfident". Thus, the first tap has a higher gain than the corresponding MMSE coefficient. This is seen from the simulation, in Figure 4-7. The prediction is reasonable given the assumptions made.

Figures 4-8 and 4-9 have the corresponding plots for soft-decision directed adaptation. They show that the soft-decision directed adaptive equalizer is somewhat closer to the MMSE solution. Once again the prediction is reasonable. In both the

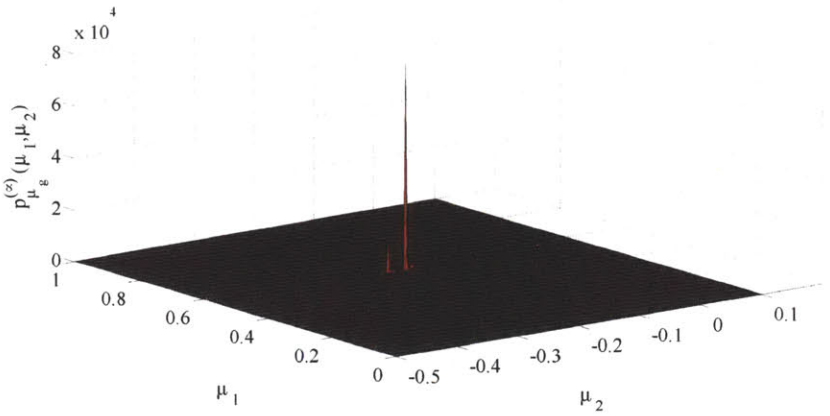


(a) Predicted Steady State Distribution of the Mean of the Coefficient Vector

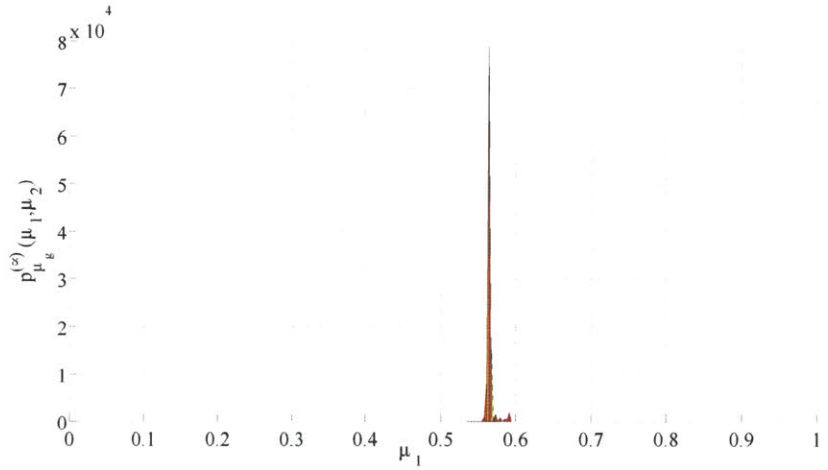


(b) Simulated Coefficient Vector Distribution

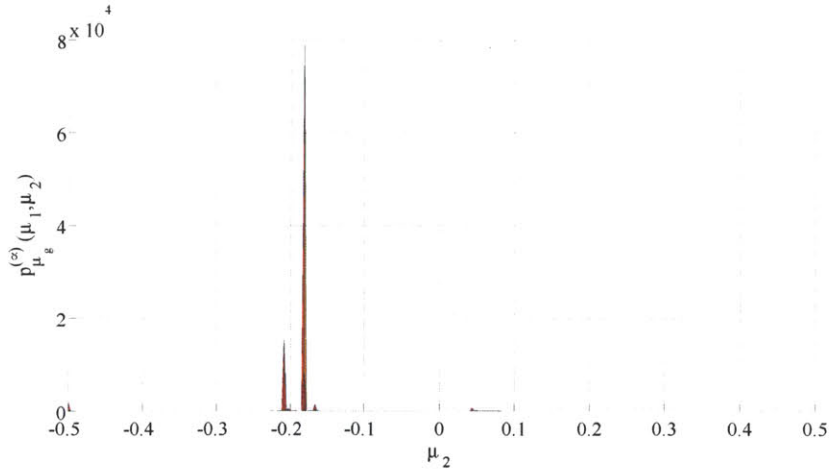
Figure 4-5: 1-tap equalizer for 2-tap channel under soft-decision directed adaptation



(a) Perspective View

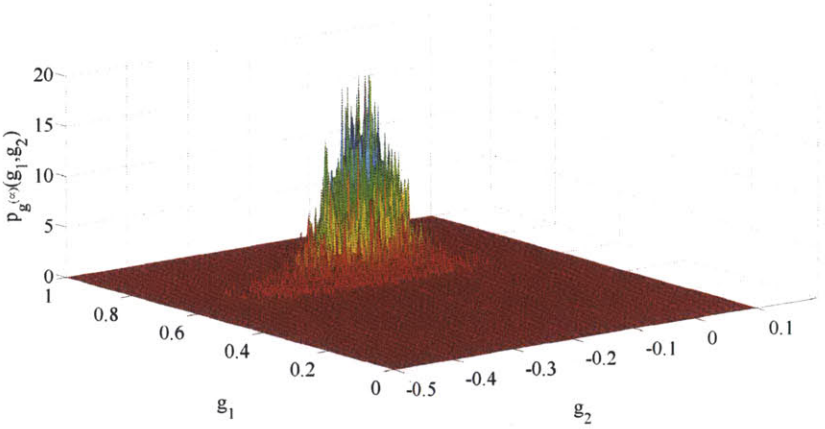


(b) Feedforward tap 1

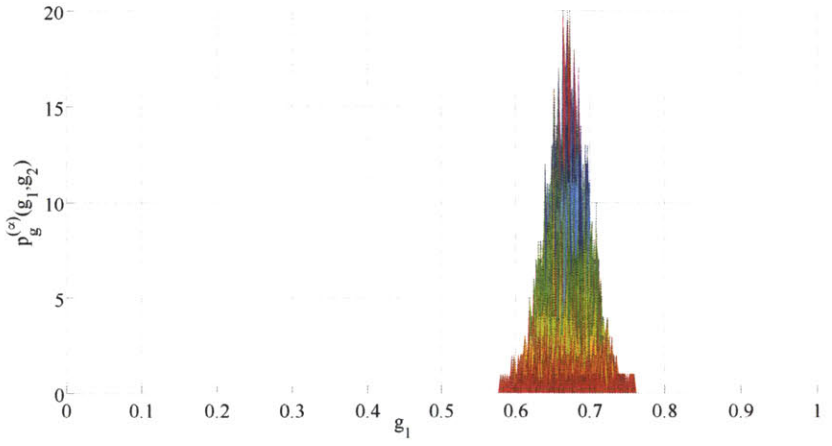


(c) Feedforward tap 2

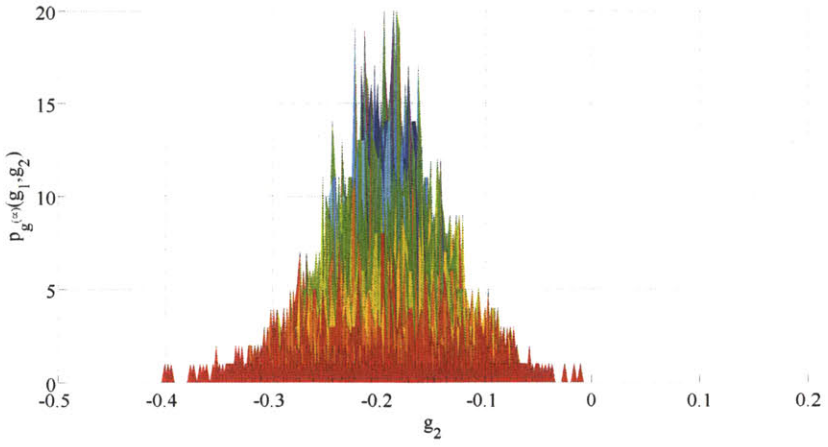
Figure 4-6: Predicted Distribution of the Mean of the Coefficient Vector with 2 Feedforward Taps, Hard-Decision Directed Adaptation



(a) Perspective View



(b) Feedforward tap 1



(c) Feedforward tap 2

Figure 4-7: Distribution of the Coefficient Vector from Simulation with 2 Feedforward Taps, Hard-Decision Directed Adaptation

hard and soft decision directed adaptive equalizers for this channel, however, there is some error in prediction, particularly in the second tap. It is still unclear why the prediction for the second tap should be worse than the first.

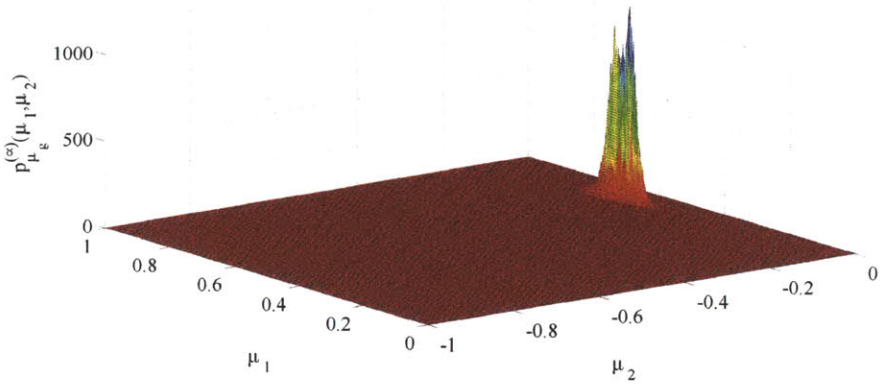
We consider one more important case that arises when feedback taps are involved. It is known that in general, equalizers with feedback (DFEs) perform better than their linear counterparts [38]. In many cases, using a DFE is necessary to achieve reasonable performance. However, DFEs are subject to a catastrophic failure mode. This has been noted in [54] and [37] in the context of Multiuser CDMA systems and in [16] for underwater communication systems, although, to the best of our knowledge, it has not been analyzed in detail. We now discuss how it occurs due to errors in the adaptation process, and demonstrate that our analysis procedure can indeed predict it.

4.5.2 The Catastrophic Failure Mode of Adaptive DFEs

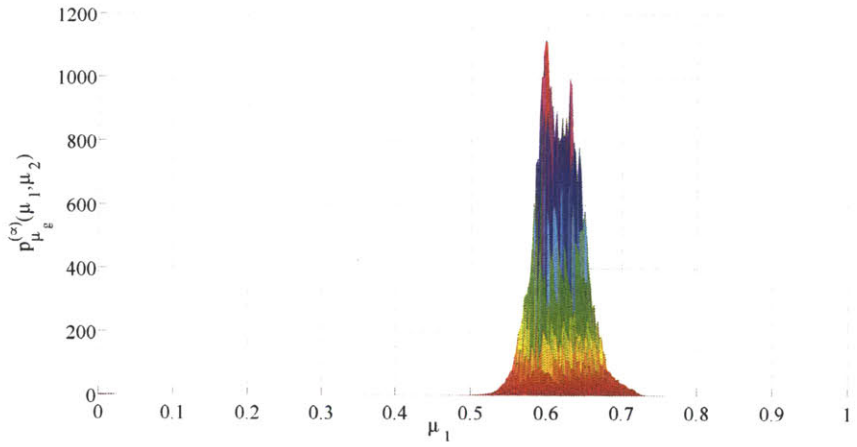
At an intuitive level, hard-decision directed adaptation assumes that the output of the equalizer has the correct sign (in BPSK). Thus, we assume that the output is never far away from the input. So, the RLS algorithm adjusts such that it always moves in the same direction.

We have seen in the linear (feedforward) equalizer examples considered above that as a consequence of this behavior, the algorithm assumes the SNR is higher than it really is. One consequence is that at very low input SNRs, the equalizer may see a stronger degree of correlation between feedback taps and the filter output than actually exists, as it assumes that the "overall" SNR is high. In the worst case, this drives the coefficient vector into a state from which it cannot recover. This state in which the equalizer stops adapting and responding to the channel, but gets stuck in a recurrent state of the coefficient evolution Markov process is what we call the *catastrophic failure mode* of the adaptive DFE.

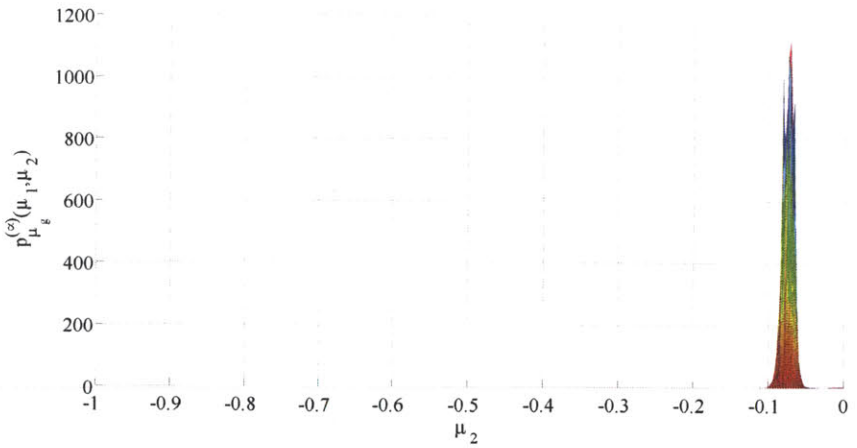
We can discuss this further in terms of our knowledge of how the RLS algorithm operates. Essentially, the RLS algorithm looks at the cross correlation between the "desired symbol" and the input to the filters. At low SNRs, when many symbols are



(a) Perspective View

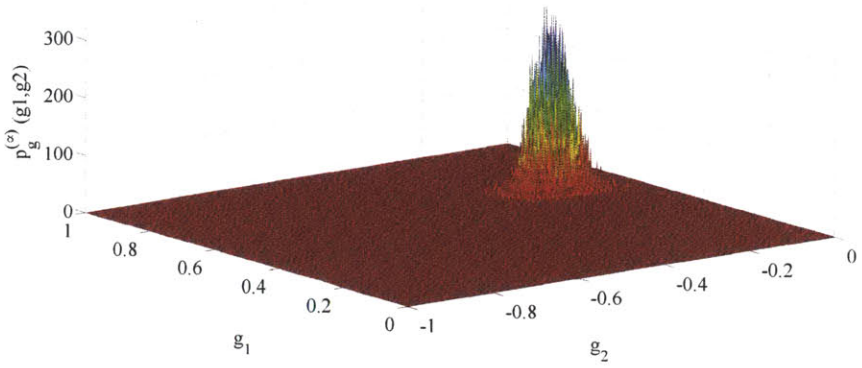


(b) Feedforward tap 1

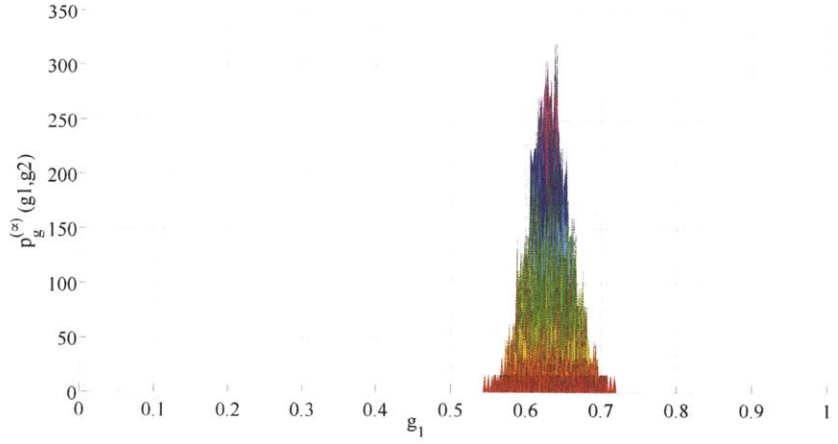


(c) Feedforward tap 2

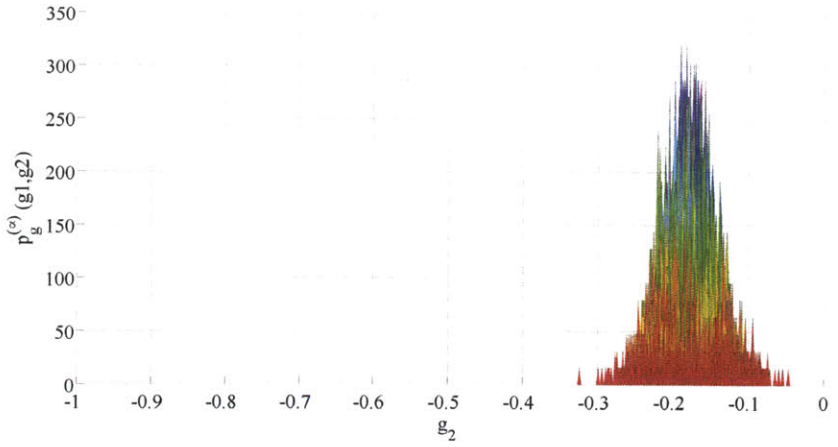
Figure 4-8: Predicted Distribution of the Mean of the Coefficient Vector with 2 Feedforward Taps, Soft-Decision Directed Adaptation



(a) Perspective View



(b) Feedforward tap 1



(c) Feedforward tap 2

Figure 4-9: Distribution of the Coefficient Vector from Simulation with 2 Feedforward Taps, Soft-Decision Directed Adaptation

in error, the feedforward filter input at time n which depends on the true transmitted symbol $s(n)$, may have a low correlation with the desired symbol at time n, $d(n)$, especially if sufficient errors are made in succession. So, the feedforward filter may start to have a low norm.

The feedback filter, however, contains a set of past symbols $s(n-1), s(n-2), \ldots, s(n-N_b)$. Recall that we assumed that the true past symbols are in the feedback filter- we will consider the actual case (when decisions are fed back) shortly. Then, if the feedforward filter taps become small at some time n, then the output of the equalizer at time n is always strongly correlated with the feedback filter input. Thus, the belief of the equalizer that the feedforward filter inputs are noisy but the feedback filter inputs have a strong correlation with the current channel output, gets reinforced.

The Weiner filter solution itself (to which, as we have shown, we converge in training mode with stationary channels) has small values for the feedfoward taps at low SNRs. However, in training mode, the feedback filter taps are also relatively small, as they subtract out the residual ISI after passing the input through the feedforward filter, so that what they optimally need to subtract are small values. In decision directed mode, this relationship is not necessarily preserved. When the correlation between the decision-directed symbol and feedforward taps is small, the feedback taps can start to dominate because the output of the filter and the decision are dependent on each other.

For example, consider a 1-tap feedforward filter g_f and a 1-tap feedback filter g_b. At low SNR, the feedforward tap starts to get small, because, as we discussed above, it may have poor correlation over a few symbols with the output. The output $y(n) \approx g_b s(n-1)$, when we assume that the correct past symbol is fed into the feedback filter. Then, if $g_{fb} > 0$, $d(n) = s(n-1)$. This is obviously quite strongly correlated with the feedback filter, so the algorithm decides to push the feedback filter higher. This continues until $g_b = 1$, at which point the innovation obviously becomes 0 and the equalizer is stuck. This is an example of the catastrophic failure mode- a state in which adaptation stops and the feedforward filter coefficients are all zero.

We need to consider, however, what happens when the feedback filter receives the hard decisions. It is evident that a very similar situation could show up, although the modes of failure could be different. Once again, it is possible that we would only weight the feedback filter taps. In this case, these could possibly even keep receiving the same data. For instance, one possible mode for the equalizer we discussed above is that the feedback tap is 1. Then the input to the filter (which would be $d(n)$) is always just 1 and the output $y(n) = 1$. However, the general definition of the failure mode as a mode in which adaptation stops and the filter coefficients are stuck still applies.

It should be noted that one important characteristic of the failure mode that is a consequence of the discussion above (and indeed has been observed in practice) is that the feedforward filter tap magnitudes become very small (L_2-norm of the feedforward filter is small when compared to that of the feedback filter), i.e., the adaptation stops trusting the channel.

This may become important in predicting when failure is happening "on the fly" as the equalizer operates, and we may be able to take advantage of it by weighting the feedback filter less when it happens. This would help because we know that the channel output has to be correlated with the symbol we wantt to demodulate, whereas the feedback filter may not be. Thus, restricting the L_2-norm of the feedback filter coefficients would likely stop catastrophic failure.

One standard approach to controlling the relative L_2-norm of filter coefficients in an optimization problem is *diagonal loading*, which adds a loading factor δ to the cost criterion for the coefficients whose value is to be restricted. However, the resulting cost function may not have a recursive solution. One method to get around this problem is to observe that diagonal loading is equivalent to adding white noise to the output of the feedback filter. This would decorrelate the output of the overall filter from the feedback filter coefficients, which would help the adaptation algorithm avoid failure.

Of course, the variance of the noise would have to be time-variant, i.e., the diagonal loading factor is not a fixed constant. As the equalizer starts to fail, we would need

add more noise to the feedback filter. This is why having a quick metric to check the current status of the equalizer is important. While this is the intuition behind one possible approach to tackling this problem, the exact mechanism to solve it is still an open question.

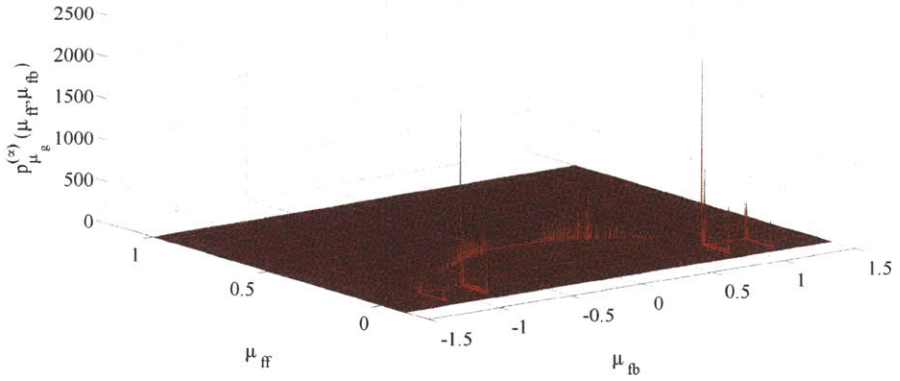
This catastrophic failure mode has been observed in the past, but it has been hard to predict and understand. We show, however, that the coefficient mean prediction procedure that we described does in fact predict the existence of the catastrophic failure mode.

We choose a scalar, 1-tap channel with unity gain, i.e., $\mathbf{h} = 1$. The channel is assumed to have A.W.G.N. with variance 10 (-10dB SNR). The equalizer is a DFE (where, as we assumed at the beginning of this chapter, the feedback filter gets the right symbols) with 1 feedforward tap and 1 feedback tap. The optimal system in this case is $\mathbf{g_0} = \begin{bmatrix} 0.091 & 0 \end{bmatrix}^T$. Again, the forgetting factor is $\lambda = 0.99$.

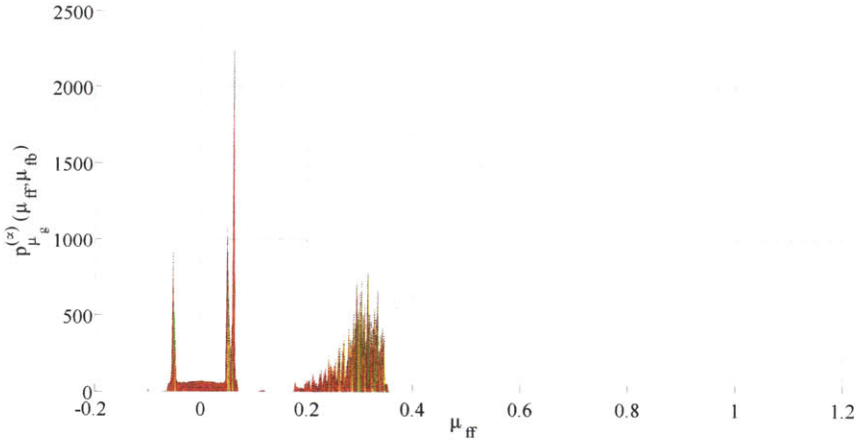
Plots of the predicted and simulated distributions of the coefficients are shown in Figures 4-10 and 4-11, respectively. The key feature of the plots is that there are 2 clusters of points about which the coefficient vector may be distributed. Further, the magnitude of the feedforward tap is nearly 0, and the feedback tap has a large (close to 1) magnitude!

From simulation, we see in Figure 4-11 that the coefficient is either $\begin{bmatrix} 0 & 1 \end{bmatrix}^T$ or $\begin{bmatrix} 0 & -1 \end{bmatrix}^T$. These are catastrophic modes. Consider that the coefficient is $\begin{bmatrix} 0 & 1 \end{bmatrix}^T$. Then, given that the input to the feedback filter at time n is $s(n-1)$, the output $y(n) = s(n-1)$. Then, the hard decision made on $y(n)$ is also $s(n-1)$, the innovation is 0, and the future coefficients are all the same. An identical argument applies in the other case as well.

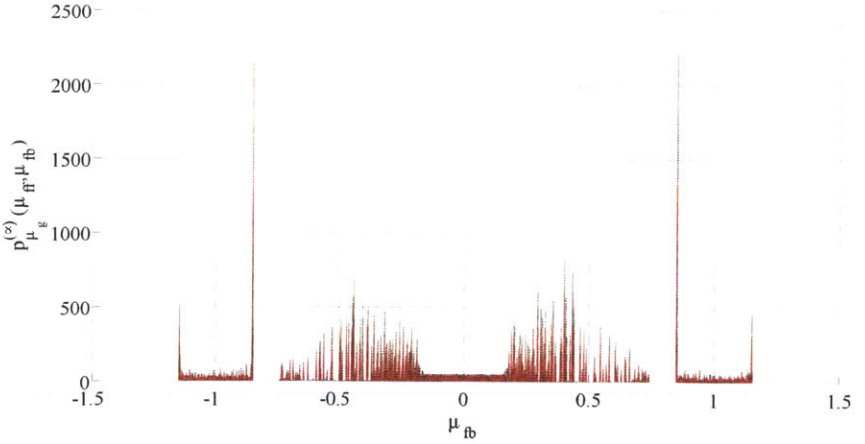
The predicted mean is not exactly one of these 2 modes. However, with high probability, the mean takes one of 2 (symmetric about $g_{fb} = 0$ values in which the $L-2$ norm of the feedback tap is much larger than that of the feedforward tap. Thus, if the feedback tap is positive, the equalizer output takes the sign of $s(n-1)$ with high probability, and thus reinforces the algorithm's opinion that the "correct" coefficient vector places more weight on the feedback tap. Thus, it further increases the value



(a) Perspective View

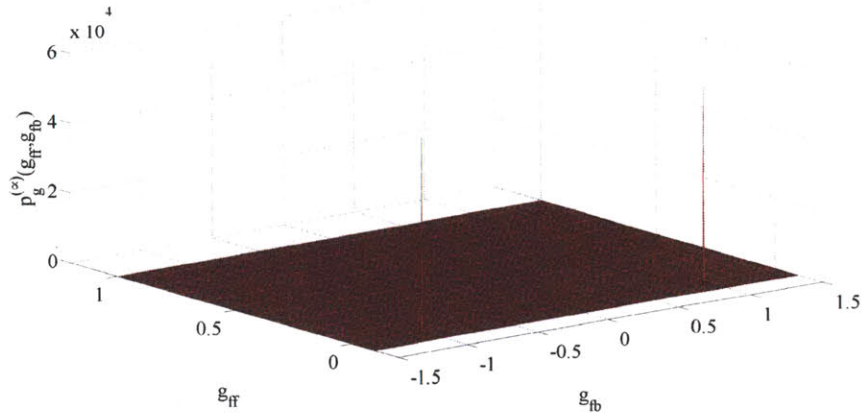


(b) Feedforward tap

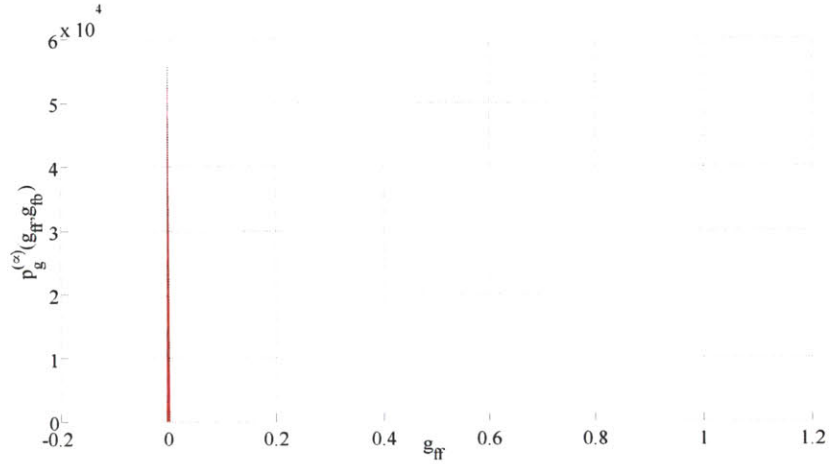


(c) Feedback tap

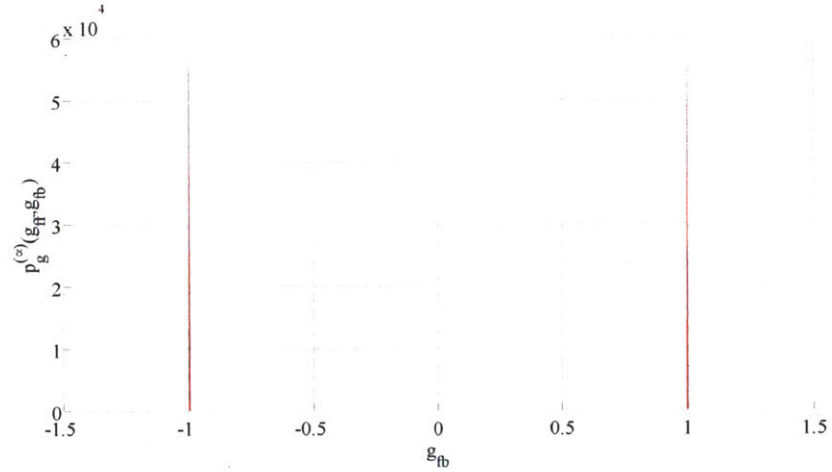
Figure 4-10: Predicted Distribution of the Mean of the Coefficient Vector of DFE with 1 feedforward and 1 feedback tap, Hard-Decision Directed Adaptation. Showing Failure Mode



(a) Perspective View



(b) Feedforward tap



(c) Feedback tap

Figure 4-11: Simulated Distribution of the Coefficient Vector of DFE with 1 feedforward and 1 feedback tap, Hard-Decision Directed Adaptation. Showing Failure Mode

until it reaches a point where the Markov chain is stuck in a recurring state.

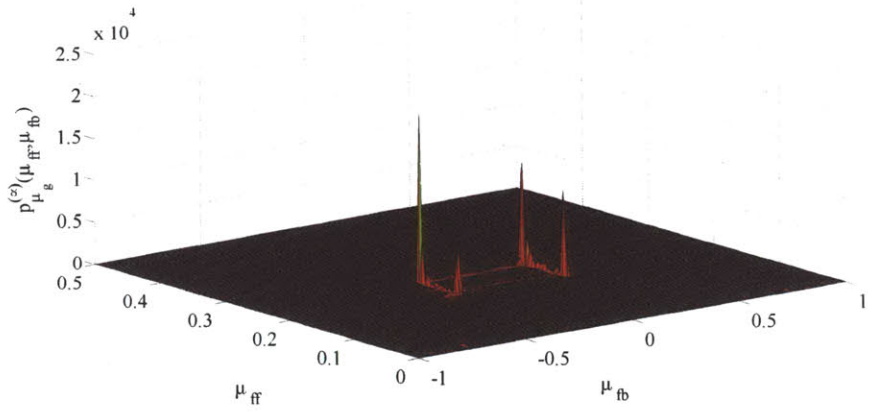
This is an important issue in practice. The DFEs that we will test on practical data in the chapters that follow will be subject to this issue. It is thus of importance to understand it intuitively and to be able to predict it, at least to some extent.

What about the soft-decision directed equalizer? Is it also subject to the failure mode? We consider Figure 4-12, which uses the mean prediction algorithm on the soft-decision directed adaptive equalizer. The results, once again, show the existence of clusters, which leads us to expect failure. However, the clusters are closer to the MMSE solution, so we would theorize that possible failure in this case takes a longer time, or happens with less probability that for hard-decision directed adaptation.

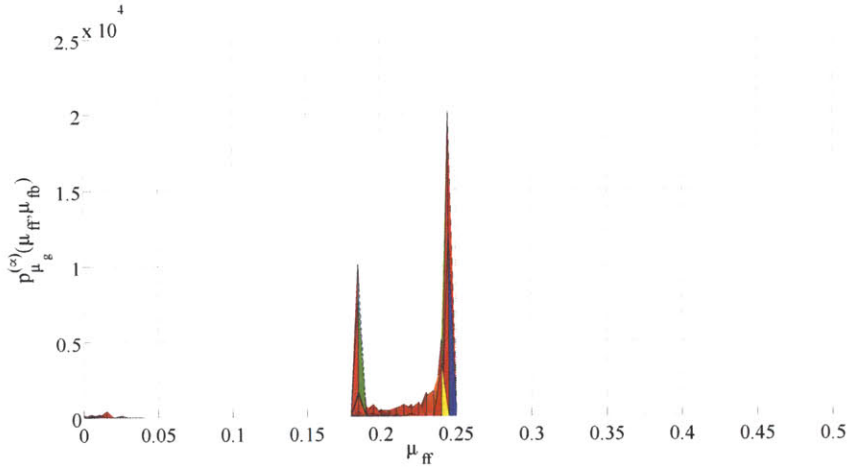
However, Figure 4-11 gives us a surprise, as it shows that simulating the soft-decision directed adaptive equalizer, the failure mode never occured. That is, the algorithm stayed close to the actual solution. This is highly encouraging, as it demonstrates that there are at least some scenarios in which the soft-decision directed adaptation algorithm prevents failure.

It should be noted at this point, though, that even the soft-decision directed algorithm fails. However, as we might infer from the results in Figures 4-12 and 4-13, the threshold SNR for failure is lower (more factors than just SNR apply to time-varying channels, which are not fully understood, but the gist is that it is more robust and fails with smaller probability than the hard-decision directed algorithm).

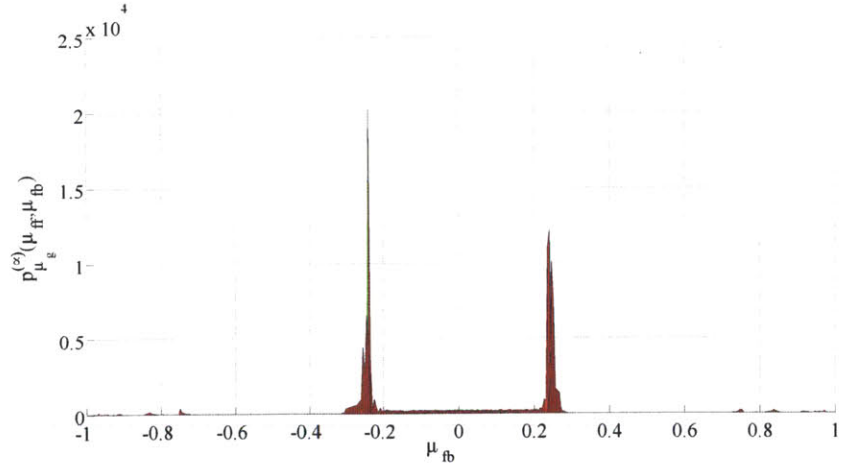
This is verified for the same channel as above in Figure 4-14. This figure indicates the simulated probability of failure- the fraction of equalizer runs at a particular SNR (with different, randomly generated noise sequences) that resulted in catastrophic failure, i.e., which resulted in a steady state innovation close to 0. This shows that the soft-decision directed adaptation approach has a significantly smaller probability of failure at these SNRs, and does not fail with high probability even at -35dB with this channel. Thus, it clearly buys us a lot of performance. On the other hand, with practical systems, the typical difference in performance has been observed to be $\sim$ 6dB or so. We will see how much it gains us in the context of practical systems in Chapter 6.



(a) Perspective View

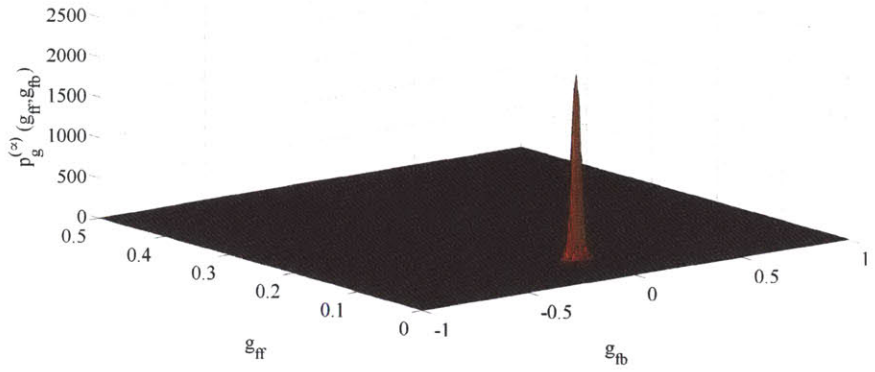


(b) Feedforward tap

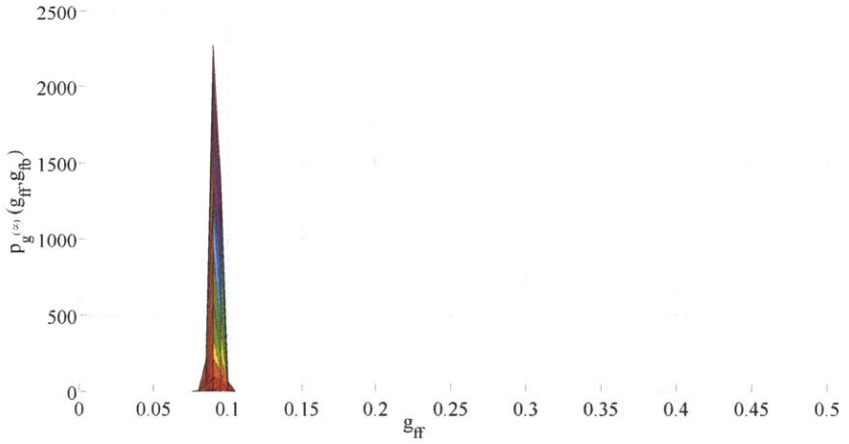


(c) Feedback tap

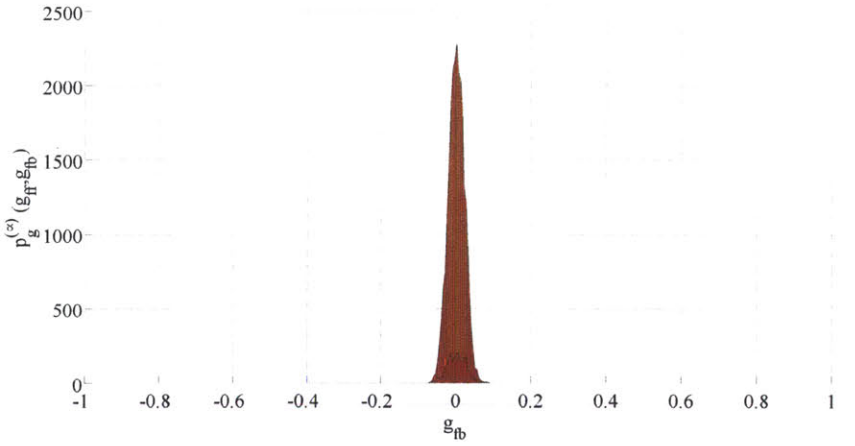
Figure 4-12: Predicted Distribution of the Mean of the Coefficient Vector of DFE with 1 feedforward and 1 feedback tap, Soft-Decision Directed Adaptation. Showing Failure Mode



(a) Perspective View



(b) Feedforward tap



(c) Feedback tap

Figure 4-13: Simulated Distribution of the Coefficient Vector of DFE with 1 feedforward and 1 feedback tap, Hard-Decision Directed Adaptation. Does not Fail Catastrophically

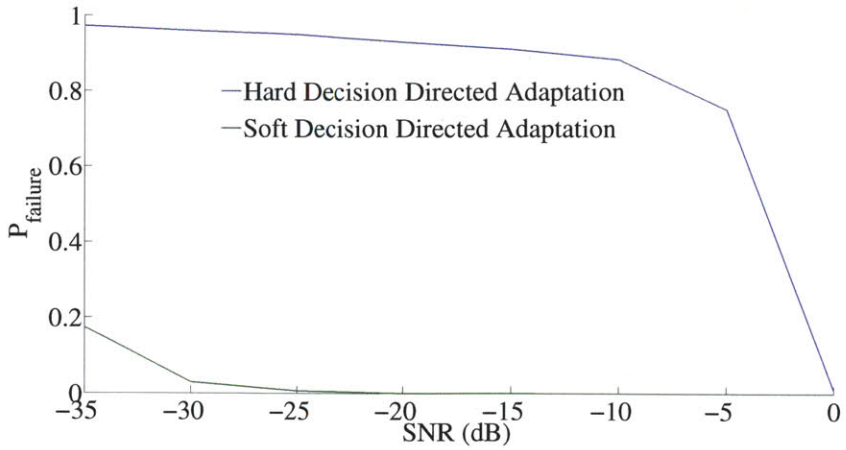


Figure 4-14: Plot of Probability of Catastrophic Failure in 7500 Symbols at Various SNRs

A possible conjecture about why the algorithm did not fail when failure was predicted is that the time over which it was run was insufficient to drive the system to the catastrophic state. It has been observed that longer runs lead to a higher probability of failure, which is reasonable, as it gives the algorithm longer to enter a catastrophic chain of events. This is still an encouraging result, as it indicates a definite improvement in performance for the same channel.

To briefly recap the discussion so far, we have tested the performance of the different adaptation strategies. In general, we have seen that the soft-decision directed algorithm is more robust in that the solution is typically closer to the MMSE solution and it is less susceptible to catastrophic failure. We have also seen how not knowing the transmitted signal affects the coefficient adaptation process and the issue of "overconfident" adaptation, how it manifests itself in both non-catastrophic and catastrophic modes.

However, one important objective of the analysis was to gain insight into developing models for the PDF of the system output, for the RELS procedure that we defined in Chapter 3. An understanding of the distribution of the coefficient vector is key to this. We will now turn to applying this understanding to the statistics of the equalizer output, and will show how that may help us in further improving the performance of the equalizer system.

4.6 Characterizing the Output Statistics

As we discussed in Section 3.4, an important issue in the Recursive Expected Least Squares algorithm is modelling the PDF of the output of the equalizer conditioned on the transmitted symbols and the past outputs. We also assumed that the output statistics are stationary and Gaussian centered about the tranmitted symbols. The soft-decision directed adaptive equalizer was the equalizer that used the RELS algorithm with this assumption.

In this section, we discuss what the statistics of the equalizer really are. This serves two purposes. First, it allows us to find regions in which the assumption we have made is valid, and where it needs to be modified. It also allows us to develop more sophisticated models for when it is not valid.

The development is relatively simple. We have seen in Section 4.5.1 that the variance of the coefficient vector about its mean is quite small when there is only one cluster of coefficients (in the training mode, specifically). We have also seen that with various adaptation strategies, we can use our prediction strategy to obtain the steady state distribution of the mean of the coefficient vector. What this implies is that the distribution of the coefficient vector has some distribution about each mean, and each of the means occurs with some probability. Our discussion essentially means that we can replace the distribution of the coefficient vector by the distribution of its mean.

We know from Section 4.3.1 that the input to the equalizer is a Gaussian mixture,

$$\mathbf{u}(n) \sim \frac{1}{2^P} \sum_{\mathbf{s}\in\{-1,1\}^P} \mathcal{N}(\mathbf{H}^T\mathbf{s}, \mathbf{S}) \tag{4.33}$$

As linear combinations of Gaussian mixtures are Gaussian mixtures, passing $\mathbf{u}(n)$ through an equalizer filter $\mathbf{g}$ gives an output $y(n)$ which is also a Gaussian mixture, when parametrized on the coefficient vector.

$$y(n) \sim \frac{1}{2^P} \sum_{\mathbf{s}\in\{-1,1\}^P} \mathcal{N}(\mathbf{g}^T(\mathbf{H}^T\mathbf{s}), \mathbf{g}^T\mathbf{S}\mathbf{g}) \tag{4.34}$$

Further, observe that a similar argument applies when we condition on $s(n)$. The only thing that changes is that we now only have 2^{P-1} components in the Gaussian mixture, as the first element is fixed. For example, conditioning on $s(n) = 1$.

$$\mathrm{p}_{y(n)|s(n)}(y|1;\mathbf{g}) = \frac{1}{2^{P-1}} \sum_{\mathbf{s}\in\{-1,1\}^{P-1}} \mathcal{N}\left(\mathbf{g}^T\left(\mathbf{H}^T\begin{bmatrix}1\\ \mathbf{s}\end{bmatrix}\right), \mathbf{g}^T\mathbf{S}\mathbf{g}\right) \tag{4.35}$$

This is almost exactly what we want, if we assume that $y(n)$ is independent of $\mathbf{y}(n-1)$ given $s(n)$. In practice, this is a reasonable assumption. Thus we immediately see that the Gaussian assumption on the output conditioned on the symbols, where the means are at the symbols, is not very accurate, particularly if we assume that the means are the transmitted symbols. Even if we estimate the mean and variance of the output data and fit a Gaussian to that, it is still a sub-optimal fit. However, the Gaussian mixture assumption with $\mathbf{g} = \boldsymbol{\mu}_{\boldsymbol{g}}^{(\infty)}$ is an excellent fit. That is, assuming the output is a Gaussian mixture as in Equation 4.35 with the coefficient vector equal to its steady state mean is a good fit.

These models are verified in Figure 4-15, for an equalizer in training mode. Here, the blue line is the normalized histogram of the filter outputs corresponding to a transmitted signal $s(n) = 1$. The green line is a Gaussian PDF with the mean estimated as the mean of the output data, when the filter coefficient is equal to the mean of the filter coefficient in training mode (which, as we have shown in Section 4.4 is simply the MMSE coefficient $\mathbf{g_0}$) and variance given by the variance of the output. Clearly there is a mismatch between the two. The red line is the Gaussian mixture PDF of Equation 4.35, with $\mathbf{g} = \mathbf{g_0}$. This is clearly an excellent match.

However, the original Gaussian PDF assumption of Section 3.4.1 is not too bad a match. In higher SNR regions, when the mean of the output conditioned on $s(n)$ is close to $s(n)$, the soft-decision directed adaptive equalizer performs quite well, as we have seen. Moreover, when we have complicated time-varying channel coefficients, and none of the statistics are stationary, it becomes evident that applying Equation 4.35 can be quite difficult, as the coefficients are all time-varying. This is why, for the

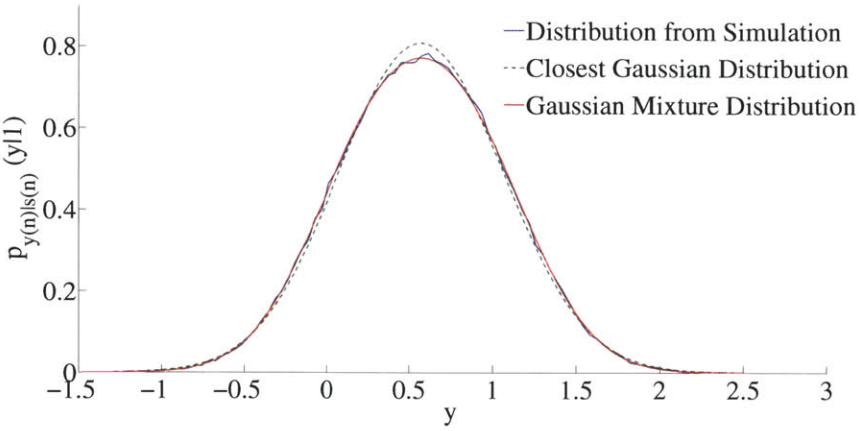


Figure 4-15: Plot of Statistics of Equalizer Output Conditioned on $s(n) = 1$

rest of the thesis in practical implementations, we will continue to use the Gaussian assumption and not make the model any more involved. But it should be realized that the underlying system dynamics are now known, and further research may provide simple ways to take advantage of them, especially considering the fact that we do parametrize on the previous value of the coefficient vector, so that knowing its full statistics is not always required.

The analysis that we have performed so far also gives us intuition into when the stationary output assumption is valid. As everything we have done so far has indicated, it is valid as long as the coefficient cluster has one mean (or one tight cluster of means) with high probability, and the variance of the cluster is not too high. In these cases, the coefficient vector takes a value around this mean with high probability. Estimating the mean and variance of the statistics of the output (or assuming the mean is $s(n)$ when the SNR is high) is valid in these cases.

The issue that comes up is if the mean cluster either gets too large or begins to split into multiple clusters. The latter, as we now know, happens in the failure mode case. This is further reason why we expect the soft-decision directed equalizer also to fail- there is still a model mismatch, which becomes highly pronounced at very low SNR values. We theorize, however, that finding the "right" probability density function of the output will not allow the system to be driven into failure, as, intuitively, in that case, we're always tracking the right cluster and not allowing the system to get stuck.

Finally, we mentioned in Section 4.5 that we could for the hard-decision directed case, compute in a closed form what the probability of output error is. Evidently, from Equation 4.35, this is easily done as a mixture of Q-functions obtained by integrating the PDF of Equation 4.35 from $-\infty$ to 0 for $s(n) = 1$. This is also helpful in general to obtain an estimate of the output probability of error we expect from an equalizer given a coefficient vector at a particular time, and in the steady state when it has converged.

Recap and Looking Ahead

In this chapter we considered the issue of analyzing the performance of the RLS adaptation algorithm.

Analysis was performed under hard-decision directed adaptation and the soft-decision directed adaptation procedure, and the insights gained from this were discussed. This is a hard problem, due to the fact that decisions made at a particular time affect future decisions. We applied the theory of Markov processes to obtain a steady-state distribution of the coefficient vector, and showed that in general this was hard to find.

We then chose to look at the steady state distribution of the mean of the coefficient vector. This has advantages over the traditional approach, which looks at the steady state mean and the steady state variance of the coefficient vector [20], as our approach shows the different points about which the coefficient vector could be clustered, and with what proabability they occur, while the approaches in the past assume that there is only one such cluster, which is not necessarily the case.

We then applied this procedure to understanding how adaptive equalizers behave under decision directed adaptation and why we expect the soft-decision directed adaptation procedure to improve this. We noted how different equalizer taps behave under different adaptation procedures, and showed that in general the conservatism of the soft-decision directed adapation guarantees that it is closer to the MMSE solution.

We then looked at an important casatrophic failure mode of adaptive DFEs. We

explained at an intuitive level why it occurs, and also demonstrated that the mean distribution prediction that we perform is capable of predicting this mode, which, to the best of our knowledge has not been done before. We also showed in simulation that there are cases in which the hard-decision directed adaptation procedure fails, but the soft-decision directed adaptation does not, and thus that it has better thresholds for failure. We will see in practice that the improvements can be very significant. We also saw that this was contrary to predictions, and suggested reasons for this.

Finally, we looked at the output statistics of the equalizer and saw that the Gaussian assumption is not a perfect match. The right model would be a Gaussian mixture. But with time-variability, this is a hard model to apply and the Gaussian model works well when there is expected to be only one coefficient cluster. The means are close to the transmitted symbols when the SNR is high. Once again, we will see how this translates into the performance of a practical system at high and low SNRs in Chapter 6. We gave reasons for sticking to the Gaussian model for the remainder of this work.

This brings us to the conclusion of the "theoritical" part of the thesis. In the following chapter, we discuss the implementation of a practical system for underwater communication- a turbo equalizer- using the soft-decision directed equalizer that we derived. Some of the elements of the analysis- in particular, whether to use $s(n)$ as the mean of the Gaussian distribution (high SNR) or to estimate it (low SNR), when the failure mode occurs, etc.- will be important in the design and results. However, with the understanding gained in this chapter, we will be in a position to interpret the results correctly and determine whether or not they are meaningful, and to take as much care as possible about the issues we have discussed in designing the practical systems.

Chapter 5

The Soft-Adaptive Turbo Equalizer

As we have stated in this thesis, a practical system for underwater communication is the turbo equalizer. The turbo equalizer is a system of equalization that takes advantage of coding in the system to iterate between the equalizer and decoder components.

In this chapter we design a turbo equalizer using the soft-decision adaptive equalizer that we designed and analyzed in the last 2 chapters, and show how the adaptation algorithm takes advantage of the priors available in the turbo setup. This turbo system is what we term the *soft-adaptive turbo* equalizer. We start with an introduction to turbo equalization and an explanation of the principles involved.

5.1 Turbo Equalization: An Introduction

This section provides a very brief introduction to turbo equalization. The theory of turbo equalization itself is very rich and could (and has) served as the subject of theses on its own. We refer the reader to [26] for more details on turbo equalization.

Turbo codes, introduced in [6] in 1993 by Berrou, Glavieux, and Thitimajshima, were among the first practical codes to approach the Shannon capacity of a channel. The turbo encoder consists of a pair of convolutional encoders, with an interleaver in between the two. The decoder is a pair of soft-decoders that exchange likelihoods until they agree on the set of input bits.

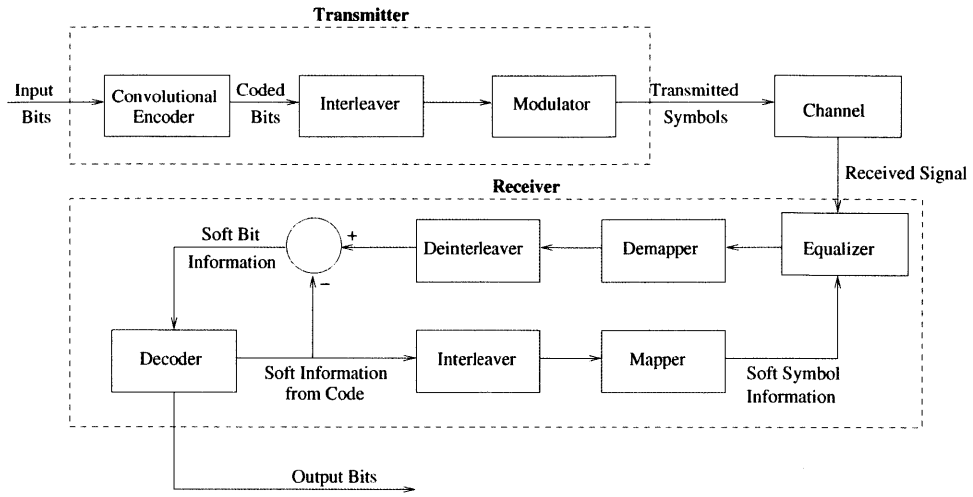


Figure 5-1: Turbo Equalizer- a High-Level Block Diagram

Turbo equalization [11] is an extension of the turbo principle to equalizer systems. The intuition behind turbo equalization is that the discrete-time ISI channel can be thought of as an encoder, except that the channel encoder is unknown. Thus, introducing an additional convolutional encoder and interleaver stage will effectively turn the system into a turbo coding system.

A first, high-level block diagram of turbo equalization is shown in Figure 5-1. This demonstrates the flow of information in the turbo system.

The transmitter side has a conventional convolutional encoder followed by an interleaver. The interleaved bits are then modulated onto a signalling scheme and transmitted. The "second encoder" is the channel, which is independent from the first encoder. So this is effectively a Serially-Concatenated Convolutional Code [4].

The interleaver is critical to the design of the turbo system. The purpose of the interleaver in turbo coding is to shuffle bits such that the errors made by the first decoder do not affect the operation of the second decoder. Thus interleaving enforces independence of the errors made by the 2 decoders.

At the receiver end, the first "decoder" is the equalizer, which essentially decodes the channel. This is where the turbo principle comes in. Rather than making decisions on the symbols and decoding, as would be done in a conventional receiver, the equalizer and decoder pass messages to each other on the likelihood of the symbols

or bits that arise as a result of decoding in each case. They then *iterate* and in each iteration, incorporate the information provided by the other in the previous iteration.

Thus, in Figure 5-1, the equalizer produces probabilities of the tranmitted symbols (rather than the output symbols themselves). The mapper converts this into bit probabilities of the interleaved bits. These are then deinterleaved, and passed to the decoder. The decoder is a soft-output type decoder, which could either be a BCJR Algorithm [2] or a Soft-Output Viterbi (SOVA) type decoder [18].

The output of the soft-decoder is a set of probabilities on the coded bits. These are passed to the interleaver, and mapped back into symbol probabilities. These go back to the equalizer. The equalizer now has to account for these symbol probabilities. In the turbo equalization literature, these are called *a-priori* probabilities, or *priors* on the symbols, as they are equivalent to having some prior knowledge on the symbols. Priors are a kind of soft information (Section 2.3.3).

The one block we have not yet explained is the subtractor before the decoder. This is a very important concept in practical turbo equalization. The purpose of this block is to eliminate the reinforcement of the same information in the turbo loop. In particular, we wish to eliminate any structure that we may have imposed on the data. Specifically, in this case, a trellis is involved in the convolutional code. Thus, the output of the decoder depends on the trellis structure. So we want the input to the decoder to contain information that does not depend on this trellis structure, otherwise reinforcement of the same information in the loop becomes an issue.

In other words, we only want the decoder and equalizer to "see" information that they have not seen before. This is called extrinsic information [26]. Note that the additions and subtractions take place on *likelihood ratios*, which are given by

$$L(n) = \log \frac{\mathrm{p}(b(n) = 1)}{\mathrm{p}(b(n) = 0)} \tag{5.1}$$

The operation is that the bit probabilities coming out from the deinterleaver are mapped into likelihood ratios of the bits. The likelihood ratios from the previous iteration are then subtracted out of this. The resulting extrinsic likelihood ratios can

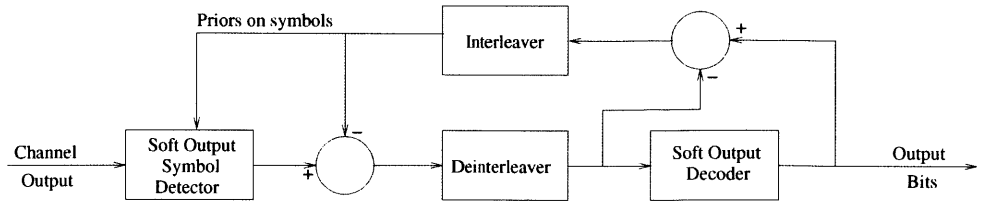


Figure 5-2: The Original Turbo Equalizer [11]

then be passed to the decoder. Depending on the way the decoder is implemented, it may either use likelihood information directly or map them back into probabilities.

We now turn to various approaches taken to incorporating priors into the equalization process in the past. This is the crux of the problem in turbo equalization. We know how to use probabilistic information in the decoding of convolutional codes. How do we do it with equalization, particularly with unknown channels?

5.2 A Brief History of Turbo Equalization

The original turbo equalization concept, proposed by Douillard et al. [11] assumes that the channel is known. For a known channel, the Viterbi algorithm decoder is known to be an optimum detector [41]. Thus, the original concept uses a Soft-Output Viterbi Algorithm as a channel equalizer, labelled in Figure 5-2 as the Soft-Output Symbol Detector (in the language of the paper), as it detects the channel. A convolutional code was applied, which was decoded by a second Soft-Output Viterbi Algorithm (SOVA), labelled the Soft-Output Decoder. These 2 component iterate with the extrinsic information. There are two subtractors here in order to ensure that the trellis structure in each code is not re-used.

The computational complexity of running 2 Viterbi Algorithms is high, so linear solutions were sought. In [56] and [55] Tuchler et al. introduced a linear equalizer based on the Minimum Mean Squared Error (MMSE) cost criterion, and showed that it could replace the Viterbi algorithm as the equalizer with similar results. Furthermore, approximations were introduced which further decreased the complexity of these algorithms. These techniques use the a-priori information as known signal statistics in the MMSE solution, and as these statistics vary with time, the MMSE

solution in this case also varies with time.

In Chapter 4, and in particular, in Equation 4.5, we introduced the MMSE equation without priors. When priors exist, for the time-invariant known channel case, with the notation introduced in Chapter 4, we can show that the MMSE solution for $\mathbf{g}$ is given by:

$$\mathbf{g_0}(n+1|n) = (\mathbf{S} + \mathbf{H}^H \mathbf{V}(n)\mathbf{H})^{-1}\mathbf{r}_{\mathbf{u}s^*} \tag{5.2}$$

where, as before we denote the MMSE solution by $\mathbf{g_0}$. However, the MMSE solution is time-variant, as the priors are time-variant. $\mathbf{V}(n)$ is the covariance matrix of the vector of the past N symbols. The estimates are given by

$$y(n+1) = \bar{s}(n+1) + \mathbf{g_0}^H(n+1|n)[\mathbf{u}(n+1) - \bar{\mathbf{u}}(n+1)] \tag{5.3}$$

where we denote the a-priori mean of a quantity by a bar. Evidently, all the means and variances are computable given the prior probabalities.

A similar solution, using a Decision Feedback Equalizer (DFE) instead of a linear equalizer, was also developed by Lopes and Barry in [30], and was termed a "Soft-Feedback Equalizer". Rather than feeding back hard decisions, as is typically done in DFEs, the expectations of the symbols given the a-priori information was fed back instead. That is, the feedback filter of the DFE contains $\bar{s}(n)$, which is the mean of the symbol. Once again, the channel was assumed to be known, and given the known channel, the coefficients are computed similar to our discussion above.

The application of these concepts to Turbo equalization was investigated in [55], along with some additional results, and practical approximations to these algorithms. More detailed reviews of all these techniques are to be found in [26].

5.2.1 Estimating the Channel

The techniques discussed above require the knowledge of the channel. When the channel is unknown or time-variant, we must first estimate the channel, and can then use the channel estimate in one of the solutions. The former may be done using

one of many well established algorithms for channel-estimation. For example, for time-varying channels, we have seen the RLS algorithm for equalization. It should be obvious that changing the problem statement slightly will allow us to use it for channel estimation as well [20]. One interesting problem, however, is taking advantage of soft-information in the estimation process.

For instance, in [47, 48], Song et al. used estimators based on Kalman filters and RLS based estimators. In this case as well, the prior statistics that may be known about the symbols are used in the algorithms. The desired signal is the channel output, as usual for a channel estimator. The input signal is the mean of the signal input, as computed by a decoder.

In [34], a new least-squares type cost function is proposed by taking expectations with respect to unknown quantities, conditioned on what is known. This becomes the estimate of $\mathbf{h}(n)$ that is computed as follows:

$$\mathbf{h}(n|n) = \arg\min_{\mathbf{h}} \sum_{m=1}^{n} \lambda^{n-m} \mathbb{E}[|x(m) - \mathbf{h}^H \mathbf{s}(m)|^2 | \mathrm{p}(\mathbf{s}(m)), \mathbf{h}(m-1|m-1), \mathbf{u}(m)] \quad (5.4)$$

where, as usual, $\mathbf{h}(n|n)$ denotes the estimate of $\mathbf{h}$ at time n given the data upto time n. In this case, we have the data up to time n while performing the update at time n. This is similar to the Expected Least Squares criterion that we derived in Section 3.2 for the channel estimation context, and is defined to make the estimator of $\mathbf{h}$ independent of the value of $s(n)$ which is unknown. Unlike in the case of the RLS algorithm for the usual least squares criterion, no simple recursive solution exists for it. The covariance matrix has full rank, and so the direct computation of this matrix has a complexity proportional to N^3.

However, we have derived an algorithm for this already, linked it to the EM algorithm and derived a recursive approximation. The difference is that this particular cost criterion is defined specifically for channel estimation. It should thus be observed that the Expected Least Squares Cost Criterion can be used in the channel estimation scenario as well, and the Recursive Expected Least Squares algorithm can be used as an approximation. We will use it in the direct-form context, however, and we now

discuss direct-form adaptation structures for adaptive turbo equalizers.

5.2.2 Direct-Form Adaptive Equalization with Turbo Systems

With known channel coefficients, knowing the mean of the data allows us to introduce a bias into the estimator, as we have seen in Section 5.2. The form of a direct adaptation structure, however, does not allow us to design an exactly similar kind of structure. Taking advantage of the prior information here, therefore is an interesting challenge.

One method, first proposed by Laot et al. in [27], is to use different stages for different iterations of the equalizer. In the method in [27], the first stage is a linear adaptive RLS equalizer, operated (after training) in decision directed mode. After the first stage, a DFE-type structure is used, and the symbol estimates from the previous stage of decoding are used in the feedback filter. For these stages, the coefficients are computed using a decision-directed RLS-based channel estimator. This is a hybrid kind of equalizer.

An equalizer with a similar structure, but based on a different cost criterion was proposed in [22]. In this case, during the first iteration, the equalizer is a conventional Decision Feedback Equalizer (which has a training period). For iterations $k > 1$ (where k is the iteration index) the soft decision $\hat{s}^{(k)}(n) = \mathbb{E}[s(n)|\mathbf{y}^{(k)}]$, is computed in an MAP decoder. This is put into the feedback section of equalizer in the next iteration, and we call the equalizer inputs with these decisions in the feedback filter as $\mathbf{u}^{(k)}(n)$ at time n. At this point, the equalizer coefficients are chosen to minimize the value of

$$J = \sum_{m=1}^{n} |\hat{s}^{(k)}(m) - \mathbf{g}^H \mathbf{u}^{(\mathbf{k})}(m)|^2 \tag{5.5}$$

This is *not* a recursive solution, and $\mathbf{g}$ is constant over one complete iteration. So the least squares solution is done once per iteration. This is therefore not a good solution for time-variant channels. Further, it should be noted that at each iteration, after computing the priors, MAP estimators are run on the resulting data to compute

the $\hat{s}^{(k)}(n)$ (which are then used in the cost function and feedback filter).

How to take advantage of the priors generated during decoding while updating the coefficients of an equalizer when we don't want to estimate the channel directly is a more difficult question to answer. One method that has been proposed is to modify the cost-criterion. One such cost criterion was defined for linear BPSK equalizers by Raphaeli and Saguy in [43]. This criterion is called the mean log-likelihood squared error,

$$J = \mathbb{E}[|y(n) - L(n)|^2] \tag{5.6}$$

where $L(n)$ is the log-likelihood ratio of the symbol. However, this is effectively trying to drive the equalizer output to the log-likelihood, which may not be appropriate as the latter is a non-linear function, while the equalizer is a linear filter.

Another attempt at defining statistical criteria was made using Kullback-Leibler divergence in [45]. The criteria defined here are not specifically for turbo systems, but general criteria for the design of adaptive systems. They take advantage of the knowledge of the p.d.f. of the known signal. Essentially, the criteria involve minimizing the KL Divergence between the P.D.F. of the signal at the output of the receiver and the transmitted signal.

Both criteria defined in the paper essentially reduce to taking expectations of the log-likelihood. The ordering of the expectations is the difference between the cost criteria. Specifically, the 2 criteria are (in terms of our symbols)

$$\begin{aligned} J_1 &= -\mathbb{E}_y \log \mathbb{E}_s \frac{1}{\pi\sigma_v^2} e^{-|y-s|^2/\sigma_v^2} \\ J_2 &= -\mathbb{E}_s \log \mathbb{E}_y \frac{1}{\pi\sigma_v^2} e^{-|y-s|^2/\sigma_v^2} \end{aligned} \tag{5.7}$$

where $\mathbb{E}_y$ and $\mathbb{E}_s$ mean expectations taken with respect to y and s, respectively. As the KL Divergence is related to the EM algorithm, as outlined in [24], we could form a similar relationship between the proposed cost criteria and the expected log-likelihood criteria. Once again, a simple recursive solution to this may not exist. However, suppose we were taking conditional expectations (conditioning on known quantities),

rather than complete unconditional expectations (expectations with respect to all possible random quantities). Then these criteria would look rather similar to the EM criterion, which is what we started from for our derivation. There are thus some interesting connections between the turbo equalization approaches in the literature and the present work.

All of the above discussions lead us to think that the general cost criterion and recursive approximation that we derived in Chapter 3 have applications in turbo equalization as well. This indeed turns out to be the case. In fact, with no modification, the soft-decision adaptive equalizer (or its generalization, the Recursive Expected Least Squares algorithm) can be used for turbo equalization.

While the purpose of this section has been to build insight into how the turbo equalization has been thought of traditionally, there are many other turbo equalizers that are known, and many other ways of using the soft information, including using means- either conditional or conditional- in the feedback filter but not in adaptation, etc. We use some of these algorithms as baselines against which we compare performance. Thus, we first discuss the implementation of this system. Following this, we discuss more in detail some of the direct-form structures that we compare its performance against.

5.3 Designing the Turbo System

In this section we consider the design of what we term the *soft-adaptive turbo equalizer*, which is the turbo system designed using the soft-decision directed adaptive equalizer. The transmitter and channel structure that we are interested in are shown in Figure 5-3.

This is a conventional transmitter. The input bits, denoted $b(n)$ to be transmitted are encoded by a convolutional code. Recursive Systematic Convolutional (RSC) Codes are used, as they are known to be better at spreading information over longer lengths, while not making the code too strong. The encoded bits are interleaved (to make channel errors independent of decoding errors) and then modulated. We

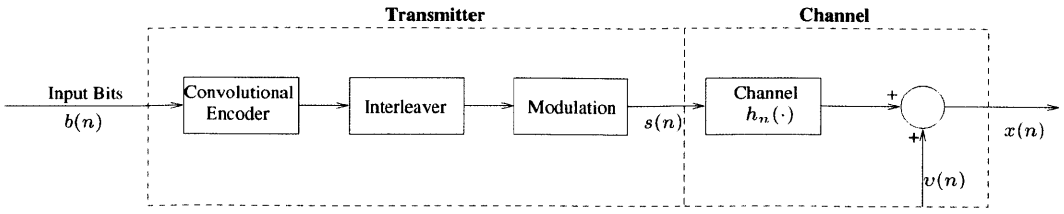


Figure 5-3: Turbo Equalizer- Transmitter

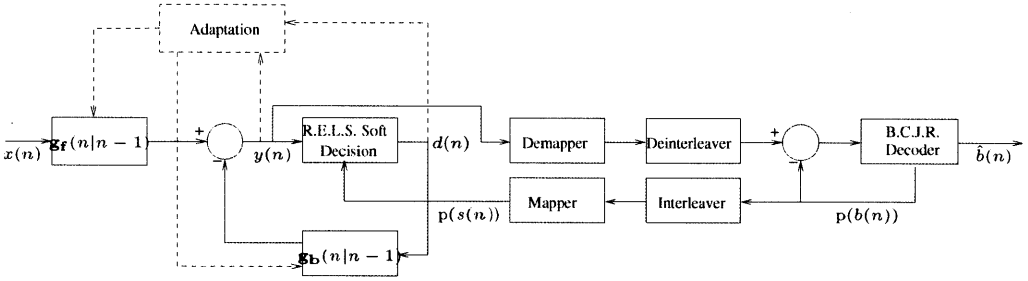


Figure 5-4: Turbo Equalizer- Receiver

use, typically, a coherant PSK scheme of modulation- BPSK, QPSK or 16-QAM are generally used, although this is not required.

At the receiver end, the complete block diagram is shown in Figure 5-4. The key feature of the system is the feedback loop from the BCJR decoder to the equalizer. We explain the features of the system iteration by iteration.

Denote the iteration index by k. In the first iteration ($k = 1$), the equalizer is the usual soft-decision directed adaptive equalizer that we have discussed in Chapter 3. However, the output mapper instead of making decisions on the symbols $s(n)$, maps the outputs into probabilities of the coded bits. We denote these probabilities by $\mathrm{p}_o^{(k)}$ (for output probabilities at iteration k, where output means output of equalizer, and this notation refers to symbol probabilities). These are deinterleaved and passed to the BCJR Decoder in the form of probabilities of the encoded bits. It is of course a simple matter to convert symbol probabilities into probabilities of the interleaved bits when we know the modulation scheme.

The BCJR Decoder is an MAP decoder for trellis based codes. Given the probabilities of the encoded bits, it outputs the data sequence, the probabilities of the bits in the data sequence and (hence) the probabilities of the coded bits. The last of these are interleaved and mapped back into symbol probabilities, which are now

called $\mathrm{p}_i^{(k+1)}$.

Now we consider Equation 3.24, and the Gaussian candidate model of Section 3.4.1. We have since assumed that the symbols are equiprobable, but it seems evident from these equations that there is nothing to stop us from using priors if these are available, as $\mathrm{p}(s(n))$ is already part of these equations! Thus, the RELS algorithm automatically accounts for a-priori information in its derivation. To the best of our knowledge this is the first instance of a systematic adaptation algorithm that accounts for prior information, as well as the soft information at the equalizer output, in the adaptation process.

Thus, all we need to do is to plug in $\mathrm{p}_i^{(k+1)}$ as the priors in the $(k+1)^{\text{th}}$ iteration. As the priors have changed, the equalizer output sequence and output (posterior) probabilities will have changed (and, we expect, improved, as they incorporate the code information), and we denote these by $\mathrm{p}_o^{(k+1)}$.

We go through the process again, but with one caveat. The algorithm should only see the equalizer probabilities that arose as a result of the coded priors $\mathrm{p}_i^{(k+1)}$, but not the coded priors themselves. So, we subtract the prior likelihood ratio from the posterior (or, equivalently, divide and renormalize the probabilities).

The purpose of the subtraction is thus to ensure that the trellis information does not affect the input to the BCJR decoder. The output of the BCJR decoder at iteration k already depends on the trellis, so we subtract this out before feeding in information at time $k+1$. However, for the equalizer, we use the full priors that the decoder outputs. This is because we want the adaptation algorithm to reflect the information from all of the previous iterations rather than just the latest iteration.

Essentially as discussed in Section 5.1, we want the code trellis structure to be incorporated only once, but the rest of the information gathered over iterations to be preserved. As the equalizer itself does not make any assumptions on the trellis structure, we do not subtract out any information before the equalizer priors. However, as the code uses the same trellis in every iteration, it becomes necessary to ensure that the decoder is not simply looking at the information due to the trellis from the previous iteration. Intuitively speaking, the subtractor block accounts for this.

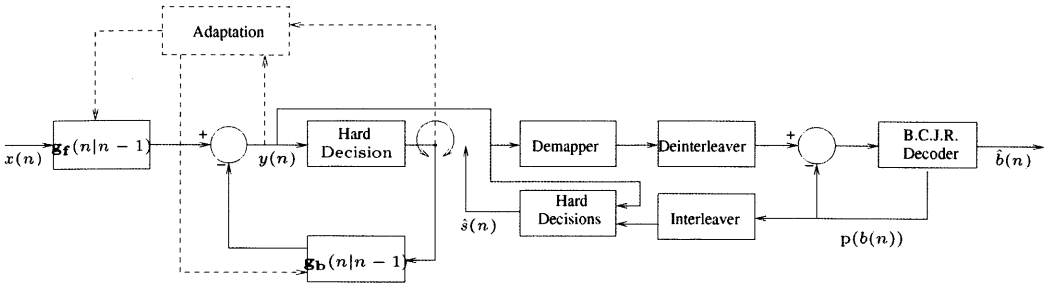


Figure 5-5: Turbo Equalizer from [7]: Soft-Iterative DFE

The process can continue for a set number of iterations, or until the probabilities converge. This is a turbo structure in which the a-priori information is used in the adaptation process. Note that the same information is also used in the feedback filter- we feedback the same symbols that we use in adaptation in this case. Naturally there are other ways of using a-priori information which have been explored in the literature. We now consider some of the most important of these, and we use these for comparison of performance. The comparison will show that using soft information in the adaptation in the way we have described can lead to significant improvements in performance, thereby confirming the importance of the adaptation procedure in the operation of the equalizer.

5.4 Some Important Turbo Equalization Structures

The first structure we present was introduced in [7] and [49]. In [7] it was termed the *Soft Iterative DFE*. The block diagram of the Soft Iterative DFE is shown in Figure 5-5.

The only major difference between the systems of Figures 5-4 and 5-5 is that in the former, the priors are incorporated into soft decisions made according to the RELS update rule. In the latter, however, the priors and the past equalizer outputs are combined to make hard decisions ($\hat{s}(n)$) on the symbol sequence, i.e., hard decisions are made on the symbols. From the second iteration onward, the switch is thrown and these hard symbols are used to drive the adaptation and in the feedback filter, using a process very like hard-decision directed adaptation.

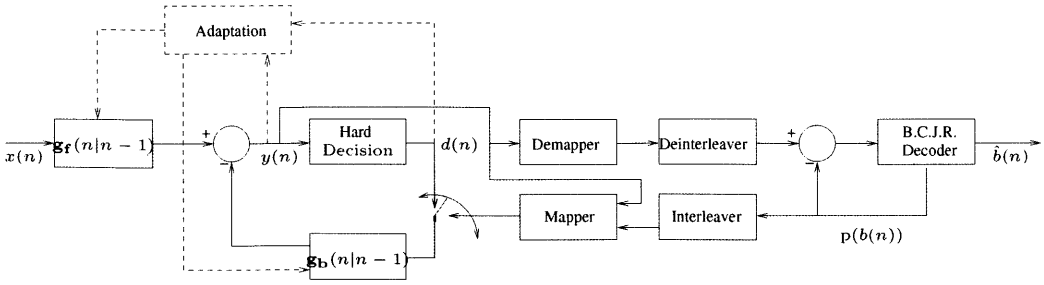


Figure 5-6: Turbo Equalizer from [8]: Soft Information in Feedback Only

Another method, or perhaps more accurately, class of methods that we could use for turbo equalization is using the prior information in just the feedback filter, and using conventional decision directed RLS for adaptation [8]. This is depicted in Figure 5-6.

Note that here the adaptation is always driven by hard decisions. However, after a first iteration in which the feedback filter is fed the hard decisions, it is fed a form of a-priori information. Generally, the a-priori mean is fed, which is obviously dependent only on the prior information, as in [8] However, we could also use either the RELS decision, or a hard decision made as in the Soft-Iterative DFE as discussed above (we don't *have* to drive the adaptation with these). Similarly, we could *only* drive adaptation with these statistics, and drive the feedback filter with hard-decisions.

We test these various methods in practice in the following chapter.

Recap and Looking Ahead

In this chapter, we discussed the design of turbo equalizer systems. We introduced the turbo equalization concepts and took at look at the history of turbo equalization with a known channel and with channel estimator. We then looked briefly at adaptation criteria used in the past for turbo equalization.

We then introduced the soft-adaptive turbo equalizer, which is a systematic algorithm that uses the priors available in turbo equalization in the adaptation process, which is a relatively new concept. We introduced the required components to surround the equalizer and showed how the adaptation and feedback filter use the priors.

We finally introduced other direct-form adaptive turbo systems that have been introduced in the past, against which we will measure the practical performance of the system.

In the next chapter, we demonstrate the practical validity of the algorithms that we've introduced in the thesis. For this, we first describe the KAM11 Acoustic Communication Experiment, the experimental setup and the signals transmitted. We test the various turbo equalizers that we designed in this chapter, and thus demonstrate that the algorithms introduced do indeed have significant practical application.

Chapter 6

Experimental Results

In this chapter we present the results of implementing the algorithms we have presented so far on experimental data. We first provide the relevant parameters for the experiment, and describe the signals transmitted during the experiment. We then show the implementation results.

6.1 The KAM11 Experiment

The Kauai Acoustic MURI 2011 (KAM11) Experiment was a large acoustics communication experiment conducted at the Pacific Missile Research Facility (PMRF) off the coast of Kauai Island, Hawaii from June 23 to July 12, 2011. KAM11 was conducted in shallow water ($40 - 200$m deep) from the RV Kilo Moana. The basic objective of the KAM11 Experiment was the collection of acoustic data for the design and characterization of underwater communication systems in shallow water.

A detailed report on the KAM11 experiment is to appear in [21]. However, we discuss certain aspects of the experiment that are relevant to the results presented. While many source and receiver arrays were used during the experiment, the results present from one pair of these are presented here.

The locations of the transmitter and receiver are shown in Figure 6-1. The transmitter and receiver were about 3km apart. The transmitter consisted of 4 hydrophones separated by 0.5m, with a sampling rate of 39.0625 kilosamples/second.

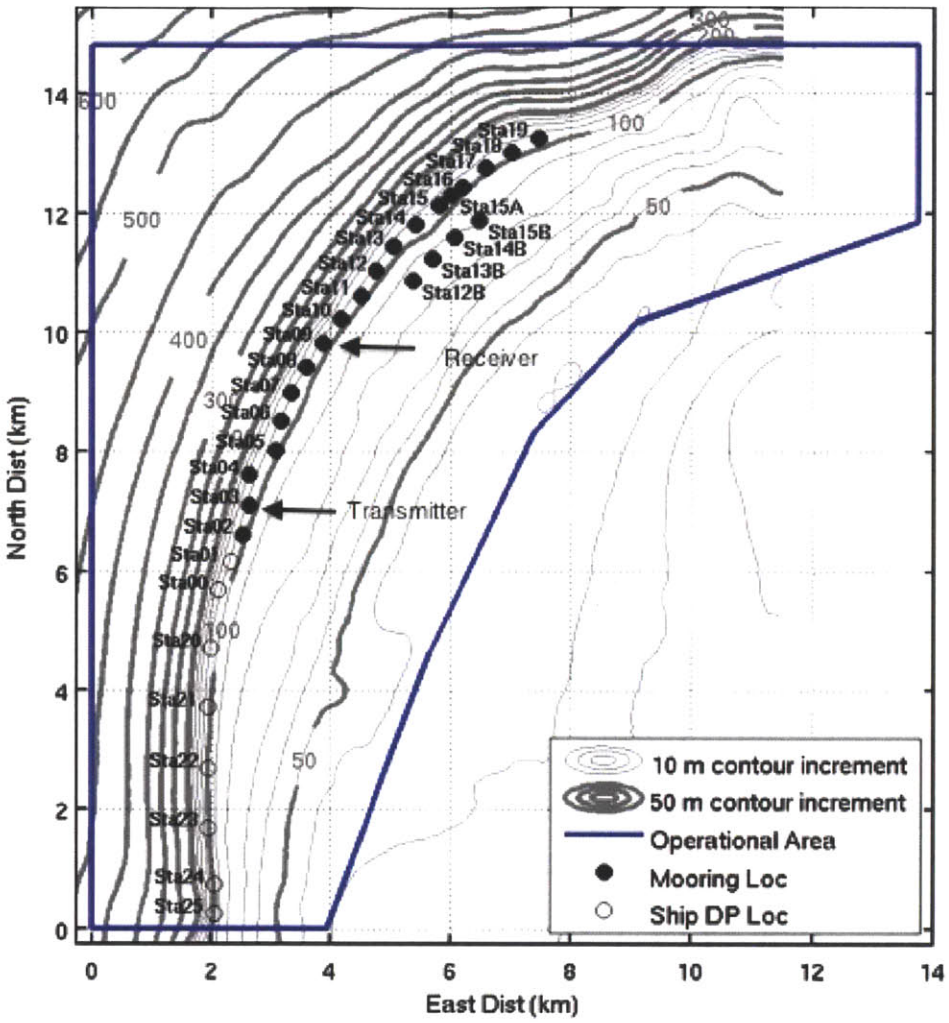


Figure 6-1: KAM11 Mooring Positions Showing Locations of Transmitter and Receiver

The transmitter bandwidth is about 8kHz, centered at 13.5kHz.

The receiver consisted of 24-element Vertical Linear Array (VLA) with 5cm interelement spacing (with $\lambda/2 \approx 15$kHz. The sampling rate of the receiver was the same as that of the transmitter at 39.0625 kilosamples/second.

For this thesis, we processed data from only four receive elements (numbers 2, 9, 16, and 23 numbered bottom to top) so the spacing between these elements was 0.35meters. A justification for doing so is seen in Figure 6-2, which is for an adaptive equalizer with known transmitted symbols. It is evident that as the spacing increases, initially, the performance increases. This is likely to be because at very low spacings,

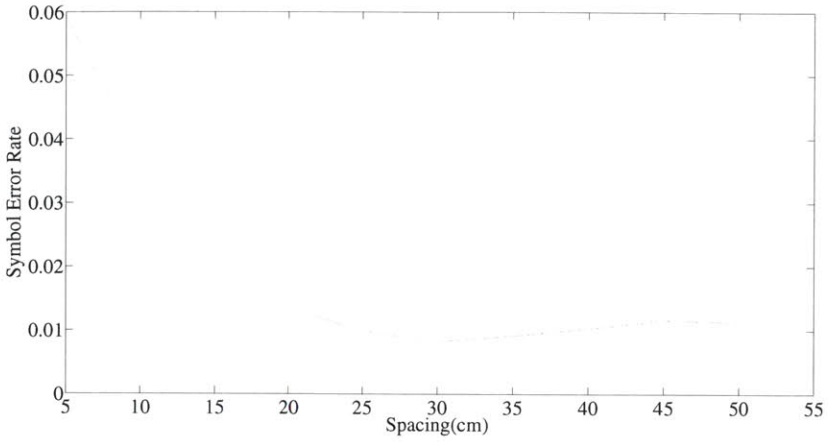


Figure 6-2: The Effect of Interelement Spacing on Performance, Training Mode Equalizer

the array does not have sufficient aperture to spatially resolve the multipath arrivals. Following this, we enter a beamforming domain in which the performance increases up to a point, as the equalizer takes advantage of beamforming. Then, the spacing becomes too large to use beamforming and the paths essentially become independent, so the equalizer can perform space-diversity combining. Evidently, 35cm is quite close to the region of best performance, and thus we nominally choose this distance as the interelement spacing.

6.2 The Practical System

Having briefly described the key features of the experimental setup, we discuss the signals that were transmitted and the practical aspects of the receiver that we use.

6.2.1 Transmitted Signals

A binary sequence was encoded with a Recursive Systematic Convolutional (RSC) Code. The codes used for the encoding were chosen from [5]. In particular, the following were transmitted

1. A rate 1/2 RSC code with constraint length 3, a generator matrix $\begin{bmatrix} 7 & 5 \end{bmatrix}_8$ and a feedback matrix $[7]_8$.

2. A rate 1/4 RSC code (low rate convolutional code) with constraint length 5, a generator matrix $\begin{bmatrix} 23 & 35 & 37 & 27 \end{bmatrix}_8$ and a feedback matrix $[23]_8$. This code is good for low SNR processing.

3. A rate 2/3 RSC code with constraint matrix $\begin{bmatrix} 4 & 4 \end{bmatrix}$, generator matrix $\begin{bmatrix} 15 & 0 & 13 \\ 0 & 15 & 13 \end{bmatrix}_8$ and feedback matrix $\begin{bmatrix} 15 \\ 15 \end{bmatrix}_8$.

4. A rate 1/2 RSC code with constraint length 6, generator matrix $\begin{bmatrix} 67 & 45 \end{bmatrix}_8$ and feedback matrix $[67]_8$.

5. A rate 1/16 Super Orthogonal Convolutional Code [57] with feedback polynomial $[45]_8$.

For the purpose of this work, we primarily present the results of the first (rate 1/2) and the second (rate 1/4) codes. This is because they demonstrate the important features of performance that we require in the medium and low SNR regimes. Other results parallel these.

The coded bits are passed through a random interleaver with interleaver seed 4000. A random interleaver uses a random permutation of the input bits which depends on the interleaver seed. The bits are then collected and modulated onto one of BPSK, gray-coded QPSK or gray-coded 16-QAM signalling schemes. The number of symbols for this transmitter is 24000. These symbols are modulated onto a Gaussian pulse of length 16, so that we transmit 16 samples/symbol. This processing is done at a sampling rate of 100 kilosamples/second. These symbols are then resampled to the system sampling rate of 39.0625 kilosamples/second, modulated to a carrier frequency of 13kHz and transmitted over the channel.

6.2.2 Receiver Details

The receiver structure is the turbo equalizer of Figure 5-4 with the feedforward equalizer being a multichannel input. The input is first synchronized, basebanded and

filtered. The baseband input is downsampled to 4 samples/symbol Thus, we use a fractionally-spaced (1/4) equalizer (future work may focus on 2 samples/symbol).

The equalizer is implemented in the frequency-domain [39] for stability reasons, and to keep the required number of taps per channel low so as to keep the complexity low. The limit frequencies of the Fourier transform are taken as ± 5500Hz.

The filter lengths are adjusted for different datasets (trial and error). However, the general feedforward filter lengths used are $1.5-7$ms per channel and the feedback filter lengths are $3-10$ms per channel. The forgetting factor λ for the RLS algorithm is 0.995.

The first 4000 of 24000 transmitted symbols are taken as known at the receiver and used for initialization and training. The decoder for the convolutional code is a BCJR decoder. Note that these are the nominal values used, and we will specify particular values where they deviate from these.

6.2.3 The Time-Updated RLS Algorithm

There is a time-varying Doppler shift in the underwater acoustic communication channel. This would need to be compensated for at the receiver. In practice, a modification of RLS termed the Time-Updated RLS Algorithm (TU-RLS) [12] is used for this. We briefly describe this algorithm here.

The idea behind the TU-RLS update algorithm is to multiply the filter coefficients by a matrix $\mathbf{F}(n)$ of coefficients that compensate the Doppler shift at time n. Thus, we have the update equations

$$\text{Time Update Step:} \quad \mathbf{g}(n|n) = \mathbf{F}(n|n-1)\mathbf{g}(n|n-1) \tag{6.1}$$

$$\text{RLS Step:} \quad \mathbf{g}(n+1|n) = \mathbf{g}(n|n) + \mathbf{k}(n)[d(n) - \mathbf{g}^H(n|n)\mathbf{u}(n)]^* \tag{6.2}$$

Note in Equation 6.1 that we used $\mathbf{F}(n|n-1)$, which, consistent with our notation up to this point, is the estimate of $\mathbf{F}(n)$ given the data up to time $n-1$. The desired symbol $d(n)$ is chosen according to the usual methods- training and hard decisions or soft decisions. As we shall see in a moment, adding the time-update step does

not in any way affect the derivation we carried out in Chapter 3. Thus, the decisions described therein can continue to be used.

Moreover, it has been observed that the matrix $\mathbf{F}(n)$ is nearly diagonal. So, it can be replaced approximately by a vector $\mathbf{f}(n)$ and the matrix multiply can be replaced by an element by element product (denoted by $\odot$), so that Equation 6.1 can be written as

$$\mathbf{g}(n|n) = \mathbf{f}(n|n-1) \odot \mathbf{g}(n|n-1) \tag{6.3}$$

Note that this step rotates the coefficient vector subspace by the phases of the elements in vector $\mathbf{f}(n|n-1)$. In order to keep the RLS algorithm consistent, it becomes necessary to also rotate the column space of the matrix $\mathbf{R}_{\mathbf{u}}^{-1}(n)$, which is done by the additional algorithmic step

$$\mathbf{R}_{\mathbf{u}}^{-1}(n) = [\mathbf{f}_p(n|n-1)\mathbf{f}_p^H(n|n-1)]\mathbf{R}_{\mathbf{u}}^{-1}(n) \tag{6.4}$$

where we define $\mathbf{f}_p(n|n-1)$ as the element-by-element phase of the complex values in $\mathbf{f}(n|n-1)$. Essentially, we normalize each element of the vector $\mathbf{f}(n|n-1)$ to have a magnitude 1 and rotate the column space of $\mathbf{R}_{\mathbf{u}}^{-1}(n)$ by the resultant vector.

The question remains of how to choose $\mathbf{f}(n|n-1)$. This vector is tracked using the NLMS (Normalized LMS) algorithm. For the Doppler compensator $\mathbf{f}(n|n-1)$, the "effective" input vector can be written as

$$\mathbf{u}_d(n) = \mathbf{g}^*(n|n-1) \odot \mathbf{u}(n) \tag{6.5}$$

and the NLMS update equations are given by

$$\hat{\mathbf{f}}(n+1|n) = \mathbf{f}(n|n-1) + \mu\mathbf{u}_d(n)[d(n) - \mathbf{g}^H(n|n-1)\mathbf{u}(n)]^*/[\mathbf{u}_d^H(n)\mathbf{u}_d(n)] \tag{6.6}$$

$$\mathbf{f}(n+1|n) = \hat{\mathbf{f}}(n+1|n)/\max(1, |\hat{\mathbf{f}}(n+1|n)|) \tag{6.7}$$

Note that by $|\mathbf{x}|$, we mean the element-by-element magnitude of the elements of $\mathbf{x}$, not the norm. That is, we take the larger of 1 and the largest magnitude

of the elements of $\hat{\mathbf{f}}(n+1|n)$ and normalize the elements by this to give us the next Doppler shift matrix. μ is the step-size of the NLMS algorithm, which for the KAM11 experiments is taken to be 3.03×10^{-4}.

One final important point regarding these algorithms is that none of the assumptions and discussions about the desired symbols and the derivation we went through in Chapter 3 are affected by the time-update step. We can go through the same derivation as before and apply this step a-posteriori as we have done, without losing generality in the algorithm we derived, as all we have done is to add a tracking factor into the algorithm. Thus, we can simply use the $d(n)$ that we derived there with this algorithm in order to track the Doppler shift.

We now present the results of the experiments, both in standalone and turbo equalizer settings, with the various modifications that we have proposed and analyze them.

6.3 Results

We come to the results of the implementation. At the output of the equalizer, we consider the Mean Square Error between the filter output of the equalizer and the transmitted symbols, given by

$$\hat{e}_{\mathrm{MS}}(n_{\mathrm{symbols}}) = \frac{1}{n_{\mathrm{symbols}}} \sum_{m=1}^{n_{\mathrm{symbols}}} |s(m) - y(m)|^2 \tag{6.8}$$

At the output of the decoder, we consider the Bit Error Rate of the decisions made on the bits. We consider these as a functions of the number of iterations at different SNRs.

We also look at the Bit Error Rate of the decoder as a function of the Mean Square Error at the equalizer output. This is a good indication of how much information there is in the equalizer output. For example, if method A has a higher bit error rate for the same Mean Square error, we could conclude that the outputs have less information that the decoder could use.

6.3.1 Adaptation, Feedback, or Both?

Mean Square Error Performance

First, we consider the results of the Soft-Adaptive Turbo System against the class of systems in [8], which is represented in Figure 5-6. The original system in the paper, as we have described, considers feedback of the *a-priori* mean (given by $\sum s\mathbb{P}(s(n) = s)$), where the priors come from the decoder. The adaptation is with conventional RLS, which is hard decision directed when the symbols are unknown.

This leads us to consider the following systems apart from the Soft-Adaptive Turbo System- we consider the hard decision directed system with means fed back (as in the paper, which we label "Hard Decision Directed Adaptation, Means Feedback") and with RELS decisions fed back (labeled "Hard Decision Directed Adaptation, Soft Feedback"). Note that when we say Soft Feedback, we mean the RELS Soft Decision (and, as we've discussed already, the Gaussian assumption with means as the symbol as the equalizer output statistics).

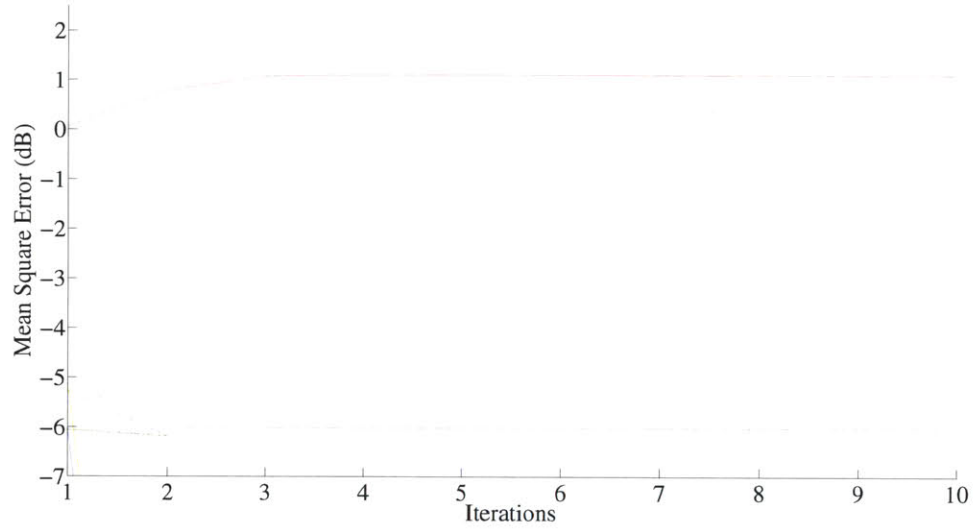
We also consider the Soft Adaptation procedure (i.e., RELS adaptation) but with hard decisions and means fed back, aside from the Soft-Adaptive Turbo Procedure, which involves Soft Adaptation and Soft Feedback. Finally for completeness, we consider feedback of means and RELS decisions, with "cheat" mode adaptation (i.e., the adaptation procedure has access to the true symbols).

This is tested in the setup discussed in Section 6.2.2 with 4000 training symbols. The native SNR of the received data was about 24dB. We test the system by adding noise to corrupt to various noise levels ranging from 24 to 9dB. The code in this case is the Rate-1/4 convolutional code that we discussed in Sections 6.2.1

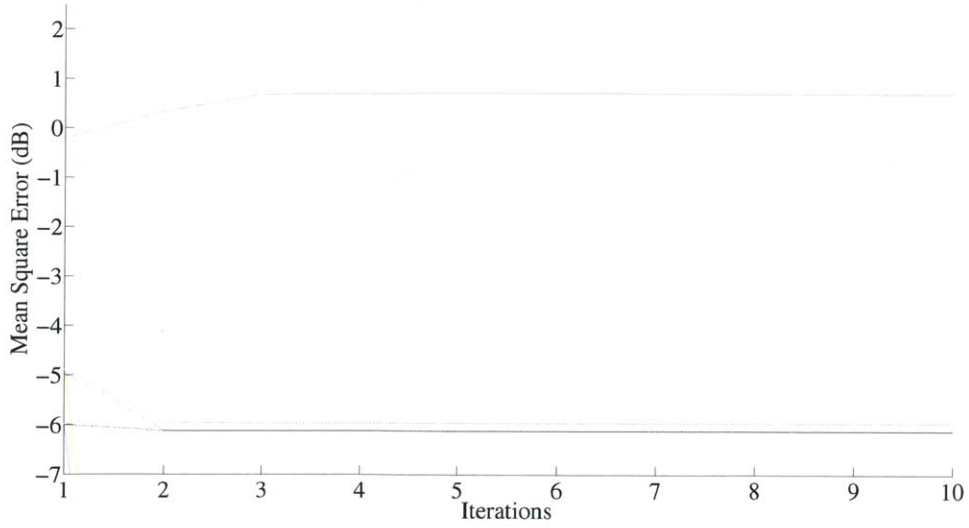
These results are presented in Figure 6-2. In this case we consider the Mean Square Errors of the equalizer output as a function of the number of iterations. The results are subject to a number of interesting observations.

First consider the different adaptation scenarios. As expected, the training mode adaptation scenarios perform far better than the others. More interestingly, the results are relatively invariant to the SNR and number of iterations. The training

Training Adaptation, Means Feedback
Training Adaptation, Soft Feedback
Hard Decision Directed Adaptation, Means Feedback
Hard Decision Directed Adaptation, Soft Feedback
Soft Decision Directed Adaptation, Hard Feedback
Soft Decision Directed Adaptation, Means Feedback
Soft–Adaptive Turbo Equalization

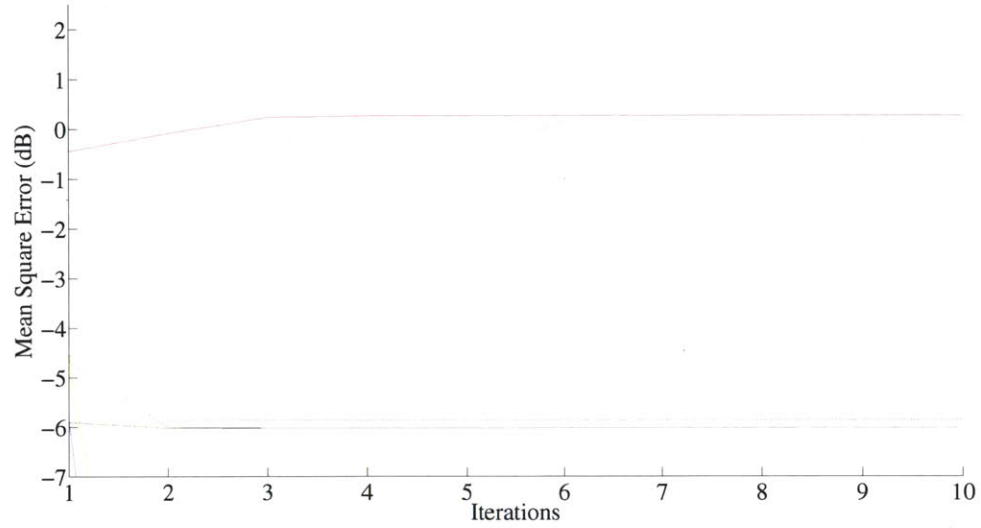


(a) SNR = 24dB

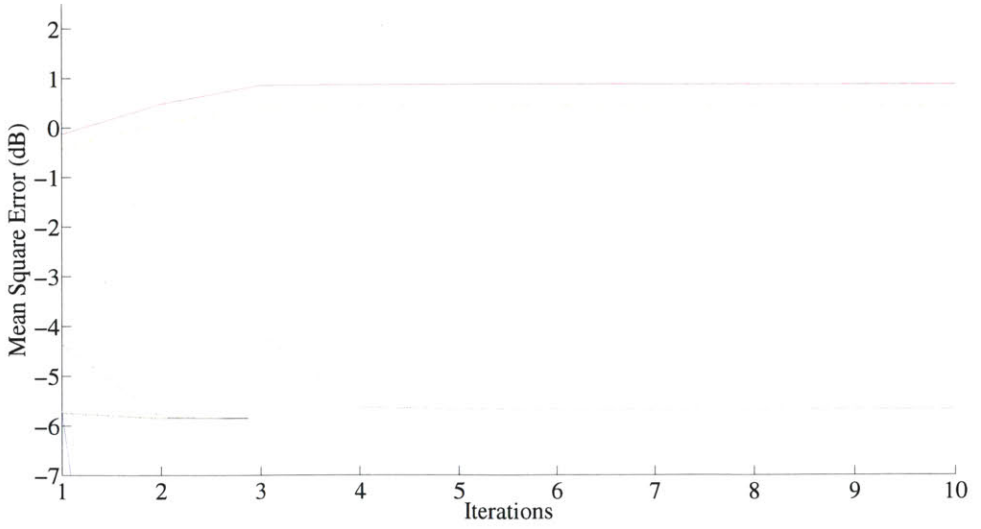


(b) SNR = 21dB

Training Adaptation, Means Feedback
Training Adaptation, Soft Feedback
Hard Decision Directed Adaptation, Means Feedback
Hard Decision Directed Adaptation, Soft Feedback
Soft Decision Directed Adaptation, Hard Feedback
Soft Decision Directed Adaptation, Means Feedback
Soft–Adaptive Turbo Equalization

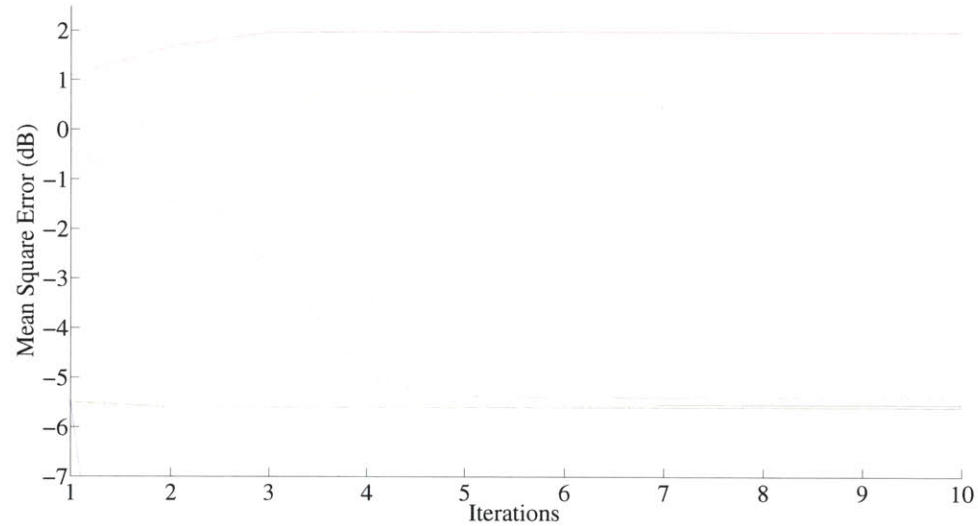


(c) SNR = 18dB

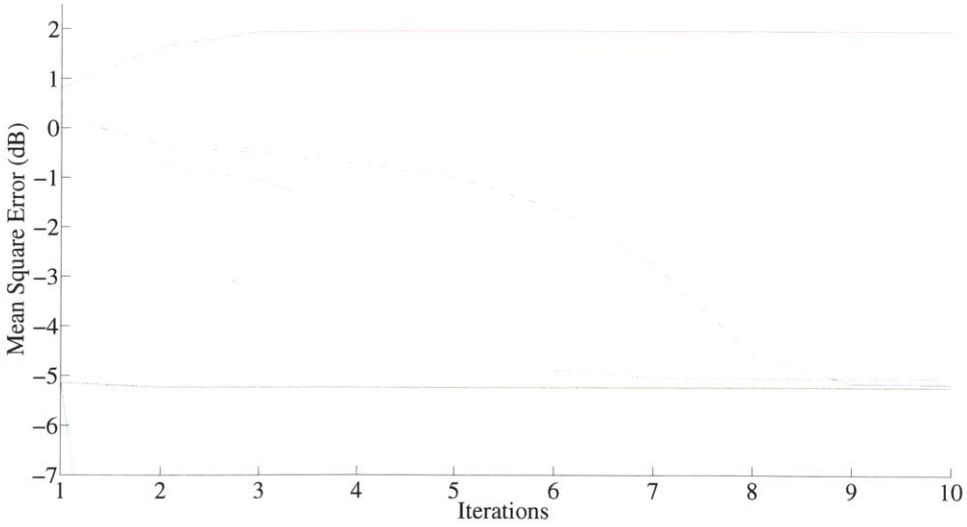


(d) SNR = 15dB

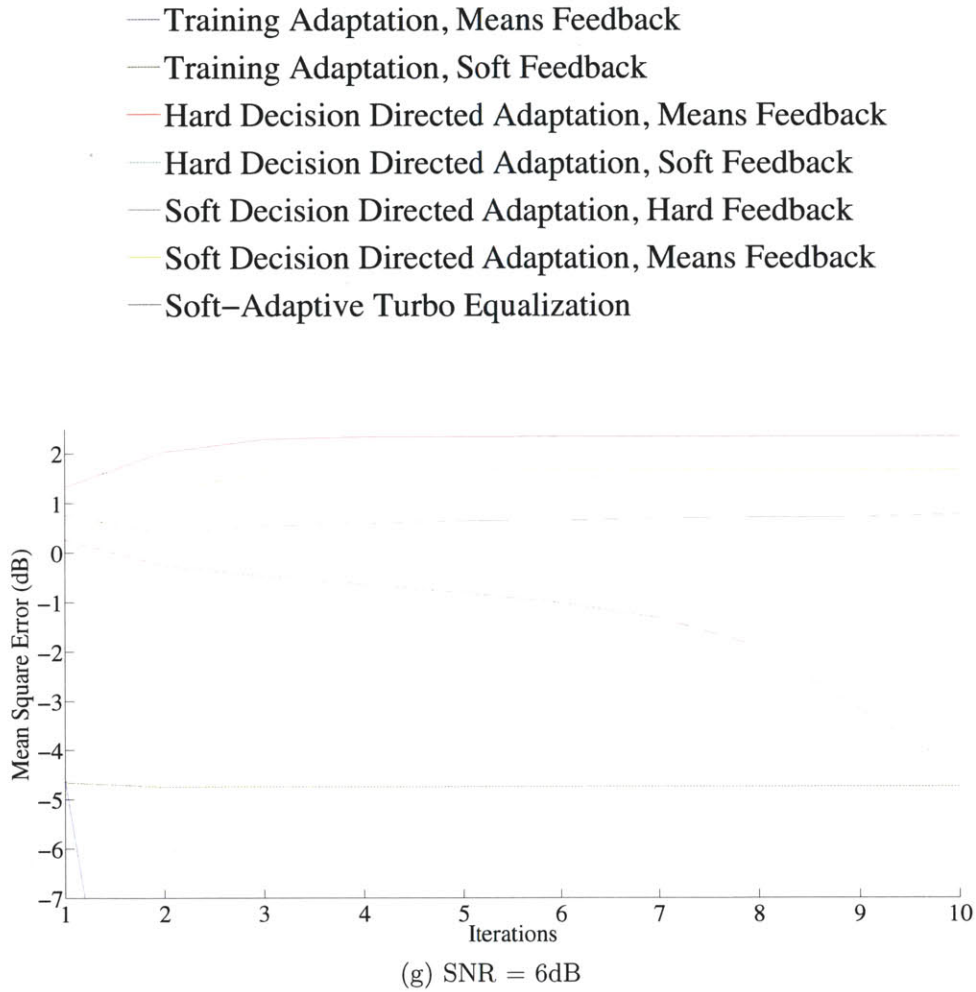
Training Adaptation, Means Feedback
Training Adaptation, Soft Feedback
Hard Decision Directed Adaptation, Means Feedback
Hard Decision Directed Adaptation, Soft Feedback
Soft Decision Directed Adaptation, Hard Feedback
Soft Decision Directed Adaptation, Means Feedback
Soft–Adaptive Turbo Equalization



(e) SNR = 12dB



(f) SNR = 9dB



(g) SNR = 6dB

Figure 6-2: Plot of Mean Square Error of the Equalizer Output as a Function of Iteration at Various SNRs and for Different Turbo Schemes

mode with means fed back hits a mean-square error floor of about -20dB and with soft decisions fed back the error floor at high SNRs is ~ -7dB. The axis is not shown all the way down to this in order to show the details of other plots. Thus, it is clear that these systems are hitting their limit of performance early. They don't take advantage of priors in the same way as the others- the convergence to the error floor happens over one or two iterations rather than improving over multiple iterations.

The hard-decision directed adaptation based systems do not seem to be able to equalize this particular channel reliably even at high SNRs. The equalizer performs so poorly in the initial iterations that the decoder converges down the wrong path of the trellis. Once again, however, changing the feedback mechanism does not change performance by as much as we might expect. So we see that while the feedback mechanism is important to the functioning of the equalizer, it does not take advantage of priors to the extent that we would like.

Another important point to note here is that while feeding back the means does indeed improve performance in the training mode adaptation case, hard-decision directed adaptation is a different scenario. We expect that the first iteration is important in this case. If the code can correct the errors in the equalizer output, we would expect the performance to improve over iterations.

Coming to the soft-adaptation equalizers, they evidently have a sufficiently small Mean Square Error that they can take advantage of multiple iterations and improve performance over a few iterations. At high SNRs with soft-decision directed adaptation, feeding back the a-priori means is extremely beneficial in terms of Mean Squared Error performance. In this regime, the means in the feedback filter evidently prevent error propagation. We see this in training mode as well.

At lower SNRs, however, this is no longer the case- the Mean Feedback, Soft Adaptation case is the first of the Soft Adaptation scenarios to break down. This is reasonable, because at lower SNRs the means are expected to be close to zero as the code has a large element of doubt in its estimates, so feeding these back essentially makes the DFE a linear equalizer. Thus, feeding the means back, which is good in terms of MSE performance at high SNRs, is no longer a good strategy. Furthermore,

as we will discuss soon, good MSE performance does not translate into good decoder performance.

The equalizers with Soft-Decision Adaptation and Hard and Soft feedback are quite close in terms of performance. At higher SNRs, soft feedback seems to have a small advantage, but at lower SNRs, hard decision feedback seems to work better. This is possibly because the soft decisions are a little too under-confident at very low SNRs, making the equalizer tend toward a feedforward equalizer. However, the important thing to note is that even in this case, the equalizer is still taking advantage of priors, where the hard-decision directed adaptation gives up. Once again, the adaptation process has an important

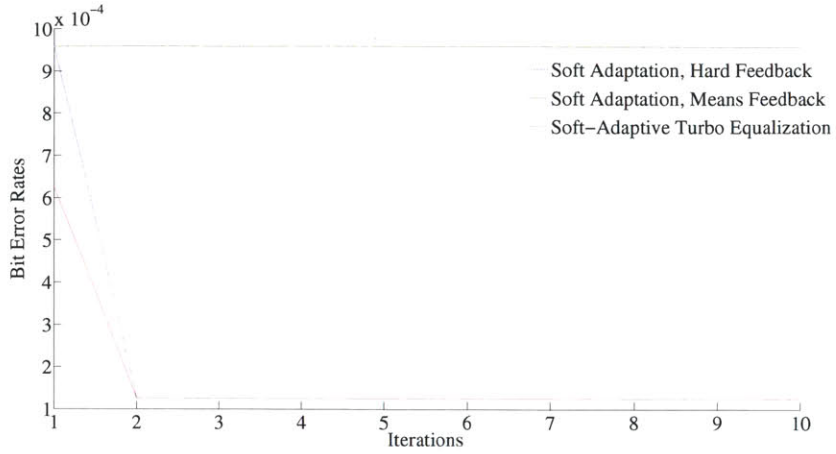
So we infer that the first iteration sets a cut off point for whether the turbo process is likely to be of use. This is one area where Soft-Adaptation helps, as even in the absence of priors, it generally performs significantly better than hard-decision directed adaptation. Even if it improves the performance only to the extent that the decoder corrects some errors and does not get stuck on the wrong path through the code trellis, this is sufficient for the turbo equalizer.

Second, it takes advantage of priors over iterations to a larger extent that just using them in feedback. This allows more rapid convergence to the best solution.

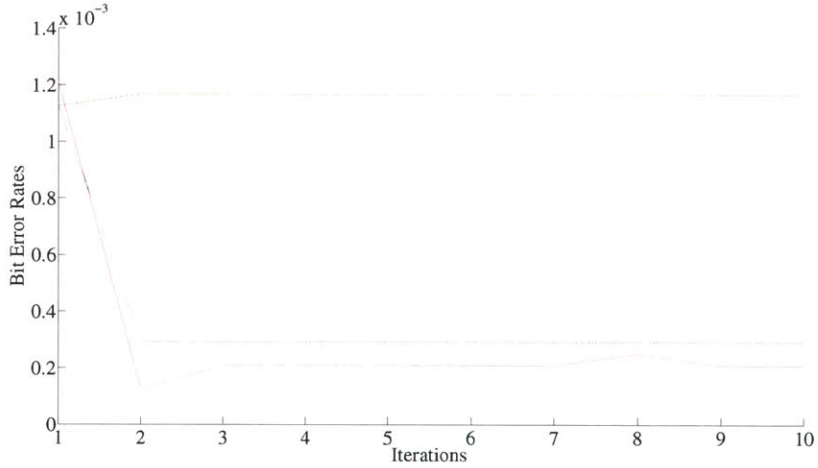
Bit Error Rates of Soft Adaptation Schemes

As we have seen, the soft-adaptation schemes are practical and work fairly well for this channel in terms of MSE performance. We present, for the same datasets as above, the bit error rates as functions of iterations for these schemes. These results, presented in Figure 6-3 back up the effects we have been talking about. Specifically, we have that feeding back soft decisions has a small advantage at high SNRs, but at low SNRs we may do better by feeding back hard decisions.

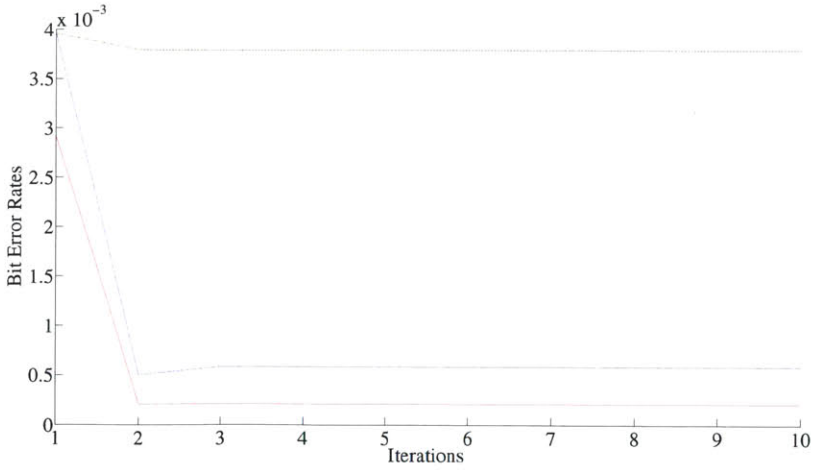
However, we spot one interesting property of the plots, on comparing Figures 6-3 and 6-2. Specifically, comparing the soft-decision directed adaptation with means fed back in the two cases seems to present a contradiction. We mentioned that there is an advantage in feeding back means at high SNRs and indeed, in terms of the Mean



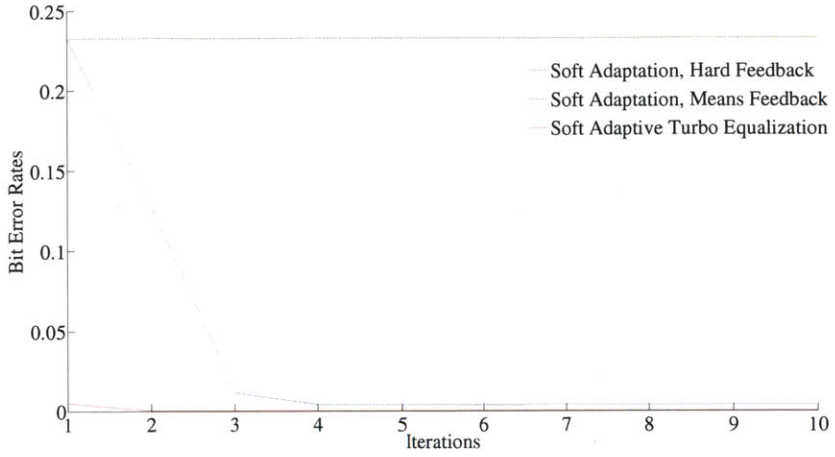
(a) SNR = 24dB



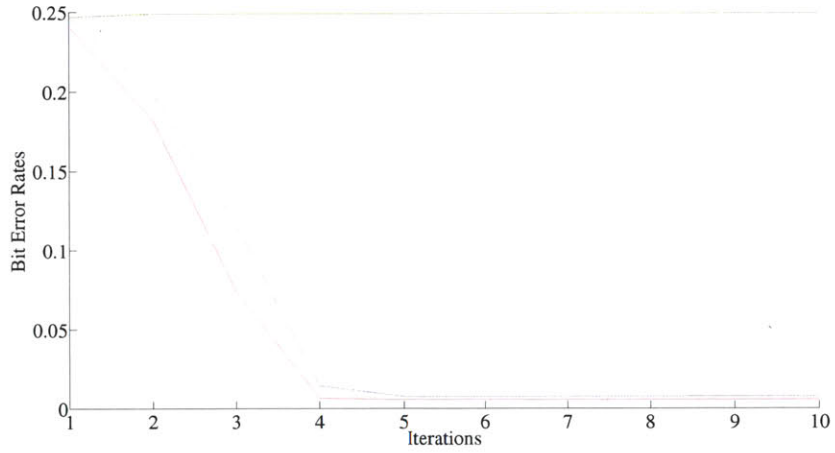
(b) SNR = 21dB



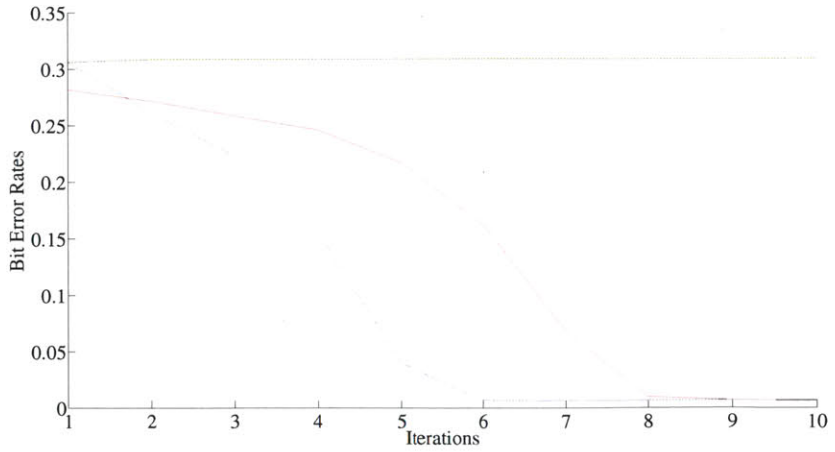
(c) SNR = 18dB



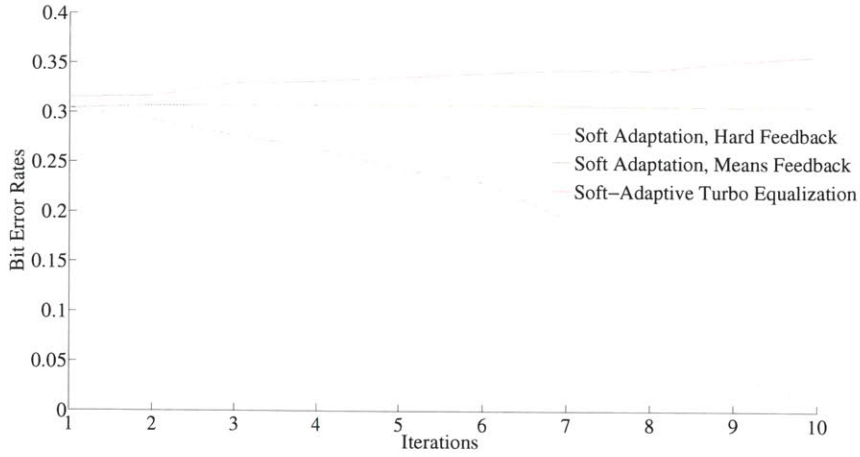
(d) SNR = 15dB



(e) SNR = 12dB



(f) SNR = 9dB



(g) SNR = 6dB

Figure 6-3: Plot of Bit Error Rate of the Decoder Output as a Function of Iteration at Various SNRs for Various Soft Adaptation Based Turbo Schemes

Squared Error at the equalizer, there is. But Figure 6-3 shows that in spite of this, there is no advantage in terms of the error rate at the decoder output.

While this is a strange result, it can be interpreted as follows: a very small mean squared error means that the signals at the output of the equalizer have extremely high confidences, and that they are usually right. However, a very strong confidence on a few wrong symbols can throw off a BCJR decoder. The moral of this is that we would prefer to have more doubt on correct symbols than confidence on wrong ones. Thus, although feeding back hard symbols and RELS decisions leads to a larger output MSE of the equalizer, this translates to a better Bit Error Rate of the decoder.

Interestingly, at 6dB, as with the mean squared error performance, the hard-feedback technique converges, while the soft-feedback technique does not. This, as we mentioned before, could be because the soft decisions are quite close to 0 when unreliable, which may be making the equalizer "look" like a feedforward equalizer. At higher SNRs, the Soft-Adaptive Turbo Equalizer typically beats the other two in terms of performance. Thus, there is possibly a threshold SNR at which the soft decisions no longer hold enough useful information to justify using them in feedback (although, as we have seen, they always help in adaptation).

Finally, the turbo concept with adaptation evidently works well in practice. The

eventual error rate is being brought down to quite a small value $\sim 10^{-3}$, even at the relatively low SNRs we have been considering.

Summary

We summarize the results of this section below:

- First, regarding the adaptation techniques investigated:
  - In the training mode, the equalizer converges quickly to an error floor
  - In hard-decision directed mode, for this particular channel at least, the first iteration does not succeed, so that the code does not get a chance to correct any errors.
  - With soft-decision directed adaptation, the equalizer does in many cases converge. The performance depends on what we choose to feed back, however, it is clear that the adaptation process itself is able to take advantage of the priors.
- Among the soft-adaptation techniques, at low SNRs, the hard-decision feedback seems to be doing better. We hypothesize that the soft-decisions and means are close to 0 at low SNRs, making the equalizers in these cases essentially feedforward.
- At very high SNRs, feeding back means does have an advantage in terms of MSE performance. This is observed for the training mode equalizer as well. However, it crashes at relatively high SNRs (stops working at, in this case, 15dB). Also, the BER performance is not very good for this scheme as seen from the second set of plots. In other words, the good MSE performance does not translate to a good BER performance.
- The Soft Adaptive Turbo Equalizer is good on average- it converges at a variety of SNRs and the performance degrades gracefully for the most part. However, this is not to say that using hard feedback or means feedback may not be good

at some SNR regimes. Without a full analysis of the feedback filter and how it relates to the turbo loop (which has not been done in this thesis) it is hard to say which one is optimal or when to switch between them.

- However, while the performance characteristics depend on the feedback schemes, the soft adaptation schemes in general can take advantage of priors. This is evident from the fact that no matter what the feedback scheme, we see some performance improvement from the soft adaptation scheme (unless, of course, it crashes).

These results and ideas are encouraging, and show that using turbo priors in the adaptation algorithm gives us a significant improvement over using it in just feedback symbols. We have also seen that it allows faster convergence to the error floor. We will now see a further instance of this, and improve the PDF model that we are using to further improve the low SNR reliability.

6.3.2 Rate 1/2 Code Results

In this section, we consider the results of implementing the system with a Rate-1/2 code, specifically, with code number 1 from Section 6.2.1. This code was also used in [55].

We consider the Soft-Adaptive Turbo Equalizer that we have designed and we compare it against the Soft-Iterative Turbo Equalizer of Figure 5-5 (presented in [7]) as a test against a similar, direct-form adaptive turbo equalizer system.

However, we test one more effect in this section. We have seen in Section 4.6 that the output of the equalizer conditioned on the transmitted symbol can be assumed to be a stationary Gaussian process with a mean whose magnitude is less than the magnitude of the symbol. Further, the mean decreases as the noise variance increases.

We have assumed that the mean of the Gaussian process is the symbol, but if the analysis is to be believed, this is suboptimal. Can we thus gain some performance by explicitly estimating the mean? In order to test this, we look at the mean square error of the equalizer output as a function of iteration, the bit error rate of the decoder

output as a function of iteration, and the bit error rate of the decoder output as a function of the mean square error of the equalizer output in a particular output.

We consider these at different SNRs as before. The native SNR of this dataset is also close to 24dB, and we consider SNRs down until 9dB. The results are in Figures 6-4 through 6-9.

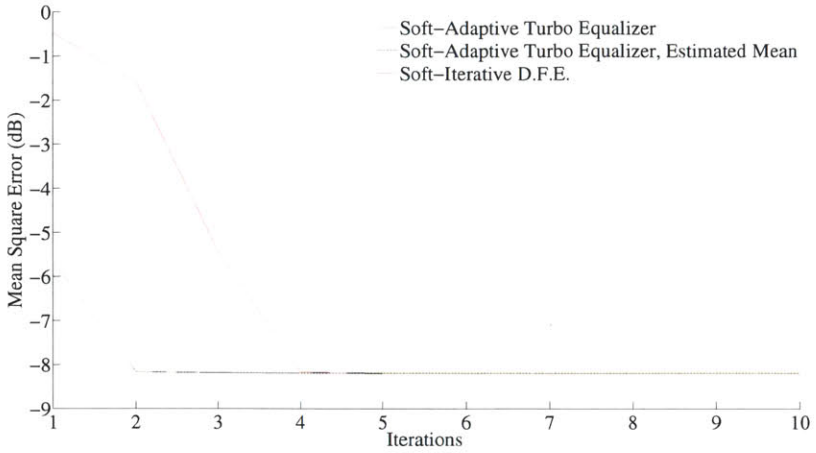
First we consider the difference between the Soft-Adaptive Turbo Equalizer with and without the estimated means. The results here are not surprising. At low SNRs, assuming the means are the symbols runs into problems as the PDF differs significantly, and the results are indicative of this, as at low SNRs estimating the mean gives us a performance improvement.

One surprising fact here, however, is that at high SNRs the assumption that symbols are the means actually performs better than estimating the mean. A possible reason for this is that the assumption makes the algorithm a little more confident about its decisions than estimating the mean, which appears to be slightly better in practice at high SNRs.

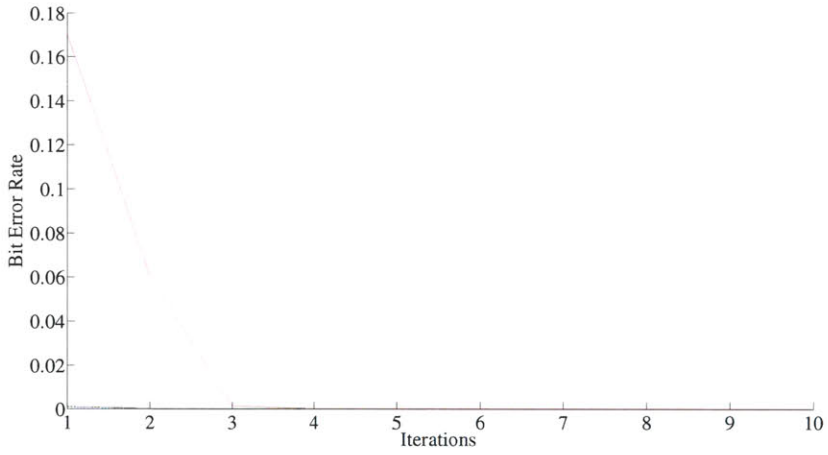
Comparing these schemes to the Soft-Iterative DFE, we observe that there is a significant improvement in the number of iterations required to achieve convergence to the steady state error rate. This is an excellent results, as it indicates we can significantly improve performance in turbo system by incorporating soft information into the adaptation process. Consider for instance Figure 6-5 at 21dB. In this case, all 3 systems converge to the same error floor of -8dB, and a bit error rate of 1.25×10^{-4}. However, while the 2 soft-adaptive turbo procedures converge in 2 iterations, the Soft-Iterative DFE takes 5 iterations to converge. Similarly at 24dB and 18dB.

At 15dB (Figure 6-7, there is an interesting effect, where the Estimated Mean Soft-Adaptive Turbo system and the Soft-Iterative DFE actually achieve a better error floor than the usual Soft-Adaptive Turbo system. This is one of the cases in which either the modeling error becomes an important issue for the error floor. We would expect that a perfect model would achieve the best possible performance.

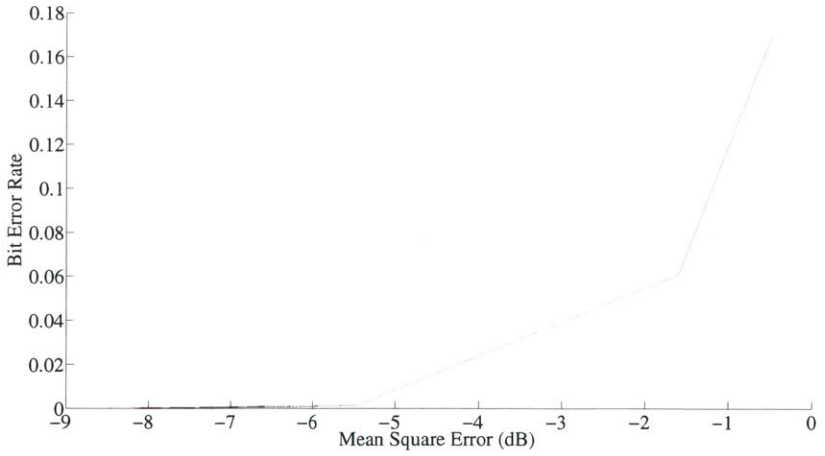
We observe that there are also cases in which the Soft-Iterative DFE fails but the Soft-Adaptive Turbo System converges (Figure 6-9). This is due to the hard-decision



(a) Mean Square Error of Equalizer Output vs. Iteration

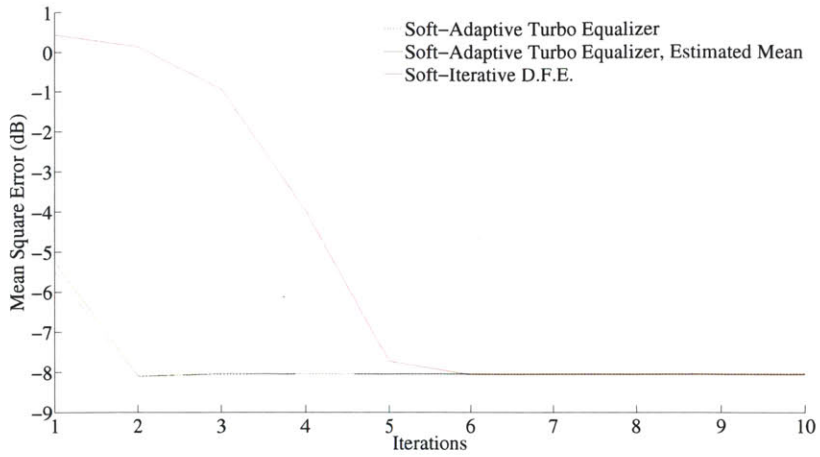


(b) Bit Error Rate of Decoder Output vs. Iteration

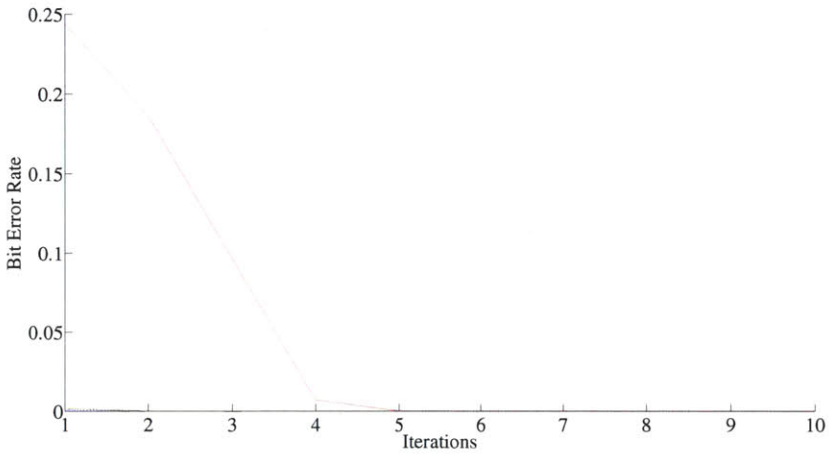


(c) Bit Error Rate as a function of Equalizer Mean Square

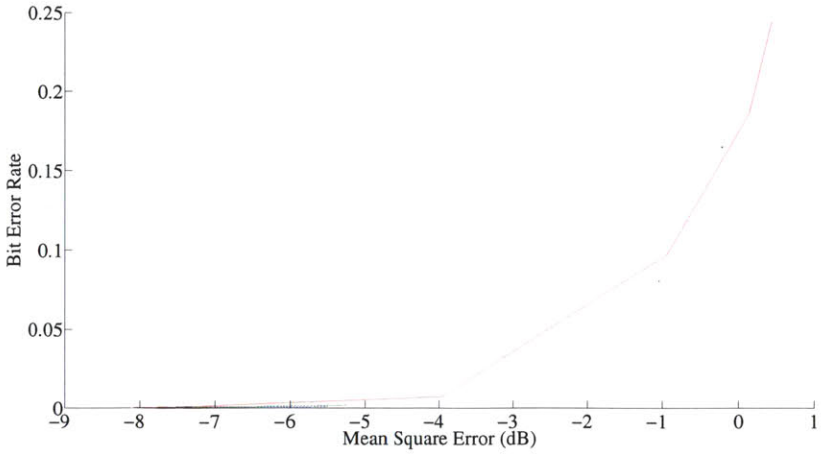
Figure 6-4: Results for Soft-Adaptive Turbo Equalizer, Soft-Adaptive Turbo Equalizer with Estimated Means and Soft-Iterative DFE at 24dB



(a) Mean Square Error of Equalizer Output vs. Iteration

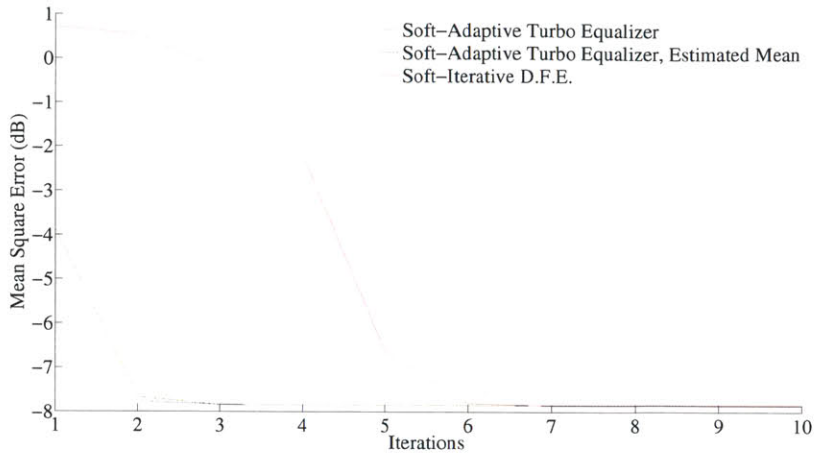


(b) Bit Error Rate of Decoder Output vs. Iteration

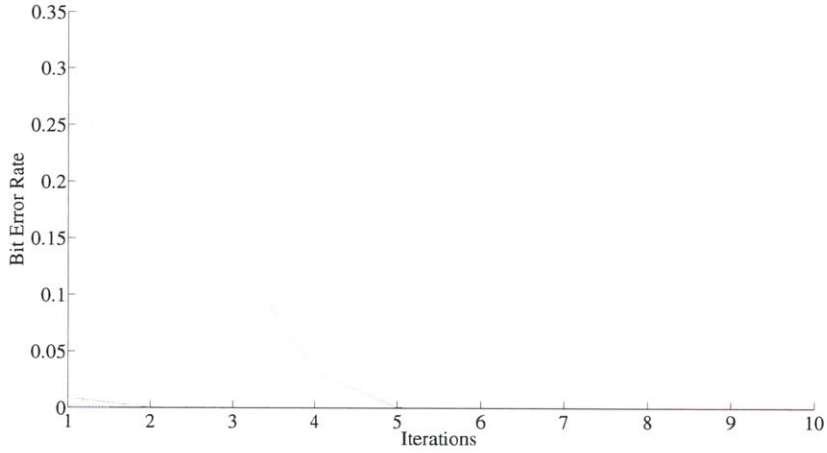


(c) Bit Error Rate as a function of Equalizer Mean Square

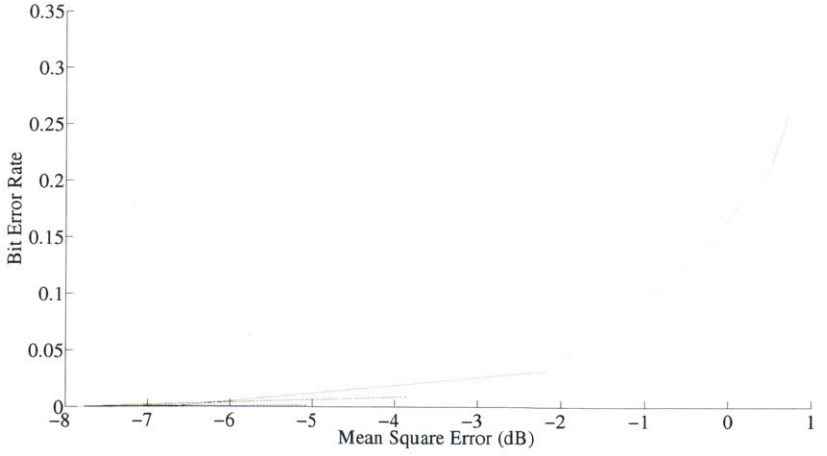
Figure 6-5: Results for Soft-Adaptive Turbo Equalizer, Soft-Adaptive Turbo Equalizer with Estimated Means and Soft-Iterative DFE at 21dB



(a) Mean Square Error of Equalizer Output vs. Iteration

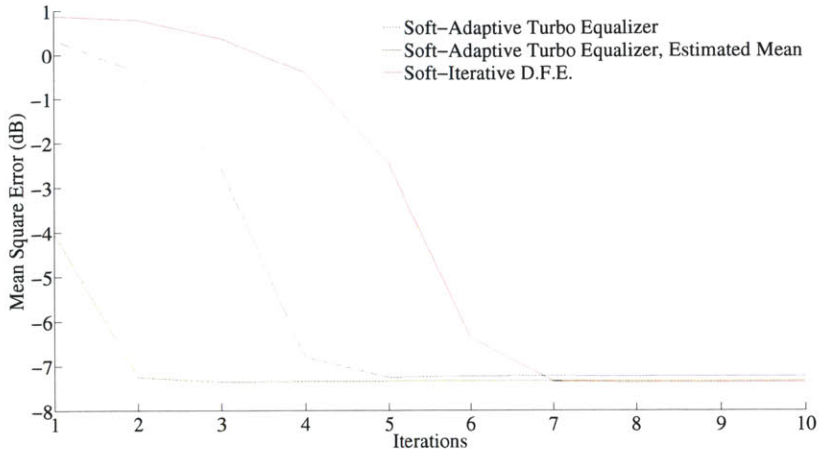


(b) Bit Error Rate of Decoder Output vs. Iteration

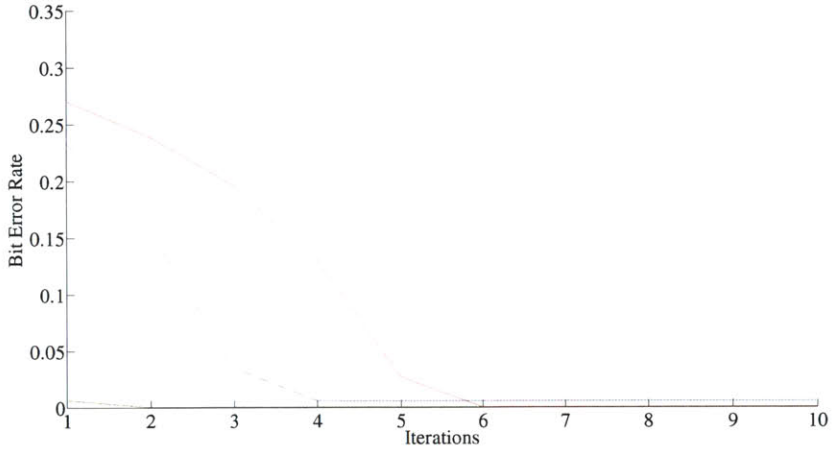


(c) Bit Error Rate as a function of Equalizer Mean Square

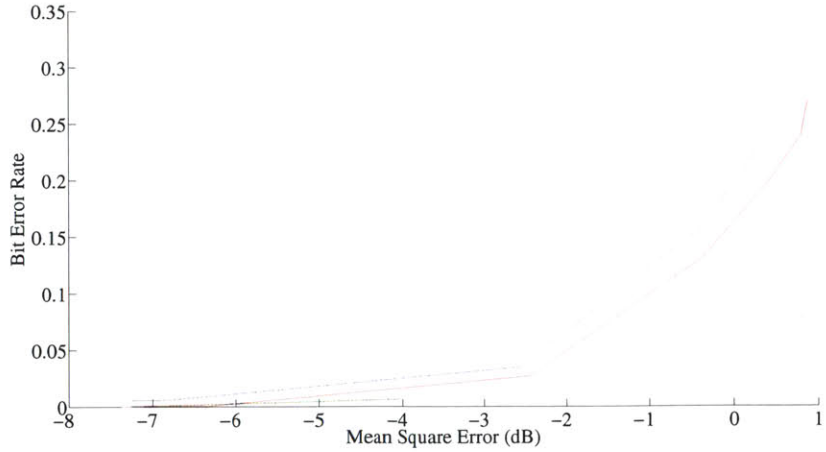
Figure 6-6: Results for Soft-Adaptive Turbo Equalizer, Soft-Adaptive Turbo Equalizer with Estimated Means and Soft-Iterative DFE at 18dB



(a) Mean Square Error of Equalizer Output vs. Iteration

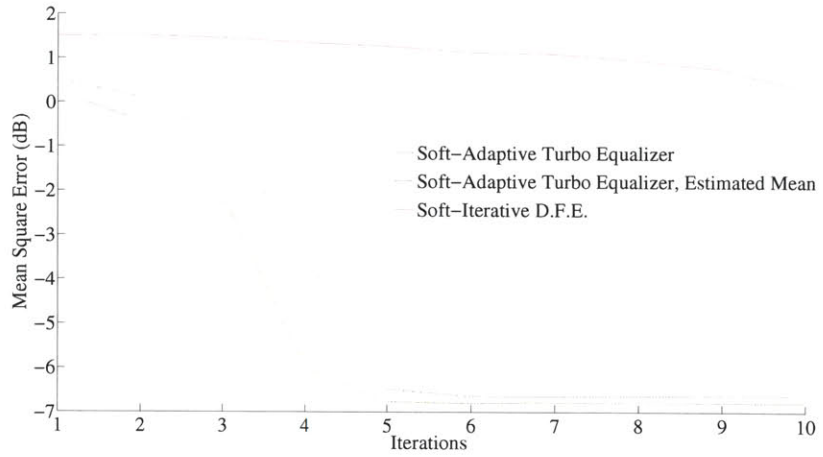


(b) Bit Error Rate of Decoder Output vs. Iteration

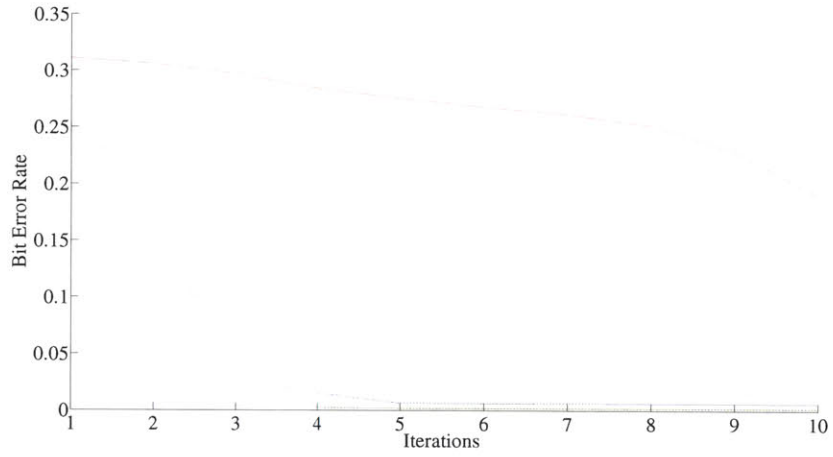


(c) Bit Error Rate as a function of Equalizer Mean Square

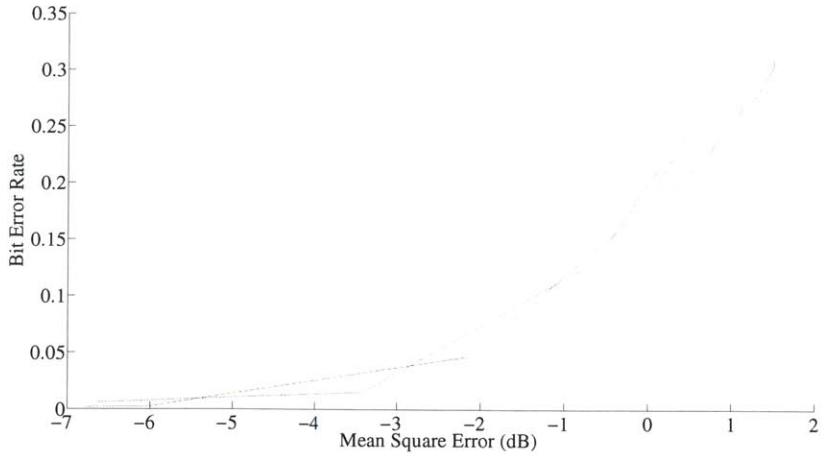
Figure 6-7: Results for Soft-Adaptive Turbo Equalizer, Soft-Adaptive Turbo Equalizer with Estimated Means and Soft-Iterative DFE at 15dB



(a) Mean Square Error of Equalizer Output vs. Iteration

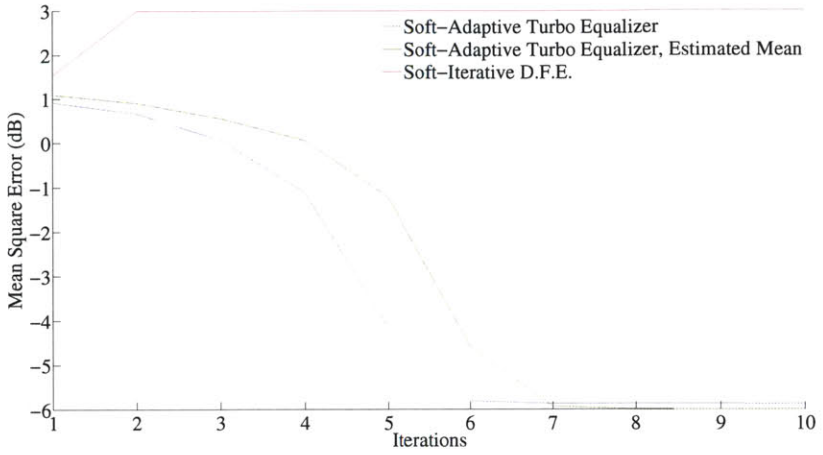


(b) Bit Error Rate of Decoder Output vs. Iteration

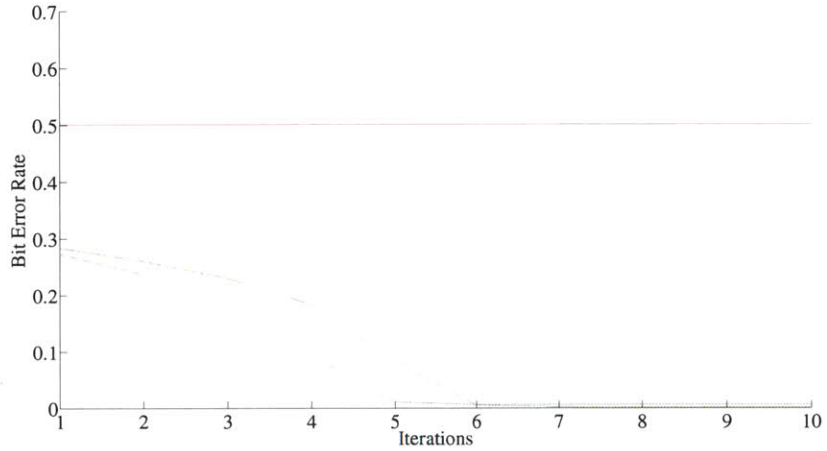


(c) Bit Error Rate as a function of Equalizer Mean Square

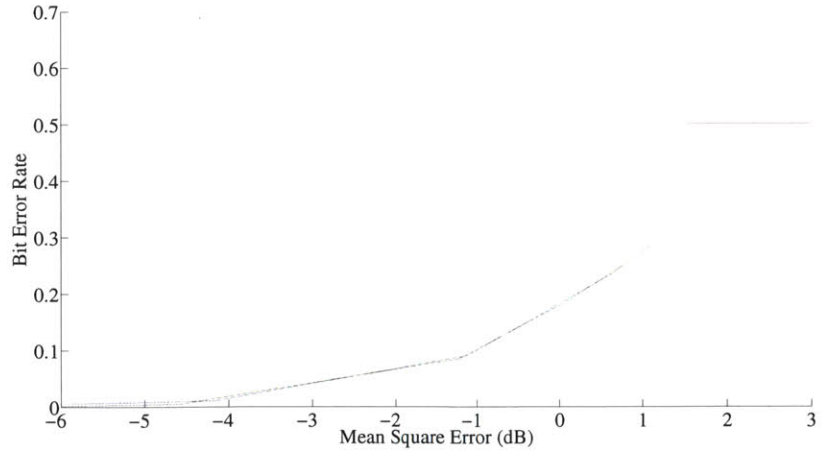
Figure 6-8: Results for Soft-Adaptive Turbo Equalizer, Soft-Adaptive Turbo Equalizer with Estimated Means and Soft-Iterative DFE at 12dB



(a) Mean Square Error of Equalizer Output vs. Iteration



(b) Bit Error Rate of Decoder Output vs. Iteration



(c) Bit Error Rate as a function of Equalizer Mean Square

Figure 6-9: Results for Soft-Adaptive Turbo Equalizer, Soft-Adaptive Turbo Equalizer with Estimated Means and Soft-Iterative DFE at 9dB

directed equalizer not converging to a point from where the decoder can find the right path, while the soft-decision directed equalizer does in the first iteration, as we have seen previously.

These results are very encouraging, as they show a significant improvement over existing turbo procedures, and show that there is indeed much practical value to introducing soft-information into adaptation in both standalone and turbo systems.

Recap and Looking Ahead

In this chapter, we have introduced the KAM11 Experiment, the signals that we transmitted and the turbo system. We provided relevant channel condition details and setup details, and a detailed listing of parameters of the receiver and transmitter.

We then presented and analyzed the results of implementing the turbo equalizer based on our understanding from the previous chapters. We showed how the Soft-Adaptive Turbo System shows an improvement over using prior information in just feedback filters, how it allows us to operate and eventually attain good error rates in which the hard-decision directed procedure causes wrong convergence in the first iteration, and how we can modify the adaptation procedure in the low SNR regime to further improve performance and attain a lower error floor.

We now recap the thesis and present avenues for future work .

Chapter 7

Conclusions

7.1 Thesis Summary

Adaptive equalization is an important aspect of underwater acoustic communication systems. One important aspect of adaptive equalization is the algorithm for adaptation. In this thesis we have considered the issue of design and analysis of adaptive equalizers in the practical scenario when the transmitted symbols are unknown at the receiver. We have presented a variety of interesting results and insights, which we briefly recap here.

First, we considered the issue of decision-directed adaptation. In Chapter 3, we derived an algorithm which we showed is "optimal" in an approximate Expectation Maximization sense when we do not have access to the transmitted symbols. We did this through the following steps:

- Deriving a recursive form of the EM algorithm, which performs one maximization of the expected log-likelihood at each time step.
- Relating the expected log-likelihood to the least squares approach by observing that these coincide for Gaussian variables, and deriving a cost criterion for optimization as a result. This was the Expected Least Squares cost criterion.
- Deriving a closed-form solution for the Expected Least Squares cost criterion and showing that this was non-causal.

- Enforcing causality on the solution to derive a purely recursive update algorithm. This is what we called the Recursive Expected Least Squares algorithm.

- Assuming a model for the output statistics that are required for the RELS algorithm. Specifically, we assumed that the output statistics are stationary and Gaussian conditioned on the transmitted symbols, with a mean equal to the symbol and a variance that could be estimated during training. This is the adaptation algorithm we used for equalization, and the equalizer was called the soft-decision directed equalizer.

We showed that this algorithm had application in practice with simulated and practical BPSK data. This demonstrated that we had a significant performance gain using the soft-decision directed equalizer over the hard-decision directed RLS algorithm, even though the Gaussian model used was unjustified.

In order to further quantify the performance of the equalizer, justify the models used and propose better models, we turned to a performance analysis of the hard-decision and soft-decision directed adaptive equalizer in Chapter 4. We started by showing that the exact problem was extremely hard and came up with a problem that is easier to solve but still provides insight- the problem of computing the steady state distribution of the mean of the coefficient vector. The problem is solved using the theory of Markov processes.

Applying the procedure, we showed the following:

- In training mode the mean of the equalizer coefficients converges to the Minimum Mean Square Error in the steady state.

- In hard-decision directed mode, the mean clusters about a point which represents the "overconfidence" of the procedure- the taps corresponding to the transmitted signal are larger (and hence blow up noise more) and taps cancel out less ISI.

- In soft-decision directed mode, the coefficient mean is closer to the MMSE solution than for the hard-decision directed mode in general.

- For a DFE in hard-decision directed mode, a catastrophic failure mode exists at low SNR, and we are able to predict it (which, as far as we know, has not been done before). We further discussed reasons that this happens.

- We further showed that this could happen for soft-decision directed adaptation as well, but that the SNR that it happens is lower, so that we gain some improvement in the performance. We showed a case in which failure occurs for hard-decision directed adaptation but not for soft-decision directed adaptation (even though it is predicted for both).

From the coefficient mean PDF, we showed how to go to the approximate statistics of the equalizer output. We showed that when the equalizer coefficient means cluster about a single point, the stationary assumption is justified. However, we showed that the equalizer outputs are Gaussian mixtures rather than Gaussians. Thus our Gaussian model is suboptimal. However, the mixing proportions in general are time-variant and depend on the variation in the channel. So for practical underwater channels, it may still be a good model due to its simplicity.

We then moved to a setting of turbo equalization. After looking at existing approaches to turbo equalization in underwater communication, we concluded that one very interesting way to use priors that has not been explored much in the past is using them in the adaptation process. We observed that the RELS adaptation procedure does in fact take priors into account if these are available. This led us to designing a turbo equalizer which uses the RELS decisions for adaptation and uses priors in them in successive iterations. We termed this the Soft-Adaptive Turbo Equalizer.

Finally, we tested the turbo equalizer and various other turbo structures using the data collected in the KAM11 Experiment. We introduced the relevant details of the experiment and the signals transmitted and then presented the results. The following points emerged from the results:

- The Soft-Adaptive Turbo Equalizer has an improved performance over procedures that use soft information only in feedback. The performance improvement can occur due to 2 reasons.

- The first iteration performance improvement: the error rate of the soft-decision directed adaptive equalizer is better than the hard-decision directed directed equalizer in the first iteration. The first iteration performance determines whether the code corrects errors at all, or whether it converges down the wrong trellis path. This first iteration effect ensures that we can converge to a low error rate even at SNRs in which algorithms not using soft information do not converge at all, as after the first iteration the code can correct more and more errors on each iteration
- In general, using the soft information in adaptation results in an adaptation track that improves with iterations quite significantly. The soft adaptive equalizer allows us to cut the number of iterations to achieve a particular error floor.

- A comparison of the Soft-Adaptive Turbo Equalizer to an identical system with means of the Gaussians estimated showed that at low SNRs there was performance to be gained by estimating the means.
  - At low SNR values, we showed that the mean of the equalizer output gets closer and closer to 0 even for an MMSE equalizer. Thus, assuming the means are equal to the transmitted signals causes a model mismatch.
  - At high SNRs, however, the assumption does not cause a large mismatch, and it allows the equalizer to adapt confidently, rather than estimating the means which in this case proves to be a little too conservative.

- We also compared these equalizers against a different direct adaptation form turbo structure, which uses soft information to form a set of hard decisions that are used for adaptation. We showed that in most cases the Soft-Adaptive Turbo structure achieved the same or a lower error floor than the other structure, and also required fewer iterations to do so.

We thus see that the algorithm has significant practical applications. However, there are various possibilities for further research, which we now consider.

7.2 Future Avenues

We have made various assumptions in the algorithm derivation. The one that we have revisited most often, of course, has been the assumption that the output statistics are stationary and normal. We have provided one justification for making it, but have acknowledged that it is not perfect.

A possible avenue of research is to determine how to track how the output statistics vary with time. This would let us form decisions as close to optimal as possible. However, this is a non-trivial problem, as we have discussed. It would be very interesting to understand how to track these statistics as the channel varies.

A related, and possibly simpler approach to this issue is to parametrize the system explicitly on the coefficient vector. Rather than making the stationary statistics assumption, if we assumed that the statistics of the coefficient vector are non-stationary and parametrize explicitly on the current value of the coefficient vector, we should be able to track the non-stationarity in the output directly. The mechanism to do this would possibly be simpler that directly forming a method to track the output statistics.

However, even in this case, the question of how to model the channel so that we "know" the equalizer input statistics remains. The input is still a multivariate Gaussian mixture with time varying components. This makes it quite hard to work with for estimation purposes.

Further there are issues of how to choose the various other parameters of the equalizer, including

- The number of multipath channels used
- The lengths of the feedforward and feedback filters
- The forgetting factor (in the RLS algorithm)

There are various possible methods that have been proposed for these in literature. In spite of the significant progress made towards this, however, definitive answers to choosing these are hard to find. The problem becomes even more complicated when

we consider the problem of jointly determining the optimal set of parameters to set for the algorithm. Much further research is required before this can be concretized.

We have also discussed the "failure mode" of the adaptive DFE, and have shown that even the soft-decision directed adaptive equalizer is subject to catastrophic failure. This is an issue, because in this mode of the equalizer, the output is completely uncorrelated to the tranmitted symbols, so the errors are unrecoverable. Two questions arise in this context:

- From an intuitive understanding of the failure mode, we have discussed that it occurs because the feedforward input is very noisy, and the equalizer assumes that the feedback filter is noiseless, which is not true. This could be avoided by injecting noise into the feedback filter inputs for the purpose of determining the adaptation symbols. This would effectively be diagonal loading. In this case, how much noise should be injected?

- If we had the optimal statistics, i.e., we did not make the stationary assumption, but somehow had access to the RELS optimal decision, would there still be a failure mode? That is, would the optimal RELS procedure be just confident enough that failure never occurs? While this is not guaranteed by the procedure, intuitively it would appear that the better the decision, the less the probability of failure.

In the context of turbo systems, it becomes harder to analyze the systems that we have designed. We have shown that there is a significant increase in performance through actual implementation. However, whether or not this is the best achievable performance is still open. More generally, finding bounds on performance and designs that achieve them for adaptive turbo systems is an unanswered question.

This thesis leads to the more general problem of designing adaptive systems for tracking time variant systems. While this thesis is focused on equalizers, the mathematics and general principles involved apply to many other scenarios and systems as well. Performance bounds and optimal strategies for general adaptive systems would be of interest to a variety of applications.

Thus, we could ask what the optimal strategies for the design of a time-varying system are, and what the best computational approximations to these are. These lead us to linking the theories of optimal inference and decision theory and information theory to the existing theory of adaptive filtering, just as we have done in this thesis. Such an understanding would generalize the work of this thesis into a form that is suitable for a variety of applications, and could be a fertile area for further research.

Bibliography

[1] K.E. Atkinson. The numerical solution of Fredholm integral equations of the second kind. *SIAM Journal on Numerical Analysis*, 4(3):337–348, 1967.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *Information Theory, IEEE Transactions on*, 20(2):284–287, 1974.

[3] Sandro Bellini. Bussgang techniques for blind equalization. In *IEEE Global Telecommunications Conference, Conference record*, pages 1634–1640, 1986.

[4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding. *Information Theory, IEEE Transactions on*, 44(3):909–926, 1998.

[5] S. Benedetto, R. Garello, and G. Montorsi. A search for good convolutional codes to be used in the construction of turbo codes. *Communications, IEEE Transactions on*, 46(9):1101–1105, 1998.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes I. In *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1064–1070. IEEE, 1993.

[7] F. Blackmon, E. Sozer, M. Murandian, J. Proakis, and M. Salehi. Performance comparison of iterative/integral equalizer/decoder structures for underwater acoustic channels. In *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, volume 4, pages 2191–2200. IEEE, 2001.

[8] J.W. Choi, T.J. Riedl, K. Kim, A. Singer, and J.C. Preisig. Adaptive linear turbo equalization over doubly selective channels. *IEEE J. Ocean. Eng*, 2011.

[9] J.M. Cioffi, W.L. Abbott, H.K. Thapar, C.M. Melas, and K.D. Fisher. Adaptive equalization in magnetic-disk storage channels. *Communications Magazine, IEEE*, 28(2):14–29, 1990.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, January 1977.

[11] C. Douillard, M. Jézéquel, C. Berrou, et al. Iterative correction of intersymbol interference: Turbo-equalization. *European Transactions on Telecommunications*, 6(5):507–511, 1995.

[12] T.H. Eggen, A.B. Baggeroer, and J.C. Preisig. Communication over Doppler spread channels. Part I: Channel and receiver presentation. *Oceanic Engineering, IEEE Journal of*, 25(1):62–71, 2000.

[13] U. Erez and G. Wornell. A Super-Nyquist architecture for reliable underwater acoustic communication. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 469–476. IEEE, 2011.

[14] A. Fertner. Improvement of bit-error-rate in decision feedback equalizer by preventing decision-error propagation. *Signal Processing, IEEE Transactions on*, 46(7):1872–1877, 1998.

[15] J.B.D. Filho, G. Favier, and J.M.T. Romano. New Bussgang methods for blind equalization. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 3, pages 2269–2272 vol.3, 1997.

[16] L. Freitag and D. Kilfoyle. Spatial modulation in the underwater acoustic channel, 2004.

[17] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes*, volume 80. Oxford University Press, 2001.

[18] J. Hagenauer and P. Hoeher. A Viterbi algorithm with soft-decision outputs and its applications. In *Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM'89., IEEE*, pages 1680–1686. IEEE, 1989.

[19] L. Hanzo, T.H. Liew, B.L. Yeap, and J. Wiley. *Turbo coding, turbo equalisation and space-time coding*. Wiley Online Library, 2002.

[20] Simon Haykin. *Adaptive Filter Theory*. Prentice Hall, 4 edition, September 2001.

[21] W. Hodgkiss and J.C. Preisig. Kauai Acomms MURI (KAM11) experiment. In *European Conference on Underwater Acoustics, 2012*, 2012.

[22] M. Honig, V. Tripathi, and Yakun Sun. Adaptive decision feedback turbo equalization. *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, page 413, 2002.

[23] M. Johnson, D. Herold, and J. Catipovic. The design and performance of a compact underwater acoustic network node. In *OCEANS'94.'Oceans Engineering for Today's Technology and Tomorrow's Preservation.'Proceedings*, volume 3, pages III–467. IEEE, 1994.

[24] Michael I. Jordon. The EM algorithm. In *An Introduction to Probabilistic Graphical Models*. Chapter 11.

[25] B.C. Kim and I.T. Lu. Parameter study of OFDM underwater communications system. In *Oceans 2000 MTS/IEEE Conference and Exhibition*, volume 2, pages 1251–1255. IEEE, 2000.

[26] R. Koetter, A.C. Singer, and M. Tuchler. Turbo equalization. *Signal Processing Magazine, IEEE*, 21(1):67–80, 2004.

[27] C. Laot, A. Glavieux, and J. Labat. Turbo equalization: adaptive equalization and channel decoding jointly optimized. *Selected Areas in Communications, IEEE Journal on*, 19(9):1744 –1752, September 2001.

[28] E.L. Lehmann and G. Casella. *Theory of point estimation*. Springer Verlag, 1998.

[29] B. Li, S. Zhou, M. Stojanovic, L. Freitag, J. Huang, and P. Willett. MIMO-OFDM over an underwater acoustic channel. In *OCEANS 2007*, pages 1–6. IEEE, 2007.

[30] R.R. Lopes and J.R. Barry. Soft-output decision-feedback equalization with a priori information. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 3, pages 1705–1709 vol.3, 2003.

[31] Manolakis, Ingle, and Kogon. *Statistical and Adaptive Signal Processing*. McGraw-Hill Education, May 2000.

[32] JE Mazo. Faster-than-Nyquist signaling. *Bell Syst. Tech. J*, 54(8):1451–1462, 1975.

[33] S. J Nowlan and G. E Hinton. A soft decision-directed LMS algorithm for blind equalization. *Communications, IEEE Transactions on*, 41(2):275–279, 2002.

[34] R. Otnes and M. Tuchler. Iterative channel estimation for turbo equalization of time-varying frequency-selective channels. *Wireless Communications, IEEE Transactions on*, 3(6):1918–1923, 2004.

[35] SU Pillai, T. Suel, and S. Cha. The Perron-Frobenius theorem: some of its applications. *Signal Processing Magazine, IEEE*, 22(2):62–75, 2005.

[36] W. Pogorzelski. *Integral equations and their applications*, volume 1. Pergamon, 1966.

[37] H.V. Poor and X. Wang. Code-aided interference suppression for DS/CDMA communications II- parallel blind adaptive implementations. *Communications, IEEE Transactions on*, 45(9):1112–1122, 1997.

[38] J.C. Preisig. Performance analysis of adaptive equalization for coherent acoustic communications in the time-varying ocean environment. *The Journal of the Acoustical Society of America*, 118:263, 2005.

[39] J.C. Preisig, A.C. Singer, and G.W. Wornell. Reduced bandwidth frequency domain equalization for underwater acoustic communications. In *Sensor Array and Multichannel Signal Processing Workshop (SAM), 2010 IEEE*, pages 93–96. IEEE, 2010.

[40] J.G. Proakis. Adaptive equalization for TDMA digital mobile radio. *Vehicular Technology, IEEE Transactions on*, 40(2):333 –341, May 1991.

[41] J.G. Proakis and M. Salehi. *Digital communications*, volume 4. McGraw-hill New York, 2001.

[42] S.U.H. Qureshi. Adaptive equalization. *Proceedings of the IEEE*, 73(9):1349 – 1387, 1985.

[43] D. Raphaeli and A. Saguy. Linear equalizers for turbo equalization: A new optimization criterion for determining the equalizer taps. In *Proc. 2nd Intern. Symp. on Turbo codes, Brest, France*, pages 371–374, 2000.

[44] M. Reuter, J.C. Allen, J.R. Zeidler, and R.C. North. Mitigating error propagation effects in a decision feedback equalizer. *Communications, IEEE Transactions on*, 49(11):2028–2041, 2001.

[45] J. Sala-Alvarez and G. Vázquez-Grau. Statistical reference criteria for adaptive signal processing in digital communications. *Signal Processing, IEEE Transactions on*, 45(1):14–31, 2002.

[46] J.E. Smee and N.C. Beaulieu. Error-rate evaluation of linear equalization and decision feedback equalization with error propagation. *Communications, IEEE Transactions on*, 46(5):656–665, 1998.

[47] S. Song, A.C. Singer, and K.M. Sung. Soft input channel estimation for turbo equalization. *IEEE Transactions on Signal Processing*, 52(10):2885–2894, 2004.

[48] Seongwook Song, A.C. Singer, and K. M. Sung. Turbo equalization with an unknown channel.

[49] EM Sozer, JG Proakis, and F. Blackmon. Iterative equalization and decoding techniques for shallow water acoustic channels. In *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, volume 4, pages 2201–2208. IEEE, 2001.

[50] M. Stojanovic. Acoustic (underwater) communications. *Encyclopedia of Telecommunications*, 2003.

[51] M. Stojanovic. Low complexity OFDM detector for underwater acoustic channels. In *OCEANS 2006*, pages 1–6. IEEE, 2006.

[52] M. Stojanovic, J. Catipovic, and J.G. Proakis. Adaptive multichannel combining and equalization for underwater acoustic communications. *The Journal of the Acoustical Society of America*, 94:1621, 1993.

[53] M. Stojanovic and J. Preisig. Underwater acoustic communication channels: Propagation models and statistical characterization. *Communications Magazine, IEEE*, 47(1):84–89, 2009.

[54] Z. Tian, K.L. Bell, and H.L. Van Trees. A quadratically constrained decision feedback equalizer for DS-CDMA communication systems. In *Signal Processing Advances in Wireless Communications, 1999. SPAWC'99. 1999 2nd IEEE Workshop on*, pages 190–193. IEEE, 1999.

[55] M. Tuchler, R. Koetter, and A. C Singer. Turbo equalization: principles and new results. *Communications, IEEE Transactions on*, 50(5):754–767, 2002.

[56] M. Tuchler, A. C Singer, and R. Koetter. Minimum mean squared error equalization using a priori information. *Signal Processing, IEEE Transactions on*, 50(3):673–683, 2002.

[57] O. Wintzell, M. Lentmaier, and K.S. Zigangirov. Asymptotic analysis of super-orthogonal turbo codes. *Information Theory, IEEE Transactions on*, 49(1):253–258, 2003.

[58] CF Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[59] A. Yellepeddi and J.C. Preisig. Direct-form adaptive equalization using soft information. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 461–468. IEEE, 2011.