Adaptive Turbo Equalization

By

Ronald J. Bryant

B.S. Massachusetts Institute of Technology, 1974

M.S. Massachusetts Institute of Technology, 1975

M.D. University of Michigan, 1979

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(in Electrical Engineering)

The Graduate School

The University of Maine

December, 2000

Advisory Committee:

Donald M. Hummels, Professor of Electrical and Computer Engineering, Advisor

Habtom Ressom, Assistant Professor of Electrical and Computer Engineering

Bruce E. Segee, Associate Professor of Electrical and Computer Engineering

Adaptive Turbo Equalization

By Ronald J. Bryant

Thesis Advisor: Dr. Donald M. Hummels

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
(in Electrical Engineering)
December, 2000

This thesis addresses the issue of reliable digital communications over a channel with time-varying inter-symbol interference. The theory of turbo coding and turbo equalization is explained and an adaptive turbo equalizer is designed. The adaptive equalizer combines a maximum *a posteriori* channel equalizer with a recursive least squares algorithm to allow adaptation in the equalization stage. This stage iterates with a soft-input, soft-output error correction stage. The adaptive ability of the equalizer is tested on unknown time-invariant and time-varying inter-symbol interference channels by computer simulation. The channels were generated by an auto regressive model.

The equalizer tracked unknown channels accurately without a penalty in terms of signal to noise ratio (SNR) compared to an equalizer given perfect knowledge of the channel. The penalty of not knowing the channel is the need for more iterations of the decoding algorithm at low SNRs. On a time-varying channel this iteration penalty disappeared when the SNR increased by approximately one decibel and only a single iteration through the algorithm was necessary to correctly decode the received data with high reliability. The cause of the iteration penalty at low SNRs was partially due to conflicting information fed back from the error

correction code to the equalization stage in early iterations of the algorithm. The computational complexity of the adaptive turbo equalizer is high.

The results of the simulation indicate that the iterative decoding that is characteristic of the turbo principle is beneficial only over a very limited range of SNR, regardless of whether the channel is known or unknown. The result is that the large computational effort required by multiple iterations of this algorithm yield minimal rewards in terms of a reduced signal energy requirement when tracking a time-varying channel. This drawback appears to be a characteristic of the iterative process involved in the turbo code design and is not related to the computational complexity of the specific equalizer stage utilized.

ACKNOWLEDGMENTS

Many thanks to Don Hummels for his extensive guidance, assistance, and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Use of the "turbo" principle to iteratively decode parallel or serial concatenated error correction codes has lead to performance closely approaching Shannon's theoretical channel capacity limit. This principle has also been used to improve the reliability of communications over a multi-path channel. In this thesis the concept is further exploited to allow communication over a time-varying multi-path channel.

1.1 Background

The problem of multi-path transmission is well known to anyone who grew up prior to the introduction of cable TV and frustrated him or herself by trying to adjust the rabbit ears antenna on the television in an attempt to merge the double images. Now the burgeoning demand for mobile communication systems has rejuvinated the search for solutions to the multi-path problem. Mobile systems add an additional dimension to the problem; the signal transmission pathways are changing with time. When the transmission uses a sequence of digital symbols, a multi-path transmission channel is also referred to as an inter-symbol interference (ISI) channel. Symbols that arrive at the receiver simultaneously by different paths are superimposed just like the images on the television. The goal of the receiver designer is to adjust the digital rabbit ears so that the integrity of the transmitted message is maintained.

This thesis arose partially out of an interest developed following an introduction to the problem of undersea submarine acoustic communications. Acoustic signals are subject to reflection by the sea floor contours, temperature gradients, and sea water salinity gradients. The multiple transmission paths created by these environmental factors are changing due to the submarine's motion and ocean currents. The multi-path transmission can be well described mathematically as the impulse response of a time-varying ISI channel. Despite the ability to accurately model the multi-path problem, design of a low power and reliable communication link has not been accomplished.

Error correction codes have significantly improved the reliability of digital signal transmission and reception. The relatively recent introduction of the "turbo" principal by Berrou *et. al.* [1] has enabled the design of communication systems with channel capacities closely approaching Shannon's theoretical limit. The turbo principle involves iteratively decoding a hugely complex code generated by two convolutional encoders separated by an interleaver.

Bauch *et. al.* [2] proposed combining a maximum likelihood ISI channel decoder in a turbo-like scheme with a convolutional error correction code. This design, referred to as a turbo equalizer, has been simulated on a known time-invariant ISI channel and provides good reliability with a low signal to noise ratio (SNR).

Turbo coding has yet to see widespread applications. A major drawback is receiver complexity and hence a long latency period before generation of output by the receiver. This has caused a focus on applications where the latency period is of little relevance such as deep space communications. Undersea acoustic communications is another area where this latency is acceptable.

Some turbo decoders have been developed in hardware, but most of the data on turbo code performance has been obtained through computer simulation [3].

1.2 Purpose of the Research

The purpose of this research was to design a turbo equalizer capable of adapting to a time-varying ISI channel. After being designed, the performance of the adaptive turbo equalizer was determined through simulated transmission over a time varying ISI channel. Based on these results, assessment of the feasibility of the design for application in the undersea acoustic environment was made.

The turbo decoding algorithm is complex in the sense of computational intensity as well as the more common sense of "complex to understand". Therefore an additional goal of this thesis was to develop a consistent and unequivocal notation of turbo decoding theory.

1.3 Thesis Organization

Chapter 2 is a tutorial that gives a detailed explanation of the theory of turbo codes, and turbo equalization. Following this, the chapter includes an explanation of the adaptive turbo equalizer design. A discussion of the computational complexity of the adaptive turbo equalizer is given. The chapter concludes with a short example that focuses on the "turbo" aspect of the decoding process.

Chapter 3 presents the criteria used for judging performance of the turbo equalizer. Performance of the turbo equalizer on a known time-invariant channel is presented. Performance results are drawn from the literature and compared to results generated by the adaptive equalizer design of Chapter 2.

In Chapter 4 the adaptive turbo equalizer is tested for its ability to estimate a time-invariant ISI channel without knowledge of the channel. The interaction of the equalizer stage and the decoder stage of the turbo equalizer between iterations is examined in detail. Criteria for setting some of the parameters of the adaptive turbo equalizer are discussed.

Chapter 5 allows the ISI channel to vary over time and the performance of the adaptive turbo equalizer is assessed in this situation. Transmission over the same sequence of time-varying channels is tested with and without knowledge of the channel impulse response.

Chapter 6 discusses the results above, and points out the strength and weakness of the adaptive equalization process. Conclusions are drawn from the experiments performed and possible improved designs are discussed along with other suggestions for further investigation.

CHAPTER 2

Tutorial

The introduction of concatenated error correction codes and decoding theory (turbo codes) in the field of communications has enabled information transmission rates that closely approach Shannon's theoretical limits. This chapter begins with an explanation of turbo coding theory. It then shows how the turbo scheme can be modified to develop a "turbo equalizer" to equalize the output from an inter-symbol interference (ISI) channel. Next the necessary modifications of a turbo equalizer to allow adaptation to a time-varying ISI channel is described and the issue of receiver complexity is discussed. Finally, a brief example is given to help clarify the interleaving that is an integral part of why turbo codes perform so well.

2.1 Turbo Coding

Turbo coding involves the concatenation of two convolutional encoders separated by an interleaver. The "turbo" principle refers to the iterative decoding of the complex code generated. Each constituent code is decoded separately, but with the benefit of a limited amount of information learned from the other code during previous iterations. This separation enables efficient, near optimal decoding. Each code is decoded at the receiver by a four-port device known as a soft-input soft-output (SISO) decoder. A firm appreciation of the SISO module is key to understanding how turbo codes are decoded.

This presentation of turbo coding theory develops a detailed notation that represents an expansion, combination, and clarification of conventions taken from the literature. It is based on notations primarily used at two labs, Benedetto *et. al.* at The Jet Propulsion Laboratory [4] and Hagenauer *et. al.* at The Technical University of Munich [5].

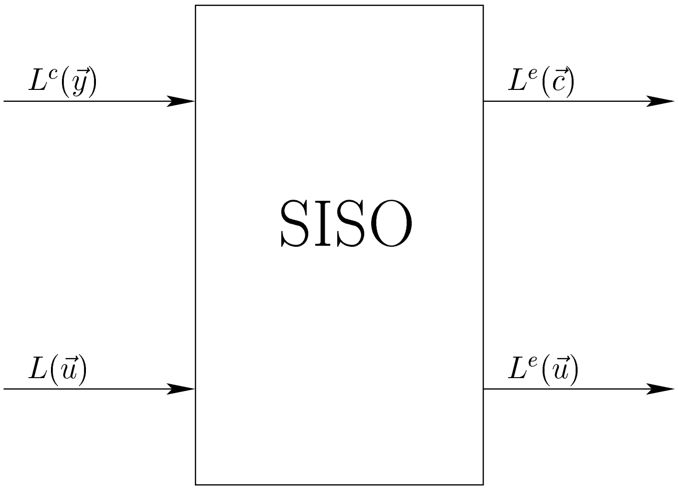

Figure 2.1: Block diagram of a SISO module.

2.1.1 SISO

When a block of encoded information is transmitted over a channel with additive white Gaussian noise (AWGN), information about the k^{th} symbol of the transmitted sequence has three independent sources.

1. The received value of the k^{th} symbol.

2. *A priori* information about the symbol sent in the k^{th} position.

3. Information about the k^{th} symbol that resides in all other received symbols within the block by virtue of the error correction coding done prior to transmission.

These three sources of information are referred to as intrinsic, *a priori*, and extrinsic information respectively.

The four-port SISO module, illustrate in Figure 2.1, receives intrinsic and *a priori* information as inputs and returns extrinsic information about the encoded information words and transmitted codewords that were sent through the channel. One input receives a sequence of independent likelihoods of transmitted codewords based on the channel outputs, designated $L^c(\vec{y})$. The other input is the *a priori* likelihood of information words, $L(\vec{u})$. The two outputs are a sequence of likelihoods of the information words ($L^e(\vec{u})$) and codewords ($L^e(\vec{c})$) from the corresponding encoder. These likelihoods can be generated using the SISO's knowledge of the encoding algorithm. For descriptive purposes these four ports are referred to as the intrinsic and *a priori* inputs and the uncoded and coded outputs respectively.

A SISO can be implemented with a variety of algorithms as long as the encoder can be represented as a trellis. The following description of a trellis is established to facilitate explanation of the function of a SISO module. The encoders used in turbo coding are systematic convolutional encoders that accept as input, a sequential block of K information words designated $\vec{u}$. The information words are comprised of small groups of k_s symbols. Thus the k^{th} information word, u_k, can be viewed on a symbol by symbol basis as $[\ u_{k,1}\ \ u_{k,2}\ \ ...\ \ u_{k,k_s}\ ]$. The information symbols are drawn from an alphabet of size N_I. The encoder generates a sequence of K parity words $\vec{p}$, with p_s parity symbols drawn from an alphabet also of size N_I. Since the codes are systematic the output of the encoder is the sequence of codewords, $\vec{c}$, that are assembled from $\vec{u}$ and $\vec{p}$. Thus, the k^{th} codeword of $\vec{c}$ contains $n_s = k_s + p_s$ symbols and can be viewed in three equivalent forms.

$$c_k = [\ u_k\ \ p_k\ ] \tag{2.1}$$

$$= [\ u_{k,1}\ \ u_{k,2}\ \ \cdots\ \ u_{k,k_s}\ \ p_{k,1}\ \ p_{k,2}\ \ \cdots\ \ p_{k,p_s}\ ] \tag{2.2}$$

$$= [\ c_{k,1}\ \ c_{k,2}\ \ \cdots\ \ c_{k,n_s}\ ]. \tag{2.3}$$

The rate of the resulting code is $\frac{k_s}{n_s}$.

A section of encoder trellis can be represented by two sets of N states $\{s_0, s_1, s_2, ..., s_{N-1}\}$ interconnected by $N \cdot N_I^{k_s}$ edges (Figure 2.2.) An edge, e_k, connects the state of the encoder, s_{k-1}, at time $k-1$ to the state, s_k, at time k. The following four functions are associated with each edge, e_k, at time k.

1. The starting state $s^S(e_k)$.
2. The ending state $s^E(e_k)$.
3. The input information word $u(e_k)$.
4. The output codeword $c(e_k)$.

If the encoder is in state $s_{k-1} = s^S(e_k)$, an input of $u_k = u(e_k)$ results in a transition to state $s_k = s^E(e_k)$ along edge e_k in association with the generation of the codeword $c_k = c(e_k)$.

The number of states and the edge functions above depend on the characteristics of the particular encoder implemented. Given an initial trellis state, there is a one-to-one correspondence between the sequence of edges traversed and each of the three sequences described by the input $\vec{u}$, the coded output $\vec{c}$, and the sequence of states occupied.

In this implementation, the SISO uses an optimal decoding algorithm that computes the maximum *a posteriori* probability of an information word at each time k given the entire received sequence of symbols, $P\{u_k|\vec{y}\}$. This trellis decoding algorithm was first described by Bahl *et. al.* [6] in 1974 and is referred to as the BCJR algorithm. Once these probabilities are known, the algorithm selects the word u_k that has the maximum $P\{u_k|\vec{y}\}$ as output. The algorithm initially calculates probabilities of transitions along each edge given the matched filter output vector $\vec{y}$, *i.e.* $P\{e_k|\vec{y}\}$. Note that the desired quantity,

$$
\begin{aligned}
P\{e_k|\vec{y}\} &= \frac{P\{e_k;\vec{y}\}}{P\{\vec{y}\}} && (2.4)\\
&= \frac{\int_{\vec{y}-\Delta y}^{\vec{y}+\Delta y} f(e_k;\vec{y})d\vec{y}}{\int_{\vec{y}-\Delta y}^{\vec{y}+\Delta y} f(\vec{y})d\vec{y}}, && (2.5)
\end{aligned}
$$

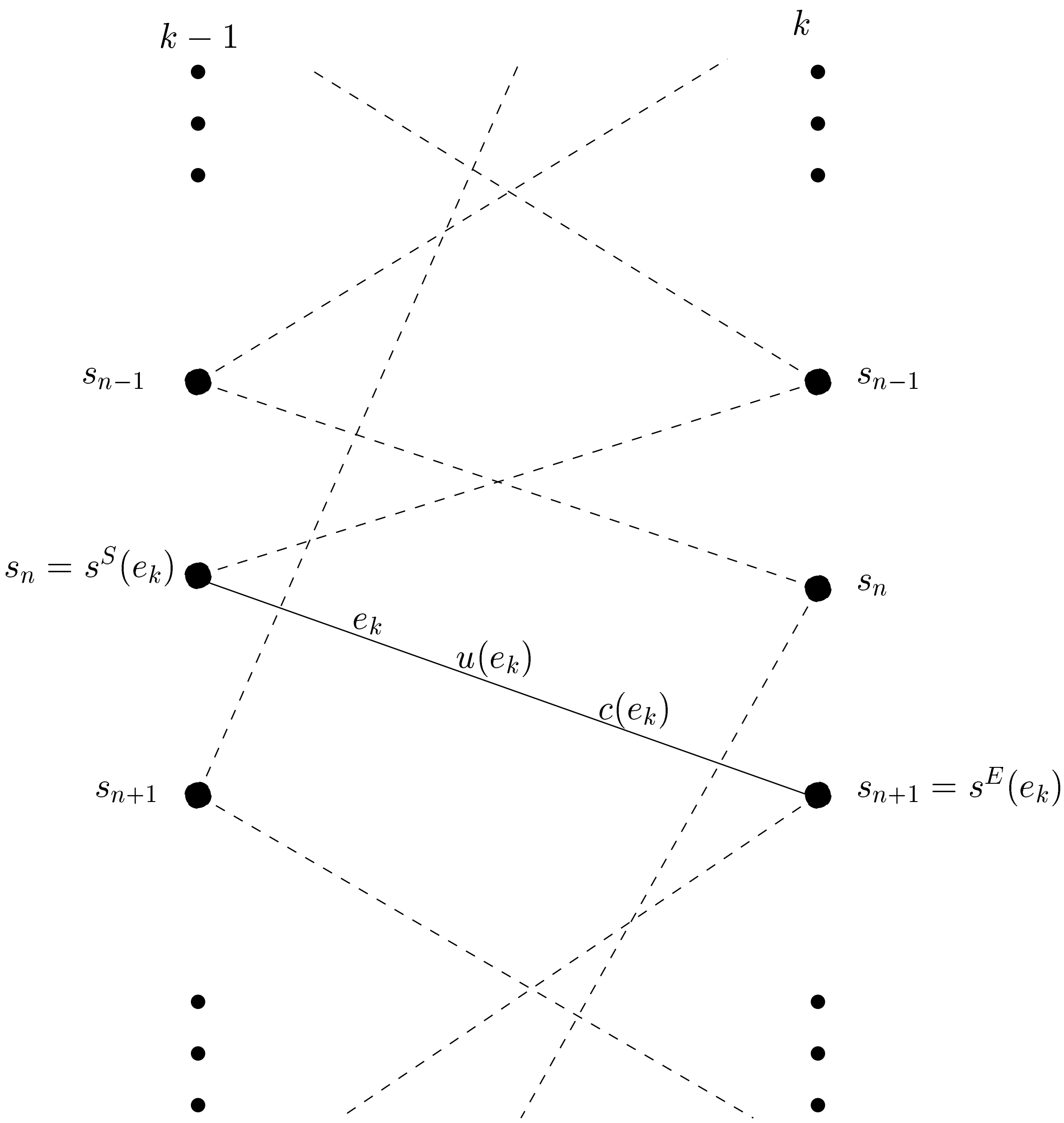

Figure 2.2: Section of encoder trellis

where the integrals of the density functions $f(\cdot)$ are over a small interval around each element of $\vec{y}$ determined by resolution characteristics of the receiver. Since the term $P\{\vec{y}\}$ is constant for all edges it does not influence the final decision and may be dropped without any loss of performance. It is convenient to derive $P\{e_k;\vec{y}\}$ and assume that the appropriate integral has been done where probabilities of continuous data are expressed.

One conclusion of Bahl *et. al.* [6] is that the joint probability of traversing a particular edge at time k and receiving the vector $\vec{y}$ is

$$P\{e_k;\vec{y}\} = \alpha_{k-1}(s^S(e_k))\ P\left\{s^E(e_k);y_k|s^S(e_k)\right\}\ \beta_k(s^E(e_k)). \tag{2.6}$$

where $\alpha_k(s)$ and $\beta_k(s)$ are computed recursively in a forward or reverse direction respectively using the middle term of Equation 2.6.

$$\alpha_k(s) = \sum_{e:s^E(e)=s} \alpha_{k-1}(s^S(e))\ P\left\{s^E(e);y_k|s^S(e)\right\} \tag{2.7}$$

$$\beta_{k-1}(s) = \sum_{e:s^S(e)=s} \beta_k(s^E(e))\ P\left\{s^E(e);y_k|s^S(e)\right\} \tag{2.8}$$

The encoder is initialized to state zero and forced to return to state zero at the end of the input sequence by calculating and appending a short string of additional input symbols. So the necessary boundary conditions for the recursions are:

$$\alpha_0(s) = \begin{cases} 1 & s = 0, \\ 0 & s \neq 0 \end{cases} \tag{2.9}$$

and

$$\beta_K(s) = \begin{cases} 1 & s = 0, \\ 0 & s \neq 0. \end{cases} \tag{2.10}$$

The term $P\left\{s^E(e_k); y_k|s^S(e_k)\right\}$ that is common to the three Equations 2.6, 2.7, and 2.8 is

$$P\left\{s^E(e_k); y_k|s^S(e_k)\right\} = P\left\{y_k|s^E(e_k); s^S(e_k)\right\}\ P\left\{s^E(e_k)|s^S(e_k)\right\}. \tag{2.11}$$

This can be simplified, after recalling the one-to-one relationships between trellis edge sequences and $\vec{u}$, $\vec{c}$, and the state sequence. Thus

$$P\left\{s^E(e_k); y_k|s^S(e_k)\right\} = P\{y_k|c(e_k)\}\ P\{u(e_k)\} = P\{y_k|c_k\}\ P\{u_k\} \tag{2.12}$$

and Equation 2.6 becomes

$$P\{e_k; \vec{y}\} = \alpha_{k-1}(s^S(e_k))\ P\{u_k\}\ P\{y_k|c_k\}\ \beta_k(s^E(e_k)), \tag{2.13}$$

and Equations 2.7 and 2.8 are similarly simplified by substituting Equation 2.12. The code word likelihood, $P\{y_k|c_k\}$, is calculated directly from the channel reception $\vec{y}$,

$$P\{y_k|c_k\} = \int_{y_k-\Delta y}^{y_k+\Delta y} f(y_k|c_k)dy_k. \tag{2.14}$$

For an AWGN channel the density function $P\{y_k|c_k\}$ is Gaussian.

To determine the probability of a given input or output word at time k the SISO module sums Equation 2.13 over all edges that generate the given word.

Thus the probability of information and coded words are:

$$P\{u_k;\vec{y}\} = \sum_{e:u(e)=u_k} \alpha_{k-1}(s^S(e))\; P\{u(e)\}\; P\{y_k|c(e)\}\; \beta_k(s^E(e)) \tag{2.15}$$

and

$$P\{c_k;\vec{y}\} = \sum_{e:c(e)=c_k} \alpha_{k-1}(s^S(e))\; P\{u(e)\}\; P\{y_k|c(e)\}\; \beta_k(s^E(e)). \tag{2.16}$$

At this time it is worthwhile taking advantage of the monotonic nature of the logarithm function and refer to the natural logarithm of probabilities. This improves decoder execution speed, as will be shown later, but more importantly for the moment, it leads to improved clarity in illustrating the rest of the SISO algorithm and the interaction of the two SISOs within a turbo decoding scheme. The notation $L(\cdot) \triangleq \ln(P\{\cdot\})$ is therefore defined.

a SISO module has two inputs, $L^c(\vec{y})$, the likelihood of the codewords given the observable information received, and $L(\vec{u})$ the *a priori* information about the encoder input words. Note that the elements of $L^c(\vec{y})$, $L^c(y_k) \triangleq L(y_k|c_k)$, are required for Equations 2.15 and 2.16. $L^c(\vec{y})$ simply represent the natural logarithm of Equation 2.14. The SISO takes these two sources of information about the encoded sequence, and using its knowledge of the encoder trellis, the SISO uses the BCJR algorithm to generate the quantities, $L(u_k;\vec{y})$ and $L(c_k;\vec{y})$. The difference between the information generated by the BCJR algorithm and the information provided to the SISO defines the extrinsic information contained within the code.

$$L^e(u_k) \triangleq L(u_k;\vec{y}) - [\; L(u_k) + L(y_k|c_k)\;] \tag{2.17}$$

$$L^e(c_k) \triangleq L(c_k;\vec{y}) - [\; L(u_k) + L(y_k|c_k)\;] \tag{2.18}$$

$L^e(u_k)$ and $L^e(c_k)$ represent information about words u_k or c_k respectively, that resides in all of the received sequence excluding the k^{th} word; information that is present by virtue of the trellis constraints imparted by the corresponding encoder. The two quantities $L^e(\vec{u})$ and $L^e(\vec{c})$ are the output of the SISO module.

The description of SISO function has been developed with inputs consisting of a vector of word likelihoods. If each vector element is replaced by a row of symbol likelihoods at the SISO inputs, as can be done when symbol transmissions are independent, extrinsic information for individual symbols can be calculated by the SISO. The algorithm is analogous to that described above if Equations 2.15 and 2.16 are summed over edges containing symbols in the appropriate position within the words.

2.1.2 Turbo Code in Parallel Concatenated Configuration

In a turbo code two convolutional codes can be concatenated in a serial or parallel fashion. The codewords are broken into their individual symbols before being sequentially transmitted. Independent transmission of all symbols through the channel is assumed.

Initially we will work with a parallel concatenated convolutional code (PCCC) as illustrated in Figures 2.3 (encoder) and 2.4 (decoder). The encoders are labeled Code 1 and Code 2, but the constituent code used may or may not differ. The information words to be transmitted are in a vector, $\vec{u}$, of length K. The first encoder generates K parity words $\vec{p}^1$. The individual information words and parity words are combined as follows to produce the vector of codewords $\vec{c}^1$ of length K.

$$c_k^1 = [\ u_k \ \ p_k^1\ ]. \tag{2.19}$$

Note that u_k is explicitly shown in the code words because the code is systematic.

Prior to input in the second encoder, $\vec{u}$ is shuffled by the pseudo-random interleaver, Π, to form $\vec{u}^{\pi}$. The interleaver is generated with a random generator, but the interleaver performs an invertible operation. The receiver has the knowledge required to interleave and de-interleave at will.

The second encoder operates on $\vec{u}^{\pi}$ and produces K parity words, $\vec{p}^2$. A vector of systematic codewords, $\vec{c}^2$ of length K, can be formed by the second encoder.

$$c_k^2 = [\ u_k^{\pi}\ \ p_k^2\ ] \tag{2.20}$$

However, only minimal coding gains arise from sending both copies of the redundant information $\vec{u}$ and $\vec{u}^{\pi}$. Thus the codewords transmitted as the vector $\vec{x}$, of length K, are formed from $\vec{c}^1$ and part of $\vec{c}^2$ as follows

$$x_k = [\ u_k\ \ p_k^1\ \ p_k^2\ ]. \tag{2.21}$$

By not transmitting the interleaved copy of the information words the code rate is $\frac{k_s}{k_s+p_s^1+p_s^2}$ rather than the lower $\frac{k_s}{2k_s+p_s^1+p_s^2}$. (The superscript on the number of parity symbols per codeword refer to the first or second encoder.)

The codewords $\vec{x}$ are transmitted one symbol at a time and the AWGN channel output $\vec{y}$, is presented to the decoder of Figure 2.4 in the form of likelihoods, $L^c(\vec{y})$. For clarity of presentation it is helpful to consider the $L(\cdot)$ operator as casting the likelihoods as a matrix when given a vector argument. The matrix size is K by n_s where K is the length of the vector and n_s is the number of symbols in the codewords that comprise the vector argument. Although the rows of $L^c(\vec{y})$ are n_s elements in length, it is sometimes convenient to view $L^c(\vec{y})$ in a form more intuitively related to the constituent codewords and reminiscent of Equations 2.1

to 2.3. The k^{th} row of $L^c(\vec{y})$ can be expressed in several equivalent forms.

$$\begin{aligned} L^c(y_k) &= [L^c(c_k^1)\ L^c(p_k^2)] && (2.22)\\ &= [L^c(u_k)\ L^c(p_k^1)\ L^c(p_k^2)] && (2.23)\\ &= [L^c(u_{k,1})...L^c(u_{k,k_S})\ L^c(p_{k,1}^1)...L^c(p_{k,p_s^1}^1)\ L^c(p_{k,1}^2)...L^c(p_{k,p_s^2}^2)] && (2.24)\end{aligned}$$

and the entire $L^c(\vec{y})$ can be partitioned in similar fashions, *e.g.*

$$L^c(\vec{y}) = [\ \ L^c(\vec{u})\ \ L^c(\vec{p}^1)\ \ L^c(\vec{p}^2)\ \ ]. \tag{2.25}$$

This notation illustrates the association between encoder and decoder components and clarifies how the interleaving of symbol likelihoods is done in the receiver.

When making final codeword decisions, the log-likelihood of all constituent symbols must be summed. This operation is indicated as

$$\sum_{All\ Symbols} L(u_k) \ \triangleq\ L(u_{k,1}) + L(u_{k,2}) + ... + L(u_{k,k_s}). \tag{2.26}$$

Given knowledge of the interleaver, the appropriate columns of $L^c(\vec{y})$ can be selected and manipulated to give correctly ordered symbol likelihoods for each SISO. The channel likelihood inputs to the two SISOs are designated $L^c(\vec{y}^1)$ and $L^c(\vec{y}^2)$. For the first SISO decoder

$$\begin{aligned} L^c(\vec{y}^1) &= [\ \ L^c(\vec{u})\ \ L^c(\vec{p}^1)\ \ ] && (2.27)\\ &= [\ \ L^c(\vec{c}^1)\ \ ]. && (2.28)\end{aligned}$$

The second SISO decoder requires interleaved information symbol likelihoods so the likelihoods $L^c(\vec{u})$ are first interleaved to form $L^c(\vec{u}^\pi)$, then

$$L^c(\vec{y}^2) = [\ L^c(\vec{u}^\pi)\ \ L^c(\vec{p}^2)\ ]. \tag{2.29}$$

In Figure 2.4 $L^o(\vec{u})$ represents *a priori* information about $\vec{u}$. In virtually all situations, source symbols are considered to be equi-probable, so $L^o(\vec{u})$ can usually be omitted, but it is included in this discussion for completeness. The *a priori* input to the first SISO is initialized with $L^{e2}(\vec{u}) = 0$, so $L(\vec{u}) = L^o(\vec{u})$.

Once these SISO inputs are formed, the BCJR algorithm is executed within SISO 1. The uncoded output of the first SISO $L^{e1}(\vec{u})$ is added to $L^o(\vec{u})$, interleaved, and used as input to the second SISO as $L(\vec{u}^\pi)$. $L^{e1}(\vec{u})$ is information that resides in the first code, and is independent of the channel likelihoods and the encoded information in Code 2. Therefore it is valid *a priori* information to give to the second SISO. Alternatively the output of the second SISO $L^{e2}(\vec{u}^\pi)$, information encoded by Code 2, is independent of encoded information in Code 1, so it can be added, after de-interleaving, to $L^o(\vec{u})$ to form $L(\vec{u})$ before executing SISO 1 a second time. This algorithm can be executed iteratively. At any time a decoder estimate of the transmitted information word $\vec{u}$, can be made by adding all available information and choosing the word with the maximum likelihood at each time index, that is:

$$\hat{u}_k = \max_{All\ Words} \left\{ \sum_{All\ Symbols} \left[\ L^c(u_k) + L^{e1}(u_k) + L^{e2}(u_k) + L^o(u_k)\ \right] \right\}. \tag{2.30}$$

As this process iterates the decoding decisions gradually improve, but the extrinsic information from each SISO slowly lose their independence from one another (starting with the second iteration through the second SISO) and the rate of improvement in decoding with each iteration soon slows and eventually ceases.

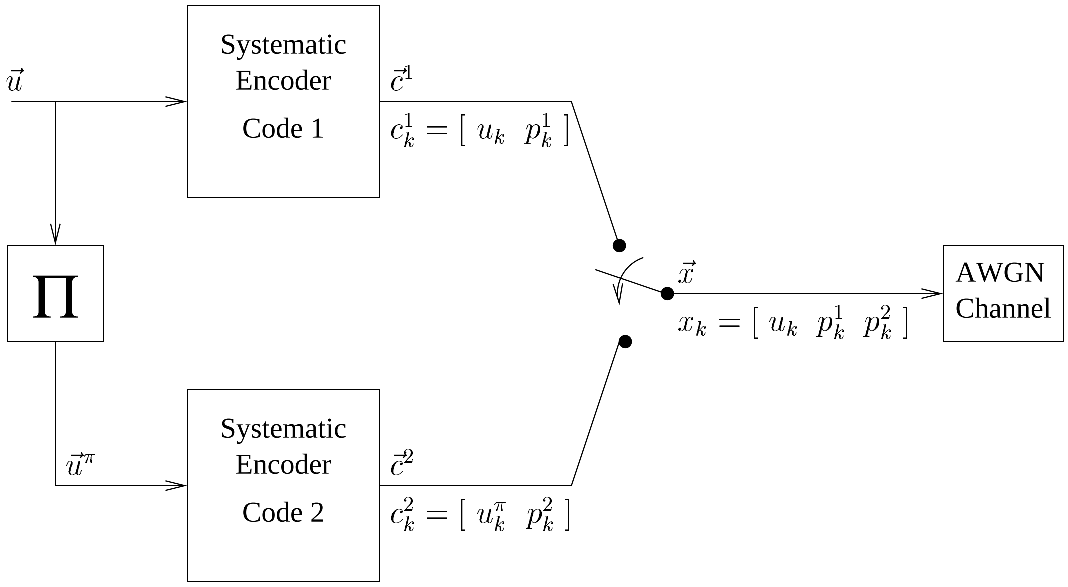

Figure 2.3: PCCC Encoder

This stopping point can be determined arbitrarily by picking a large number of iterations to assure convergence, or by examining the cross entropy of likelihoods from iteration to iteration [5]. When decoding accuracy is improving with each iteration the magnitude of the cross entropy will steadily drop but once cross entropy fails to improve, the turbo decoder has converged to its solution and repeated iterations will not change decoding decisions.

There is a large body of theoretical work about choice of interleaver, block length, and encoder polynomials. The optimal interleaver is unknown, however, Benedetto and Montorsi [3] have determined an upper error bound for a turbo code using a theoretical interleaver called a uniform interleaver. Furthermore they showed that a random interleaver will perform close to the theoretical uniform interleaver. It is important for the interleaver to interleaves code symbols and not simply code words. This is especially important when transmitting over a channel with memory. It is advantageous for the block length of the coded information (and interleaver) to be long, but this leads to a proportionately long latency at

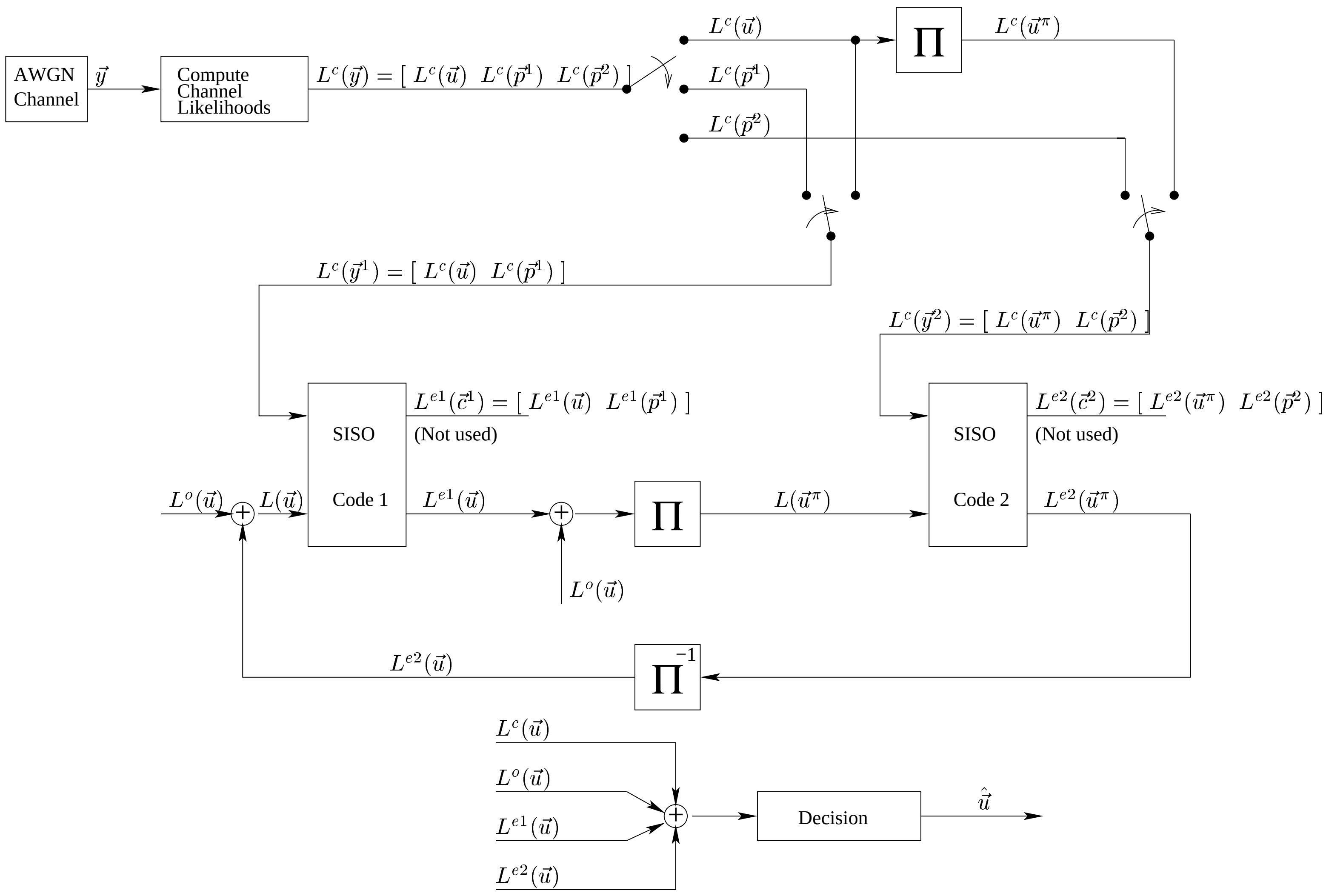

Figure 2.4: Iterative PCCC Decoder

the receiver. Finally the constituent codes used in turbo coding do not need to be complex, but do need to be recursive. The advantage of a recursive code is that it results in a uniform distribution of weights after interleaving of low weight input sequences [3].

It is important to take note that the turbo algorithm is not optimal. However, evidence [3] suggests that the suboptimal algorithms used in turbo codes can perform very close to optimal. Actual maximum likelihood decoding is impossible because of the huge number of states generated by interleaving a large block of data between two simple convolutional encoders.

2.1.3 Turbo Code in Serial Concatenated Configuration

Serial concatenated convolutional codes (SCCC) are more easily grasped after the discussion of the parallel case above. Figures 2.5 and 2.6 show block diagrams of a serial concatenated encoder and decoder respectively. The components in the figures perform the same functions as in the parallel case although the interconnections differ in the serial case. The first and second constituent codes of the encoder are often referred to as the outer and inner codes, respectively. This terminology arises because it avoids referring to encoders and SISOs by number. The later practice is confusing because the inner code (code 2) is decoded first at the receiver.

In the transmitter, note that the entire codeword of the outer encoder is interleaved to form the input to the inner encoder, $\vec{u}^2 = \vec{c}^{\pi}$, and the output of the inner encoder $\vec{c}^2 = \vec{x}$, is the vector used to modulate the signal for transmission. The encoders work exactly as in the parallel case. The intermediate codewords are formed as follows:

$$c_k^1 = [\ u_k^1\ \ p_k^1\ ], \tag{2.31}$$

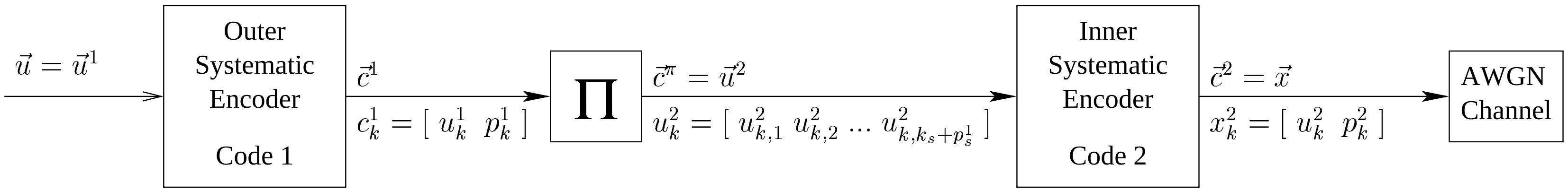

Figure 2.5: SCCC Encoder

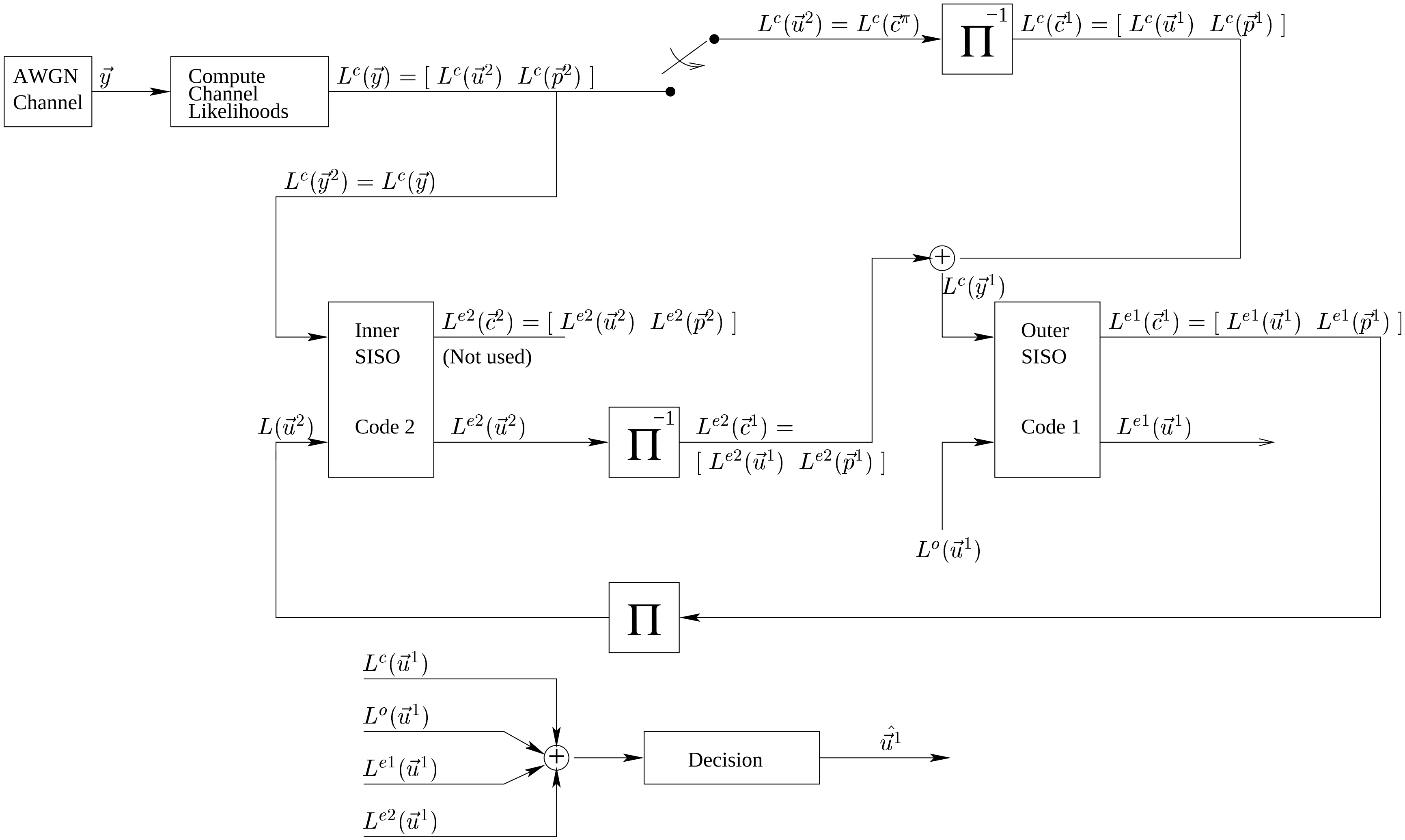

Figure 2.6: Iterative SCCC Decoder

and

$$c_k^2 = x_k = [\ u_k^2 \ \ p_k^2 \ ]. \tag{2.32}$$

The number of the information symbols input into the encoder and number of parity symbols added by the outer and inner encoders are k_s, p_s^1, and p_s^2 respectively. The overall rate of the code is $\frac{k_s}{k_s+p_s^1+p_s^2}$. Note, there is no overtly redundant information that can be discarded prior to transmission as was possible in the parallel case.

In the serial decoder as in the parallel case, likelihoods $L^c(\vec{y})$ are formed from the received signal $\vec{y}$. Again a likelihood can be attached to each symbol individually because of independence of transmission through the channel. For the inner SISO the entire $L^c(\vec{y})$ is used as input and written for clarity as

$$L^c(\vec{y}^2) = [\ L^c(\vec{u}^2) \ \ L^c(\vec{p}^2) \ ]. \tag{2.33}$$

For $L^c(\vec{y}^1)$, the intrinsic input to the outer SISO, a component of information comes from the channel and another comes from the inner SISO. The channel component is formed by taking $L^c(\vec{u}^2) = L^c(\vec{c}^\pi)$ from $L^c(\vec{y})$ and de-interleaving to form

$$L^c(\vec{c}^1) = [\ L^c(\vec{u}^1) \ \ L^c(\vec{p}^1) \ ]. \tag{2.34}$$

The inner SISO, contributes $L^{e2}(\vec{u}^2) = L^{e2}(\vec{c}^\pi)$ which is de-interleaved to form

$$L^{e2}(\vec{c}^1) = [\ L^{e2}(\vec{u}^1) \ \ L^{e2}(\vec{p}^1) \ ]. \tag{2.35}$$

These are summed to form the intrinsic input to the outer SISO, $L^c(\vec{y}^1) = L^c(\vec{c}^1) + L^{e2}(\vec{c}^1)$. Note that this differs from the parallel case where the intrinsic SISO input comes entirely from the channel.

The *a priori* input for the outer SISO is usually absent since it is limited to $L^o(\vec{u}^1)$. Extrinsic information about the coded words from the outer SISO, $L^{e1}(\vec{c}^1)$, correspond after interleaving to independent information about the words $\vec{u}^2$ and is thus suitable as *a priori* inputs to the inner SISO. The symbol estimates at time index k are formed similarly to the parallel case as

$$\hat{u}_k = \hat{u}_k^1 = \max_{All\ Words} \left\{ \sum_{All\ Symbols} \left[L^c(u_k^1) + L^{e1}(u_k^1) + L^{e2}(u_k^1) + L^o(u_k^1) \right] \right\}. \quad (2.36)$$

Iteration around the loop will initially improve overall decoding accuracy, but improvement eventually ceases. Stopping criteria based on cross entropy are similar to the parallel case.

2.2 Turbo Equalization

Equalization involves compensating for receptions over multiple transmission paths. This section describes modification of a turbo code to equalize data received over an ISI channel.

An ISI channel is most commonly modeled by a tap delay line as shown in Figure 2.7. In the figure, the M channel coefficients, h_m, and Gaussian noise, n_k, are complex. The input $\vec{u}$ is a real sequence of symbols of alphabet size N_I and the output vector $\vec{y}$ is complex. Given the input $\vec{u}$ the ISI channel can be considered to be in one of N_I^{M-1} states that are characterized by the past $M-1$ symbols sent into the channel.

This channel model is readily adapted to the trellis model that was introduced in Figure 2.2. An edge, e_k, starts in the state, $s^S(e_k)$, labeled by the

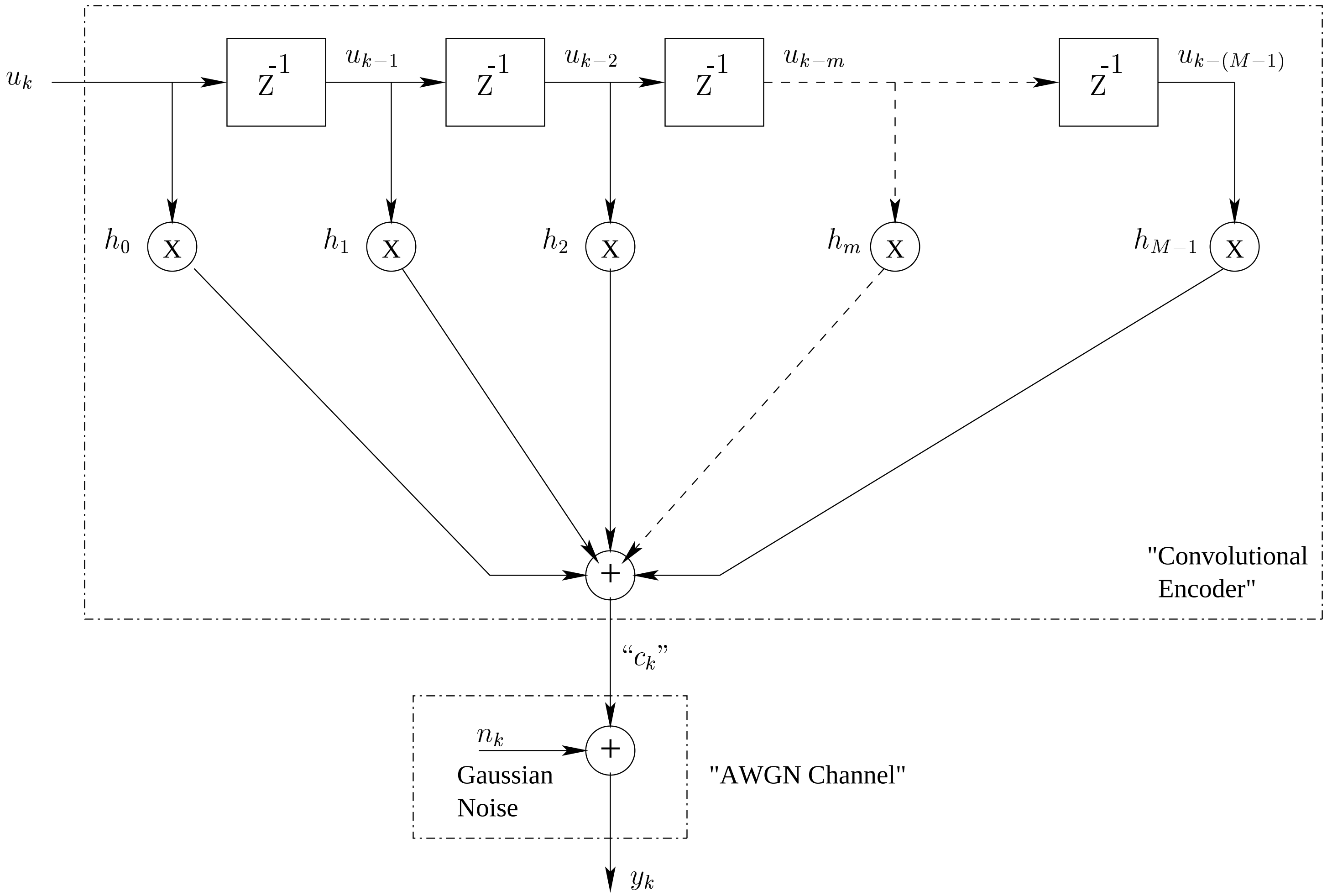

Figure 2.7: ISI Channel Modeled as Tap Delay Line

ordered set of inputs, $\{u_{k-1}, u_{k-2}, u_{k-3}, ..., u_{k-(M-1)}\}$ and ends in the state $s^E(e_k)$, labeled by the ordered set of inputs, $\{u_k, u_{k-1}, u_{k-2}, ..., u_{k-(M-2)}\}$. Associated with this transition is the input symbol, $u(e_k) = u_k$ and the output code symbol $c(e_k) = \sum_{m=0}^{M-1} h_m u_{k-m}$.

The same BCJR algorithm employed in the SISO module can be used to decode the channel (Equation 2.13) [2] [7]. The two key quantities to determine are $P\{u_k\}$, the *a priori* probability of each input symbol and $P\{y_k|e_k\}$, the probability of receiving the value y_k given the current and past $M-1$ channel inputs. If it is assumed the channel impulse response is known, at each time k, it is expected that the channel will produce the codeword c_k corrupted by channel noise [8]. Thus $P\{y_k|e_k\}$ can be calculated from the Gaussian distribution of y_k about the symbol associated with the given edge.

$$\begin{aligned} P\{y_k|e_k\} &= \frac{1}{\sqrt{2\pi}\sigma}\int_{y_k-\Delta y}^{y_k+\Delta y} e^{-\frac{1}{2\sigma^2}\cdot\|y-c(e_k)\|^2} dy \\ &= \frac{1}{\sqrt{2\pi}\sigma}\int_{y_k-\Delta y}^{y_k+\Delta y} e^{-\frac{1}{2\sigma^2}\cdot\|y-\sum_{m=0}^{M-1} h_m u_{k-m}\|^2} dy. \end{aligned} \quad (2.37)$$

Given the terms, $P\{u_k\}$ and $P\{y_k|e_k\}$, the equalization process needs to be integrated into a turbo coding scheme.

The dotted lines in Figure 2.7 indicates how the ISI channel model can be broken into two parts, the first behaving as a convolutional encoder and the second part being a simple AWGN channel. Figure 2.8 shows a block diagram of a serial encoder with the tap delay line functioning as the inner encoder and the Gaussian section of the channel taking on its usual role.

Figure 2.9 shows the turbo equalizer configuration. Two modifications relative to the SCCC decoder of Figure 2.6 have been made. First, the SISO equalizer that replaces the inner SISO accepts the channel input, $\vec{y}$, and calculates the channel likelihoods internally. Second, the extrinsic outputs are modified. The

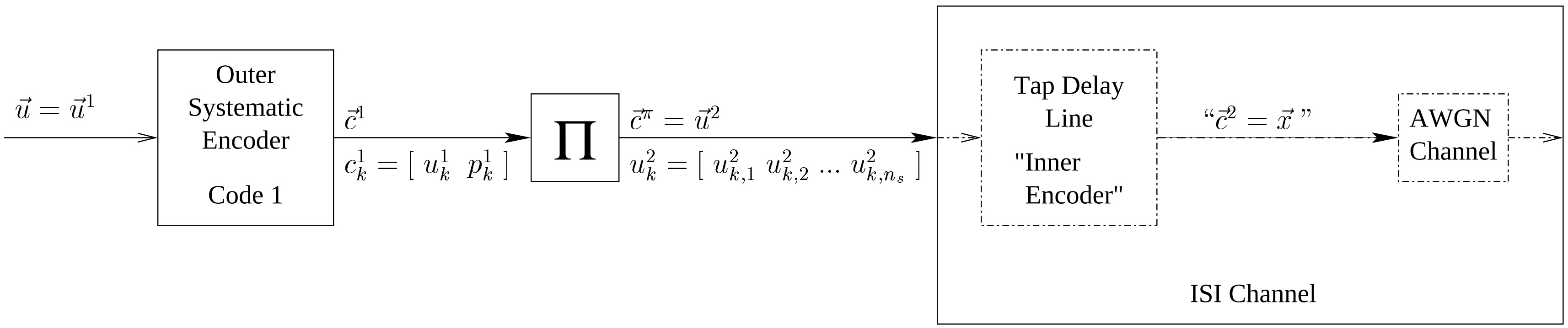

Figure 2.8: Encoder and ISI channel configured as a serial turbo encoder and AWGN channel.

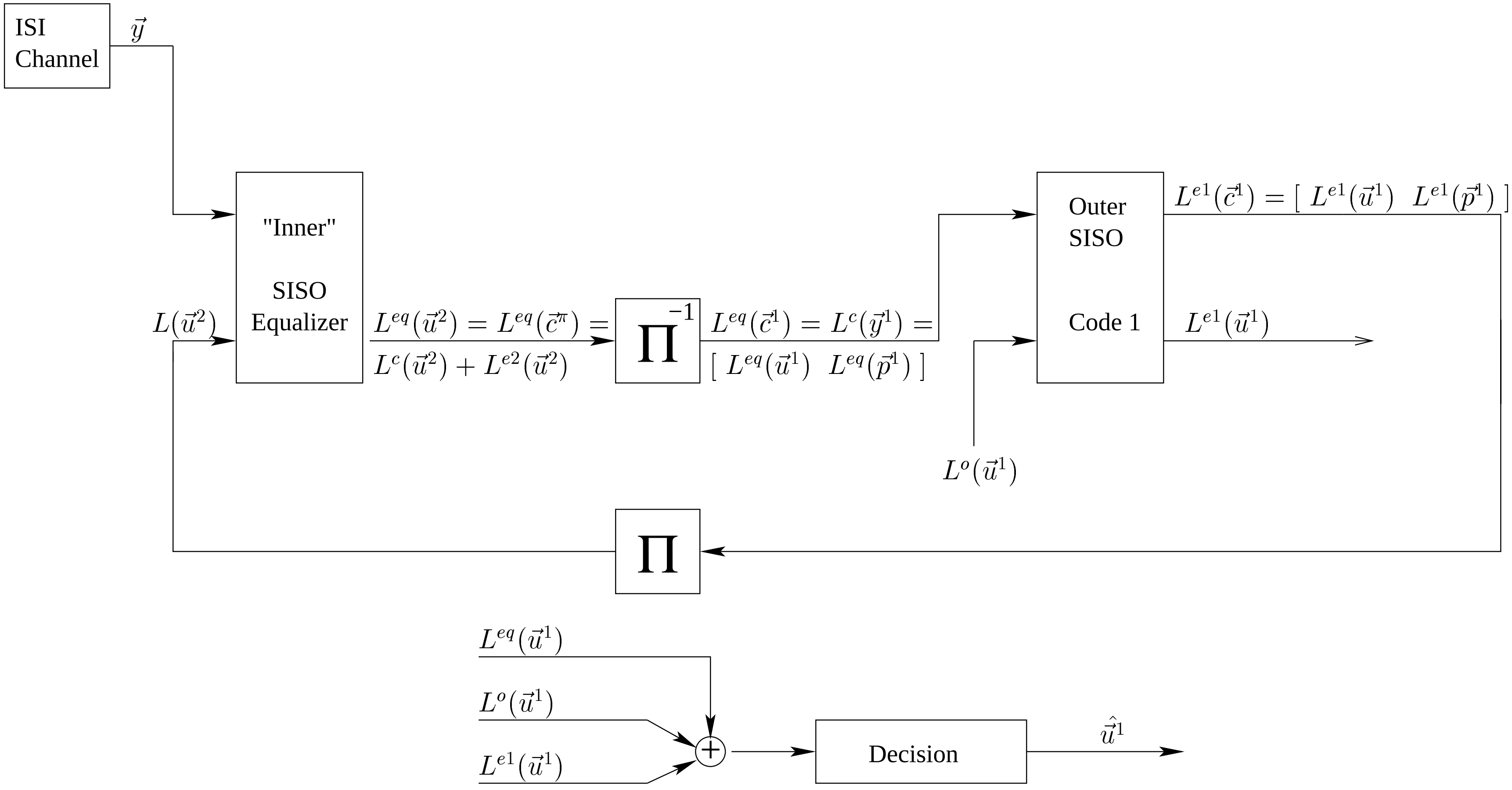

Figure 2.9: Turbo Equalizer.

upper output values are not needed so the output is eliminated. The lower output is critical.

Inside the SISO equalizer, symbol likelihoods are calculated using Equation 2.37, then the BCJR algorithm is executed as in the normal SISO. Once $L(u_k; \vec{y})$ has been calculated by Equation 2.15 the SISO equalizer output is calculated as the sum of

$$L^{eq}(u_k^2) = L^c(u_k^2) + L^{e2}(u_k^2) = L(u_k^2; \vec{y}) - L(u_k^2) \tag{2.38}$$

by Equation 2.17. The superscript on u_k is now used since the SISO equalizer is being used within the context of the turbo equalizer of Figure 2.9.

The decoding decision for the turbo equalizer, is made as in the SCCC,

$$\hat{u}_k = \hat{u}_k^1 = \max_{All\ Words} \left\{ \sum_{All\ Symbols} \left[L^{eq}(u_k^1) + L^{e1}(u_k^1) + L^o(u_k^1) \right] \right\}, \tag{2.39}$$

where $L^{eq}(\vec{u}^1)$ is extracted from $L^{eq}(\vec{c}^1)$. Symbol estimates are typically made using Equation 2.39 at the end of the last iteration. However, it is possible to obtain intermediate estimates of information words at any point in the decoding process using Equation 2.39. In fact, by using only one of the three likelihood arguments in the equation, an assessment of how well one or the other of the stages is decoding the information words. The ability to do this is valuable in gaining understanding of the interaction of the two stages as will be demonstrated later.

By sending $L^c(\vec{u}^2)$ through the SISO equalizer the upper pathway in the SCCC decoder of Figure 2.6 can be eliminated, thus leading to the structure of the turbo equalizer. Note that computation of channel likelihoods and execution of the BCJR can be done independently, so channel likelihoods could be calculated external to the SISO equalizer in a configuration analogous to the SCCC decoder. However, when attempting adaptation to a time varying channel the likelihood calculations are more intimately related to trellis decoding and the structure given is required.

2.3 Adaptive Turbo Equalization

The turbo equalizer uses Equation 2.37 requiring that the channel is known in terms of impulse response, h_m, and channel variance, $\sigma^2 = \frac{N_0}{2}$. However, even if these channel descriptors are known at the onset of transmission, in a mobile system they are time-varying and may change between blocks of encoded data or even within a long block. This section describes the modification of the SISO equalizer necessary to allow tracking of a time-varying channel.

The BCJR algorithm used in the SISO equalizer (and SISO) module is optimal in a maximum *a posteriori* sense, but it requires that the likelihoods for the edge transitions (Equation 2.37) be calculated for all time indices before any algorithm output can be generated. This is because the calculation of the values $\beta_k(s)$ is done recursively by Equation 2.8 starting from the last symbol in the transmitted sequence. Therefore the BCJR algorithm allows adaptation of channel estimates only after the input block fully emerges from the SISO equalizer stage. An approach based on this algorithm would not allow the SISO equalizer to benefit from improved estimation of the channel impulse response during the equalization step. Decoding with poor estimation of the channel would tend to increase the rate of erroneously decoded symbols which would in turn reduce the effectiveness of the adaptation process.

The BCJR algorithm can be modified to give suboptimal decoding that has only a minimal effect on accuracy of the algorithm output but allows generation of the output with only a short delay. The loss of decoding accuracy from this modified BCJR algorithm is insignificant relative to the gains obtained from providing the algorithm with a more up-to-date channel description.

The terms $\alpha_k(s^S(e_k))$ (Equation 2.7) and the channel transition probabilities of Equation 2.37 can be computed during the forward recursion, but the $\beta_k(s^E(e_k))$ terms (Equation 2.8) are computed in a reverse direction from the end of the transmitted block. However, the state of the encoder at time $k + D$ has minimal effect on the edge transmission probabilities at time k, when D is significantly larger than the memory size of the convolutional encoder [9]. Thus the BCJR algorithm is modified as follows. Rather than initializing $\beta_K(s)$ according to Equation 2.10 set

$$\beta_{k+D}(s) = \alpha_{k+D}(s) \;\; \forall\, s \tag{2.40}$$

and do the backwards recursion for D periods to determine $\beta_k(s)$. Using this approach symbols can be estimated each period with a time delay of D, and the decoded symbol $\hat{u}_k$ is available to estimate channel coefficients for use at time $k + D + 1$. In this modified BCJR algorithm the corresponding joint symbol and channel output likelihoods are designated $L(u_k^2; \vec{y}_D)$ rather than the usual $L(u_k^2; \vec{y})$, where the subscript D indicates that the likelihood is only based on the first $k+D$ symbols in $\vec{y}$.

Having the symbol estimates available with a time delay of only D allows use of a recursive least squares (RLS) algorithm inside the SISO equalizer to calculate the time-varying channel coefficients. As each received symbol is generated by the equalizer, the RLS algorithm uses the symbol estimate to update the channel model that is used in Equation 2.37.

Figure 2.10 is a block diagram of the adaptive SISO equalizer stage used in the adaptive turbo equalizer. The figure illustrates how the modified BCJR algorithm is integrated with the RLS algorithm to adapt to a time-varying ISI channel.

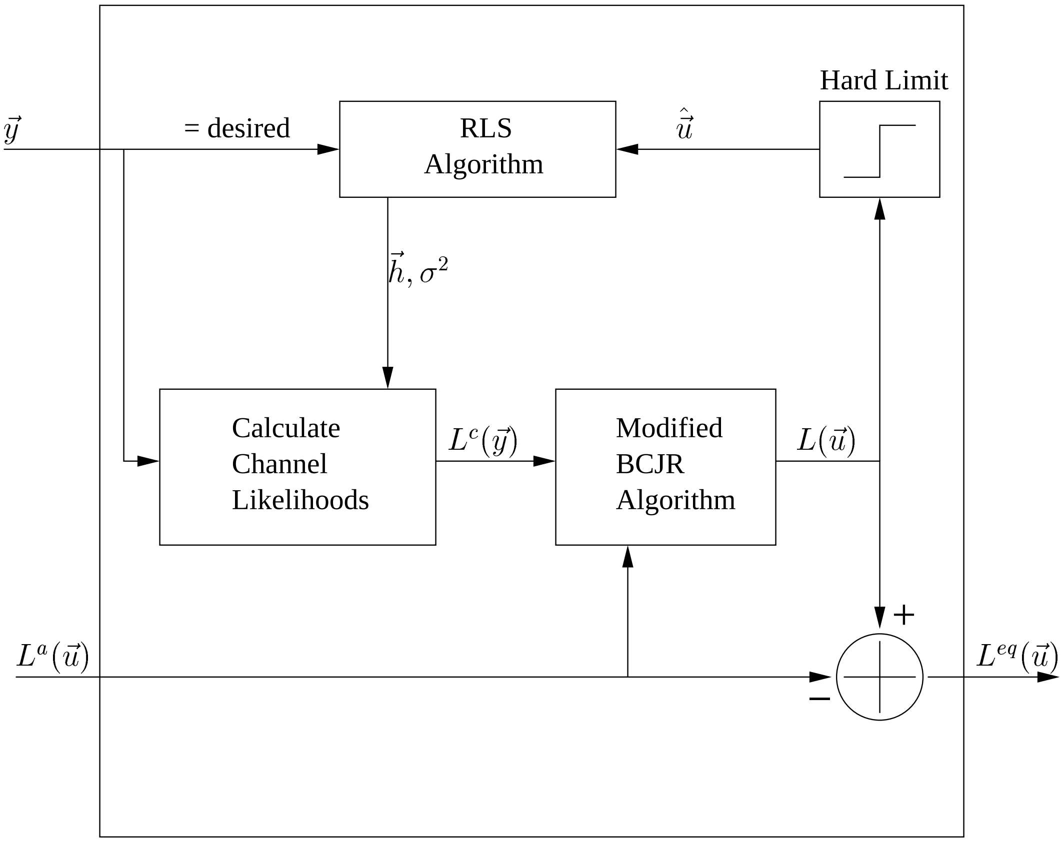

Figure 2.10: Adaptive SISO equalizer stage

The method of RLS is well described in Haykin [10]. This description of RLS will only cover enough detail to illustrate how RLS is integrated within the turbo equalizer. Normally, an RLS algorithm is given channel outputs and determines the coefficients of an inverse filter that produces the desired symbol outputs. In the case of the SISO equalizer the input and output to the RLS algorithm are reversed and the algorithm generates channel coefficient estimates. That is: estimated symbol outputs based on the modified BCJR algorithm are given as input and the RLS algorithm estimates a series of coefficients that will generate the received channel outputs. Below are the four recursive equations, used after each symbol reception, to estimate the channel impulse response.

$$\xi(k) = y(k) - \hat{\vec{h}}^{*}(k-1)\hat{\vec{u}}^{2}_{M}(k) \tag{2.41}$$

$$\vec{g}(k) = \frac{\lambda^{-1}\mathbf{P}(k-1)\hat{\vec{u}}^{2}_{M}(k)}{1+\lambda^{-1}\hat{\vec{u}}^{2^*}_{M}(k)\mathbf{P}(k-1)\hat{\vec{u}}^{2}_{M}(k)} \tag{2.42}$$

$$\hat{\vec{h}}(k) = \hat{\vec{h}}(k-1) + \vec{g}(k)\xi^{*}(k) \tag{2.43}$$

$$\mathbf{P}(k) = \lambda^{-1}\mathbf{P}(k-1) - \lambda^{-1}\vec{g}(k)\hat{\vec{u}}^{2^*}_{M}(k)\mathbf{P}(k-1). \tag{2.44}$$

At time k, Equation 2.41 gives the error vector $\xi(k)$, calculated as the difference between the received vector $\vec{y}$ and the estimated channel output. The estimated channel output is the convolution of the most recent estimated channel coefficients, $\hat{\vec{h}}^{*}(k-1)$ ($()^*$ denotes the Hermitian transpose), and the estimate of the M most recently transmitted symbols of the vector $\vec{u}^2$, designated $\hat{\vec{u}}^{2}_{M}(k)$, where $\hat{\vec{h}}^{*}(k-1) = [\ \hat{h}_0(k-1),\ \hat{h}_1(k-1),\ \cdots,\ \hat{h}_{M-1}(k-1)\ ]$ and

$$\hat{\vec{u}}^{2}_{M}(k) = \begin{bmatrix} \hat{u}^2_k \\ \hat{u}^2_{k-1} \\ \vdots \\ \hat{u}^2_{k-(M-1)} \end{bmatrix} \tag{2.45}$$

In calculation of the gain vector, $\vec{g}(k)$, $\mathbf{P}(k-1)$ is the inverse of the $M \times M$ correlation matrix, and λ is the exponentially weighted forgetting factor. If $\lambda = 1$ this method reduces to ordinary least squares. Typically λ is slightly less than one and it governs how quickly the effect of past symbol receptions and estimations is lost. The interaction of this factor with the turbo equalization algorithm will be illustrated later.

The vector $\hat{\vec{u}}^2_M(k)$, used at each time interval by the RLS algorithm, is calculated from all information available at the time within the SISO equalizer. This information, $L(u^2_k, \vec{y}_D)$, is generated by the modified BCJR algorithm.

Channel noise variance was calculated as [10]

$$\sigma^2(k) = \sigma^2(k-1)[1-\lambda] + \xi^*(k)[y(k) - \hat{\vec{h}}^*(k)\hat{\vec{u}}^2_M(k)]. \tag{2.46}$$

The output of the SISO equalizer is calculated using Equation 2.38 with $L(u^2_k; \vec{y}_D)$ substituted for $L(u^2_k; \vec{y})$.

2.4 Receiver Complexity

A significant disadvantage of a turbo code is the receiver complexity. The number of operations necessary within a SISO is derived below followed by a similar assessment for the SISO equalizer. Recall that there are two SISOs within a turbo code and repeated iterations are made through each SISO. The complexity grows exponentially with the size of the encoder memory plus input word length. This is not a major constraint with a SISO since the encoder memory and word length can be controlled and typically do not need to be large. Alternatively the memory in a SISO equalizer is equal to $M-1$, where M is the length of the channel impulse response, and M is beyond the control of the designer and potentially large.

Once the channel symbol probabilities are determined, there are three sets of intermediate values to be calculated at each time period k. First the state transition probabilities, $P\left\{s^E(e_k); y_k|s^S(e_k)\right\}$ are calculated then $\alpha_k(s)$ and $\beta_k(s)$ are calculated recursively.

Recall that there are k_s and n_s encoded and transmitted symbols respectively for each time k of the total K periods in a block. The symbol alphabet is size N_I and for a convolutional code with N_M memory locations the encoder has $N = N_I^{N_M}$ different states leading to a total of $N_E = NN_I^{k_s} = N_I^{N_M+k_s}$ edges at each time. The number of operations needed to calculated the symbol probabilities at each time period k is discussed below.

Edge transition probabilities require $n_s + k_s - 1$ multiplications per edge. The k_s multiplications are not needed if no *a priori* input is given to the SISO. The results of each calculation along an edge (N_E results) must be stored for use in calculating $\alpha_k(s)$, $\beta_k(s)$, and the final symbol likelihoods.

The values $\alpha_k(s)$ and $\beta_k(s)$ each require similar computational efforts. There are $N_I^{k_s}$ multiplications and $N_I^{k_s} - 1$ additions for each state. Thus at each time k a total of $NN_I^{k_s} = N_E$ multiplications followed by $N(N_I^{k_s} - 1)$ additions. The $\alpha_k(s)$ data must be stored (N values), the $\beta_k(s)$ data can be used immediately to calculate the final symbol likelihoods, after which they can be discarded.

To calculate the symbol probabilities from the above three sets, along each edge there are 2 multiplications followed by $\frac{N_E}{N_I^{k_s}} - 1 \cong N$ additions to calculate each symbol probability. Calculation of each of the two extrinsic SISO outputs requires one multiplication (division) per symbol. Thus this entire step involves $2N_E + 2k_s$ multiplications and $N = N_I^{N_M}$ additions.

Table 2.1 contains a summary of the above operations and storage requirements and gives the highest order factor contributing to the complexity of the SISO. Additionally, the data is given for an adaptive SISO equalizer.

	Multiplications	Additions	Storage
SISO	$KN_I^{N_M+k_s}(n_s+k_s+3)$	$2KN_I^{N_M+k_s}$	$KN_I^{N_M+k_s}$
SISO equalizer	$K\frac{n_s}{k_s}N_I^{N_M+k_s}(4+D)$	$(K\frac{n_s}{k_s}+D)N_I^{N_M+k_s}$	$DN_I^{N_M+k_s-1}$

Table 2.1: Major factors effecting computational complexity of the SISO and SISO equalizer stages of the adaptive turbo equalizer.

The SISO equalizer is more complex for several reasons. For the SISO equalizer note that the number of states is $N = N_I^{N_M}$ where $N_M = M - 1$. This is the same as for the standard SISO stage. Typically a convolutional code will have N_M in the range 2 to 5. While the channel used in these experiments set $N_M = 4$, actual channels can have significantly longer impulse responses. The modification of the BCJR algorithm to adapt to a time-varying channel also adds to the complexity. The backwards recursion step for calculating the $\beta_k(s)$ values requires calculating $\beta_k(s)$ D times at each time period versus just once in the regular BCJR algorithm. There is only a single multiplication to calculate edge transition probabilities, but the length of the equalizer trellis increases inversely with the code rate so all operations are done $\frac{K}{R} = K\frac{n_s}{k_s}$ times rather than just K times. The memory requirements are reduced significantly by $\frac{D}{K}$ with this algorithm since only D periods of edge transition probabilities need to be stored. Despite the storage savings possible, the increased number of calculations required can make the adaptive SISO equalizer much more complex than the standard SISO. Overall the complexity of this stage is proportional to $K\frac{n_s}{k_s}N_M N_I^{N_M+k_s}$ since D is directly proportional to N_M. For a long channel impulse response $N_M \cong M$ so complexity of the equalizer stage is mainly determined by the characteristic of the channel once encoder input word length and symbol alphabet has been chosen.

There is additional complexity within the SISO equalizer due to the RLS algorithm. The operations in Equations 2.41 to 2.44 are repeated only once per time period and their complexity grows with the square of channel memory size.

Thus the order of their complexity $K\frac{n_s}{k_s}M^2$ is small compared to the probability calculations on all but the channels with the shortest impulse response.

In summary the complexity of both receiver stages grows with the alphabet size to the $N_M + k_s$ power. The equalizer stage is more complex than the SISO stage, being proportional to $\frac{n_s}{k_s}MN_I^{M+k_s-1}$. The complexity can be controlled to some extent by encoding messages using a small symbol alphabets, short words, and small convolutional code memory size. However the equalizer stage's complexity is determined by the length of the communication channel's impulse response and this complexity rapidly becomes unmanageable. Figure 2.11 illustrates the number of multiplications per transmitted bit in the SISO equalizer versus the order of the channel's impulse response, given binary transmission symbols. For a channel of length five, which is used through out this thesis, there are approximately 750 multiplications required per transmitted bit. Recall that these operations are required during each iteration of the SISO equalizer.

Some simplifications of the receiver can be made by taking the logarithm of the probabilities [4] [5]. First, multiplication of probabilities can be changed to additions of logarithms of probabilities. Second, additions of probabilities can be simplified to maximum operations when logarithms are used. The logarithm of a sum of exponentials is to a good approximation, the maximum exponent of the exponentials, that is

$$a = \ln\left[\sum_i e^{a_i}\right] \cong a_M \tag{2.47}$$

where

$$a_M = \max_i a_i. \tag{2.48}$$

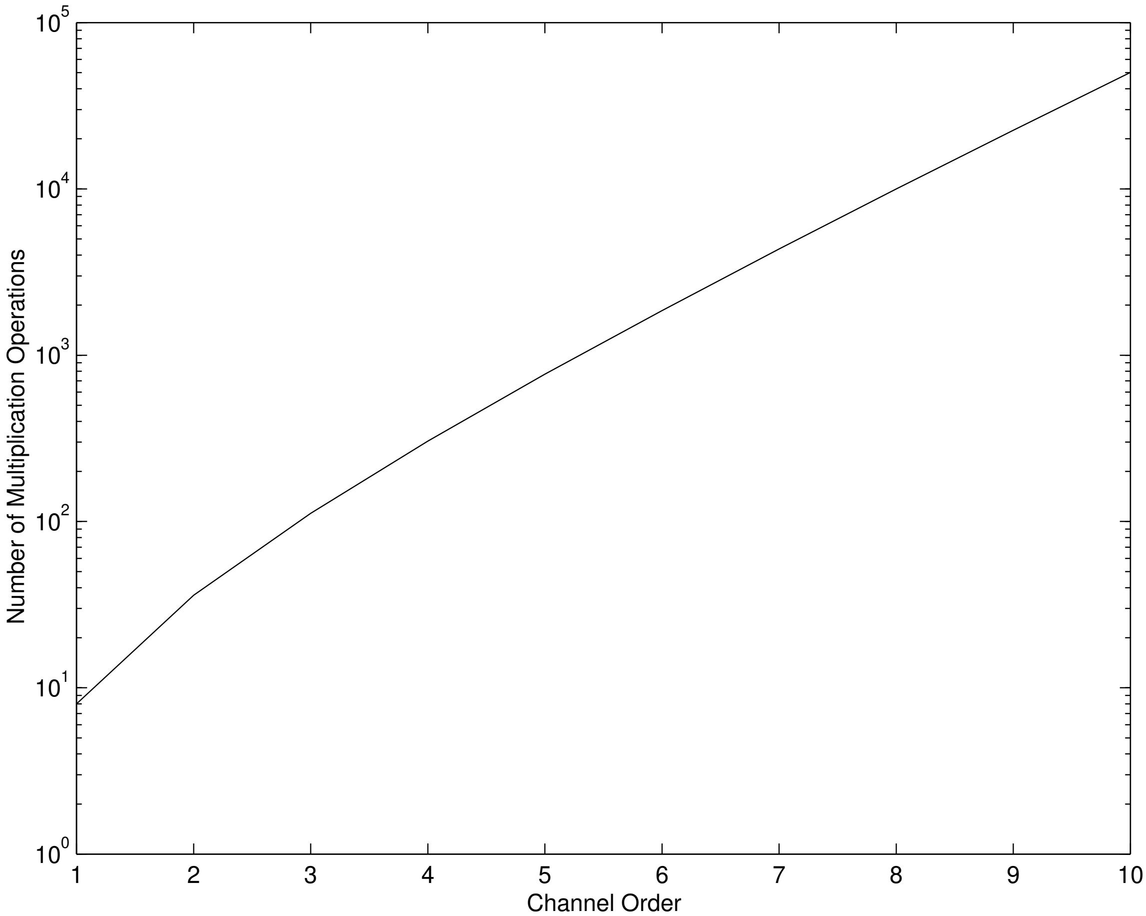

Figure 2.11: The number of multiplication operations per bit transmitted in the SISO equalizer versus the length of the channel impulse response (binary channel).

This approximation assumes that the exponentials differ in value and is of sufficient accuracy for the SISO algorithm, but it can be improved upon with minimal computational effort. A correction term can be added to significantly improve accuracy by using a look-up table of as few as eight values. The index values for the table are calculated based on the difference between a_M and the second largest exponent [4].

Finally, one further computational simplification is possible for binary input symbols. Recall that all symbol decisions ultimately come down to taking the ratio of two symbol likelihoods or the difference of log-likelihoods. With binary data there is only a single ratio to form, thus by forming the log-likelihood ratio initially and performing all calculations with the ratio the number of additions, maximizations, and stored values are halved. Thus by using log-likelihood ratios for binary symbols the SISO complexity grows with $K2^{N_M+k_s-1}$ and multiplications are converted to additions and additions to maximizations.

In all the experiments described in this thesis binary symbols were used and the above computational savings were used including use of log-likelihood ratios. Maximization operations were corrected with an eight value look-up table. With these modifications the key equations (Equation 2.13, Equation 2.7, Equation 2.8, Equation 2.15, and Equation 2.16) of the BCJR algorithm become:

$$L\left(P\{e_k;\vec{y}\}\right) = L\left(\alpha_{k-1}[s^S(e_k)]\right) + L(u_k) + L^c(y_k) + L\left(\beta_k[s^E(e_k)]\right), \tag{2.49}$$

$$L\left(\alpha_k[s]\right) = \max_{e:s^E(e)=s} \left\{ L\left(\alpha_{k-1}[s^S(e)]\right) + L(u_k) + L^c(y_k) \right\}, \tag{2.50}$$

$$L\left(\beta_{k-1}[s]\right) = \max_{e:s^S(e)=s} \left\{ L\left(\beta_k[s^E(e)]\right) + L(u_k) + L^c(y_k) \right\}, \tag{2.51}$$

$$\begin{aligned} \text{Ł}\left(P\{u_k;\vec{y}\}\right) = & \max_{e:u(e)=+1} \left\{ L\left(\alpha_{k-1}[s^S(e)]\right) + L(u_k) + L^c(y_k) + L\left(\beta_k[s^E(e)]\right) \right\} \\ & - \max_{e:u(e)=-1} \left\{ L\left(\alpha_{k-1}[s^S(e)]\right) + L(u_k) + L^c(y_k) + L\left(\beta_k[s^E(e)]\right) \right\}, \end{aligned} \tag{2.52}$$

and

$$\begin{aligned} L\left(P\{c_{k,i};\vec{y}\}\right) = & \max_{e:c_i(e)=+1} \left\{ L\left(\alpha_{k-1}[s^S(e)]\right) + L(u_k) + L^c(y_k) + L\left(\beta_k[s^E(e)]\right) \right\} \\ & - \max_{e:c_i(e)=-1} \left\{ L\left(\alpha_{k-1}[s^S(e)]\right) + L(u_k) + L^c(y_k) + L\left(\beta_k[s^E(e)]\right) \right\} \end{aligned} \tag{2.53}$$

respectively. Note that in these equations $L(u_k)$ and $L^c(y_k)$ are the log-likelihood ratio inputs of the SISO and Equation 2.53 is calculated for each symbol in the code word as designated by the subscript "i". The boundary conditions for Equations 2.50 and 2.51 set $s_0 = 0$ and all other states to $-\infty$ at $k = 0$ and $k = K$ respectively. SISO outputs, $\text{Ł}^e(u_k)$ and $L^e(c_k)$, are simply Equations 2.52 and 2.53 less the SISO inputs.

2.5 Example SCCC

One of the major difficulties in understanding turbo coding and decoding is that the data is literally shuffled and un-shuffled multiple times. This example is primarily designed to clarify the interconnections between the two encoders and the SISO modules in the turbo coding process.

Assume a SCCC with symbols drawn from a binary alphabet. The outer encoder is rate $\frac{1}{2}$ and inner encoder is rate $\frac{2}{3}$. Assume that encoded data is transmitted through an AWGN channel and log-likelihood ratios are calculated at the receiver. The block we wish to encode and transmit is only five words long and contains one-symbol words. To clearly identify the involved symbols, they are all labeled with lower case letters. The input to the outer encoder,

$$\vec{u} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} \tag{2.54}$$

generates the a vector of two-symbol codewords

$$\vec{c^1} = \begin{bmatrix} a & f \\ b & g \\ c & h \\ d & i \\ e & j \end{bmatrix}. \tag{2.55}$$

with parity symbols generated by the outer encoder ($\vec{p^1}$) represented by the letters f through j. A pseudo-random interleaver then shuffles these symbols and reforms a vector of five two-symbol words

$$\vec{c}^{\pi} = \begin{bmatrix} c & b \\ h & i \\ d & f \\ j & a \\ e & g \end{bmatrix}, \tag{2.56}$$

which is also called $\vec{u}^2$ because it is an input to inner encoder. The inner encoder is rate $\frac{2}{3}$ so it adds a third column of parity symbols ($\vec{p}^2$)to form

$$\vec{c}^2 = \begin{bmatrix} c & b & k \\ h & i & l \\ d & f & m \\ j & a & n \\ e & g & o \end{bmatrix}. \tag{2.57}$$

$\vec{c}^2$ is also known as the vector $\vec{x}$ of three-symbol codewords that modulates the carrier. Symbols are sent through the channel and corrupted with noise one symbol at a time in the order

$$[\, c\; b\; k\; h\; i\; l\; d\; f\; m\; j\; a\; n\; e\; g\; o\,]. \tag{2.58}$$

As this bit stream is received the log-likelihood ratios are formed and designated by upper case letters with the superscripts "c" to indicated their origin from the channel. So initially, the stream of log-likelihood ratios

$$[\, C^c\; B^c\; K^c\; H^c\; I^c\; L^c\; D^c\; F^c\; M^c\; J^c\; A^c\; N^c\; E^c\; G^c\; O^c\,] \tag{2.59}$$

is calculated and immediately used to form the matrix

$$L^c(\vec{y}) = \begin{bmatrix} C^c & B^c & K^c \\ H^c & I^c & L^c \\ D^c & F^c & M^c \\ J^c & A^c & N^c \\ E^c & G^c & O^c \end{bmatrix}. \tag{2.60}$$

This entire matrix is equal to $L^c(\vec{y}^2)$, the input to the inner SISO. For the first iteration the *a priori* log-likelihood ratios, $L(\vec{u}^2)$, are all 0 for the inner SISO. The inner SISO generates

$$L^{e2}(\vec{u}^2) = \begin{bmatrix} C^{e2} & B^{e2} \\ H^{e2} & I^{e2} \\ D^{e2} & F^{e2} \\ J^{e2} & A^{e2} \\ E^{e2} & G^{e2} \end{bmatrix}, \tag{2.61}$$

the estimated log-likelihood ratios of the extrinsic information within the inner code. The superscript "$e2$" indicates that this is extrinsic information from Code 2. This matrix of log-likelihood ratios is de-interleaved to form

$$L^{e2}(\vec{c}^1) = \begin{bmatrix} A^{e2} & F^{e2} \\ B^{e2} & G^{e2} \\ C^{e2} & H^{e2} \\ D^{e2} & I^{e2} \\ E^{e2} & J^{e2} \end{bmatrix}. \tag{2.62}$$

In the upper path of Figure 2.6 the first two columns of $L^c(\vec{y})$ are also de-interleaved to form $L^c(\vec{c}^1)$ and then added to $L^{e2}(\vec{c}^1)$ to form the input to the outer SISO

$$L^c(\vec{y}^1) = L^{e2}(\vec{c}^1) + L^c(\vec{c}^1) = \begin{bmatrix} A^{e2} + A^c & F^{e2} + F^c \\ B^{e2} + B^c & G^{e2} + G^c \\ C^{e2} + C^c & H^{e2} + H^c \\ D^{e2} + D^c & I^{e2} + I^c \\ E^{e2} + E^c & J^{e2} + J^c \end{bmatrix}. \tag{2.63}$$

Note the log-likelihood ratios all refer to the same symbols and are in the same order as the original output from the outer encoder. The *a priori* input to the outer SISO is usually $\vec{0}$, but if not, it would be correctly ordered to correspond to the original information vector $\vec{u} = \vec{u}^1$. The two outputs of the outer SISO are

$$L^{e1}(\vec{c}^1) = \begin{bmatrix} A^{e1} & F^{e1} \\ B^{e1} & G^{e1} \\ C^{e1} & H^{e1} \\ D^{e1} & I^{e1} \\ E^{e1} & J^{e1} \end{bmatrix} \tag{2.64}$$

and

$$L^{e1}(\vec{u}^1) = \begin{bmatrix} A^{e1} \\ B^{e1} \\ C^{e1} \\ D^{e1} \\ E^{e1} \end{bmatrix}. \tag{2.65}$$

After interleaving, $L^{e1}(\vec{c}^1)$, yields

$$L(\vec{u}^2) = \begin{bmatrix} C^{e1} & B^{e1} \\ H^{e1} & I^{e1} \\ D^{e1} & F^{e1} \\ J^{e1} & A^{e1} \\ E^{e1} & G^{e1} \end{bmatrix} \tag{2.66}$$

with the log-likelihoods correctly aligned for the inner SISO and the start of the second iteration.

When it is necessary to make a symbol estimation the first column of $L^{e2}(\vec{c}^1)$ and $L^c(\vec{c}^1)$ are added to $L^{e1}(\vec{u}^1)$ (assuming $L^o(\vec{u}^1) = \vec{0}$),

$$\begin{bmatrix} A^{e2} + A^c + A^{e1} \\ B^{e2} + B^c + B^{e1} \\ C^{e2} + C^c + C^{e1} \\ D^{e2} + D^c + D^{e1} \\ E^{e2} + E^c + E^{e1} \end{bmatrix}, \tag{2.67}$$

and the sign of the log-likelihood ratio indicates the most likely value of the corresponding bit.

CHAPTER 3

Turbo Equalization - Performance

This chapter presents the performance of turbo equalization on a known time invariant channel. The data presented here will serve as a benchmark for assessing the adaptive turbo equalizer in later chapters. Initially, criteria for measuring performance are described and then literature results of modeling transmission through an ISI channel are reviewed.

3.1 Metrics of Performance

A major goal of any communication link is to have reliable communication with a minimized signal to noise ratio (SNR). A low SNR is typically associated with the benefits of low power consumption, small size, and hence portability. While error correction coding schemes such as turbo codes make possible dramatic reductions in SNR, these gains are acquired at the price of increased complexity at the receiver. The complexity manifests itself by increased power consumption at the receiving end, but more importantly it can introduce a significant delay due to the time required to decode the received signal. In some applications this latency is acceptable. The delay can be minimized by adding hardware decoding circuitry in the receiver.

To represent reliability of a transmission scheme, the bit error rate (BER) of received data is graphed versus SNR. The SNR is expressed in terms of $\frac{E_b}{N_0}$, where E_b is the average energy in each transmitted bit of information and $\frac{N_0}{2}$ is the intensity of the white noise process active on the channel. The ratio $\frac{E_b}{N_0}$ is the signal power-to-noise ratio where the power of the noise is defined in a bandwidth equivalent to that of the bit rate.

This presentation of the reliability is common in the literature [7] [2] [9] because it enables establishment of an SNR threshold for a given BER. Unfortunately this method is very imprecise at estimating very low BERs (especially BERs at SNRs above the SNR threshold) because the error events are so rare and the processing time necessary to encounter a significant number of these [7] rare events is very long. The SNR threshold is defined at a specified BER, but after the last iteration of decoding, the BER versus $\frac{E_b}{N_0}$ curve is steep. Thus, for a time-invariant ISI channel, the SNR threshold is not very dependent on SNR in the BER $= 10^{-5}$ range.

Complexity is expressed in terms of the number of additions, multiplications, maximization operations, and memory requirements. The rate of growth in the number of these operations when decoding increases exponentially with the memory size of the constituent codes of the turbo code. In the case of a turbo equalizer the memory size grows with the length of the channel impulse response.

3.2 Literature

Bauch *et. al.* [2] and Hagh [7] have both published evaluation of turbo equalization on a time invariant ISI channel with known impulse response and noise variance. In both studies they modeled the transmission of binary symbols using a rate $\frac{1}{2}$ encoder with memory of length 4, a 4096 information message block length, and the same channel impulse response. Both used a pseudo-random interleaver. The ISI channel had an impulse response of

$$h(n) = 0.227\delta(n) + 0.460\delta(n-1) + 0.688\delta(n-2) + 0.460\delta(n-3) + 0.227\delta(n-4). \tag{3.1}$$

This channel has two zeros close to the unit circle, both near the frequency $f = \frac{R}{3}$, where R is the bit rate.

Figure 3.1 show the results of a simulation of transmission over the same ISI channel used by Bauch *et. al.* [2] and Hagh [7]. The results were obtained using the code generating matrix ([$1, \frac{1+D+D^2+D^4}{1+D^3+D^4}$]) within the encoder, the same information block length (4096), and a pseudo-random interleaver. For each value of $\frac{E_b}{N_0}$ block transmissions were repeatedly modeled and decoded until seven blocks, containing errors after the fifth iteration, were encountered. The overall bit error rate after each iteration is plotted in the figure against $\frac{E_b}{N_0}$. After 5 iterations a BER of 10^{-4} was achieved with a $\frac{E_b}{N_0} = 6.15\ dB$. This compares to SNR thresholds of $6.4\ dB$ and $5.4\ dB$ reported by Hagh [7] and Bauch *et. al.* [2] respectively. All three results were derived using the parameters listed above except the code generating matrix used by Bauch *et. al.* [2] are is known, and other minor methodological differences may also exist among the three studies. The precision of BER estimates falls rapidly at levels $< 10^{-3}$ due to the large number of simulations necessary to accurately estimate the number of rare error events. The results shown in Figure 3.1 are comparable to Hagh [7], but do not reflect the apparently better performance obtained by Bauch *et. al.* [2]. In comparisons that follow in this thesis, the results of the adaptive equalizer will be compared with those presented in Figure 3.1 since all parameters and methods are known and can be varied individually.

The relative position of the curves in the Figure 3.1 are typical. The family of curves reported by Bauch *et. al.* [2] and Hagh [7] are shifted in the $\frac{E_b}{N_0}$ direction by the amount of the differences in SNR thresholds reported above. Also of note in Figure 3.1 is the asymptotic improvement of performance of the turbo equalizer with iterations. Early iterations produce rapid improvement of the BER, but more than five iterations does little to enhance receiver accuracy.

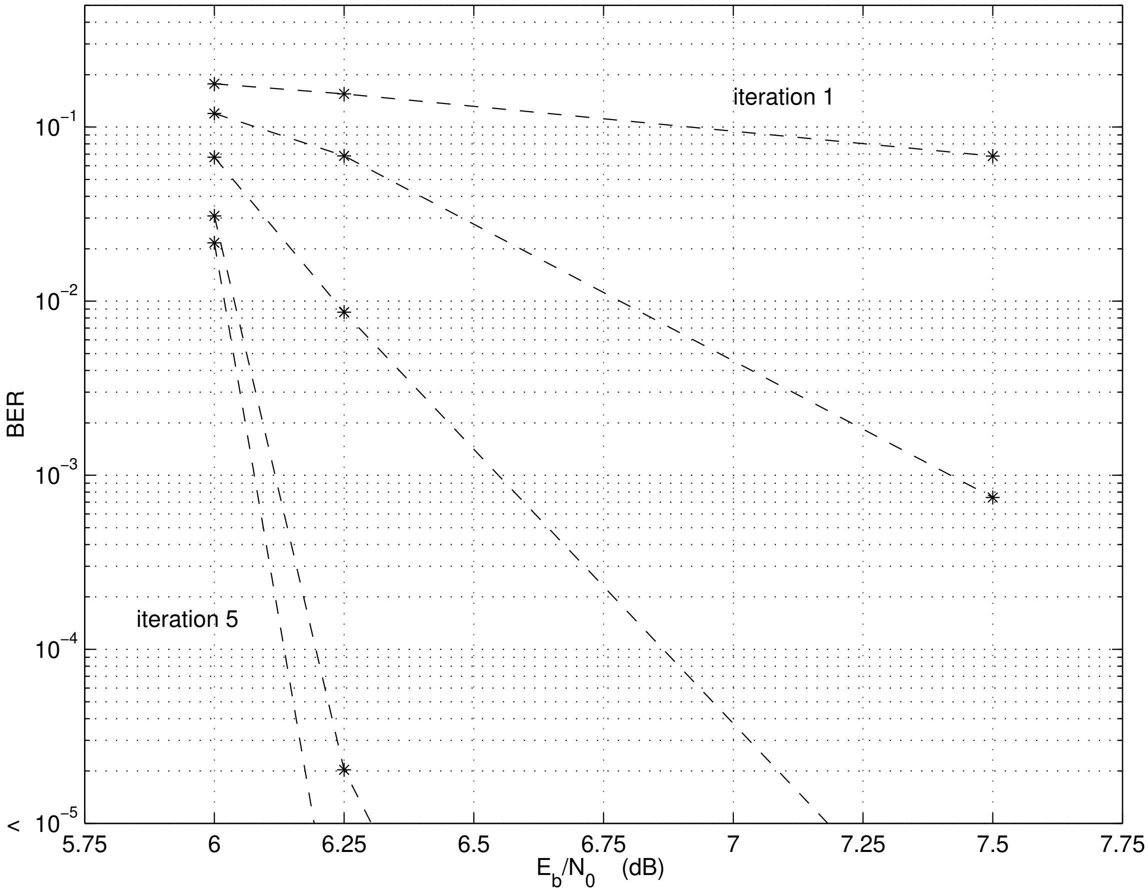

Figure 3.1: Turbo Equalizer Bit Error Rate versus $\frac{E_b}{N_0}$ after Iterations 1 to 5 on Known ISI Channel.

CHAPTER 4

Adaptive Turbo Equalization with a Time-invariant ISI Channel

An obvious test of an adaptive equalizer is to allow the ISI channel to drift randomly and assess the ability of the turbo equalizer to decode transmitted blocks of information while tracking the channel. However, insight into the operation of the turbo equalizer can be obtained by examining how the turbo equalizer finds a static but unknown channel given an initial channel estimate. This chapter demonstrates how the RLS algorithm, SISO equalization stage, and the SISO decoder stage interact with one another.

4.1 Simulation Parameters

In this chapter the turbo equalizer was operated with the settings given in Table 4.1. The settings in the top part of the table are the same as in the time-invariant case discussed in Chapter 3. In the lower part of the table are additional parameters that are necessary for the turbo equalizer to operate in an adaptive mode. Pseudo-random information sequences of 4096 bits were encoded, interleaved, and transmitted through the channel. Decoding was initialized with a channel estimate, $\hat{\vec{h}}(0)$, that assumed energy was equally distributed in each of the five real channel coefficients. Up to ten iterations through the decoder were made. At the beginning of the first iteration of each block of data, the inverse correlation matrix, $\mathbf{P}(0)$, is created using a soft-constrained initialization as recommended in Haykin [10] and the initial channel estimate from the table is used. A soft-constrained initialization involves forming $\mathbf{P}(0)$ by inverting an initial correlation matrix $\delta\mathbf{I}$ where δ is small relative to $0.01\sigma_{u_k}^2$ and $\mathbf{I}$ is an appropriately sized

Parameter	Setting
Information word size	1 bit
Code rate	$\frac{1}{2}$
Encoded block size	4096 bits
Transmitted block size	8200 bits
Code generating matrix	$[\ 1,\ \frac{1+D+D^2+D^4}{1+D^3+D^4}\ ]$
Interleaver	Pseudo-random, bitwise.
Channel impulse response	[0.227 0.46 0.688 0.46 0.227]
$\frac{E_b}{N_0}$ on channel	7.25 dB
Initial channel estimate	[0.45 0.45 0.45 0.45 0.45]
Initial estimate of $\frac{E_b}{N_0}$ on channel	8.0 dB
Decoding delay (D)	20
δ	0.004
λ	Varied

Table 4.1: Adaptive turbo equalizer settings.

identity matrix. In all the experiments in this thesis $\delta = 0.004$. At the beginning of subsequent iterations, $\mathbf{P}(k)$, $\hat{\vec{h}}(k)$, and the estimated channel noise are set equal to the values determined at the end of the previous iteration.

4.1.1 Selecting λ, the Exponential Forgetting Factor

Initially, choice of a suitable value for the exponential forgetting factor, λ, of the RLS algorithm is considered. The factor λ is a number slightly less than or equal to one. The term "forgetting factor" relates to the time dependent, exponentially diminishing effect that an observation has on the inverse correlation matrix in the RLS algorithm. After k time intervals have passed, an observation has only λ^k as much effect on $\mathbf{P}$ as when it was the most recent observation. Figure 4.1 shows the BER after 10 iterations with $\frac{E_b}{N_0} = 7.25\ dB$ for λ ranging from 0.99105 to 1. To give a more intuitive sense of what these λ values mean, the number of time periods that have elapsed when the observation has fallen to

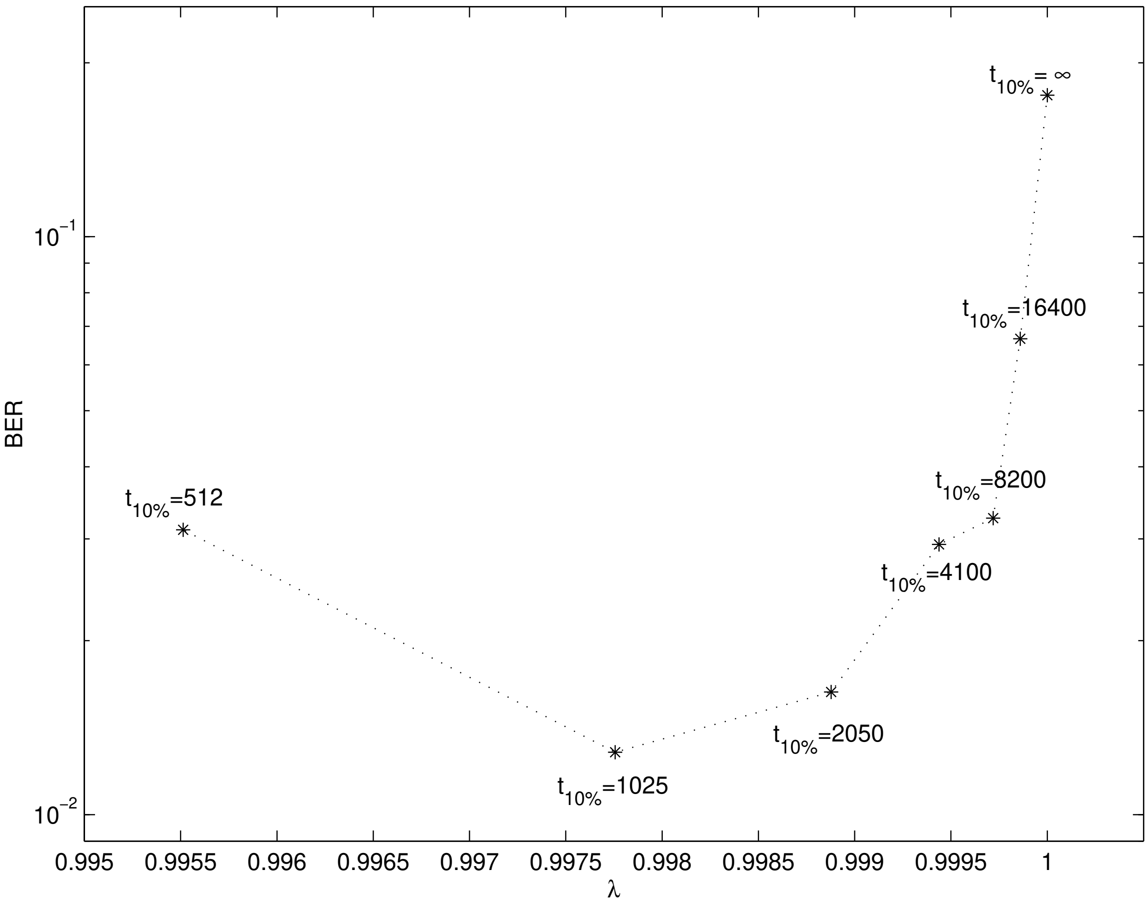

Figure 4.1: Bit error rate (BER) versus λ after 10 iterations with $\frac{E_b}{N_0} = 7.25dB$.

10% of its original effect, designated $t_{10\%}$, is also given on the graph. The λ values above correspond to $t_{10\%}$ from 256 to ∞.

Despite the fact that the channel is time-invariant in this experiment, the BER is reduced substantially for a large reduction in λ relative to the transmitted block size of 8200 bits. The BER is minimized when all but the most resent thousand bits of received data have less than 10% of their initial effect.

When using RLS on a static channel it is usually advantageous to let $\lambda = 1$. Since the input to the RLS algorithm is the channel output, averaging the channel noise over the maximum time interval will lead to the best filter. However in the case of a turbo equalizer, the estimated symbol values used as input to the RLS are determined by two sources: the equalized channel output (Equation 2.37) and the error correction code. During the first iteration only the equalizer stage contributes, but during subsequent iterations the error correction code provides extrinsic information through the feedback loop from the outer SISO. After the first iteration, a better estimate of the transmitted symbols can be made based on the combined information from equalized channel outputs and the error correction code. It is beneficial to forget the RLS data that was based on sub-optimal symbol estimates from early iterations and rely on more recent and more correct information. This phenomenon accounts for the improved performance of the adaptive turbo equalizer illustrated in Figure 4.1 when λ is initially reduced. But, as the figure shows, if λ becomes too small there is insufficient averaging of noise by the RLS algorithm and overall performance falls.

4.1.2 Convergence of the Turbo Equalizer

It should be noted that in Figure 4.1 the overall BER is poor; in all circumstances an average of more than one bit in a hundred is received incorrectly. However, if the results of the experiment are presented in a different fashion the results are more promising. Recall that an estimate of each information bit can be made during each iteration after the equalizer stage has been completed and again after the SISO has decoded the information message. With each iteration we expect the disagreement between the equalizer stage and the SISO stage to diminish. The number of differences in estimated information bits between stages is defined as Δ. The differences that comprise Δ may be due to errors in the equalizer stage, or due to SISO stage decoding errors. Regardless of which stage errs and thus contributes to Δ, if the message is successfully decoded, the magnitude of Δ must decrease with each iteration and reach a stable value. Also note that Δ is a value

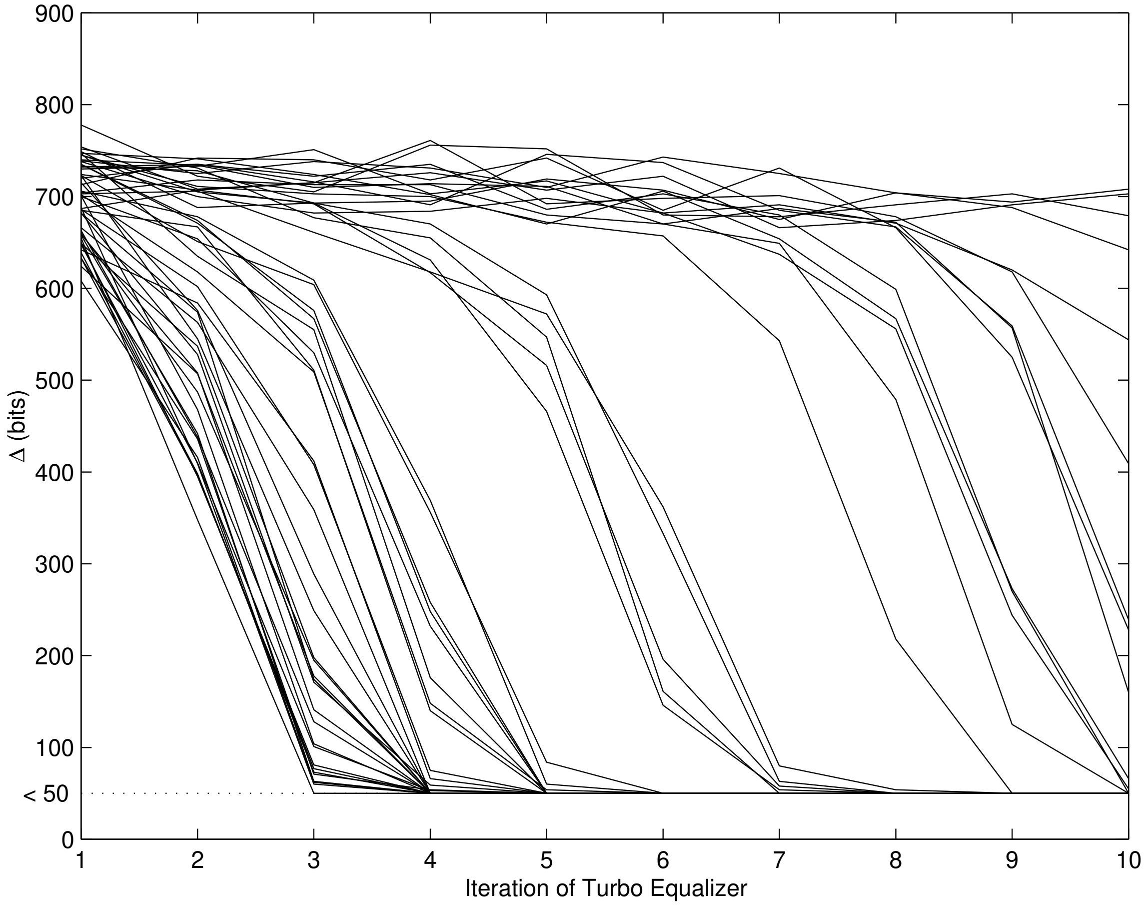

Figure 4.2: Number of bit discrepancies between stages of turbo equalizer versus iteration. $\lambda = 0.999$, $t_{10\%} = 2050$, $\frac{E_b}{N_0} = 7.25\ dB$, and $BER = 0.016$.

that can be computed at the receiver, it requires no knowledge of the information message sent. The value Δ relates to the cross entropy of the two stages described previously in Chapter 2.

Figure 4.2 presents a rather busy graph, but its overall pattern is what is important, not the specifics of the data presented. This figure plots Δ versus iteration for 49 blocks of transmitted data. It represents all the data that contributed to the point at $\lambda = 0.999$ ($t_{10\%} = 2050$) in Figure 4.1. The figure shows that after one iteration the two stages of the turbo equalizer disagree on about 700 of the 4096 transmitted information bits in each block. For the majority of blocks, Δ

starts to fall on the second iteration, then rapidly decreases to less than fifty in three to five iterations. Occasionally the rapid decent of Δ is delayed for eight or more iterations, but once the decent begins, Δ will reach a minimum of less than 50 in just a few iterations. Not apparent in Figures 4.1 and 4.2 is the fact that all of the errors that account for the dismal BER of 0.016 occurred in seven of the eight information blocks where Δ is above 200 at the end of the 10^{th} iteration.

A similar pattern is present for the other choices of λ. The relationship of BER to λ was determined entirely by whether of not Δ was "stuck" at a high level and initiation of rapid convergence of Δ was delayed long enough to prevent convergence by the tenth iteration.

This experiment was repeated with $\lambda = 0.999$ for 20 iterations. During the transmission of 100 blocks of information, two blocks accounted for all decoding errors. The resulting BER was 0.0020, but in one of the two blocks where errors occurred rapid convergence had started but was not complete at the 20^{th} iteration and the other block appeared to be on the verge of entering rapid convergence. These results suggest that when $\frac{E_b}{N_0} = 7.25\ dB$ and given sufficient iterations, transmission errors are rare indeed ($BER < 10^{-5}$.)

To evaluate this phenomenon further it is necessary to examine the character of the information shared between stages. Figure 4.3 shows histograms of information contributed by the equalizer stage ($L^{eq}(\vec{u}^1)$) and the SISO ($L^{e1}(\vec{u}^1)$) at each iteration for an early converging block of data (ECB). Initially, the information contributed by both stages is unimodal and of low magnitude. When convergence becomes noticeable in terms of decreasing Δ, the SISO histogram begins to broaden. Then during the next two iterations the stages quickly reinforce one another and two distinct populations are established, marking the completion of convergence. Note that the equalizer stage does not completely separate the two populations, but decoding is done correctly by the dominance of $L^{e1}(\vec{u}^1)$ over $L^{eq}(\vec{u}^1)$ when convergence is complete.

For a delayed converging block of data (DCB), the histograms from early iterations are unimodal with low magnitude likelihoods, virtually indistinguishable from the first iteration in Figure 4.3. This pattern of histogram persists until Δ heralds the onset of rapid convergence. Once convergence begins the histograms of all blocks, whether an ECB or a DCB, appear as in iterations two through four of Figure 4.3.

The mean magnitude of the information generated by the two stages, $\overline{|L^{eq}(\vec{u}^1)|}$ and $\overline{|L^{e1}(\vec{u}^1)|}$, for an ECB and a DCB are shown in Figure 4.4 plotted against iteration. During the period of rapid convergence the two sets of curves are indistinguishable. The initial magnitude of the information from each stage is low for both the ECB and the DCB, as expected from Figure 4.3 and the above description of a DCB, but there is a rapid increase in $\overline{|L^{e1}(\vec{u}^1)|}$ for the ECB. Careful inspection of Figure 4.4 will show that for a DCB, $\overline{|L^{eq}(\vec{u}^1)|}$ and $\overline{|L^{e1}(\vec{u}^1)|}$, are increasing slightly during the early iterations. This is a consistent finding for all DCBs. Initial growth is imperceptible, then the rate of increase in $\overline{|L^{e1}(\vec{u}^1)|}$ exceeds $\overline{|L^{eq}(\vec{u}^1)|}$. This accelerated growth in $\overline{|L^{e1}(\vec{u}^1)|}$ is a precursor to rapid convergence.

Understanding what the turbo equalizer is doing during the early iterations when there is minimal growth of $\overline{|L^{eq}(\vec{u}^1)|}$ and $\overline{|L^{e1}(\vec{u}^1)|}$ is difficult if only the received data is examined. Figure 4.5 gives an alternate view of the early iterations that can be obtained given knowledge of the data transmitted. The figure graphs the of number of bit errors during each iteration made by the equalizer or SISO stage alone and by the turbo equalizer as a whole. This figure of a typical ECB and DCB gives insight into why the turbo equalizer is sometimes slow to initiate convergence. The two parts of the figure illustrate the same two blocks of data from Figure 4.4, the ECB starting rapid convergence immediately, and the

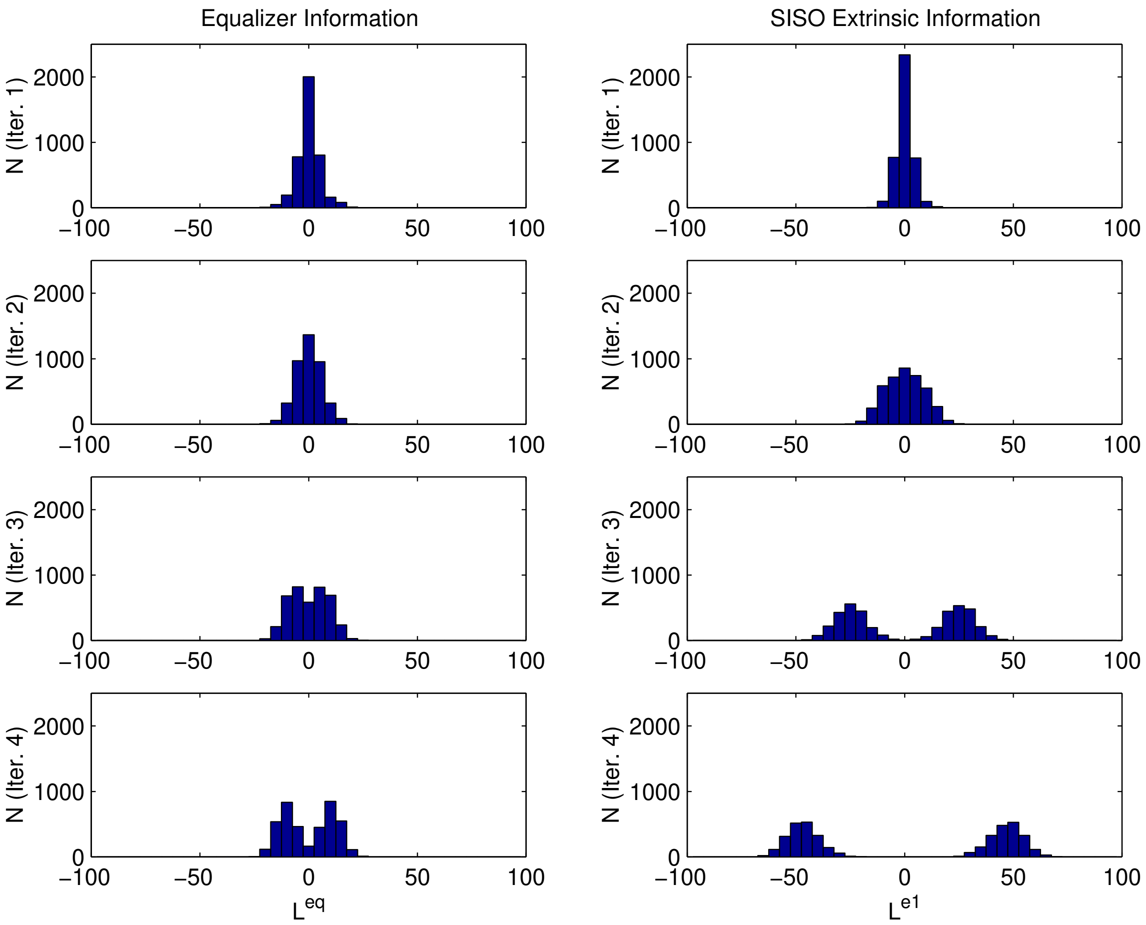

Figure 4.3: Histograms of $L^{eq}(\vec{u}^1)$ and $L^{e1}(\vec{u}^1)$ for iterations one to four. This block of data converged in four iterations.

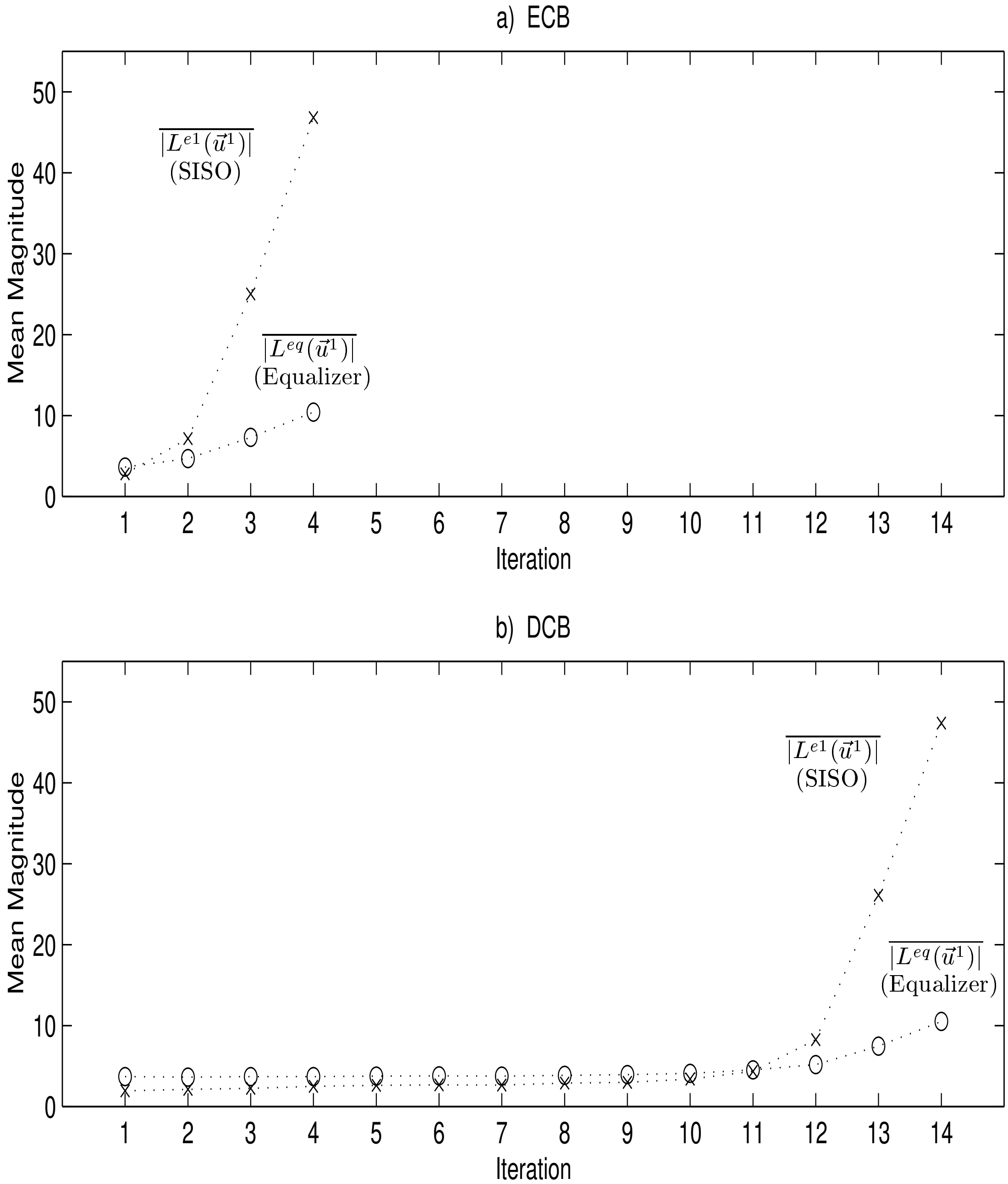

Figure 4.4: $\overline{|L^{eq}(\vec{u}^1)|}$ (o) and $\overline{|L^{e1}(\vec{u}^1)|}$ ($\times$) versus iteration for an early (a) and a delayed (b) converging block of data.

DCB initiating rapid convergence after 10 iterations. In both cases, after the first iteration Δ was in the typical range of 600 to 800. During convergence, the sets of curves are remarkably similar. However, the DCB has a much higher initial number of errors and the number of bit errors is decreasing during much of the period when Δ is fluctuating between 600 and 800 and the histogram is static. In general DCBs are characterized by an initial high number of bit errors. Also characteristic of the early iterations of a DCB is that the overall decoding success of the turbo equalizer is obtained primarily through the equalizer stage's output. Note that the number of correctly decoded bits in early iterations is higher based on information from the equalizer alone than on the combined equalizer and SISO information; the SISO's contribution to decoding initially increases the number of bits errors made by the turbo equalizer. During the early iterations of a DCB the equalizer stage decreases the overall number of bit errors despite the often misleading information it receives from the SISO.

Whether or not a block of data will show early or delayed convergence is dependent on the degree of success of the first equalizer iteration. If the estimate of the impulse response of the ISI channel converges quickly and is accurate, appreciable information is passed to the SISO. Then the SISO can in turn correct significant numbers of errors and feedback information that reinforces the equalizer during the second iteration. This scenario leads to early convergence. Alternatively, if the received vector $\vec{y}$, is such that the equalizer has greater difficulty in estimating the channel and provides the SISO with a lot of erroneous information, the SISO will be overwhelmed. That is, the capacity of the error correction code of the SISO to repair the data is exceeded and the SISO provides no significant information to the equalizer at the end of the first iteration. The low magnitude likelihoods returned to the equalizer represent little more than added noise for the equalizer. This would be a hopeless case were in not for the fact that at the

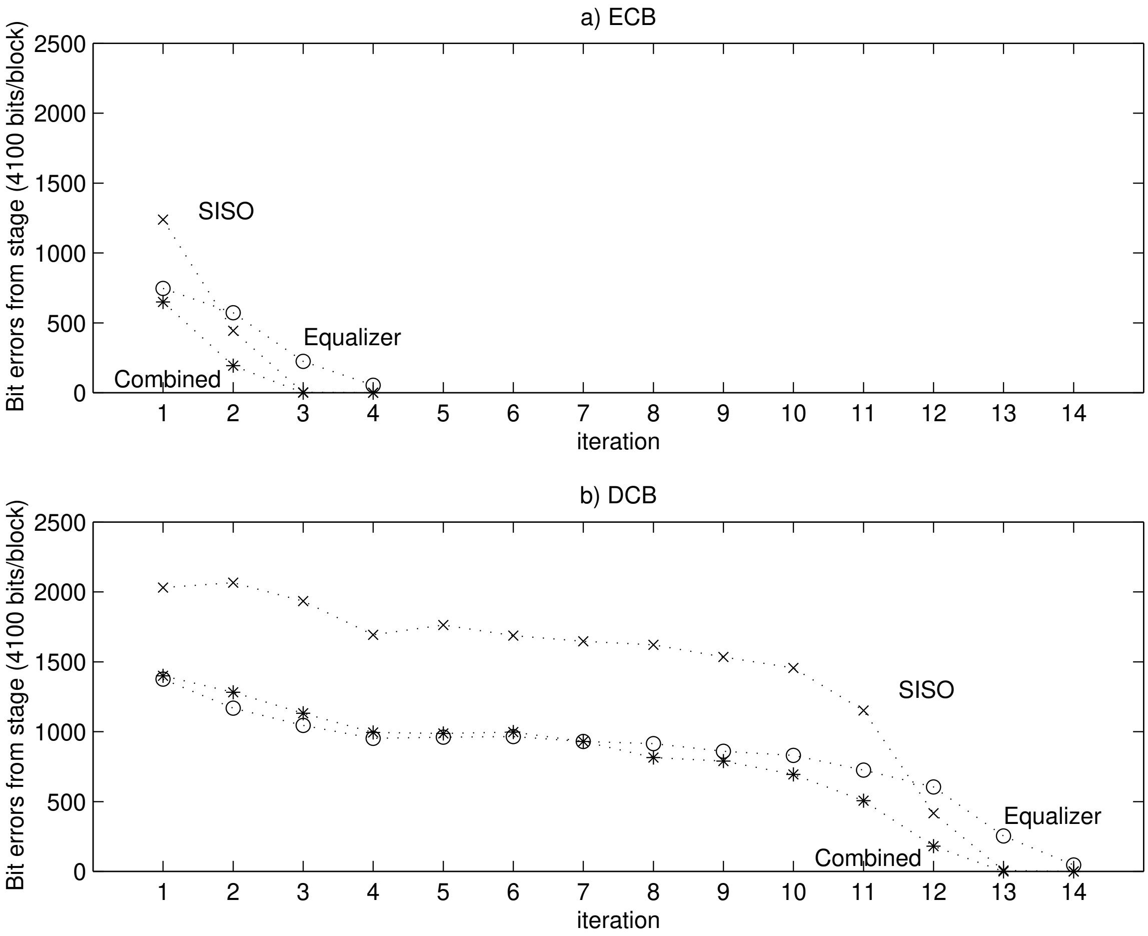

Figure 4.5: Number of information bit errors at the end of each iteration. The two blocks are examples of an early (a) and a delayed (b) converging block that were completely decoded in 4 and 14 iterations respectively. The three curves show the bit errors based on the output of the equalizer stage alone (o), the SISO stage alone ($\times$), or the output of the two stages together ($*$).

beginning of the second iteration an improved, albeit still poor, channel estimate is available for the equalizer. This new channel estimate is enough to enable the equalizer to improve the information it generates during the second iteration, despite having to deal with increased noise in the form of extrinsic information from the SISO. The noisy information from the SISO has a limited effect on the equalizer's estimate of the channel during the second iteration since this noise is averaged. This low magnitude erroneous information from the SISO is primarily manifest as an increased estimate of the noise variance of the channel and slower convergence of the channel coefficient estimates. This situation is repeated for a number of iterations with moderate improvement of data and channel estimates by the equalizer stage. During each iteration the SISO shows only a minimal improvement of its data estimates and a minimal increase in the magnitude of its extrinsic information. Eventually this "SISO noise" offsets any remaining ability of the equalizer stage to improve its channel estimates. This brings the entire turbo equalizer to a plateau with respect to the number of bit errors.

While on the plateau, each iteration results in small changes in the individual likelihoods given to the SISO, but small changes in the input can lead to significantly different paths on the trellis and improvement in decoding over short intervals can occur in bursts. In a sense these bursts can be interpreted as a random search for the correct information sequence. Once the SISO finds a short sequence that reinforces the equalizer, this short segment's likelihood values increase and it becomes stable. Over time these affected short sequences reduce the overall error rate below the error rate from the equalizer alone, and convergence is assured, gradually at first but then rapidly as the magnitude of the information from the SISO exceeds that from the equalizer. The number of iterations between the beginning of the plateau and the crossing of the equalizer and SISO bit error curves can be variable; the number dependents on how many iterations the above "random search" requires.

It appears that the progression of the decoding process by the turbo equalizer is well defined. The variation in the number of iterations necessary for convergence is just a function of where an individual block of data finds itself after the first iteration. Conceivably, the corruption of the transmitted block of data could result in the first iteration being anywhere along the curve of Figure 4.5b. The initial number of errors made by the equalizer stage and the length of the first part of the plateau determine the total number of iterations needed to decode a block of data. Overall $\frac{E_b}{N_0}$ will determine which end of this bit error curve a block of data is most likely to enter the decoding process. A high $\frac{E_b}{N_0}$ will cause a low error count from the equalizer stage that enables significant error correction by the SISO and rapid convergence will commence immediately. At a low $\frac{E_b}{N_0}$ the equalizer stage will initially make many errors and many iterations will be required to complete decoding. If $\frac{E_b}{N_0}$ is too low the equalizer stage will be unable to improve decoding during the second iteration and decoding will fail. If the equalizer can reduce the number of errors to about 25%, even in the face of noisy output from the SISO, then complete and highly accurate decoding appears almost certain for this particular convolutional code, given sufficient iterations.

4.2 Improving Turbo Equalizer Performance by Selectively Removing Feedback

Based on the observations and explanations of the previous section a potential improvement in the turbo equalizer can be proposed. Since the information that is being generated by the SISO is often detrimental, the equalizer stage should function more effectively without SISO feedback in some instances. If the appropriate instances can be identified, SISO feedback can be removed. Although the data presented in Figure 4.5 clearly identifies when the SISO is and is not helping, it requires knowledge of the correct information sequence transmitted and so can not be used in practice.

The next best measure of convergence, Δ, is available at the receiver and can be used. The value of Δ does not predict convergence within the next three iterations with high specificity or sensitivity. The rate of change of Δ is a better predictor of impending convergence. A test of impending convergence is given by

$$\frac{\Delta_i - \Delta_{i-2}}{\Delta_{i-2}} > \alpha. \tag{4.1}$$

For example, when $\alpha = 0.1$, if Δ decreased by less than 10% over the course of two iterations then the turbo equalizer was in a delayed converging pattern.

The following modification of the turbo equalizer was made. If the above test of Δ failed at the third or sixth iterations, for $\alpha = 0.1$, at the beginning of the fourth or seventh iteration respectively, the extrinsic information from the SISO was reset to $\vec{0}$. This modification was evaluated with data blocks decoded for ten iterations and $\frac{E_b}{N_0}$ reduced from the 7.25 dB used above to 6.5 dB. Despite the significantly lower SNR on the channel, the overall BER is reduced by a factor of over four to 0.0038. Figure 4.6 graphs Δ versus iteration for 100 block transmissions. The asterisks on the figure indicate those blocks that were reset after iteration three and/or six. Comparing this figure to Figure 4.1 readily demonstrates the efficacy of resetting the extrinsic information to $\vec{0}$. At a reduced SNR, twice as many blocks converged within 10 iterations.

Although the immediate effect of reseting the SISO extrinsic information is not readily apparent in Figure 4.6, Figure 4.7 illustrates that a DCB can be moved

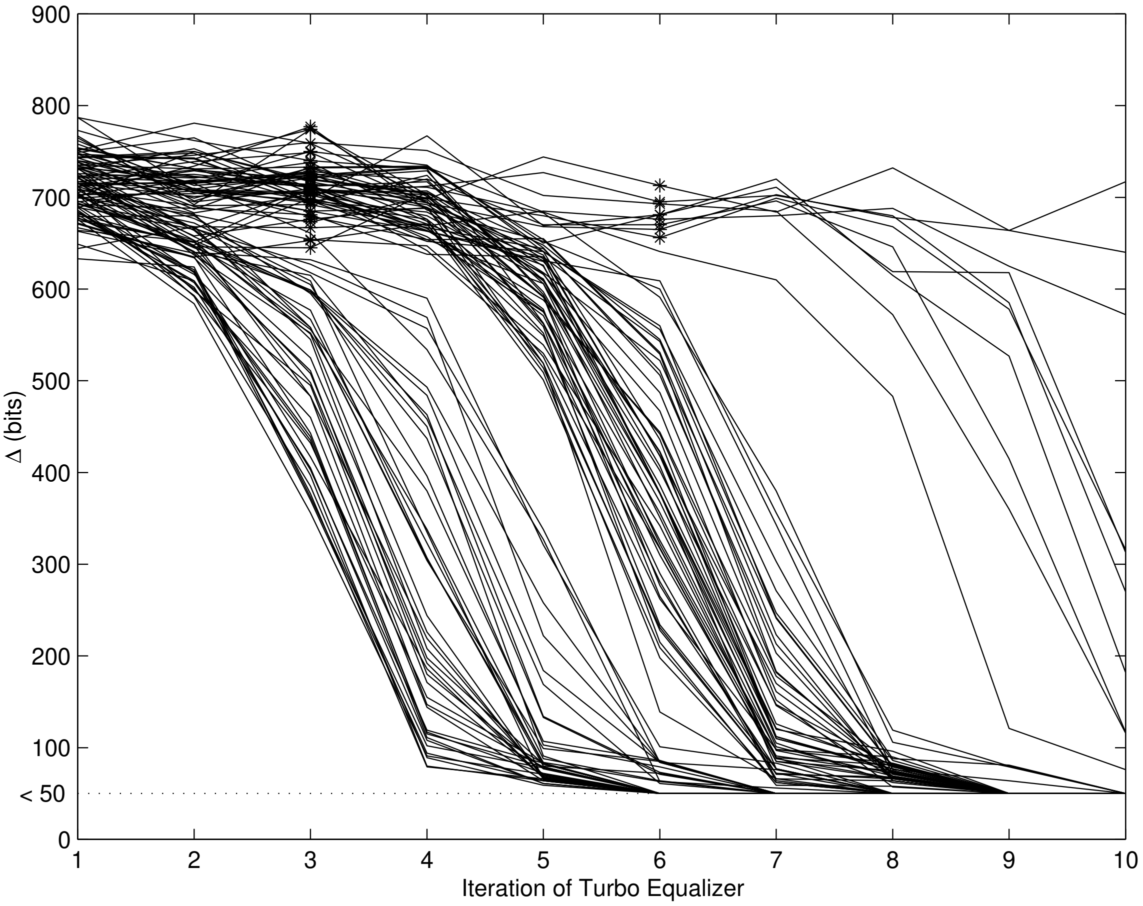

Figure 4.6: Number of bit discrepancies between stages of turbo equalizer versus iteration when $L(\vec{u}^2)$ is reset to $\vec{0}$ at iteration 3 or 6 based on Equation 4.1, $\lambda = 0.999$, $t_{10\%} = 2050$, $\frac{E_b}{N_0} = 6.5\ dB$, and $BER = 0.0038$. The three curves show the bit errors based on the output of the equalizer stage alone (o), the SISO stage alone ($\times$), or the output of the two stages together ($*$).

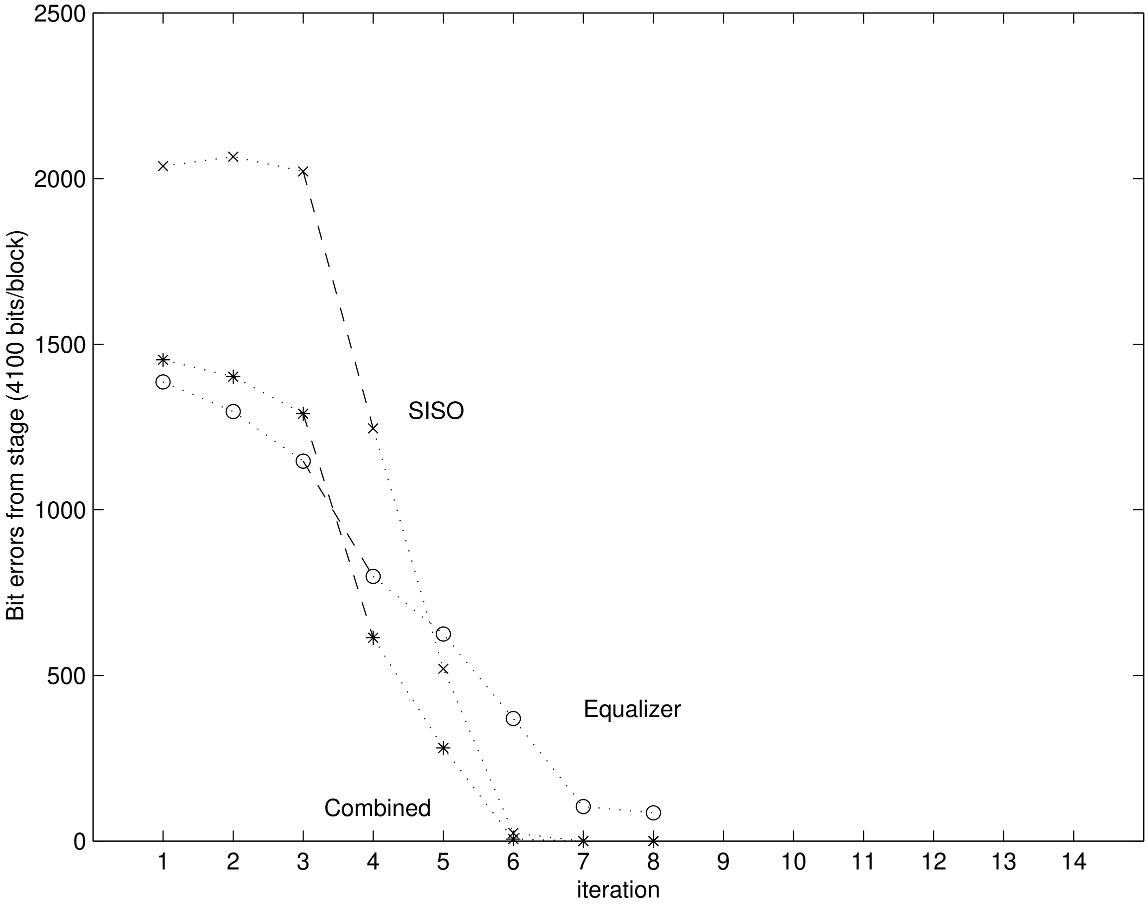

Figure 4.7: Number of information bit errors versus iteration for a block with $L(\vec{u}^2)$ set to $\vec{0}$ at the end of the third iteration. Note that the expected plateau over the dashed line segment has been eliminated. $\lambda = 0.999$, $\frac{E_b}{N_0} = 6.5\ dB$.

from the initial slowly converging section, across the entire plateau section, and begin rapid convergence when incorrect information in the form of $L^{e1}(\vec{c}^1) = L(\vec{u}^2)$ is removed. Initially the high number of bit errors of the turbo equalizer are decreasing, but the plateau has not been reached by the third iteration. Resetting all symbol likelihoods of $L^{e1}(\vec{c}^1)$ to 0 moves the block directly to the beginning of the rapidly converging section and the block was correctly decoded by iteration seven.

4.3 Summary

This chapter has demonstrated the role of the RLS algorithm within the SISO equalizer and the interaction of the SISO equalizer with the SISO decoder when communicating over a time-invariant unknown ISI channel. The initial burden of decoding is placed on the equalizer stage. The equalizer must successfully decode almost three quarters of the bits before the SISO stage can have any success in reducing the number of decoding errors.

The forgetting factor, λ, must be chosen small enough to assure that information derived by early iterations of the equalizer stage is forgotten. During later iterations through the equalizer, more accurate symbol estimates are possible because of better information returned in the feedback loop from the SISO stage. This leads to improved channel estimates by the RLS algorithm embedded in the equalizer. If λ is too large the inverse correlation matrix of the RLS algorithm is influenced by inaccurate symbol estimates generated using old information from previous SISO iterations.

The convolutional code used prior to transmitting the data has a limited ability to correct errors. If a SISO is presented with too much noise, its ability to improve on information provided by the equalizer is limited and the feedback it provides is sometimes detrimental to the equalizer stage. In this circumstance the equalizer must improve its estimate of the channel and the transmitted data in the face of added noise from the SISO stage. It was demonstrated that under these circumstances temporarily eliminating the feedback from the SISO stage is advantageous. Furthermore the cross-entropy or Δ can be used to identify those circumstances when this approach is likely to be of benefit.

Performance of the turbo equalizer for transmissions over an unknown time-invariant channel is difficult to directly compare to the performance when the channel is known as was presented in Chapter 3. Performance is obviously dependent on the quality of the initial channel estimate. However once a "reasonably good" channel estimate is obtained, it is not surprising that the decoding of a block of data appears to happen as quickly and accurately as in the case of the known channel. The additional variable of an unknown channel adds another step for the turbo equalizer: determining a "reasonably good" channel estimate. These two steps are largely performed sequentially and the first step is the burden of the SISO equalizer stage alone. A step sometimes performed in spite of, rather than, in cooperation with the SISO decoder. Evidence presented suggests that the initial channel estimation step can be accomplished given sufficient iterations and no initial SISO feedback at 6.5 dB and likely also at an $\frac{E_b}{N_0}$ close to the 6.15 dB that produces a BER of 10^{-4} when the channel is known.

The cost of not knowing the channel is the requirement of additional, possibly many, iterations. The cost in terms of SNR is small.

CHAPTER 5

Adaptive Turbo Equalization with a Time-varying ISI Channel

In this chapter the adaptive turbo equalizer is applied to an unknown time-varying channel. First the channel model is described. Then the ability of the adaptive turbo equalizer to track an autoregressive channel model is demonstrated.

5.1 Channel Model

The particular static channel model used in Chapter 4 was a rigorous test of the turbo equalizer, but as a consequence almost any change in the channel impulse response improves the BER. The channel model used in the static case of the previous chapter was about the most difficult channel that could be encountered with an impulse response of length five. The model has zeros at $0.96\angle \pm \frac{2\pi}{3}$ and $1.04\angle\pm\frac{2\pi}{3}$. This channel strongly attenuates signals containing repeated sequences of $1, 0, 0$ or $1, 1, 0$. The only more difficult five coefficient channels would place the zeros at $\angle \pm \pi$ or move the existing zeros even closer to the unit circle. Zeros on the unit circle at angles more acute than $\frac{2\pi}{3}$ are less troublesome because the repetitious sequences that these zeros severely attenuate are longer than three symbols per period and are less likely to occur than repeats of doublet or triplets of symbols. Thus, dynamic performance is difficult to directly compare to the static results because as soon as the channel drifts away from the above zeros, performance improves based on the change in channel impulse response. The BER of the turbo equalizer is as much a function of channel as it is of the SNR.

To evaluate performance of the turbo equalizer on a time-varying ISI channel an autoregressive process was initialized to the static channel given in Equation

3.1. The five channel coefficients were assumed to be complex and each of the real and imaginary components was computed as a first order autoregressive process

$$h_i(k) + \alpha h_i(k-1) = v_i(k), \tag{5.1}$$

where $i \in \{1 \ldots 5\}$ and $v_i(k)$ is a complex, zero mean white noise-process with variance (sum of real and imaginary variance) of $\sigma^2_{v_i} = \frac{1-\alpha^2}{5}$ for all i. The value α was chosen to be equal to -0.9999. The channel coefficients were normalized to unit energy so that $\frac{E_b}{N_0}$ could be fixed. The autoregressive process was continued long enough to simulate transmission of 101 blocks of encoded data. The process was continuous from block to block. The resulting autoregressive process was saved and used for testing the turbo equalizer at all SNRs. To give a sense of how fast the channel is drifting Figure 5.1 shows the real part of the channel coefficients generated for the first block of data. The imaginary parts of the channel have the same variance, but they were all initially set to zero.

This autoregressive channel model changes fast enough that the set of ISI channel coefficients represents a random selection of all five coefficient samples except there is a small bias in that the first block of data is transmitted over a channel close to the difficult channel used for the static evaluation.

5.2 Simulated Transmission over a Time-varying ISI Channel

Each of 101 random blocks of information was encoded and sequentially sent through the ISI channel with a constant $\frac{E_b}{N_0}$. The received data was decoded by two methods. First, as a control, the turbo equalizer was given the channel coefficients and $\frac{E_b}{N_0}$. Second, in the testing scenario, the turbo equalizer was given the channel only for the first symbol transmission of the first block of data. All subsequent channel information was generated by the equalizer stage of the turbo

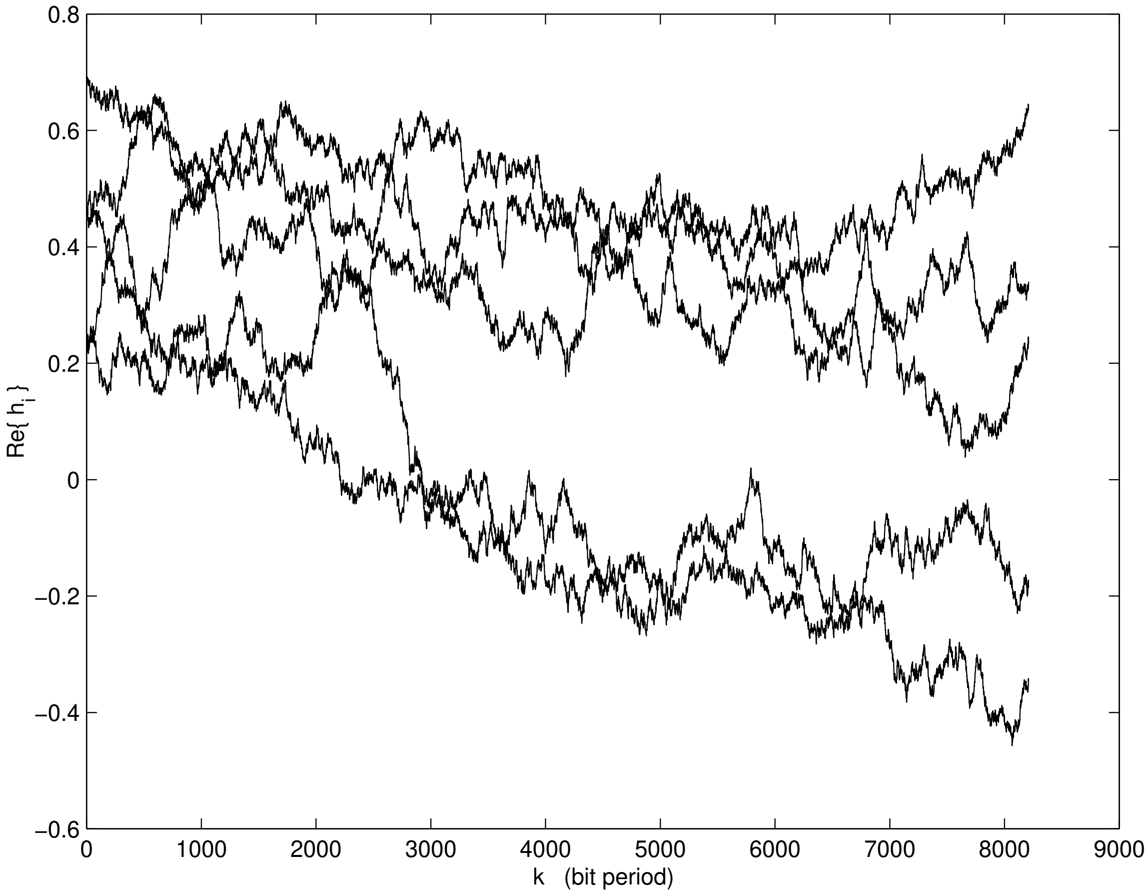

Figure 5.1: The real part of the channel coefficients for the first transmitted block generated by the autoregressive process.

equalizer. Knowledge of the channel was the only variable that differed between the two turbo equalizing algorithms. The experiment was run multiple times with $\frac{E_b}{N_0}$ varying from 4.25 dB to 6.5 dB.

In the testing scenario the equalizer's inverse correlation matrix $\mathbf{P}(0)$ was initialized using the same criteria as the in Chapter 4 for the first block of data. Since the channel drifted over the duration of a block of data the channel estimate at the end of the block was not a good estimate for the beginning of the block. Thus, unlike the in the static channel case, when starting another iteration on the current block, the estimated channel coefficients, noise, and $\mathbf{P}$ were reset to their values at the start of that block. When moving on to a new block of data the estimated channel coefficients, noise, and $\mathbf{P}$ were left unchanged.

Figures 5.2 and 5.3 show the BER for transmission over the known and unknown ISI time-varying channel. Note that the only the first and third or the first, third, and sixth iteration of decoding is shown for the known and unknown channel respectively.

When the channel is known, the turbo equalizer fully converged by the third iteration. In fact, there was usually no change in the BER between the second and third iteration. The third iteration has the same or higher BER than the first iteration at $\frac{E_b}{N_0} = 5.5\ dB$ and $\frac{E_b}{N_0} = 6.5\ dB$ respectively. The reason for this anomaly is that for some blocks the first iteration makes no errors, but with convergence a few errors appear. Convergence is characterized by minimal and stable disagreement between stages, not minimum decoding errors.

When the turbo equalizer is required to track the channel in addition to equalizing and decoding the transmitted data there is a cost in terms of decoding iterations near the SNR threshold. Convergence was not complete until the sixth iteration, although at $\frac{E_b}{N_0} = 5.75\ dB$ and above, convergence required only three iterations. Knowledge of the channel makes a large difference on the first iteration.

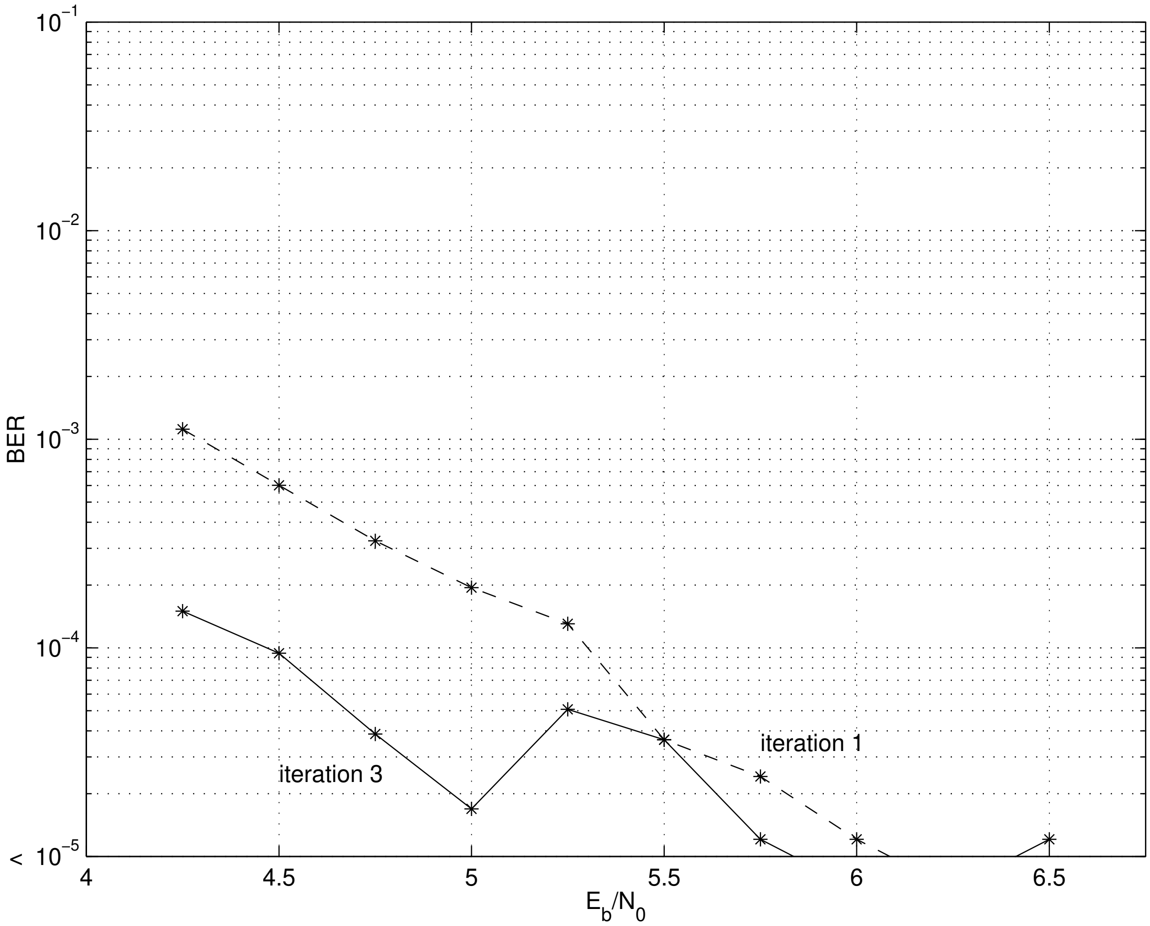

Figure 5.2: Bit error rate (BER) versus $\frac{E_b}{N_0}$ for transmission of 101 consecutive blocks of data over a time-varying ISI channel with known impulse response and channel noise.

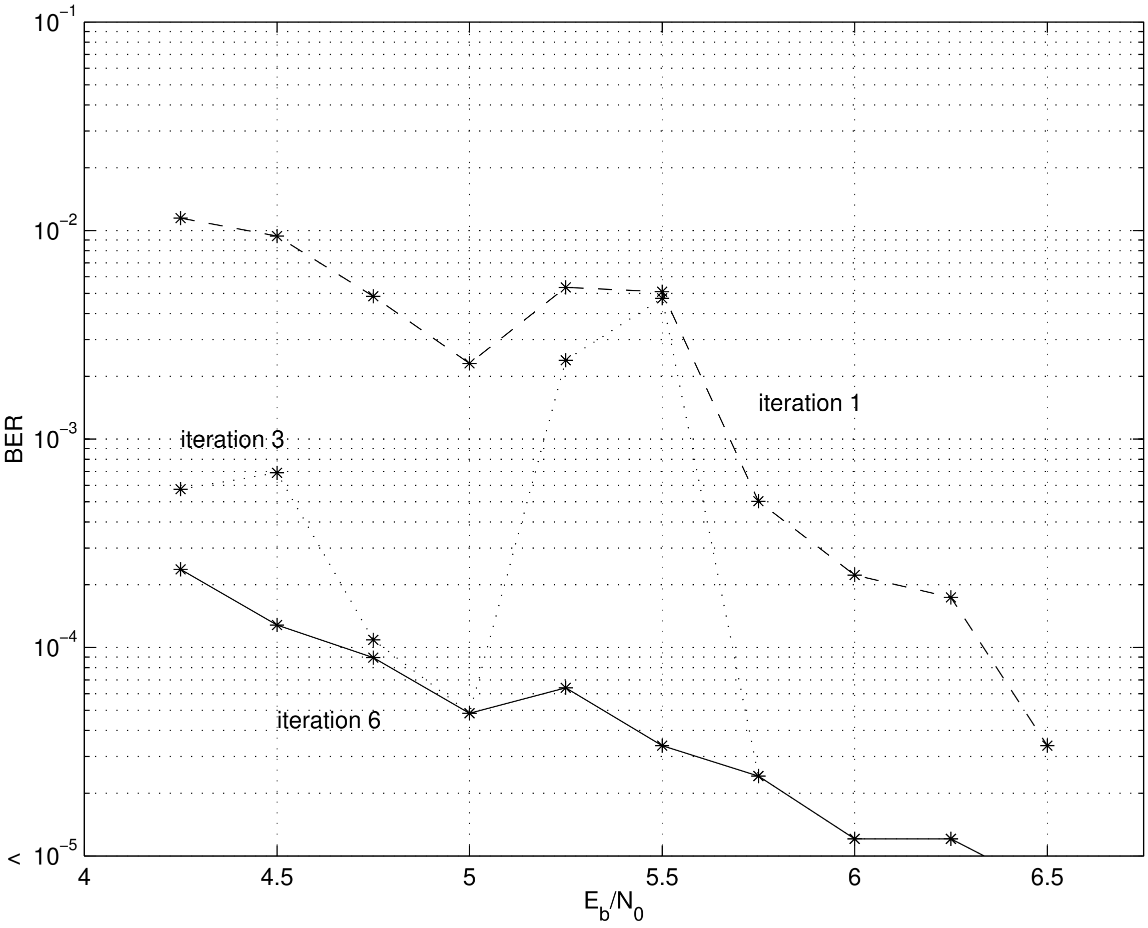

Figure 5.3: Bit error rate (BER) versus $\frac{E_b}{N_0}$ for transmission of 101 consecutive blocks of data over an unknown time-varying ISI channel.

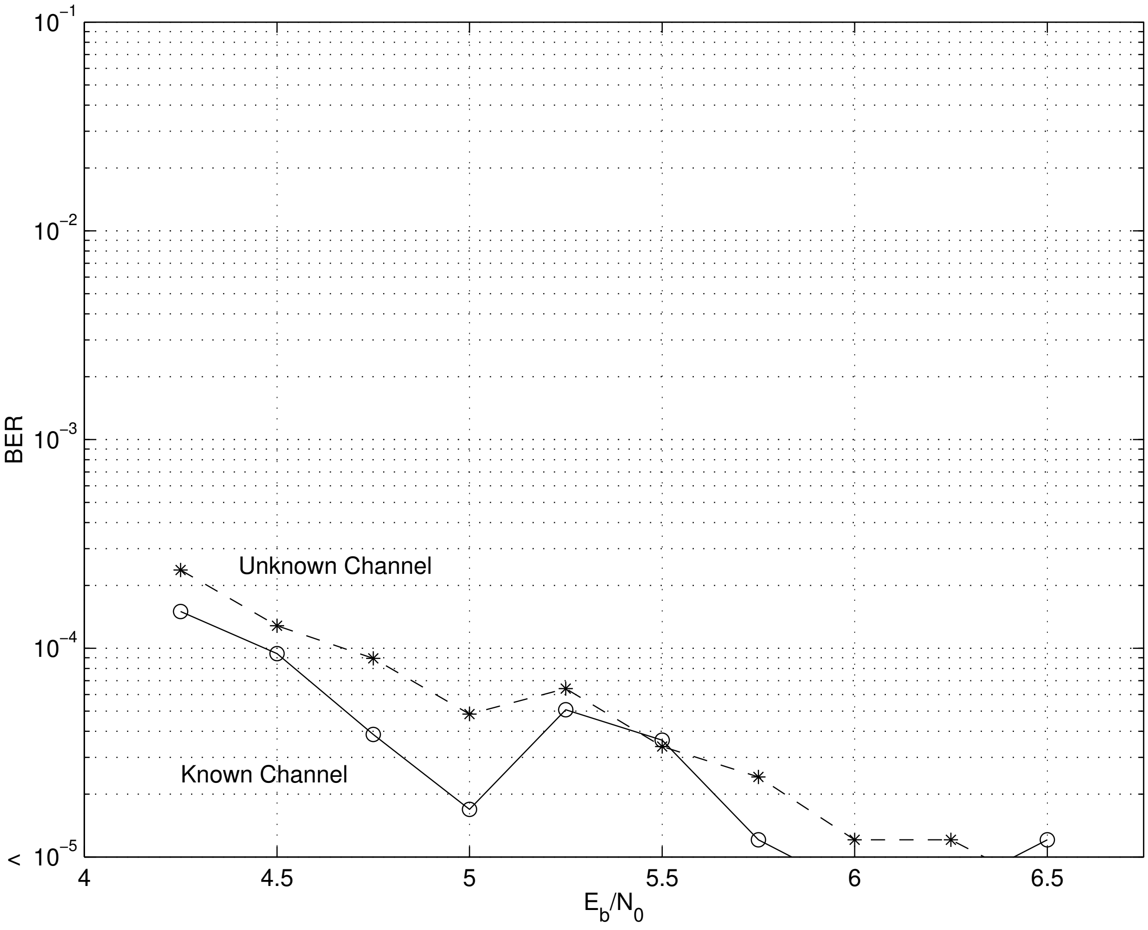

Figure 5.4: Bit error rate (BER) versus $\frac{E_b}{N_0}$ after convergence of the turbo equalizer using know (third iteration) and unknown (sixth iteration) channel data.

When the channel is known the equalizer produces optimal information during the first iteration. Alternatively, if the channel is unknown the efforts of the equalizer stage are divided between estimating the channel and the received data.

Careful examination of Figure 5.3 shows that the slope of the curve representing iteration 1 is steeper than the slope for iteration 6. If this is true, testing at a higher $\frac{E_b}{N_0}$ should lead to completely correct decoding at the end of the first iteration. To test this idea the experiments were repeated with $\frac{E_b}{N_0}$ equal to 7.0, 7.5, and 8.0 dB. In all but the first block of data at 7.5 dB, the decoding was completely correct at the end of the first iteration for the known and unknown channel. Note that the first block is transmitted over the channel that most closely resembled the "difficult channel" used in the static case.

Figure 5.4 shows the third and sixth iteration of the known and unknown channel data superimposed on one graph. The irregularity of this graph illustrates the imprecision of the estimates of the BER, especially at high SNR where the BER is based on a few errors in only one or two blocks of data. Allowing for this imprecision of BER estimation and including data from the higher SNR levels from 7.0 to 8.0 it is difficult to conclude that there is any clear SNR penalty for not knowing the channel. If there is a penalty, it appears to be no more than 0.25 dB.

5.3 Summary

This chapter demonstrates the performance of the turbo equalizer when communicating over a time-varying unknown ISI channel. The results presented in this chapter directly compare transmission of data when there is simultaneous channel estimation and decoding to transmission with perfect channel knowledge. Decoding near the SNR threshold when there is no channel knowledge requires more iterations than when the channel is known. However, this iteration penalty disappears at higher SNRs. The cost in terms of SNR is small.

If Figures 5.2 and 5.3 are compared to Figure 3.1 of Chapter 3 the character of the graphs show some differences. First, the sharp SNR threshold at about 6.25 dB on the time-invariant known channel is not present with a time-varying channel: at least not with the BER above 10^{-5}. Second, the orderly downward movement of the curves with each iteration is lacking when transmitting over the time-varying channel. Third, the BER after the first iteration decreases more rapidly with SNR than the corresponding curve in the static channel case. Each of these features are a result of the dependence of the BER on the channel impulse response in addition to the illustrated dependence on the SNR. The data in the current figures was derived from a random selection of channels. Although each point on the graph was determined using the same sample of this random process, the varying channel adds a dimension to the experiment that was not present in Figure 3.1.

Figure 5.5 shows an idealized curve illustrating the BER versus $\frac{E_b}{N_0}$ for the first and final iteration of a turbo code. The solid portion of the curves is typical of the static turbo equalizer illustrated in Chapter 3 and also of the convergence pattern of turbo codes in general. Turbo codes are typically only tested over a limited range since performance will certainly improve at higher $\frac{E_b}{N_0}$ and the interest is in establishing the SNR threshold. The dotted part of the curves is an extrapolation. The first iteration often appears linearly related to $\frac{E_b}{N_0}$ because of the limited range of $\frac{E_b}{N_0}$ usually displayed. However, Bauch [2] illustrated the curve for the first iteration as in Figure 5.5. This is the usual relationship of BER to $\frac{E_b}{N_0}$ on a Gaussian channel.

On a time-variant channel, at each time instant, a similar pair of curves exists for the channel present at that time. This set is drawn from a large family of channels, each displaced to the left or right on the $\frac{E_b}{N_0}$ axis relative to one another depending on the capacity of the particular channel. If the SNR is large enough

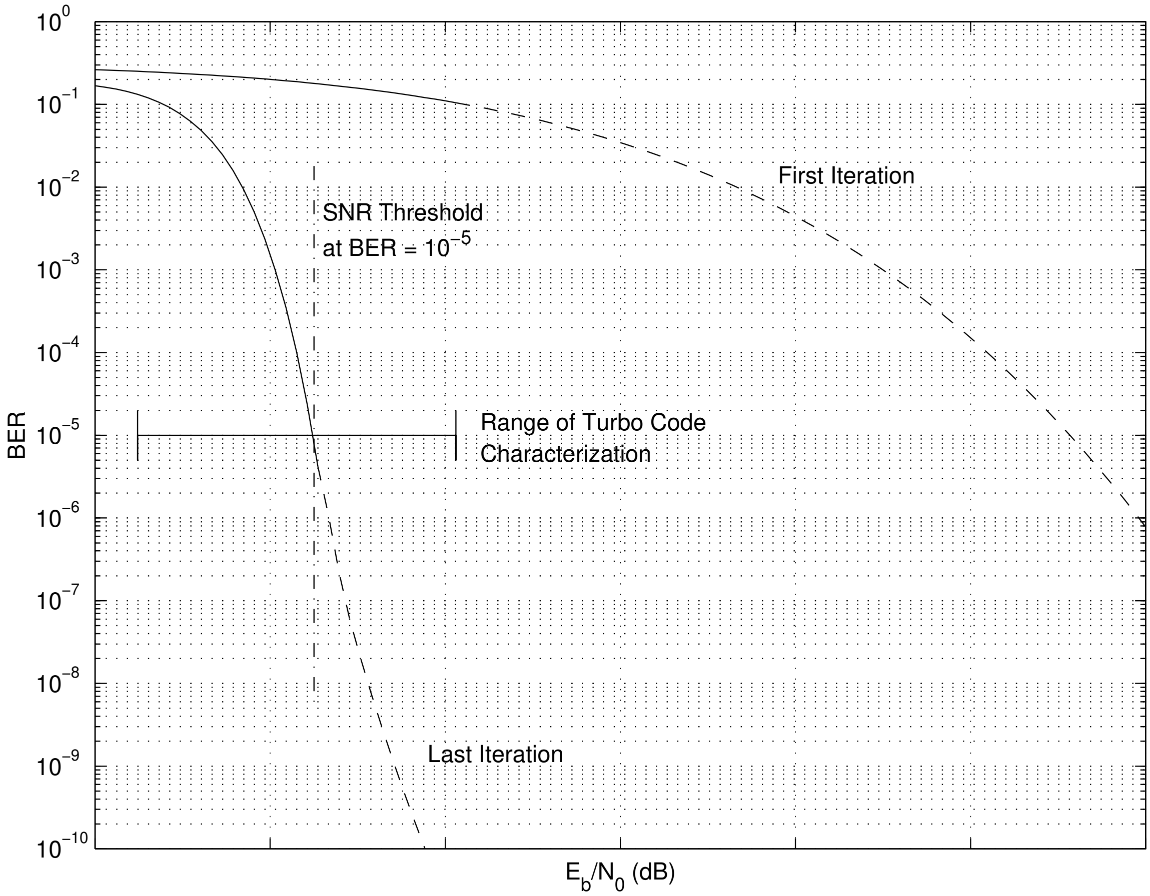

Figure 5.5: Typical bit error rate (BER) versus $\frac{E_b}{N_0}$ trends for the first and last iteration of a turbo equalizer operating on a time-invariant ISI channel. The dashed lines are usually not characterized when the performance of a turbo equalizers is evaluated. The $\frac{E_b}{N_0}$ values are relative. Benign channel converge for low $\frac{E_b}{N_0}$ values, difficult channels require high $\frac{E_b}{N_0}$ values. The relationship between iterations is characteristic of the turbo process.

to be well to the right of most channels, the adaptive turbo equalizer will usually correctly decode each block of data in just one iteration. But if the "first iteration" curves of any of the most difficult, low capacity channels still lies to the right of the SNR then, should that channel be encountered, there is a chance that more than one iteration will be required for decoding. This occurred in the experiment carried out at 7.5 dB. The first block was near the low capacity static channel of Chapter 3 and additional iterations were required to correctly decode it. Once the autoregressive process generating the channel drifted toward more benign channels the point described by $\frac{E_b}{N_0} = 7\ dB$, $BER = 10^{-5}$ appears to be to the right of all the first iteration curves of the channels. The adaptive turbo equalizer then correctly decodes each block in just one iteration.

The conclusions of Chapter 4 are not directly applicable to tracking the time-varying ISI channel. Choice of λ should be based entirely on the expected rate of change of the channel since at the beginning of each iteration of a given block the channel estimate and inverse correlation matrix of the RLS algorithm must be set to the value it had at the end of the last block received. This is a consequence of the fact that the channel is time-varying, unlike the case in Chapter 4.

The concept of resetting the extrinsic information from the SISO stage can not be implemented as simply as in Chapter 4 for a related reason. Since the channel estimate is always the same at the beginning of each iteration for a given block, the only thing that changes from one iteration to the next is the extrinsic information from the SISO. If it were set to $\vec{0}$ the decoding would simply restart at the first iteration. Still, the idea of removing unhelpful information has merit.

Resetting an entire block of extrinsic information is not an option. It is possible that segments of the extrinsic information vector that are not contributing to the decoding in early iterations could be identified. These might be identified by examining Δ over segments of $L^{e1}(\vec{u}^1)$ from each iteration. Once unhelpful segments of $L^{e1}(\vec{u}^1)$ are eliminated, the convergence of equalization and decoding might accelerate and be more accurate. Examination of $L^{e1}(\vec{u}^1)$ during early iterations suggest that likelihood magnitudes sometimes vary in segments, but these observations are preliminary and the concept of segmental reseting of the extrinsic information from the SISO stage will have to wait for future investigations.

CHAPTER 6

Discussion and Conclusions

Turbo equalization is predicated on the concept that the error correction code and equalizer stages can iteratively assist one another in performing their respective duties. While this concept is true to some extent, the results of this thesis suggest that it is not always the case. The useful SNR range over which the "turbo" effect operates appears to be small (about 1 dB) on a time-varying ISI channel. Given the high complexity of turbo equalization, iterations where the feedback is not contributing to decoding are a significant waste of computational effort. At the low end of the range near the SNR threshold the feedback is sometimes detrimental to channel equalization while at the high end of the SNR range feedback is needless since decoding is usually complete at the end of the first iteration. Additional iterations, a major part of the computational effort, are necessary only to determine that the turbo equalizer has converged. These facts indicate that the adaptive turbo equalization rarely benefits from the turbo effect of iterative decoding when adapting to a time-varying channel.

Study of the performance of the adaptive turbo equalizer on static and dynamic ISI channels shows similar behavior. The major penalty of having to adapt to an unknown channel in both cases is the requirement of more iterations to produce convergence of the decoding algorithm when near the SNR threshold. This iteration penalty is less severe as SNR increases and disappears on a time-varying channel if the SNR increases more than 1 dB above the SNR threshold for a given BER. Given adequate iterations, the BER obtained at any SNR appears to be independent of whether or not the channel is known.

The effectiveness of a single iteration in correctly decoding a data block at about 1 dB above the SNR threshold is not typical of turbo codes or turbo equalization of a time-invariant channel. Near the SNR threshold on a time-invariant channel the first iteration of the turbo equalizer shows only a small improvement with increasing SNR, but this rate of increase accelerates with much larger SNR increases. When tracking a time-varying channel, in order to obtain an acceptable BER, most of the instantaneous channels encountered must have a SNR threshold well to the left of the SNR at which the adaptive equalizer operates. Under these circumstances most of the blocks are easily decoded in one iteration. Those difficult channels that require more than one iteration are rarely encountered. If the SNR is reduced, a few channels are quickly encountered that cannot be completely decoded and the BER rapidly becomes unacceptable.

In experiments using a time-invariant, but unknown channel with transmission at the SNR threshold, the adaptive turbo equalizer sometimes separates the two functions of equalization and decoding. The turbo equalizer may iterate repeatedly while only slowly improving its channel estimate and the number of correctly decoded symbols. Then once the channel has been approximated, the turbo equalizer converges to a final output in just a few iterations. The rapid convergence that characterizes the turbo equalizer on a known time-invariant channel is often delayed because the feedback from the SISO decoder is adversely affecting the equalization process. Resetting of extrinsic information from the SISO is sometimes beneficial, but even when effective, significant computational effort is wasted on the initial iterations through the complex equalization stage before the need to reset is identified. This phenomenon may also be present in the time-varying case, but it was not investigated in detail.

When testing the adaptive turbo equalizer at higher SNRs the wasted computational effort is evident at the end of the algorithm. In practice, operation of a receiver would occur significantly above the SNR threshold to provide a margin of error that would assure the required BER is obtained. This thesis primarily investigated operation of the adaptive turbo equalizer at the threshold required for a BER in the range of 10^{-4} to 10^{-5}, but in the cases where the SNR was above the threshold level the number of required iterations dropped quickly. With an SNR at 7 dB and above the equalizer tracked the channel, converged and correctly decoded 99.7% of blocks on the first iteration, only requiring more iterations on the channel near the most difficult ISI channel. Despite the success of the decoding, one to two additional iterations are necessary to identify convergence and terminate the algorithm. It appears that at an SNR only slightly higher than that required to yield an acceptable BER, the turbo equalizer can almost correctly decode the information message in a single iteration. Yet at least one and possibly two additional iterations are required to determine that the decoding algorithm could be terminated. The second iteration through the equalizer stage may serve to fine tune the channel estimate for the next block of data received, but the importance of this additional step was not fully evaluated.

The cost of iterations is high and unnecessary ones should be avoided. Turbo codes are characterized by a long latency period due to their complexity that is proportional to $N_I^{N_M+k_s}$. (N_I, N_M, n_s, and k_s are the symbol alphabet size, encoder memory size, symbols per codeword and symbols per information word respectively.) The complexity of the adaptive turbo equalizer is even higher for two reasons. First, the equalizer's complexity is proportional to $M\frac{n_s}{k_s}N_I^{M+k_s-1}$ where M is the length of the channel impulse response. Second, in practice the channel memory is much larger than the memory of a convolutional encoder.

The adaptive turbo equalizer as designed is of limited value. The turbo effect is active only over a narrow SNR range. Above this narrow range half or more of the computational effort is not used for decoding, but to verify convergence.

The high complexity of the adaptive turbo equalizer makes it imperative to avoid any unnecessary iterations of the decoding process. If the adaptive turbo equalizer is to be utilized effectively, a decision regarding the need for additional iterations must be made after the first iteration. Decoding disagreement between stages, Δ, may provide a test but this needs to be fully evaluated as a stopping criteria. It is important to note that the limitation of the adaptive turbo equalizer design is not a result of the specific equalizer stage chosen, but a basic limitation of the turbo principle on a time-varying channel.

Alternative receiver designs should focus on reducing the complexity of the equalizer stage and avoiding iterations between equalizer and error correction stages. Reduced complexity undoubtedly would be accompanied by lessor performance from the equalizer stage. Perhaps a stronger error correction code might compensate for this trade of reduced performance for less complexity. However, regardless of the type of equalizer stage utilized, it seems that the narrow range over which interstage feedback is effective limits the value of information feedback when tracking a time-varying ISI channel. For this reason it does not seem fruitful to place the equalization stage and decoding stage in a turbo-like architecture. A proposal that seems worthy of investigation is a low complexity equalizer outputting symbol likelihoods to a SCCC or PCCC. In this scenario the iterations would be within a powerful error correction code where complexity could be controlled. Following decoding by the error correction code, the equalizer could be rerun with accurate desired outputs to fine tune the channel estimates in preparation for the next block of data to be received.

For undersea acoustic applications, the relationship of equalizer stage complexity to channel impulse response length disqualifies the turbo equalization process from consideration without a significant reduction in equalizer stage complexity. The alternative design proposed above may be applicable to the undersea environment.

Because of the rapid channel variations encountered, the difficult channels encountered may be present for only a short part of any block of data. Under these circumstances iterative information sharing between the equalizer and error correcting stage is of no benefit to decoding.

6.1 Summary of Conclusions

1. The equalizer stage and feedback from the SISO stage of the adaptive turbo equalizer are sometimes in conflict when simultaneously estimating the channel and decoding the message.

2. Removal of conflicting feedback can increase the rate of convergence of the adaptive turbo equalizer.

3. The cost of having to adapt to a time-varying ISI channel is increased complexity (more iterations) at a low SNR.

4. The cost of channel adaptation in terms of SNR to obtain a given bit error rate is minimal.

5. The range of SNRs over which the "turbo" effect occurs is narrow when tracking a time-varying channel.

6.2 Future Work

1. Evaluate the use of decoding disagreement between stages, Δ, at the end of one iteration as a stopping criterion at high SNRs where the first iteration usually completely decodes the message.

2. Evaluate architectures where the equalizer stage and receiver do not iteratively feedback information.

3. Evaluate the importance of a second equalizer pass following complete decoding to improve channel estimates at high SNRs.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *IEEE Int. Conf. on Communications*, pp. 1064–1070, 1993.

[2] G. Bauch, H. Khorram, and J. Hagenauer, "Iterative equalization and decoding in mobile communications systems," in *IEEE GLOBECOM*, 1997.

[3] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Tran. on Information Theory*, vol. 42, pp. 409–429, Mar. 1996.

[4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output maximum a posteriori (map) module to decode parallel and serial concatenated codes," TDA Progress Report 42-127, JPL, November 1996.

[5] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Tran. on Information Theory*, vol. IT-42, pp. 429–445, March 1996.

[6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Tran. on Information Theory*, pp. 284–287, Mar. 1974.

[7] M. Hagh, M. Salehi, and J. Proakis, "Turbo equalization in frequency selective time invariant channels," CDSP TEG Progress Report TEG-1, Northeastern University, APRIL 2000.

[8] J. G. Proakis, *Digital Communications.* McGraw-Hill, Inc., 2001.

[9] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Soft-output decoding algorithms in iterative decoding of turbo codes," TDA Progress Report 42-124, JPL, February 1996.

[10] S. Haykin, *Adaptive Filter Theory.* Prentice-Hall Inc, 1996.

Biography of the Author

Ron Bryant was born in London, England on May 26, 1952. He immigrated to the U.S. in 1957 and grew up in Pontiac, Michigan.

He entered the Massachusetts Institute of Technology in 1971 and obtained his Bachelor of Science degree in Chemistry in 1974 and Master of Science Degree in Biochemistry in 1975.

In September 1975 he enrolled in Medical School at the University of Michigan. He graduated in 1979 and began specialty training in Pathology at the University of Vermont. In 1983 he received specialty certification from the American Board of Pathology and began working at Eastern Maine Medical Center in Bangor, Maine. He retired from the position of Chief of Pathology at the medical center in December 1996. In January 1997, he was enrolled for graduate study in Electrical Engineering at the University of Maine. His current research interests include communications and signal processing. Ron is a candidate for the Master of Science degree in Electrical Engineering from The University of Maine in December 2000.