PERFORMANCE EVALUATION

OF TURBO CODES

by

Guangchong Zhu

A project submitted to the

Department of Mathematics and Statistics

in conformity with the requirements

for the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

September 1998

Acknowledgments

I would like to give my sincere thanks to my supervisor, Dr. Fady Alajaji, for his advice, guidance and valuable help along the fulfillment of this project. I would also like to thank Dr. Jon Davis, for his guidance and help. My further thanks go to Dr. Nam Phamdo at State University of New York at Stony Brook for helpful discussions. I would also like to thank Dr. Glen Takahara and our Sun workstation system manager, Mr. Steve Butterworth, for their friendly assistance in using the computing equipment.

Personally, I would like to thank Dr. Bob Erdahl, for giving me the precious opportunity to join the graduate program at the Department of Mathematics and Statistics.

I say "Thank you" to all the friends who have been concerned about me during my study, especially to my friend Jean Au, for many useful discussions.

Finally, I would like to thank the Department of Mathematics and Statistics, the School of Graduate Studies and Research, and NSERC for their financial support.

Abstract

Since the early days of communications, approaching the Shannon limit has been a primary goal for coding theorists. In 1993, Berrou *et al.* proposed a structure named "Turbo codes" which claims an extraordinary performance with reasonable decoding complexity. This is regarded as one of the most exciting and potentially important developments in communications.

In this project, we begin with a study on the structure and principle of Turbo codes. We then investigate the application of Turbo codes for three different channels: the binary symmetric channel (BSC), the additive white Gaussian noise (AWGN) channel and the Rayleigh fading channel (with and without the side information at the receiver). Both the AWGN channel and the Rayleigh fading channel are used in conjunction with coherent BPSK modulation. The Shannon limit is also computed for the different channels and for different coding rates.

We next evaluate the performance of Turbo codes via simulation. More specifically, we systematically examine the effects of the sequence length N, interleaving and the number of iterations. For the three different channels, we investigate the performance of Turbo codes for the following rates: 1/3, 1/2, 2/3, 4/5 and 8/9. From our simulations, we observe that when the rate increases, the gap between the Shannon limit and the E_b/N_0 required to achieve a desired BER level increases. This degradation phenomenon is common for all three channels. We also observe that, within the

first six iterations in decoding high rate Turbo codes over the BSC and the AWGN channels, the coding gain between successive iterations dramatically decreases. This is not the case for the Rayleigh fading channel. For this channel, we also remark that when the rate increases, the slope of the performance curve decreases considerably, and that the lack of side information deteriorates the performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Shannon limit

In 1948, Shannon made a breakthrough in his landmark paper "A Mathematical Theory of Communication" [19], which opened a new area in the research of communications named error-correcting coding or channel coding. This paper gave two important results known as the *Source Coding Theorem* and the *Channel Coding Theorem*. The two theorems culminated in the *Lossy Information Transmission Theorem*, also named the *Joint Source-Channel Coding Theorem with Fidelity Criterion* [13]. From the lossy information transmission theorem, we know that for a given Bernoulli (1/2) source and a given channel with capacity C, to guarantee an end-to-end bit error rate of P_e, the code rate R_c and the capacity C must satisfy

$$R_c \cdot [1 - h_b(P_e)] < C, \tag{1.1}$$

where $h_b(\cdot)$ is the binary entropy function defined as:

$$h_b(x) = -x \log_2 x - (1-x)\log_2(1-x). \tag{1.2}$$

In (1.1), $[1 - h_b(P_e)]$ is actually the *rate distortion function* for a Bernoulli (1/2) source with Hamming distortion [7] given by

$$d(x, \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x}, \\ 1 & \text{if } x \neq \hat{x}. \end{cases}$$

As will be discussed in Chapter 4, the channel capacity is a function of the signal-to-noise ratio E_b/N_0, i.e., $C = C(E_b/N_0)$. Therefore, the optimum value of E_b/N_0 to get a bit error rate of P_e can be solved using (1.1) assuming equality. This optimum value of E_b/N_0 is called the Shannon limit.

Motivated by Shannon's promise, over the last five decades, researchers have been devoting great efforts to construct efficient code structures to approach the limit.

Theoretically, to get an arbitrarily small probability of error, we only need to increase the sequence length, but practically, the decoding algorithm becomes more complicated when the length increases. For the commonly used maximum-likelihood decoding algorithm, when the length is increased up to some number N, the computational complexity will make the decoding infeasible. Thus, many proposals are aiming at constructing powerful codes which work with large block length, but claim reasonable complexity.

In 1993, C. Berrou, A. Glavieux and P. Thitimajshima surprised the communication society by presenting a structure named "Turbo codes" which can achieve the bit error rate (BER) level of 10^{-5} with E_b/N_0 only 0.7 dB, as well as with a reasonable complexity of decoding [4]. This astonishing performance soon attracted researchers' attention and generated an extremely active area.

1.2 Literature review

The first proposal of Turbo codes was in [4] by Berrou *et al.* in 1993. One year later, P. Robertson [17] and Hagenauer *et al.* [10] clarified some important aspects that were ambiguous in [4]. A new version of the modified BCJR algorithm was also introduced, which contributed greatly to reducing computational complexity. In 1996, S. Benedetto and G. Montorsi shed some light on the structure of Turbo codes [2], [3], while in the same year, L. Perez *et al.* offered some explanation by introducing a theory named *spectral thinning* [15]. Their analysis showed that the Turbo codes was invented as the result of a clever intuition building on several concepts already known [2]. This is recently verified by Berrou *et al.* themselves [6].

Unfortunately, up to now, there is still no rigorous theoretical explanation about the Turbo codes. However, a wide range of simulation work is necessary to reveal different aspects of Turbo codes. Related to our project are the work of K. Tepe and J. Anderson [21], and the work of E. Hall and S. Wilson [11].

1.3 Contributions of the project

In this project, we mainly concentrated on investigating the performance of Turbo codes for different channels : the binary symmetric channel, the additive white Gaussian channel, and the Rayleigh fading channel. For the Rayleigh fading channel, we investigated two cases: one is that the channel side information is known to the decoder, the other is that the side information is unavailable.

In the original design of the Turbo encoder, puncturing is used to achieve high rates. We can see that when the puncturing rate increases, less parity bits are available. Intuitively, when the rate is too high, the redundancy will become inadequate for correcting errors. To verify this effect, we performed simulations for the rates: 1/3, 1/2, 2/3, 4/5 and 8/9 for each channel, and observed different degradation phenomena.

The effect of interleaving, sequence length, as well as iterations are also examined.

1.4 Overview of the project

In Chapter 2, the structure of the Turbo encoder is presented. A performance bound is also discussed, which offers an explanation of the performance of Turbo codes at a primary level.

In Chapter 3, the structure of the Turbo decoder is presented first, then the modified BCJR algorithm is analyzed in comparison with the original BCJR algorithm. The principle of iterative decoding in accordance with a serial concatenated structure is studied.

In Chapter 4, the application of Turbo codes to different channels is investigated. These channels are: the binary symmetric channel, the additive white Gaussian noise channel and the Rayleigh fading channel (with and without the side information at the decoder). For each channel, modifications in the Turbo decoder are discussed, and the Shannon limit for each channel is computed.

Our simulation results are presented in Chapter 5. The effects of the sequence length, interleaving and the number of iterations are examined. For each channel, the performance of Turbo codes is investigated for different rates.

Chapter 6 concludes our report and points out possible future work.

Chapter 2

Turbo encoding

2.1 Convolutional encoder: NSC and RSC encoder

Before we start to discuss Turbo codes in detail, it is necessary to present some basic concepts about convolutional codes.

2.1.1 Preliminary concepts

A *convolutional code* is generated by passing the information sequence to be transmitted through a linear finite-state shift register. In general, in an (n, k, ν) convolutional code, a k-bit input sequence produces an n-bit output sequence where each output bit is a function of the current input sequence as well as previous input blocks. ν is the number of registers the encoder has, called the *memory*. The rate of the code is defined by $R = k/n$ [12].

Two important concepts for convolutional codes are the Hamming distance and the Hamming weight. The *Hamming distance* between two codewords is the number of positions where they differ from each other. The *Hamming weight* of a codeword is the total number of non-zero components it contains. The convolutional codes are generated by a linear system [16], therefore, the set of Hamming weights of all possible codewords is equivalent to the set of Hamming distances between codewords, since the Hamming weight of a codeword can be regarded as the Hamming distance between this codeword and the all-zero codeword.

Furthermore, the *free distance* is the minimum Hamming distance between any two codewords in the code [12], equivalently, it is the minimum Hamming weight of all possible non-zero codewords [15].

In the design of Turbo codes, a special kind of convolutional encoder named the *recursive systematic convolutional (RSC)* encoder is used. The RSC encoder will be presented with its counterpart, the *non-systematic convolutional (NSC)* encoder, for comparison.

2.1.2 Non-systematic convolutional encoder

Shown in Fig. 2.1 is a non-systematic convolutional encoder. The symbol Σ represents the modulo 2 summation. Usually the encoder is denoted according to its generator

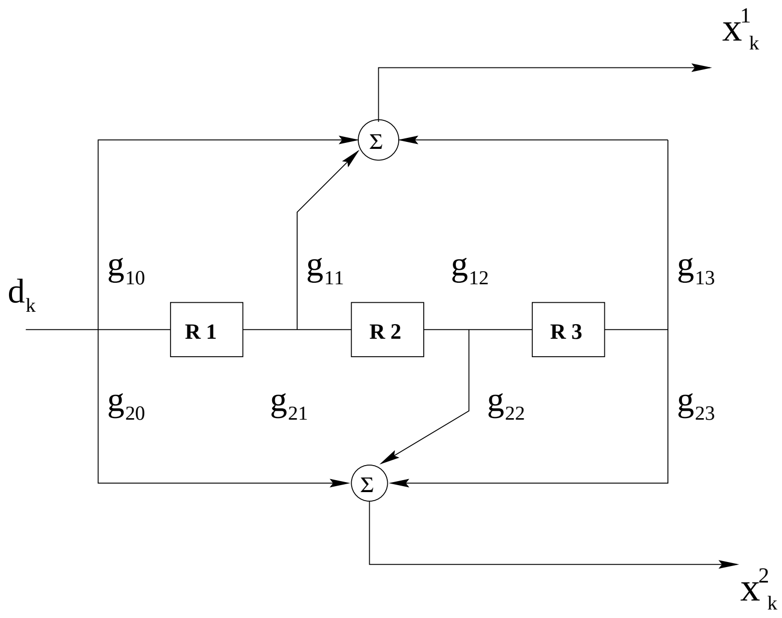

Figure 2.1: Non-systematic convolutional encoder.

structure.

Encoder generators: The encoder generators refer to the structure of circuit connections by which the values in some certain registers contribute to the output, while others are not used. For example, the generators of the encoder in Fig. 2.1 are:

$$G_1 = \{g_{10}, g_{11}, g_{12}, g_{13}\} = \{1, 1, 0, 1\}, \tag{2.1}$$

$$G_2 = \{g_{20}, g_{21}, g_{22}, g_{23}\} = \{1, 0, 1, 1\}. \tag{2.2}$$

By introducing the *time delay operator* D, we have the polynomial form [16]

$$(G_1, G_2) = (1 + D + D^3, 1 + D^2 + D^3). \tag{2.3}$$

For convenience, we usually use the equivalent octal form. The non-systematic convolutional encoder in Fig. 2.1 is denoted in octal form as (15, 13). For the encoder in Fig. 2.1, the memory $\nu = 3$.

Data flow: The relations between the output bits (x_k^1, x_k^2) and the input bit d_k can be written as follows.

$$x_k^1 = \sum_{i=0}^{\nu} g_{1i} \cdot d_{k-i} \mod 2 \tag{2.4}$$

$$x_k^2 = \sum_{i=0}^{\nu} g_{2i} \cdot d_{k-i} \mod 2. \tag{2.5}$$

In polynomial form, denote the input sequence $\{d_k\}$ as $d(D)$, the output pair $(x^1(D), x^2(D))$ in Fig. 2.1 can be written as

$$(x^1(D), x^2(D)) = d(D)(1 + D + D^3, 1 + D^2 + D^3). \tag{2.6}$$

The input bit d_k goes directly into register R_1, whose original value d_{k-1} is shifted into the next register R_2, and so on.

Encoder state: Before the information sequence $\{d_k\}$ is fed into the encoder, it is assumed that each register contains a "0". In this case, we say that the encoder is in the all-zero state. Then the state of the encoder is defined by its shift register contents [12]. The encoder in Fig. 2.1 has eight possible states: 000, 001, 011, ... , 111. At time k, when d_k is about to go into the encoder, we say that the encoder is in state S_{k-1}, and after the data are shifted, the state is S_k. The output pair (x_k^1, x_k^2) is determined by both the input d_k and the state S_{k-1}.

2.1.3 Recursive systematic convolutional encoder

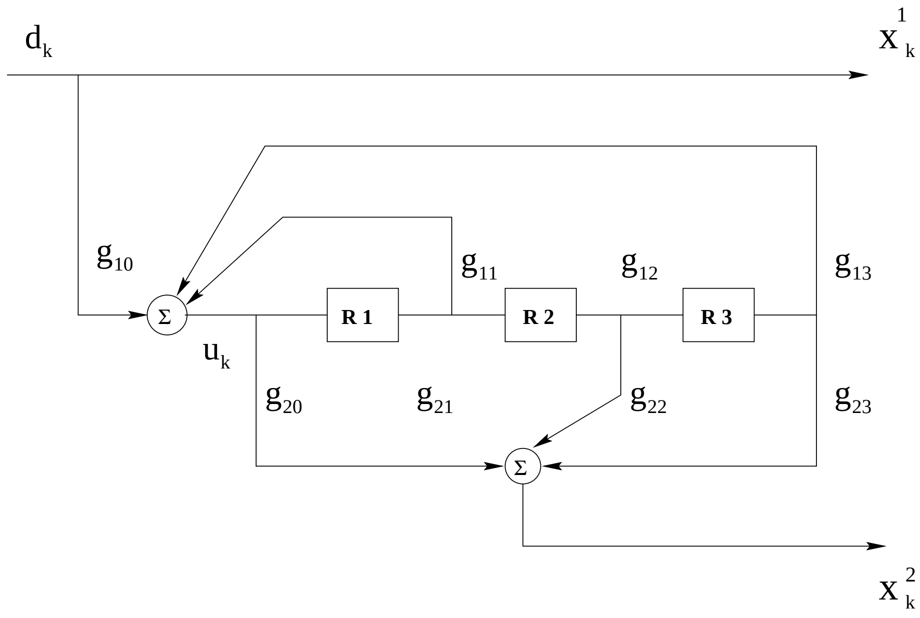

Figure 2.2: Recursive systematic convolutional encoder.

Fig. 2.2 shows a recursive systematic convolutional encoder whose NSC counterpart is the one shown in Fig. 2.1. *Systematic* means that part of the encoded sequence is identical to the input sequence. Furthermore, unlike the NSC encoder, whose generators are both feed-forward, the summation node of the generator G_1 is moved before the registers through a feedback loop, hence the name *recursive*.

Encoder generators: The encoder generators are still written as:

$$G_1 = \{g_{10}, g_{11}, g_{12}, g_{13}\} = \{1, 1, 0, 1\}, \tag{2.7}$$

$$G_2 = \{g_{20}, g_{21}, g_{22}, g_{23}\} = \{1, 0, 1, 1\}. \tag{2.8}$$

The difference is that g_{10} refers to the branch before the node of summation, while g_{20} refers to the one between the node of summation and the first register R_1. Therefore, for an RSC encoder, g_{10} is always "1", and all others can be either "0" or "1".

In polynomial form, the RSC encoder in Fig. 2.2 is:

$$\left(1, \frac{G_2}{G_1}\right) = \left(1, \frac{1+D^2+D^3}{1+D+D^3}\right). \tag{2.9}$$

The equivalent octal form is therefore (1, 13/15). But in the literature, it is often denoted as (15, 13) for convenience [4].

Data flow: As *systematic* implies, for each bit,

$$x_k^1 \equiv d_k \quad for \quad all \quad k.$$

The parity bit x_k^2 is determined by d_k and the contents in the shift registers. Unlike the NSC encoder, the data contained in the registers of the RSC encoder are affected by the generator G_1. Note that the data before the node of summation is d_k, but after the summation, the notation is u_k. The relations between d_k, u_k, and the parity check bit x_k^2 are as follows:

$$u_k = d_k + \sum_{i=1}^{\nu} g_{1i} \cdot u_{k-i} \mod 2 \tag{2.10}$$

$$x_k^2 = \sum_{i=0}^{\nu} g_{2i} \cdot u_{k-i} \mod 2. \tag{2.11}$$

In polynomial form, the output pair $(x^1(D), x^2(D))$ in Fig. 2.2 can be written as

$$(x^1(D), x^2(D)) = d(D)\left(1, \frac{1+D^2+D^3}{1+D+D^3}\right). \tag{2.12}$$

State termination: To make the decoding easier, we usually let the encoder start at the all-zero state, and terminate at the all-zero state [15]. For an NSC encoder with memory ν, it can be terminated at all-zero state by simply inputting ν zeros consecutively. To terminate an RSC encoder, we also need only ν bits, but due to the recursive structure, this ν-bit tail cannot be all-zero. Each bit, either "1" or "0", is determined by the encoder state at the moment. For example, to terminate the (15, 13) RSC encoder in Fig. 2.2 when the current state is 110, it can be verified that the 3-bit tail needed is 101. Therefore, for an N-bit sequence, when the last ν bits are used to terminate the encoder state, there is only $(N - \nu)$-bit useful information transmitted.

The problem of termination also implies that for an RSC encoder, the input sequence has at least weight 2.

2.2 Structure of Turbo encoder

The encoder of Turbo codes, in one word, is a parallel concatenation of several recursive systematic convolutional (RSC) encoders separated by interleavers, combined with a puncturing device. Fig. 2.3 shows the structure of a specific Turbo encoder which is a concatenation of two identical (37, 21) RSC encoders. This is the one Berrou *et al.* used in 1993 [4]; some encoders with similar structure have been used by other researchers [15], [2]. In general, there can be more than two constituent RSC encoders; also, they do not have to be identical. Analysis on the encoder in Fig. 2.3 can be easily extended to more general structures.

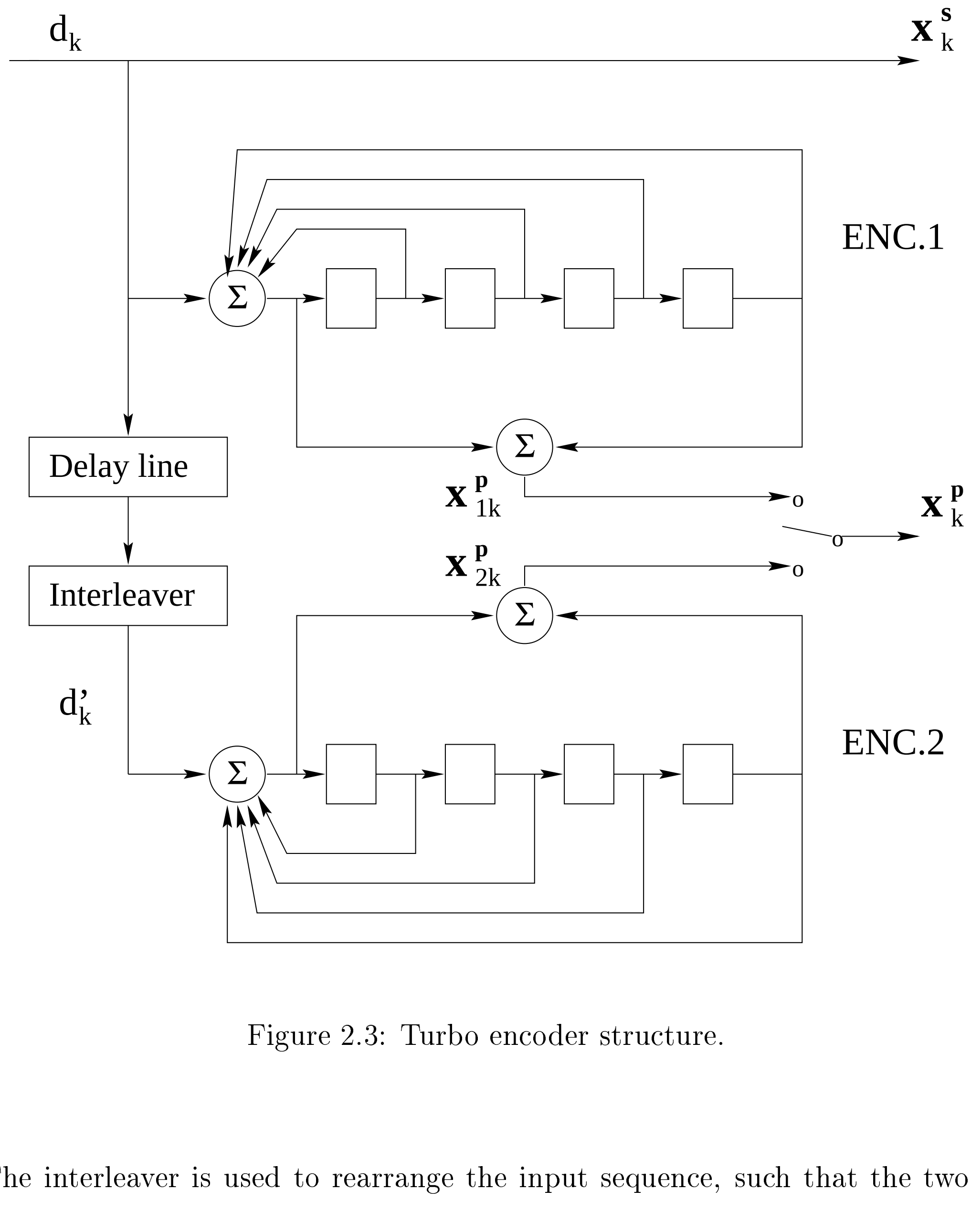

Figure 2.3: Turbo encoder structure.

The interleaver is used to rearrange the input sequence, such that the two constituent encoders are operating on two sequences consisting of the same bits, but in different order. The delay line concatenated before the interleaver is a kind of buffer used to gather the sequence $\{d_k\}$ before interleaving. After interleaving, the sequence is denoted by $\{d'_k\}$. For each input bit d_k, the outputs are the systematic bit x_k^s and two parity bits x_{1k}^p and x_{2k}^p, which can be obtained by the formula (2.11) given in the previous section. The superscripts s and p are used to distinguish the systematic bit from the parity bit.

At the end, a puncturing device is used on the two parity check sequences $\{x_{1k}^p\}$ and $\{x_{2k}^p\}$ in order to obtain the desired rate. These two sequences are deleted alternatively according to some pattern before being transmitted.

In summary, the main features of the Turbo encoder are:

1) Parallel concatenation; 2) RSC encoders; 3) Interleaving; 4) Puncturing.

2.3 Interleaving

Interleaving itself is not a new concept, its early application is to combat burst errors. Channels that have fading effect can cause errors which are statistically correlated, and thus result in a large number of consecutive errors [16]. For such cases, an interleaver is placed before the channel, such that the input sequence bits are reordered before transmission. If this reordered sequence is corrupted by a burst of channel errors, then a de-interleaver is used at the receiver to permute the sequence bits back into their original positions; thus dispersing the errors on consecutive bits.

For the purpose of dealing with channel burst errors, the interleaver is of course placed between the encoder and the channel. Turbo codes illustrate the first occurrence when an interleaver is used between the constituent encoders. Surprisingly, such an interleaver turns out to play an important role in the extraordinary performance of Turbo codes.

There are two kinds of commonly used interleavers: the *uniform interleaver* and the pseudo-random interleaver.

The *uniform interleaver* is realized in the form of an $m \times n$ matrix. An mn-bit sequence is fed into the matrix row by row and read out column by column. The *pseudo-random interleaver* performs a permutation on the input sequence bits in a random manner. For Turbo codes, a pseudo-random interleaver can perform much better than a uniform one [15].

Another important factor is the length of the interleaver, N, which is identical to the length of the input sequence. It has been observed that as N increases, the performance of Turbo codes improves.

2.4 Puncturing

Clearly, each constituent encoder in Fig. 2.3 is of rate 1/2, while the overall rate of the encoder is 1/3. We know that the lower the rate is, the larger is the amount of redundancy sent along with the information bits. For the encoder presented here, the second constituent encoder originally has two outputs $x_k^{s\prime}$ and x_{2k}^p. Since the systematic part of ENC.2, $x_k^{s\prime}$, is nothing but a permutation of x_k^s, there is no need to transmit it. Therefore, by omitting $x_k^{s\prime}$, a rate of 1/3 is obtained.

Furthermore, if a rate higher than 1/3 is needed, a puncturing device is used. By periodically deleting some selected parity check bits of both ENC.1 and ENC.2, we can get the desired rate. For example, to achieve a rate of 1/2, we can delete all odd parity bits of ENC.1 and all even bits of ENC.2. In practice, any high rate can be obtained by deleting additional parity check bits.

2.5 Performance bound discussion

In this section, we give a performance bound for convolutional codes which is also valid for Turbo codes [15]. Finding out performance bounds for Turbo codes has been actively investigated recently [8], [23], but no tight bound has yet been established. The bound herein presented, loose as it is, can serve as a good example to illustrate the criteria for designing good codes, thus it can give us some idea of why Turbo code can perform so well.

2.5.1 A performance bound

Let the input sequence for an RSC encoder have length N. When the rate is 1/2, the output of an N-bit input sequence has length $2N$.

By using the union bound, the bit error rate (BER) performance of a finite-length convolutional code with maximum-likelihood (ML) decoding over an additive white Gaussian noise channel with an SNR of E_b/N_0 can be bounded above [15] by

$$P_b \leq \sum_{i=1}^{2^N} \frac{w_i}{N} Q\left(\sqrt{d_i \frac{R_c E_b}{2N_0}}\right), \tag{2.13}$$

where w_i is the Hamming weight of the i^{th} information sequence, d_i is the total Hamming weight of the i^{th} codeword, and $Q(\cdot)$ is the complimentary error function [16], defined as

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt, \qquad x \geq 0.$$

Denote the number of codewords with Hamming weight d as N_d, and the total information weight of codewords with Hamming weight d as W_d. Define the *average information weight per codeword* as:

$$\tilde{w}_d = \frac{W_d}{N_d}. \tag{2.14}$$

The upper bound can be re-written as:

$$P_b \leq \sum_{d=d_{free}}^{2N} \frac{N_d \tilde{w}_d}{N} Q\left(\sqrt{d \frac{R_c E_b}{2N_0}}\right), \tag{2.15}$$

where d_{free} is the free distance of the code.

The performance of a Turbo code with maximum-likelihood decoding can also be bounded by (2.15) [15]. Observing this bound, we can have the following conclusions which point out the direction for designing good codes.

1. Since the Q function decreases drastically when its parameter increases, the larger the information weight d, the less significant the term with weight d affects the performance. Therefore, to improve the performance, one way is to maximize the *free distance*.

2. For each term, the *multiplicity* N_d and the length N also affect the performance. We want to enlarge the length N while maintaining N_d small.

2.5.2 Discussion on Turbo encoder

The free distance of a convolutional code can be determined by the structure of the encoder. An analytical tool is the *state transfer function* [16]. For Turbo codes, the free distance depends both on the structure of the constituent encoder and the pattern of interleaving. Therefore, with a pseudo-random interleaver, an analytical approach for finding the free distance of Turbo codes seems infeasible. J. Seghers proposed an algorithm for finding the free distance of a given Turbo encoder in [18]. It has been shown that the free distance of Turbo codes is usually not very large. However, At low SNR, when the sequence length N is large, minimizing the multiplicity is even more important than maximizing the free distance [15].

In the Turbo encoder, the interleaver maps an input sequence $\{d_k\}$ to $\{d'_k\}$. With high probability, a low-weight parity sequence in the first constituent encoder corresponds to a high-weight parity sequence in the second constituent encoder. With a pseudo-random interleaver, $N_d \tilde{w}_d$ is much less than N for low-weight codewords. For example, by the algorithm for finding the free distance of Turbo codes in [18], it was found that for a Turbo code using a pseudo-random interleaver and (37, 21) RSC encoders, when $N = 65536$, $d_{free} = 6$, $\tilde{w}_{free} = 2$, $N_{free} = 3$ [15].

For a uniform interleaver, there are some certain sequences that will be permuted into themselves. For example, consider a weight-2-input sequence which has the two "1"s located on the diagonal of the matrix. By reading in row by row and outputting column by column, the sequence is permuted into itself. This means that the two parity sequences generated by the two constituent encoders are exactly the same. So for these sequences, the interleaver has no effect. Similarly we can consider the weight-4-input sequences. We can remark that when N is large, the number of such possible sequences is also large, which results in a large multiplicity of N_{free}. Therefore, increasing the length N does not improve the performance significantly.

At the beginning of this chapter, we have evoked many differences between an NSC encoder and an RSC encoder. When they are used individually, there are not many aspects in which an RSC encoder is superior to an NSC counterpart. However, as to the application to Turbo codes, RSC encoders are preferable [4]. S. Benedetto and G. Montorsi shed some light on this issue in the discussion of a bound derived from using the *conditional weight enumerating function* [2], and showed that the RSC structure is essential for achieving large performance gain [3].

Chapter 3

Turbo decoding

3.1 Structure of the Turbo decoder

First, it is necessary to clarify the notations for the data before and after transmission in the Turbo decoding problem.

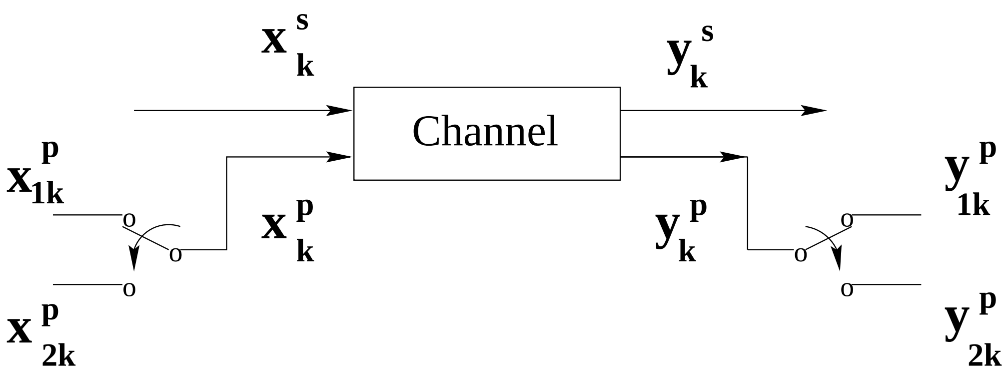

Figure 3.1: Data before and after transmission.

As shown in Fig. 3.1, before transmission, the systematic information bit is x_k^s, the parity bit is x_k^p, which is obtained from x_{1k}^p and x_{2k}^p by a puncturing device. After the transmission, correspondingly, the systematic information bit is denoted as y_k^s, and the parity bit is y_k^p, which is then separated into y_{1k}^p and y_{2k}^p by a switch. For the sake of simplicity, we denote the pair (x_k^s, x_k^p) as x_k, and the pair (y_k^s, y_k^p) as y_k in the analysis. When puncturing is used to achieve high rates, some parity bits are not available. In this Chapter, we give analysis on the case when the rate is 1/3, i.e., no puncturing is used. The punctured bits will be treated differently in the discussion of each channel, and the extension of applying the modified algorithm for higher rates achieved by puncturing is straightforward.

The decoder of Turbo codes is made up of two elementary decoders serially concatenated, as shown in Fig. 3.2. Each constituent decoder employs the *modified BCJR algorithm*. The interleaver between the two decoders is identical to the one used in the encoder, and a de-interleaver is used after the second decoder DEC.2 to perform the inverse permutation of the interleaver, such that the output sequence of DEC.2 is permuted back into the original order. Between the two decoders, the value of $L_{1e}(d_k)$ or $L_{2e}(d_k)$ is called the *extrinsic information*, which is a part of the *log-likelihood ratio (LLR)* $\Lambda(d_k)$. Its expression will be given in the section of iterative decoding (cf. (3.50) and (3.52)).

The decoding process is realized in an iterative manner by exchanging the *extrinsic*

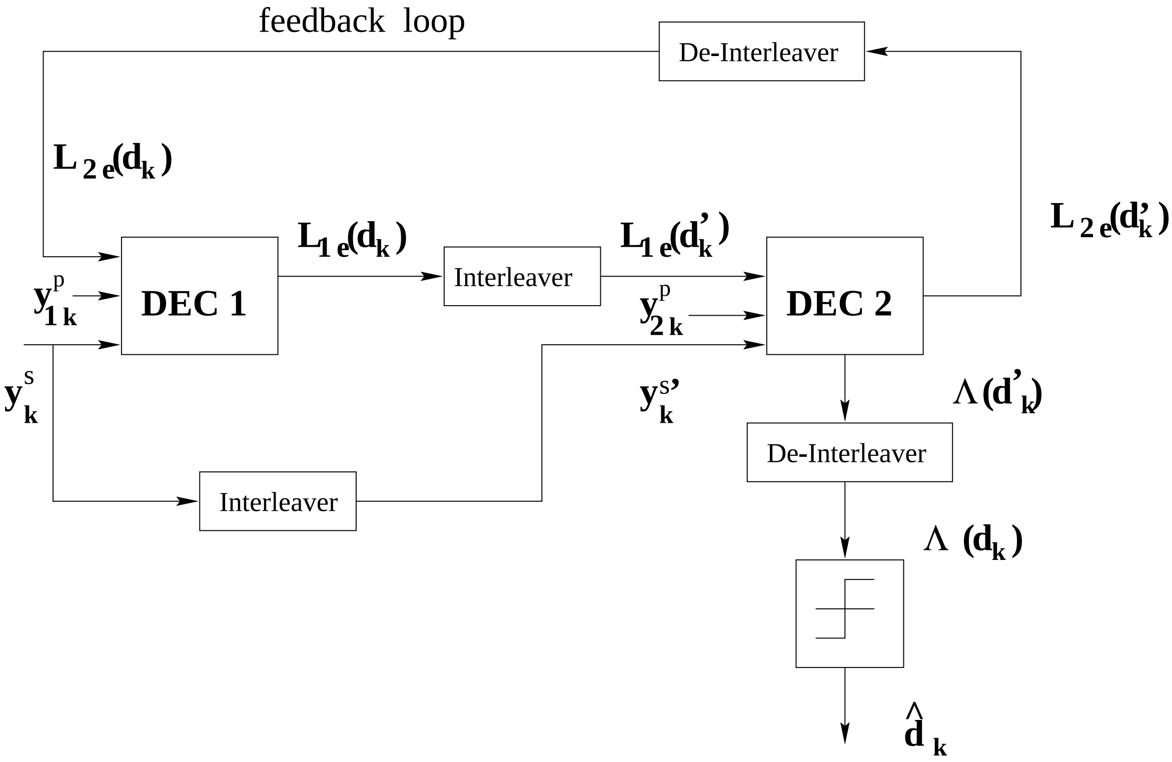

Figure 3.2: Turbo decoder structure.

information between the two decoders. For the first iteration, DEC.1 has two inputs: the information bit y^s_k and the parity bit y^p_{1k} generated from ENC.1. DEC.2 has three inputs: the information bit after interleaving, $y^{s\prime}_k$, the parity bit y^p_{2k} from ENC.2, and the interleaved version of the output of DEC.1, $L_{1e}(d'_k)$. After the first iteration, DEC.1 also has three inputs: in addition to y^s_k, y^p_{1k}, the third input is the output of DEC.2, $L_{2e}(d_k)$. The final decision is made from $\Lambda(d_k)$,

$$\hat{d}_k = 1, \quad if \quad \Lambda(d_k) > 0, \tag{3.1}$$

$$\hat{d}_k = 0, \quad if \quad \Lambda(d_k) \leq 0. \tag{3.2}$$

We can summarize the main features of the Turbo decoder as follows.

1) Serial concatenation; 2) Modified BCJR algorithm;

3) Extrinsic information; 4) Iterative decoding.

The *modified BCJR algorithm* is the most important part to understand the Turbo decoding process, and is prerequisite to the *extrinsic information* and the principle of iterative decoding. Therefore, a detailed analysis on the modified BCJR algorithm is necessary. In order to see why and how it was modified for Turbo decoding, we first present the original BCJR algorithm for comparison.

3.2 BCJR algorithm

For convolutional codes, a well known decoding method is the Viterbi algorithm, which is optimal in the sense of minimizing the probability of sequence error [16]. However, the Viterbi algorithm is not able to yield the *a priori probability (APP)* directly for each decoded bit. In 1974, L. Bahl, J. Cocke, F. Jelinek, and J. Raviv proposed a new algorithm [1] which works in a forward-backward recursive fashion and minimizes the bit error probability. This algorithm, now usually called the BCJR algorithm, was successfully used by Berrou *et al.* in 1993 for Turbo codes. Primarily, Berrou *et al.* applied the BCJR algorithm directly to Turbo codes with few modifications, thus rendering the forward recursion part unnecessarily complicated [4].

A modified version of the BCJR algorithm, which reduces a considerable amount of computational work, was presented in [17], [5].

In this section, we introduce the BCJR algorithm. We mainly concentrate on analyzing how the algorithm is derived; this will later guide us in modifying it and applying it to Turbo codes.

3.2.1 Problem formulation

In the paper of Bahl *et al.* [1], the problem was proposed as follows.

For a transmission system which consists of a Markov source, a discrete memoryless channel (DMC) and a decoder, as shown in Fig. 3.3, denote the state of the Markov source at time k as S_k, the output of the source at time k as X_k, which is also the input to the channel, while the output of the channel (the input to the decoder) at time k is denoted as Y_k. The time index k varies from 1 up to N. The possible states are 0, 1, $\cdots$, M.

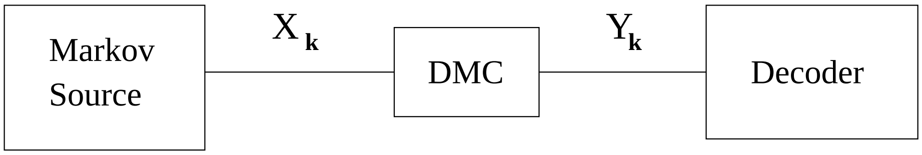

Figure 3.3: Schematic diagram of transmission system.

The state transitions of the Markov source are governed by the transition probabilities

$$\pi_k(m|m') \stackrel{\triangle}{=} Pr\{S_k = m|S_{k-1} = m'\},$$

and the output by the probabilities

$$q_k(x|m', m) \stackrel{\triangle}{=} Pr\{X_k = x|S_{k-1} = m', S_k = m\}.$$

The Markov source starts with $S_0 = 0$ and terminates with $S_N = 0$. The transition probabilities of the DMC are defined by $p(\cdot|\cdot)$, such that

$$p(y_1^N|x_1^N) \stackrel{\triangle}{=} Pr\{Y_1^N = y_1^N|X_1^N = x_1^N\} = \prod_{i=1}^{N} p(y_i|x_i),$$

where $Y_1^N = (Y_1, \cdots, Y_N)$ is the received sequence, and $X_1^N = (X_1, \cdots, X_N)$ is the input sequence to the DMC.

The goal is to find the joint probability

$$\begin{aligned} \lambda_k(m) \stackrel{\triangle}{=} Pr\{S_k = m, y_1^N\}, \qquad & \forall k = 1, ..., N, \\ and \quad & \forall m = 0, ..., M. \end{aligned} \tag{3.3}$$

3.2.2 Derivation of the BCJR algorithm

The general idea of calculating (3.3) is to factorize $\lambda_k(m)$ into two factors, each of which can be computed recursively. First,

$$\lambda_k(m) \quad = \quad Pr\{S_k = m, y_1^N\}$$

$$\begin{aligned}
&= Pr\{S_k = m, y_1^k, y_{k+1}^N\} && (3.4)\\
&= Pr\{S_k = m, y_1^k\} \cdot Pr\{y_{k+1}^N | S_k = m, y_1^k\} && (3.5)\\
&= \alpha_k(m) \cdot Pr\{y_{k+1}^N | S_k = m\} && (3.6)\\
&= \alpha_k(m) \cdot \beta_k(m), && (3.7)
\end{aligned}$$

where $\alpha_k(m) \stackrel{\triangle}{=} Pr\{S_k = m, y_1^k\}$, $\beta_k(m) \stackrel{\triangle}{=} Pr\{y_{k+1}^N | S_k = m\}$, and the simplification in (3.6) follows from the fact that if S_k is known, events after time k do not depend on y_1^k.

The recursive relations for calculating $\alpha_k(m)$ and $\beta_k(m)$ are obtained as follows.

$$\begin{aligned}
\alpha_k(m) &= Pr\{S_k = m, y_1^k\}\\
&= \sum_{m'} Pr\{S_k = m, S_{k-1} = m', y_1^{k-1}, y_k\}\\
&= \sum_{m'} Pr\{S_k = m, y_k | S_{k-1} = m', y_1^{k-1}\} \cdot Pr\{S_{k-1} = m', y_1^{k-1}\}\\
&= \sum_{m'} Pr\{S_k = m, y_k | S_{k-1} = m'\} \cdot \alpha_{k-1}(m'), \qquad (3.8)
\end{aligned}$$

where the law of total probability is used in step 2, and the simplification in step 4 is justified as in (3.6). Similarly, we have the following recursive relation for $\beta_k(m)$.

$$\begin{aligned}
\beta_k(m) &= Pr\{y_{k+1}^N | S_k = m\}\\
&= \sum_{m'} Pr\{S_{k+1} = m', y_{k+1}, y_{k+2}^N | S_k = m\}\\
&= \sum_{m'} Pr\{y_{k+2}^N | S_k = m, S_{k+1} = m', y_{k+1}\} \cdot Pr\{S_{k+1} = m', y_{k+1} | S_k = m\}\\
&= \sum_{m'} Pr\{y_{k+2}^N | S_{k+1} = m'\} \cdot Pr\{S_{k+1} = m', y_{k+1} | S_k = m\}
\end{aligned}$$

$$= \sum_{m'} \beta_{k+1}(m') \cdot Pr\{S_{k+1} = m', y_{k+1} | S_k = m\}. \tag{3.9}$$

Notice that exactly the same term $Pr\{S_k = m, y_k | S_{k-1} = m'\}$ appears in (3.8) and (3.9). Therefore, it is appropriate to define it as

$$\gamma_k(m', m) \triangleq Pr\{S_k = m, y_k | S_{k-1} = m'\}. \tag{3.10}$$

Then the recursive relations become :

$$\alpha_k(m) = \sum_{m'} \alpha_{k-1}(m') \gamma_k(m', m), \tag{3.11}$$

$$\beta_k(m) = \sum_{m'} \beta_{k+1}(m') \gamma_{k+1}(m, m'). \tag{3.12}$$

The function $\gamma_k(m', m)$ can be factored into three parts

$$\begin{aligned} \gamma_k(m', m) &= \sum_x Pr\{S_k = m | S_{k-1} = m'\} \\ &\quad \cdot Pr\{X_k = x | S_{k-1} = m', S_k = m\} \cdot Pr\{y_k | X_k = x\} \qquad (3.13) \\ &= \sum_x \pi_k(m|m') q_k(x|m', m) p(y_k|x), \qquad (3.14) \end{aligned}$$

where $\pi(\cdot|\cdot)$, $q(\cdot|\cdot,\cdot)$ and $p(\cdot|\cdot)$ are already known.

To solve (3.11) and (3.12), boundary conditions are needed. Since the Markov source starts at the initial state $S_0 = 0$ and terminates at the state $S_N = 0$, we can define the initial and terminal conditions for $\alpha_k(m)$ and $\beta_k(m)$ as

$$\alpha_0(0) = 1, \quad \text{and} \quad \alpha_0(m) = 0 \quad \text{for} \quad m \neq 0, \tag{3.15}$$

$$\beta_N(0) = 1, \quad \text{and} \quad \beta_N(m) = 0 \quad \text{for} \quad m \neq 0. \tag{3.16}$$

From (3.15), $\alpha_k(m)$ can be computed recursively from the previous values, and from (3.16), $\beta_k(m)$ can be computed in a backward recursive manner.

3.3 Modified BCJR algorithm for Turbo decoding

3.3.1 Problem formulation

In the Turbo decoding problem, we want to figure out the original information sequence $\{d_k\}$ based on the received sequence $\{y_k\}$.For convenience, the sequence before transmission is denoted as $x_1^N \triangleq \{x_1, \cdots, x_N\}$, and the sequence after the channel as $y_1^N \triangleq \{y_1, \cdots, y_N\}$. So the goal is to find

$$\lambda_k^i \triangleq Pr\{d_k = i | y_1^N\} = \sum_{m=0}^{M} \lambda_k^i(m)$$
$$\forall k = 1, \cdots, N, \quad and \quad \forall i \in \{0, 1\},$$

where

$$\lambda_k^i(m) \triangleq Pr\{d_k = i, S_k = m | y_1^N\}. \tag{3.17}$$

At first, in the paper of Berrou *et al.* in 1993 [4], $\lambda_k^i(m)$ was solved in a way that directly imitated the BCJR algorithm; i.e., factorize $\lambda_k^i(m)$ as

$$\lambda_k^i(m) = \alpha_k^i(m)\beta_k(m), \tag{3.18}$$

and then derive similar recursive relations for $\alpha_k^i(m)$ and $\beta_k(m)$. $\lambda_k^i(m)$ was finally obtained by taking the product of $\alpha_k^i(m)$ and $\beta_k(m)$. However, since $\lambda_k^i(m)$ and $\alpha_k^i(m)$

have an additional superscript i, the forward recursive relation of $\alpha_k^i(m)$ becomes very complicated. Careful analysis reveals that $\alpha_k^i(m)$ can still be simplified as $\alpha_k(m)$. Obviously,

$$\lambda_k^i(m) \neq \alpha_k(m)\beta_k(m),$$

as the left hand side depends on i, while the right hand side does not. But this problem can be avoided by calculating the log-likelihood ratio instead of $\lambda_k^i(m)$ for each $i \in \{0, 1\}$ [17].

The log-likelihood ratio is defined as

$$\Lambda(d_k) \triangleq \log \frac{\sum_m \lambda_k^1(m)}{\sum_m \lambda_k^0(m)}. \tag{3.19}$$

The decoder can make a decision by comparing $\Lambda(d_k)$ to a zero threshold. Let $\hat{d}_k$ be the hard decision that the decoder makes for d_k; then

$$\hat{d}_k = 1, \quad \text{if} \quad \Lambda(d_k) > 0, \tag{3.20}$$

$$\hat{d}_k = 0, \quad \text{if} \quad \Lambda(d_k) \leq 0. \tag{3.21}$$

The modified BCJR (M-BCJR) algorithm is to factorize $\Lambda_k(m)$ properly and get its value by calculating each factor recursively.

3.3.2 Derivation of the modified BCJR algorithm

To understand why the BCJR algorithm must be modified for Turbo codes, we begin by pointing out the specific problem, and then we describe how we will address it.

1) Factorization of $\lambda_k^i(m)$ and $\Lambda(d_k)$:

For each $\lambda_k^i(m)$, we have

$$\begin{aligned}
\lambda_k^i(m) &= Pr\{d_k = i, S_k = m | y_1^N\} \\
&= \frac{Pr\{d_k = i, S_k = m, y_1^k, y_{k+1}^N\}}{Pr\{y_1^N\}} \\
&= \frac{Pr\{y_{k+1}^N | d_k = i, S_k = m, y_1^k\} \cdot Pr\{d_k = i, S_k = m, y_1^k\}}{Pr\{y_1^N\}} \\
&= \frac{Pr\{y_{k+1}^N | S_k = m\} \cdot Pr\{d_k = i, S_k = m, y_1^k\}}{Pr\{y_1^N\}}, \qquad (3.22)
\end{aligned}$$

where the simplification in the last step follows from the fact that given $S_k = m$, the observation y_{k+1}^N does not depend on d_k and y_1^k. For the second part of the numerator, using the law of total probability, we have

$$\begin{aligned}
& Pr\{d_k = i, S_k = m, y_1^k\} = Pr\{d_k = i, S_k = m, y_k, y_1^{k-1}\} \\
&= \sum_{m'} Pr\{d_k = i, S_k = m, y_k, S_{k-1} = m', y_1^{k-1}\} \\
&= \sum_{m'} Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m', y_1^{k-1}\} \cdot Pr\{S_{k-1} = m', y_1^{k-1}\} \\
&= \sum_{m'} Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m'\} \cdot Pr\{S_{k-1} = m', y_1^{k-1}\}. \qquad (3.23)
\end{aligned}$$

Substituting (3.23) into (3.22) yields

$$\lambda_k^i(m) = \frac{\sum_{m'} Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m'\} Pr\{S_{k-1} = m', y_1^{k-1}\} Pr\{y_{k+1}^N | S_k = m\}}{Pr\{y_1^N\}}. \qquad (3.24)$$

We next define

$$\alpha_k(m) \triangleq Pr\{S_k = m | y_1^k\}, \qquad (3.25)$$

$$\beta_k(m) \triangleq \frac{Pr\{y_{k+1}^N|S_k = m\}}{Pr\{y_{k+1}^N|y_1^k\}}, \tag{3.26}$$

$$\gamma_i(y_k, m', m) \triangleq Pr\{d_k = i, S_k = m, y_k|S_{k-1} = m'\}. \tag{3.27}$$

In (3.24), if the denominator were the product of $Pr\{y_1^{k-1}\}$ and $Pr\{y_{k+1}^N|y_1^k\}$, then $\lambda_k^i(m)$ could be expressed as

$$\lambda_k^i(m) = \sum_{m'} \gamma_i(y_k, m', m)\alpha_{k-1}(m')\beta_k(m). \tag{3.28}$$

However, the denominator in (3.24) can only be factored as

$$Pr\{y_1^k, y_{k+1}^N\} = Pr\{y_{k+1}^N|y_1^k\} \cdot Pr\{y_1^k\}. \tag{3.29}$$

Therefore (3.28) is not available.

However, eventually it is the LLR rather than each $\lambda_k^i(m)$ that is wanted. Then, from the definitions in (3.25), (3.26) and (3.27), the LLR can be expressed as

$$\Lambda(d_k) = \log \frac{\sum_m \sum_{m'} \gamma_1(y_k, m', m)\beta_k(m)Pr\{S_{k-1} = m', y_1^{k-1}\}/Pr\{y_1^k\}}{\sum_m \sum_{m'} \gamma_0(y_k, m', m)\beta_k(m)Pr\{S_{k-1} = m', y_1^{k-1}\}/Pr\{y_1^k\}}. \tag{3.30}$$

Since $Pr\{y_1^k\}$ has nothing to do with the summation of m and m', we can simply replace it by $Pr\{y_1^{k-1}\}$ without effecting the result, therefore

$$\Lambda(d_k) = \log \frac{\sum_m \sum_{m'} \gamma_1(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)}. \tag{3.31}$$

2) Recursive relations of $\alpha_k(m)$ and $\beta_k(m)$:

Next, the recursive relations for calculating $\alpha_k(m)$ and $\beta_k(m)$ can be obtained as follows.

$$\alpha_k(m) = Pr\{S_k = m|y_1^k\}$$

$$
\begin{aligned}
&= \frac{Pr\{S_k = m, y_1^{k-1}, y_k\}}{Pr\{y_1^{k-1}, y_k\}} \\
&= \frac{Pr\{S_k = m, y_k | y_1^{k-1}\}}{Pr\{y_k | y_1^{k-1}\}}.
\end{aligned} \tag{3.32}
$$

Under the conditions that $S_{k-1} = m'$ and $d_k = i$, the numerator can be further expressed as

$$
\begin{aligned}
& Pr\{S_k = m, y_k | y_1^{k-1}\} \\
&= \sum_{m'} \sum_{i=0}^{1} Pr\{S_k = m, y_k, S_{k-1} = m', d_k = i | y_1^{k-1}\} \\
&= \sum_{m'} \sum_{i=0}^{1} Pr\{S_k = m, y_k, S_{k-1} = m', d_k = i, y_1^{k-1}\} / Pr\{y_1^{k-1}\} \\
&= \sum_{m'} \sum_{i=0}^{1} Pr\{S_k = m, y_k, d_k = i | S_{k-1} = m', y_1^{k-1}\} \\
& \quad \cdot Pr\{S_{k-1} = m', y_1^{k-1}\} / Pr\{y_1^{k-1}\} \\
&= \sum_{m'} \sum_{i=0}^{1} Pr\{S_k = m, y_k, d_k = i | S_{k-1} = m'\} \cdot \alpha_{k-1}(m') \\
&= \sum_{m'} \sum_{i=0}^{1} \gamma_i(y_k, m', m) \alpha_{k-1}(m').
\end{aligned} \tag{3.33}
$$

Comparing the denominator with the numerator of $\alpha_k(m)$, we have

$$
Pr\{y_k | y_1^{k-1}\} = \sum_m Pr\{S_k = m, y_k | y_1^{k-1}\}, \tag{3.34}
$$

therefore

$$
\alpha_k(m) = \frac{\sum_{m'} \sum_{i=0}^{1} \gamma_i(y_k, m', m) \alpha_{k-1}(m')}{\sum_m \sum_{m'} \sum_{i=0}^{1} \gamma_i(y_k, m', m) \alpha_{k-1}(m')}. \tag{3.35}
$$

Similarly we can get the recursive relation for $\beta_k(m)$:

$$
\beta_k(m) = \frac{\sum_{m'} \sum_{i=0}^{1} \gamma_i(y_{k+1}, m, m') \beta_{k+1}(m')}{\sum_m \sum_{m'} \sum_{i=0}^{1} \gamma_i(y_{k+1}, m', m) \alpha_k(m')}. \tag{3.36}
$$

3) Boundary conditions of $\alpha_k(m)$ and $\beta_k(m)$:

Since the first constituent encoder starts and terminates at the all-zero state, the initial and terminal conditions of the first decoder are simply

$$\alpha_0(0) = 1, \quad \text{and} \quad \alpha_0(m) = 0 \quad \text{for} \quad m \neq 0, \tag{3.37}$$

$$\beta_N(0) = 1, \quad \text{and} \quad \beta_N(m) = 0 \quad \text{for} \quad m \neq 0. \tag{3.38}$$

For the second decoder, due to interleaving, it is highly unlikely that both encoders will be terminated at the all-zero state. Because at the end of a codeword, when we use a ν-bit tail to terminate the first constituent encoder, it is almost impossible that this ν-bit tail can be permuted into exactly the ν-bit tail that is required to terminate the second constituent encoder. Therefore, when the first encoder is terminated at the all-zero state, the state of the second encoder is left undetermined. The initial and terminal conditions for the second decoder thus become

$$\alpha_0(0) = 1, \quad \text{and} \quad \alpha_0(m) = 0 \quad \text{for} \quad m \neq 0, \tag{3.39}$$

$$\beta_N(m) = 1/M \quad \text{for} \quad m = 0, 1, ..., M, \tag{3.40}$$

where M is the number of possible states of the second constituent encoder.

The ambiguity of the terminal state of the second decoder will affect the decoding, but it has been shown that when the interleaver length is large, the performance degradation is negligible [17], [15].

4) Computation of $\gamma_i(y_k, m', m)$:

As to $\gamma_i(y_k, m', m)$, it can be directly determined from

$$\begin{aligned}\gamma_i(y_k, m', m) &= Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m'\} \\ &= p(y_k | d_k = i, S_k = m, S_{k-1} = m') \\ &\quad \cdot q(d_k = i | S_k = m, S_{k-1} = m') \cdot \pi(S_k = m | S_{k-1} = m'), \end{aligned} \tag{3.41}$$

where $p(\cdot|\cdot)$ is the transition probability of the discrete memoryless channel, which can be determined by the channel characteristics. The conditional probability $q(d_k = i | S_k = m, S_{k-1} = m')$ is either 0 or 1, since given $S_{k-1} = m'$ and $S_k = m$, the input at time k is determined. In practice, $q(\cdot|\cdot)$ is always computed first. When it is 0, we do not have to calculate other probabilities. The last term $\pi(\cdot|\cdot)$ is the state transition probability. For the first decoder in the first iteration, we assume that the information bit d_k can be 0 or 1 equally likely, i.e.,

$$Pr\{d_k = 1\} = Pr\{d_k = 0\} = 1/2.$$

Since there are only two possible transitions from each state, for those transitions which satisfy $q(d_k = i | S_k = m, S_{k-1} = m') = 1$, the transition probabilities are

$$\pi(S_k = m | S_{k-1} = m') = 1/2. \tag{3.42}$$

For the second decoder, the state transition probability $\pi(\cdot|\cdot)$ is no longer simply the same as (3.42). It depends on the *a priori probability* of the information bit d_k, which can be obtained from the decision of the first decoder.

Denote the LLR of the *a priori probability* of d_k as

$$L(d_k) = \log\left(\frac{Pr\{d_k = 1\}}{Pr\{d_k = 0\}}\right). \tag{3.43}$$

Then the transition probabilities are determined by

$$\pi(S_k = m|S_{k-1} = m') = \frac{e^{L(d_k)}}{1 + e^{L(d_k)}}$$
$$\text{when} \quad q(d_k = 1|S_k = m, S_{k-1} = m') = 1, \tag{3.44}$$
$$\pi(S_k = m|S_{k-1} = m') = 1 - \frac{e^{L(d_k)}}{1 + e^{L(d_k)}}$$
$$\text{when} \quad q(d_k = 0|S_k = m, S_{k-1} = m') = 1. \tag{3.45}$$

3.4 Iterative decoding

In this section, we explain the principle of iterative decoding in accordance with a serial concatenation structure. After understanding how each decoder works by using the modified BCJR algorithm, we will see how the two decoders work cooperatively by exchanging information in an iterative way.

3.4.1 Extrinsic information

In the previous section, we have obtained the factorized form of $\Lambda(d_k)$ as in (3.31). To compute $\gamma_i(y_k, m', m)$, we need to compute three probabilities: $p(\cdot|\cdot)$, $q(\cdot|\cdot)$ and $\pi(\cdot|\cdot)$, respectively. In $p(\cdot|\cdot)$, $y_k = (y_k^s, y_k^p)$, where y_k^s is the received systematic information bit, and y_k^p is the received parity bit. Given $d_k = i, S_k = m$ and $S_{k-1} = m'$, y_k^s and y_k^p are conditionally independent, therefore

$$
\begin{aligned}
& p(y_k|d_k = i, S_k = m, S_{k-1} = m') \\
= \quad & p(y_k^s|d_k = i, S_k = m, S_{k-1} = m') \\
& \cdot p(y_k^p|d_k = i, S_k = m, S_{k-1} = m'). \qquad (3.46)
\end{aligned}
$$

Furthermore, since the encoder is systematic ($x_k^s = d_k$), given $d_k = i$, y_k^s is conditionally independent of $S_k = m$ and $S_{k-1} = m'$, thus

$$p(y_k^s|d_k = i, S_k = m, S_{k-1} = m') = p(y_k^s|d_k = i). \qquad (3.47)$$

For the first iteration, the first decoder DEC.1 computes $\Lambda_1(d_k)$ from y_k^s and y_{1k}^p. Assuming d_k can be 0 or 1 equally likely, then

$$\pi(S_k = m|S_{k-1} = m') = 1/2. \qquad (3.48)$$

Substituting (3.46), (3.47) and (3.48) into (3.41), $\Lambda(d_k)$ can be written in two parts

$$
\begin{aligned}
\Lambda_1(d_k) \quad = \quad & \log \frac{\sum_m \sum_{m'} \gamma_1'(y_{1k}^p, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0'(y_{1k}^p, m', m)\alpha_{k-1}(m')\beta_k(m)} \\
& + \log \frac{p(y_k^s|d_k = 1)}{p(y_k^s|d_k = 0)} \qquad (3.49) \\
\triangleq \quad & L_{1e}(d_k) + L_c(d_k), \qquad (3.50)
\end{aligned}
$$

where the new function $\gamma_i'(\cdot, \cdot, \cdot)$ is defined as

$$\gamma_i'(y_k^p, m', m) = p(y_k^p|d_k = i, S_k = m, S_{k-1} = m') \cdot q(d_k = i|S_k = m, S_{k-1} = m').$$

The first term in (3.50), $L_{1e}(d_k)$, is called the *extrinsic information*. The second term $L_c(d_k)$ is the LLR of the channel transition probability, called the *channel measurement* [20]. $L_c(d_k)$ is an evaluation purely based on the statistical characteristics of the channel, so it is not always reliable. The extrinsic information, however, is an additional measure of the transmitted sequence based on the parity sequence, and is essentially independent of the received systematic sequence. By adding the extrinsic information to the channel measurement, the summation can improve the decision.

3.4.2 Information exchange between the two decoders

In the discussion of computing $\gamma_i(y_k, m', m)$ in (3.42), (3.44) and (3.45), we have seen that for the second decoder, the transition probability $\pi(\cdot|\cdot)$ depends on the *a priori probability* of d_k. Substituting (3.44), (3.45) together with (3.46) and (3.47) into (3.41) yields

$$\begin{aligned}\Lambda_2(d_k) &= \log \frac{\sum_m \sum_{m'} \gamma_1'(y_{2k}^p, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0'(y_{2k}^p, m', m)\alpha_{k-1}(m')\beta_k(m)} \\ &\quad + L(d_k) + \log \frac{p(y_k^s|d_k = 1)}{p(y_k^s|d_k = 0)} \qquad (3.51) \\ &\equiv L_{2e}(d_k) + L(d_k) + L_c(d_k). \qquad (3.52)\end{aligned}$$

Therefore, the LLR of the second decoder consists of three parts. In addition to the extrinsic information and the channel measurement, there is also a term $L(d_k)$, which is the LLR of the *a priori probability* of d_k.

Now, if $\Lambda_1(d_k)$ is passed on to the second decoder as the LLR of the *a priori probability*, we will have

$$\begin{aligned}\Lambda_2(d_k) &= L_{2e}(d_k) + \Lambda_1(d_k) + L_c(d_k) \quad (3.53)\\ &= L_{2e}(d_k) + L_{1e}(d_k) + 2 \cdot L_c(d_k). \quad (3.54)\end{aligned}$$

Since $L_c(d_k)$ is not always reliable, if its value is doubled, the extrinsic information will be less significant in correcting the error, and thus a wrong decision may be made. Furthermore, if the output of DEC.2 is fed back into DEC.1, when iterations are going on, $L_c(d_k)$ will keep on accumulating, and eventually be the dominant factor. Then the contribution of the extrinsic information will be drastically devalued. To avoid this, the channel measurement $L_c(d_k)$ should not be passed from one decoder to the other. Therefore

$$\Lambda_2(d_k) = L_{2e}(d_k) + L_{1e}(d_k) + L_c(d_k). \quad (3.55)$$

A fundamental principle for feeding back information to another decoder is: never feed a decoder with information that stems from itself, because the input and output corruption will be highly correlated [20]. Therefore, among the three terms in 3.55, only $L_{2e}(d_k)$ is passed on to the first decoder through the feedback loop. Then for the next iteration, the first decoder uses L_{2e} as the *a priori* LLR to evaluate $\Lambda_1(d_k)$, and passes L_{1e} on to the second decoder, and so on. This can be illustrated in Fig. 3.4.

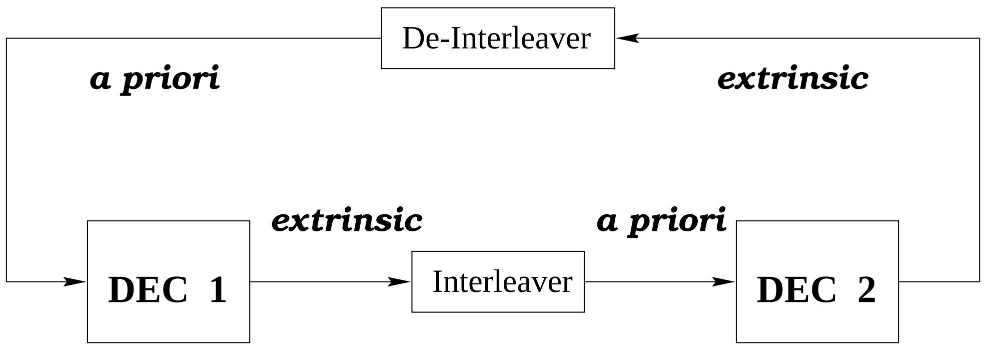

Figure 3.4: Information exchange between the two decoders.

Therefore, between the two elementary decoders, it is only the extrinsic information that is passed on to the other decoder, and it acts as the *a priori* LLR of the other decoder.

By exchanging the extrinsic information between the two decoders in an iterative manner, the performance can be greatly improved. However, the most significant coding gain is achieved by the first several iterations. And the improvement decreases quickly when the number of iterations increases. Therefore, in practice, we can stop the iterations when the improvement is negligible.

Chapter 4

Turbo codes for different channels

So far, we have studied the structure of the encoder and the decoder of Turbo codes. Next we examine the application of Turbo codes to the following channels: the binary symmetric channel (BSC), the additive white Gaussian noise (AWGN) channel, and the Rayleigh fading channel. For the Rayleigh fading channel, we discuss two cases: one is that the *side information (SI)* is known to the decoder, the other is that the side information is unknown (NSI).

For each channel, the decoder of Turbo codes needs to be modified to incorporate the appropriate channel statistics. In the Modified BCJR algorithm, this corresponds to formulating the probability function $\gamma_i(y_k, m', m)$. Also, in order to see how good the performance of Turbo codes is, we need to calculate the Shannon limit for each channel and for different rates. Throughout this chapter, the modulation is assumed to be coherent binary phase shift keying (BPSK).

4.1 Binary symmetric channel

4.1.1 Channel model

The binary symmetric channel is a special case of a discrete memoryless channel (DMC); i.e., it has discrete inputs and outputs, the output depends only on the input at that time and is conditionally independent of previous channel inputs or outputs [7]. As shown in Fig. 4.1, at the input of the channel, when a "0" is transmitted, it can be received as a "1" with probability ϵ, or still be "0" with probability $1 - \epsilon$. Since it is symmetric, the same holds when the input is "1".

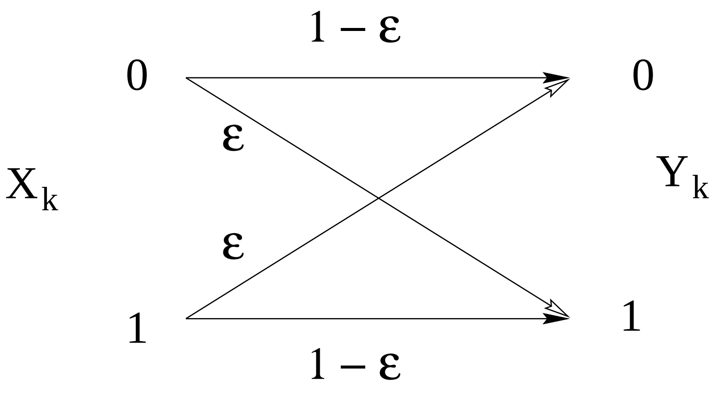

Figure 4.1: Binary symmetric channel.

4.1.2 Modifications in the Turbo decoder

In the previous chapter, we have obtained:

$$
\begin{aligned}
\gamma_i(y_k, m', m) = p(y_k | d_k = i, S_k = m, S_{k-1} = m') \\
\cdot q(d_k = i | S_k = m, S_{k-1} = m') \cdot \pi(S_k = m | S_{k-1} = m'). \\
= \quad p(y_k^s | d_k = i) \cdot p(y_k^p | d_k = i, S_k = m, S_{k-1} = m') \\
\cdot q(d_k = i | S_k = m, S_{k-1} = m') \cdot \pi(S_k = m | S_{k-1} = m').
\end{aligned}
\tag{4.1}
$$

For the BSC, the conditional probability of the information bit d_k is

$$
p(y_k^s | d_k = i) = \begin{cases} 1 - \epsilon & \text{if} \quad y_k^s = 0 \quad i = 0; \\ \epsilon & \text{if} \quad y_k^s = 1 \quad i = 0; \\ 1 - \epsilon & \text{if} \quad y_k^s = 1 \quad i = 1; \\ \epsilon & \text{if} \quad y_k^s = 0 \quad i = 1. \end{cases} \tag{4.2}
$$

For an undeleted parity bit, the conditions $(d_k = i, S_k = m, S_{k-1} = m')$ can determine the parity output of the encoder; i.e., for some certain $j \in \{0, 1\}$,

$$
p(y_k^p | d_k = i, S_k = m, S_{k-1} = m') = Pr\{y_k^p | x_k^p = j\}, \tag{4.3}
$$

and the same results hold as in (4.2).

When the rate is greater than 1/3, puncturing is used. Then in the decoder, some parity bits are not available. For these deleted parity bits, we set their values as another number different from 0 and 1. For example, set all the deleted parity bits as 2. Then whichever the original parity bits are deleted, the conditional probability

$$p(y_k^p = 2|d_k = i, S_k = m, S_{k-1} = m') = 1. \tag{4.4}$$

4.1.3 Shannon limit for the BSC

The capacity of BSC can be easily derived.

$$\begin{aligned} C &= \max_{p_X(x)} I(X;Y) \\ &= \max_{p_X(x)} \{H(Y) - H(Y|X)\} \\ &= \max_{p_X(x)} \{H(Y) - \sum_x p(x) \cdot H(Y|X=x)\} \\ &= 1 - h_b(\epsilon), \end{aligned} \tag{4.5}$$

where $h_b(\cdot)$ is the binary entropy function. The last step follows since Y is a binary random variable, and the maximization is achieved only when the inputs are uniformly distributed.

As stated in Chapter 1, the Shannon limit on ϵ for a given code rate R_c and a tolerable BER P_e can be obtained by solving

$$R_c \cdot [1 - h_b(P_e)] = C = 1 - h_b(\epsilon). \tag{4.6}$$

The optimum values of ϵ for different rates are shown in Table 4.1. The values in this table and also in the following sections are obtained via a C code.

Rate	$P_e = 0$	$P_e = 10^{-5}$	$P_e = 10^{-4}$	$P_e = 10^{-3}$	$P_e = 10^{-2}$
1/3	0.173952	0.173979	0.174171	0.175651	0.186268
1/2	0.110028	0.110058	0.110272	0.111928	0.123875
2/3	0.061490	0.061521	0.061740	0.063437	0.075798
4/5	0.031124	0.031154	0.031362	0.032981	0.044971
8/9	0.014793	0.014820	0.015010	0.016490	0.027740

Table 4.1: Optimum values of ϵ for different rates (BSC).

Rate	$P_e = 0$	$P_e = 10^{-5}$	$P_e = 10^{-4}$	$P_e = 10^{-3}$	$P_e = 10^{-2}$
1/3	1.212	1.210	1.202	1.150	0.077
1/2	1.775	1.772	1.763	1.703	1.258
2/3	2.516	2.513	2.503	2.423	1.882
4/5	3.369	3.367	3.354	3.250	2.547
8/9	4.254	4.250	4.227	4.080	3.145

Table 4.2: Optimum values of E_b/N_0 (dB) for different rates (BSC).

Also, to make it consistent with the other channels, we transfer the value of ϵ into the equivalent value of E_b/N_0 (for an AWGN channel with hard decision BPSK modulation) given by

$$\epsilon = Q\left(\sqrt{\frac{2R_cE_b}{N_0}}\right), \tag{4.7}$$

where $Q(\cdot)$ is the complimentary error function. The equivalent optimum values of E_b/N_0 in dB are shown in Table 4.2.

4.2 Additive white Gaussian noise channel

4.2.1 Channel model

The additive white Gaussian noise (AWGN) channel is a commonly used model for communication systems. The signals are distorted by the noise which follows a Gaussian distribution. Denote the output of the channel as Y_k, the input as X_k, then

$$Y_k = X_k + Z_k, \tag{4.8}$$

where Z_k is a zero-mean Gaussian noise, described by its p.d.f.

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{z^2}{2\sigma^2}}. \tag{4.9}$$

The variance is often given in the form as the power spectral density $N_0/2$, where

$$\sigma^2 = N_0/2.$$

The noise Z_k and the signal X_k are assumed to be independent. By using coherent BPSK, the signal X_k is modulated as $\pm\sqrt{E_s}$. Mainly, we are concerned about the signal-to-noise ratio (SNR), E_s/N_0, or E_b/N_0, where $E_s = R_c \cdot E_b$ [16].

4.2.2 Modifications in the Turbo decoder

When Turbo codes are used for the AWGN channel, in the γ function

$$
\begin{aligned}
\gamma_i(y_k, m', m) &= p(y_k^s | d_k = i) \\
&\cdot p(y_k^p | d_k = i, S_k = m, S_{k-1} = m') \\
&\cdot q(d_k = i | S_k = m, S_{k-1} = m') \cdot \pi(S_k = m | S_{k-1} = m'),
\end{aligned} \tag{4.10}
$$

the conditional probability $p(y_k^s | d_k = i)$ follows a Gaussian distribution,

$$
p(y_k^s | d_k = i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k^s - (2i-1))^2}{2\sigma^2}}. \tag{4.11}
$$

Then in the expression of $\Lambda(d_k)$,

$$
L_c(d_k) = \frac{2}{\sigma^2} \cdot y_k^s.
$$

Also for the undeleted parity bits, we can use $(d_k = i, S_k = m, S_{k-1} = m')$ to determine $x_k^p = j$ for some $j \in \{0, 1\}$, and the conditional probability is

$$
\begin{aligned}
& p(y_k^p | d_k = i, S_k = m, S_{k-1} = m') \\
&= Pr\{y_k^p | x_k^p = j\} \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k^p - (2j-1))^2}{2\sigma^2}}.
\end{aligned} \tag{4.12}
$$

For the deleted parity bits, it is reasonable to set their values to 0, and still use (4.12) to perform decoding.

4.2.3 Shannon limit for the AWGN channel with binary input

In practice, we adjust the value of E_s to achieve the required SNR. In the analysis, for the sake of simplicity, we let $E_s = 1$ without loss of generality.

The capacity formula of the optimum base-band Gaussian channel [7] given by

$$C = \frac{1}{2}\log_2(1 + 2R_c\frac{E_b}{N_0}) \tag{4.13}$$

does not apply here because this can only be achieved when the input is Gaussian. Here we are restricted to binary inputs: either 1 or -1, so we have to derive the capacity for this case.

The capacity of the binary input AWGN channel is

$$\begin{aligned} C &= \max_{p_X(x)} I(X;Y) & (4.14)\\ &= \max_{p_X(x)}\{H(X) - H(X|Y)\}. & (4.15)\end{aligned}$$

Since the channel input has only two possible values ± 1, and we know that the capacity can be achieved (i.e., the mutual information can be maximized) only when the channel inputs are equiprobable, i.e., $p_X(x = +1) = p_X(x = -1) = 1/2$, then

$$H(X) = -\sum_x p(x)\cdot\log_2 p(x) = 1.$$

The conditional entropy $H(X|Y)$ can be calculated by using Baye's rule:

$$\begin{aligned}
H(X|Y) &= -\sum_x \int_y p(x,y)\log_2 p(x|y)dy && (4.16)\\
&= -\sum_x \int_y p(y|x)\cdot p_X(x)\cdot \log_2 \frac{p(y|x)p_X(x)}{p(y)}dy && (4.17)\\
&= -\int_y p(y|+1)\cdot\frac{1}{2}\cdot\log_2\left[\frac{p(y|+1)(1/2)}{\sum_{x'} p(y|x')\cdot p_X(x')}\right]dy\\
&\quad -\int_y p(y|-1)\cdot\frac{1}{2}\cdot\log_2\left[\frac{p(y|-1)(1/2)}{\sum_{x'} p(y|x')\cdot p_X(x')}\right]dy && (4.18)\\
&\equiv C_1 + C_{-1}. && (4.19)
\end{aligned}$$

It can be verified (by symmetry) that $C_1 = C_{-1}$. Hence

$$\begin{aligned}
H(X|Y) &= -\int_y p(y|+1)\cdot\log_2\left[\frac{p(y|+1)}{\sum_{x'} p(y|x')}\right]dy\\
&= \int_y p(y|+1)\cdot\log_2\left[\frac{p(y|+1)+p(y|-1)}{p(y|+1)}\right]dy\\
&= \int_y p(y|+1)\cdot\log_2\left[1+e^{-\frac{2y}{\sigma^2}}\right]dy.
\end{aligned}$$

Therefore, the capacity of AWGN channel with binary input is

$$\begin{aligned}
C &= \max_{p_X(x)}\{H(X)-H(X|Y)\} && (4.20)\\
&= 1-\int_y p(y|+1)\cdot\log_2\left[1+e^{-\frac{2y}{\sigma^2}}\right]dy. && (4.21)
\end{aligned}$$

In (4.21), the variance $\sigma^2 = N_0/2$, and $E_s = R_c \cdot E_b$, so the capacity C is a function of E_b/N_0. Combining (4.6) with (4.21), the optimum value of E_b/N_0 can be determined for different P_e. The form is very complicated in terms of E_b/N_0 and can only be solved numerically. We give the optimum values of E_b/N_0 in Table 4.3, which is obtained via numerical integration.

Rate	$P_e = 0$	$P_e = 10^{-5}$	$P_e = 10^{-4}$	$P_e = 10^{-3}$	$P_e = 10^{-2}$
1/3	-0.495	-0.496	-0.504	-0.559	-0.960
1/2	0.187	0.185	0.177	0.111	-0.357
2/3	1.06	1.057	1.047	0.963	0.382
4/5	2.04	2.038	2.0229	1.909	1.152
8/9	3.033	3.030	3.008	2.844	1.833

Table 4.3: Optimum values of E_b/N_0 (dB) for different rates (AWGN channel).

4.3 Rayleigh fading channel with side information

4.3.1 Channel model

In many applications of wireless communications, the channels are more complicated than the AWGN channel. One effect is known as fading, which is a result of the dispersion of transmitted signals in the channel. For example, in a wireless communication system, even without any distortion due to the noise, the signal will become weaker and weaker due to multi-path propagation, atmospheric absorption, etc.

Here we use the model of Rayleigh slowly fading channel [16]. At time k, the relation between the channel input X_k and the output Y_k is given by

$$Y_k = A_k \cdot X_k + Z_k,$$

where $X_k = \pm\sqrt{E_s}$. Here we also set $E_s = 1$, then $X_k = \pm 1$. The term Z_k is an additive white Gaussian noise with zero mean and variance $\sigma^2 = N_0/2$. The term A_k is a random variable following the Rayleigh distribution

$$p_A(a) = 2ae^{-a^2}, \qquad a \geq 0.$$

Knowledge of A_k is called the *channel side information*, also referred to as the *channel gain*. Here the channel gain is scaled to have a mean-square value of $E[A^2] = 1$ [11].

In this section, we assume that the channel side information A_k is known to the decoder.

4.3.2 Modifications in the Turbo decoder

For Rayleigh fading channel, when the side information can be detected, the γ function in the decoder should also depend on a_k^s and a_k^p, where a_k^s is the side information of the systematic bit y_k^s, and a_k^p is the side information of the parity bit y_k^p, respectively. Then the γ function is modified as

$$\begin{aligned} \gamma_i(y_k^s, y_k^p, a_k^s, a_k^p, m', m) = Pr\{y_k^s | d_k = i, a_k^s\} \\ \cdot p(y_k^p | d_k = i, a_k^p, S_k = m, S_{k-1} = m') \\ \cdot q(d_k = i | S_k = m, S_{k-1} = m') \cdot \pi(S_k = m | S_{k-1} = m'). \end{aligned} \tag{4.22}$$

Given a_k^s, along with $d_k = i$, the channel output y_k^s is a Gaussian random variable with mean $a_k^s(2i-1)$, and variance $N_0/2$. Therefore,

$$Pr\{y_k^s | d_k = i, a_k^s\} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k^s - a_k^s(2i-1))^2}{2\sigma^2}}. \tag{4.23}$$

Similarly for the undeleted parity bits

$$
\begin{aligned}
& p(y_k^p | d_k = i, a_k^p, S_k = m, S_{k-1} = m') \\
= \ & Pr\{y_k^p | x_k^p = j, a_k^p\} \\
= \ & \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_k^p - a_k^p(2j-1))^2}{2\sigma^2}},
\end{aligned} \tag{4.24}
$$

for some certain j.

For all the deleted parity bits, we set $y_k^p = 0$, and perform decoding by (4.24).

4.3.3 Shannon limit for the Rayleigh fading channel with SI

The capacity of the Rayleigh fading channel with side information is [11]

$$
\begin{aligned}
C_{SI} &= \max_{p_X(x)} \{I(X;Y|A)\} \\
&= \max_{p_X(x)} \left\{ E_p(x,y,a) \left[\log_2 \left(\frac{p(x,y|a)}{p(x|a)p(y|a)} \right) \right] \right\} \\
&= \max_{p_X(x)} \left\{ E_p(x,y,a) \left[\log_2 \left(\frac{p(y|x,a)}{p(y|a)} \right) \right] \right\} \\
&= \max_{p_X(x)} \left\{ E_p(x,y,a) \left[\log_2 \left(\frac{p(y|x,a)}{\sum_{x'} p_X(x') \cdot p(y|x',a)} \right) \right] \right\},
\end{aligned} \tag{4.25}
$$

where $I(x, y|a)$ is the mutual information between x and y conditioned on the knowledge of the side information a. Step three follows from the conditional probability rule, and the last step is based on the law of total probability.

Since A and X are independent, the p.d.f. of $p(x, y, a)$ can be factored as

$$p(x, y, a) = p(y|x, a) \cdot p_X(x) \cdot p_A(a).$$

Similar to the AWGN channel in the previous section, the capacity can be achieved only when the inputs are uniform, we have

$$\begin{aligned}
C_{SI} &= \sum_x \int_a \int_y 1/2 \cdot p_A(a) \cdot p(y|x,a) \cdot \log_2 \left[\frac{p(y|x,a)}{\sum_{x'} 1/2 \cdot p(y|x',a)} \right] dy\, da \\
&\equiv C_1 + C_{-1} \\
&= \int_a \int_y p_A(a) \cdot p(y|+1,a) \cdot \log_2 \left[\frac{p(y|+1,a)}{\sum_{x'} 1/2 \cdot p(y|x',a)} \right] dy\, da, \qquad (4.26)
\end{aligned}$$

where the last step follows from the fact that the integration with respect to a does not effect the symmetry of C_1 and C_{-1}.

The log part can be simplified as :

$$\log_2 \left[\frac{p(y|+1,a)}{\sum_{x'} 1/2 \cdot p(y|x',a)} \right] = -\log_2[1/2 \cdot (1 + \Psi(y,a))], \qquad (4.27)$$

where $\Psi(y,a)$ is defined as

$$\begin{aligned}
\Psi(y,a) &\triangleq \frac{p(y|x=-1,a)}{p(y|x=+1,a)} \\
&= e^{-2ay/\sigma^2}. \qquad (4.28)
\end{aligned}$$

Therefore, the capacity can be written in the form

$$\begin{aligned}
C_{SI} &= -\int_a \int_y p_A(a) \cdot p(y|+1,a) \cdot \log_2[1/2 \cdot (1 + \Psi(y,a))] dy\, da \\
&= 1 - \int_a \int_y p_A(a) \cdot p(y|+1,a) \cdot \log_2[1 + \Psi(y,a)] dy\, da. \qquad (4.29)
\end{aligned}$$

The optimum values of E_b/N_0 are also solved numerically. The values are shown in Table 4.4 for reference.

Rate	$P_e = 0$	$P_e = 10^{-5}$	$P_e = 10^{-4}$	$P_e = 10^{-3}$	$P_e = 10^{-2}$
1/3	0.489	0.487	0.479	0.412	-0.066
1/2	1.830	1.829	1.817	1.729	1.107
2/3	3.667	3.664	3.647	3.516	2.627
4/5	5.936	5.932	5.904	5.690	4.331
8/9	8.520	8.512	8.462	8.087	5.964

Table 4.4: Optimum values of E_b/N_0 (dB) for different rates (Rayleigh with SI).

4.4 Rayleigh fading channel without side information (NSI)

4.4.1 Channel model

In the previous section, the side information is assumed to be known to the decoder. We obtained the channel capacity by taking advantage of the fact that when the side information is known, the channel output follows a Gaussian distribution. However, in many applications, the side information is unknown to the decoder. This makes a considerable difference both in the Shannon limit and the performance of the system [11]. In this section, we discuss the case of the Rayleigh fading channel without side information.

4.4.2 Modifications in the Turbo decoder

For Rayleigh fading channel without side information, the γ function is still written in the form

$$
\begin{aligned}
\gamma_i(y_k, m', m) &= p(y_k^s|d_k = i) \\
&\cdot p(y_k^p|d_k = i, S_k = m, S_{k-1} = m') \\
&\cdot q(d_k = i|S_k = m, S_{k-1} = m') \cdot \pi(S_k = m|S_{k-1} = m'), \qquad (4.30)
\end{aligned}
$$

but the probabilities are different. To formulate $p(y_k^s|d_k = i)$, we use the law of total probability

$$
\begin{aligned}
p(y_k^s|d_k = i) &= \int_0^\infty p_A(a)p(y_k^s|d_k = i, a)da \\
&= \int_0^\infty 2ae^{-a^2}\left[\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y_k^s - a(2i-1))^2/2\sigma^2}\right]da. \qquad (4.31)
\end{aligned}
$$

This integral also can only be done numerically due to the lack of a closed form. In the modified BCJR algorithm, we have to do numerical integration for each bit, then the iterative decoding process will be slowed down dramatically. In [9], an approximation is proposed to avoid numerical integration. We consider the region Γ in which a_k^s is most probable. Within this region, we replace the random variable a_k^s by its expectation $E[A]$. Since the Rayleigh distribution function decreases drastically, outside this region, a_k^s is approximately 0. Therefore

$$
\begin{aligned}
p(y_k^s|d_k = i) &\approx \int_\Gamma 2ae^{-a^2}\left[\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y_k^s - E[A](2i-1))^2/2\sigma^2}\right]da \qquad (4.32) \\
&\approx \frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y_k^s - E[A](2i-1))^2/2\sigma^2}, \qquad (4.33)
\end{aligned}
$$

which means that $p(y_k^s|d_k = i)$ is approximately Gaussian in the region Γ.

Similarly, for an undeleted parity bit,

$$p(y_k^p|d_k = i, S_k = m, S_{k-1} = m') \approx \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_k^p - E[A](2j-1))^2/2\sigma^2} \tag{4.34}$$

for some j. When the channel gain has a mean-square value of $E[A^2] = 1$, it has been shown that the expected value $E[A] = 0.8862$ [11]. All the deleted parity bits are set to be 0 same as before.

4.4.3 Shannon limit for the Rayleigh fading channel (NSI)

When the side information A is not available, the capacity of Rayleigh fading channel is

$$\begin{aligned} C_{NSI} &= \max_{p_X(x)} \{I(X;Y)\} \\ &= \max_{p_X(x)} \left\{ E_{p(x,y)} \left[\log_2 \left(\frac{p(x,y)}{p_X(x) \cdot p(y)} \right) \right] \right\} \\ &= \max_{p_X(x)} \left\{ E_{p(x,y)} \left[\log_2 \left(\frac{p(y|x)}{p(y)} \right) \right] \right\}. \end{aligned} \tag{4.35}$$

Same as before, the capacity can be achieved only when the channel inputs are uniform. Here the p.d.f. is $p(x, y)$, which can be calculated by

$$\begin{aligned} p(x,y) &= \int_a p(x,y,a) da \\ &= \int_a p_A(a) \cdot p_X(x) \cdot p(y|x,a) da \\ &= \int_a \frac{1}{2} \cdot 2ae^{-a^2} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-ax)^2}{2\sigma^2}} da. \end{aligned} \tag{4.36}$$

Because of symmetry, the two integrations of $x = +1$ and $x = -1$ in (4.35) are equal, therefore

$$C_{NSI} = \int_a \int_y p_A(a) \cdot p(y|x = +1, a) \cdot \log_2 \left(\frac{p(y|x = +1)}{p(y)} \right) dy\, da.$$

Furthermore, the log part can also be simplified as

$$\begin{aligned} \log_2 \left(\frac{p(y| + 1)}{p(y)} \right) &= \log_2 \left(\frac{p(y| + 1)}{1/2(p(y| + 1) + p(y| - 1))} \right) &(4.37) \\ &= -\log_2 \left[1/2 \left(1 + \Phi(y) \right) \right], &(4.38) \end{aligned}$$

where $\Phi(y)$ is defined as

$$\begin{aligned} \Phi(y) &= \frac{p(y|x = -1)}{p(y|x = +1)} &(4.39) \\ &= \frac{\int_a p_A(a) \cdot p(y|x = -1, a) da}{\int_a p_A(a) \cdot p(y|x = +1, a) da}. &(4.40) \end{aligned}$$

Therefore, the capacity is

$$C_{NSI} = 1 - \int_a \int_y p_A(a) \cdot p(y|x = +1, a) \cdot \log_2 (1 + \Phi(y)) dy\, da. \quad (4.41)$$

The corresponding optimum values of E_b/N_0 are shown in Table 4.5.

Rate	$P_e = 0$	$P_e = 10^{-5}$	$P_e = 10^{-4}$	$P_e = 10^{-3}$	$P_e = 10^{-2}$
1/3	1.291	1.290	1.281	1.216	0.748
1/2	2.567	2.565	2.553	2.466	1.857
2/3	4.347	4.345	4.328	4.198	3.325
4/5	6.576	6.572	6.544	6.332	4.990
8/9	9.145	9.138	9.087	8.715	6.609

Table 4.5: Optimum values of E_b/N_0 (dB) for different rates (Rayleigh NSI).

Chapter 5

Simulation results

In this chapter, we present our simulation results on the effects of interleaving, sequence length and the number of iterations, as well as the investigation on the applications of Turbo codes of different rates for different channels.

5.1 Simulation environment

In our simulations, the constituent encoders are chosen to be rate 1/2, 16-state RSC's with generator (37, 21). For most of our simulations, the sequence length N is set to be 65536, since this offers the best performance. The interleaver is a pseudo-random one, whose performance is superior to that of a uniform interleaver. The number of iterations is six. For the investigations on different channels, the rates are: 1/3, 1/2, 2/3, 4/5 and 8/9. All the results are plotted as BER versus E_b/N_0 (in dB).

Due to the limitation of computation, we use 20 blocks for most of the simulations with $N = 65536$, which can offer a reliable bit error rate of 10^{-4} within 130 errors. Therefore, the comparisons are made at the level of 10^{-4}. For the investigation on the effect of iterations, 80 blocks are used, which can guarantee a reliable BER at the level of 10^{-5} within 52 errors.

Our simulation program is written in C. For the binary symmetric channel, an incorrect transmission can occur with the crossover probability ϵ. This is realized by

$$y = \begin{cases} x & \text{if} \quad \text{drand48()} \geq \epsilon, \\ (x+1) \mod 2 & \text{if} \quad \text{drand48()} < \epsilon. \end{cases}$$

where "drand48()" is a function which returns a floating-point value uniformly distributed over the interval [0.0, 1.0). For the AWGN channel, the Gaussian noise n is obtained via [14]

$$n = \sqrt{-2\sigma^2 \cdot \ln(U_1)} \cdot \cos(2\pi U_2), \tag{5.1}$$

where the two independent random variables U_1 and U_2 are uniformly distributed over [0.0, 1.0) and are generated by "drand48()".

For the Rayleigh fading channel, the Rayleigh random variable A with a mean-square value of $E[A^2] = 1$ is generated by [14]

$$A = \sqrt{-\ln(U)}, \tag{5.2}$$

where U is uniform over [0.0, 1.0) generated by "drand48()".

5.2 Effect of interleaving

In the Turbo encoding part, we have discussed the importance of the interleaver. To illustrate the effect of interleaving, we performed simulations with a uniform interleaver and a pseudo-random interleaver. In comparison, we also performed simulation without using an interleaver. In this case, the whole encoder is equivalent to a single constituent encoder when the rate is 1/2.

From Fig. 5.1 to 5.3 we can see that the performance is improved by interleaving, and a pseudo-random interleaver can do better than a uniform one. At the first iteration, as shown in Fig. 5.1, at the BER level of 10^{-4}, a uniform interleaver can improve the performance by 0.849 dB, and a pseudo-random interleaver can improve it by 2.358 dB. While the iteration is going on, the effect is further enhanced. As shown in Fig. 5.3, at the sixth iteration, a uniform interleaver can improve the performance by 2.028 dB at the BER level of 10^{-4}, while a pseudo-random interleaver can make a significant improvement of 5.094 dB.

5.3 Effect of sequence length

The sequence length N is an important factor in the performance of Turbo codes with a pseudo-random interleaver. Fig. 5.4 to 5.8 show the performance when the length N equals to 256, 1024, 4096, 16384 and 65536, respectively.

When $N = 256$, 2.479 dB of E_b/N_0 can achieve the BER level of 10^{-4} at the sixth iteration. This is 2.302 dB away from the Shannon limit 0.177 dB. When $N = 65536$, at the sixth iteration, the BER of 10^{-4} can be achieved at 1.015 dB, which is only 0.838 dB away from the Shannon limit. Therefore, increasing the length N improves the performance.

We can also notice that when N increases, decoding with the same number of iterations becomes more effective in improving the performance. This can be compared at the BER level of 10^{-4}. When $N = 256$, the difference between the second and the sixth iteration is only 0.458 dB. However, when $N = 65536$, the difference is enlarged to 1.176 dB.

To make all the simulations test the same number of bits, the number of blocks is changed according to the length N. When $N = 256$, we use 4096 blocks; when $N = 65536$, 16 blocks are used. Therefore, the total number of bits is $256 \times 4096 = 1048576$, which can guarantee a reliable bit error rate of 10^{-4} within 104 errors.

5.4 Effect of iterations

In Chapter 3, we have seen that Turbo decoding is implemented in an iterative manner. The effect of the first six iterations is already shown in Fig. 5.1 to Fig. 5.8. In this section, we present the simulation result on the effect of 18 iterations, as shown in Fig. 5.9.

From the first iteration to the 18^{th} iteration, the performance is greatly improved. The most significant coding gain is achieved in the first six iterations. We can see that the performance curve at the first iteration is incomparable to those after six iterations. The BER level of 10^{-4} is achieved at nearly 1 dB with only 6 iterations. At the 18^{th} iteration, our simulation shows that when $E_b/N_0 = 0.75$ dB, the bit error rate is 1.26×10^{-5}. This is not as good as the result of Berrou *et al.* in [4], in which the BER level of 10^{-5} achieved at 0.7 dB is reported. This is probably due to the lack of sufficient number of blocks we tested. In the simulation of Berrou *et al.*, 256 blocks were used while $N = 65536$. Altogether there is 1.67×10^7 bits, which can guarantee a reliable bit error rate of 10^{-5} within 167 errors. In our simulation, the input sequence length $N = 65536$ is the same. But due to the limitation of computation, only 80 blocks are used. This 5.2×10^6 bits in total can guarantee the BER level of 10^{-5} to be reliable within 52 errors.

In Fig. 5.9, we found that when the value of E_b/N_0 is higher than 0.75 dB, the curve of the 18^{th} iteration becomes flat. This can be explained by the performance bound we introduced in Chapter 2.

When SNR is moderate or high, the bit error rate performance is dominated by the free-distance term in the union bound due to the drastic decreasing of the Q function. Thus at moderate and high SNR, the performance bound asymptotically approaches:

$$P_b \approx \frac{N_{free}\tilde{w}_{free}}{N} Q\left(\sqrt{d_{free}\frac{2R_c E_b}{N_0}}\right), \tag{5.3}$$

which is called the *free-distance asymptote*. By the algorithm in [18], it was found that when $N = 65536$, $d_{free} = 6$, $\tilde{w}_{free} = 2$, $N_{free} = 3$ [15]. Thus the free-distance asymptote is:

$$P_{free} = \frac{3 \times 2}{65536} Q\left(\sqrt{6\frac{E_b}{N_0}}\right). \tag{5.4}$$

The comparison of the performance curve of the 18^{th} iteration and the free-distance asymptote is shown in Fig. 5.10.

5.5 Turbo codes for different channels

In this section, we present our simulation results on the four different channels discussed in Chapter 4 with different rates: 1/3, 1/2, 2/3, 4/5 and 8/9. All the simulations use $N = 65536$ with 20 blocks.

For different channels and for different rates, the optimum values of E_b/N_0 to reach the Shannon limit change correspondingly. Since we are mainly concerned about the region in which the performance is improved most significantly by iterative decoding, the starting points of E_b/N_0 are adjusted according to those optimum values.

5.5.1 Turbo codes for the BSC

In the simulations on the BSC, the points are selected by the values of the crossover probabilities, ϵ : 0.1, 0.08, 0.06, etc. According to the optimum values of ϵ to reach the Shannon limit presented in Table 4.1, when the rate is 1/3, we choose ϵ to be: 0.18, 0.16, 0.14, 0.12 and 0.1; for rate equal to 1/2, we choose ϵ to be: 0.12, 0.1, 0.08, 0.06 and 0.04; ...; for the highest rate 8/9, we performed simulations with ϵ : 0.04, 0.02, 0.01, 0.008, 0.006, 0.005, 0.004, 0.003, 0.002 and 0.001. To be consistent with the other BPSK-modulated channels, we convert the values of ϵ into E_b/N_0 (in dB) and plot BER versus E_b/N_0 (in dB).

From Fig. 5.11 to 5.15 we can see that when the rate increases, the value of E_b/N_0 needed to reach the BER level of 10^{-4} also increases. However, its increment is bigger than the increment of the Shannon limit. At the sixth iteration, when rate=1/3, to get the BER of 10^{-4}, only 2.195 dB is needed, which is 0.993 dB away from the Shannon limit of 1.202 dB. When the rate is increased to 8/9, the value of E_b/N_0 has to be 5.667 dB, which is 1.440 dB away from the limit of 4.227 dB. Therefore, with the same number of iterations, when the rate is high, the performance curve is not as close to the Shannon limit as that when the rate is low.

Fig. 5.11 to 5.15 also show that when the rate increases, iterations become less effective in improving the performance. This can be clearly seen in the difference between the fourth and the sixth iteration. When the rate is 1/3, a coding gain of 0.339 dB at BER of 10^{-4} can be obtained from the fourth iteration to the sixth iteration. This improvement decreases drastically as the rate increases. When the rate is 8/9, the improvement from the fourth iteration to the sixth iteration is negligible. Also apparent is the difference between the first iteration and the sixth iteration. When the rate is 8/9, at the BER level of 10^{-4}, a coding gain of 1.233 dB is obtained from the first to the six iteration, which cannot be compared to the effect when the rate is 1/3.

5.5.2 Turbo codes for the AWGN channel

For the AWGN channel, as shown in Table 4.3, the Shannon limit at the BER level of 10^{-4} increases from -0.504 dB to 3.008 dB when the rate increases from 1/3 to 8/9. Therefore, for different rates, we vary the region of E_b/N_0 for simulations correspondingly.

We found that for the fourth and the sixth iteration, the curves drop abruptly within a very short interval of E_b/N_0. To present the performance more accurately, we performed simulations on some additional points in such regions. For example, when the rate is 1/3, the performance curve of the sixth iteration is very steep when E_b/N_0 is between 0.3 dB and 0.4 dB; therefore, the values of E_b/N_0 we choose are: -0.25, 0, 0.25, 0.3, 0.35, 0.375, 0.4, 0.6, 0.8 and 1.25 (dB).

From Fig. 5.16 to 5.20 we can also draw the following two conclusions:

1) Increasing the rate results in a higher E_b/N_0 away from the limit to reach a desired BER level. For example, when the rate is 1/3, 0.345 dB of E_b/N_0 can achieve the BER level of 10^{-4}, which is only 0.849 dB away from the limit -0.504 dB. When the rate is 8/9, 5.522 dB is needed, which is 2.514 dB away from the limit 3.008 dB. Therefore, increasing the rate from 1/3 to 8/9 results in a degradation of 1.665 dB.

2) The effectiveness of iterations decreases greatly when the rate is high. This is clearly shown in Fig. 5.16 to 5.20. We can see that when the rate is 1/3, each iteration makes a great contribution to the improvement of the performance. When the rate increases, this effect becomes weaker and weaker. When the rate is 8/9, from the first iteration to the sixth iteration, there is only 0.311 dB improvement at the BER level of 10^{-4}, and the difference between the fourth and the sixth iteration is hard to distinguish.

5.5.3 Turbo codes for the Rayleigh fading channel (with SI)

For the Rayleigh fading channel, we found that when the rate is 1/3, 1/2, the performance looks similar to that of BSC or AWGN channel, but a decrease of the slope can be noticed. For example, when the rate is 1/3, to reach the BER level of 10^{-4} from the level of 10^{-2}, 0.293 dB is needed. However, for the AWGN channel, only 0.128 dB is needed. When the rate is high, for example, 4/5 and 8/9, the performance is quite different from that of BSC or AWGN channel. We can see an apparent decrease of the slope when the rate is 4/5 or 8/9.

Also, the value of E_b/N_0 is increased to reach a desired BER level due to the increment of the limit. When the rate is 1/3, to achieve BER of 10^{-4}, the value of E_b/N_0 is about 1.464 dB, which is 0.985 dB away from the limit of 0.479 dB. When the rate is 8/9, the value of E_b/N_0 is increased to 9.890 dB, which means that 1.428 dB greater than the limit (8.462 dB) is needed to achieve the BER of 10^{-4}. The coding gain between the fourth and the sixth iteration does not change too much: at the BER level of 10^{-5}, it is 0.406 dB when the rate is 1/3, and it becomes 0.339 dB when the rate is 8/9. (it may appear to be bigger in Fig. 5.21 due to a bigger scale.)

5.5.4 Turbo codes for the Rayleigh fading channel (NSI)

The performance of Turbo codes for Rayleigh fading channel (NSI) shown in Fig. 5.26 to 5.30 is similar to that for Rayleigh fading channel with side information. In general, when the side information is not available, the performance becomes worse.

When the rate is 1/3, $E_b/N_0 = 2.402dB$ can achieve the BER level of 10^{-4}, which is 1.121 dB from the Shannon limit of 1.281 dB. When the rate is 8/9, to reach the BER level of 10^{-4}, E_b/N_0 has to be 11.634 dB. The distance to the Shannon limit (9.087 dB) then becomes 2.547 dB. Therefore, when the rate is 8/9, lack of side information causes an additional 1.119 dB degradation.

Also remarkable is the decrease of the slope in Fig. 5.29 and 5.30. To reach the BER level of 10^{-4} from the level of 10^{-2}, in Fig. 5.25, 0.847 dB is needed with the aid of the side information, while in Fig. 5.30, we need 1.829 dB when the side information is not available.

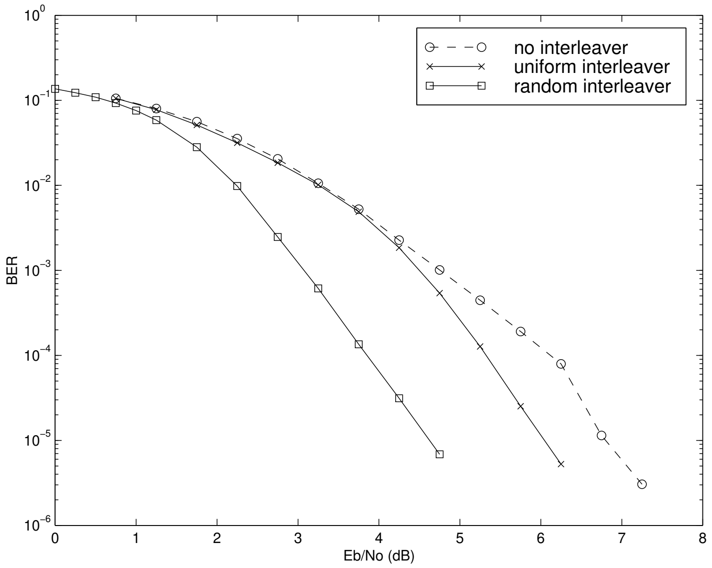

Figure 5.1: Effect of interleaving on the first iteration, (37, 21) encoder, N=65536, 20 blocks, rate=1/2, AWGN channel.

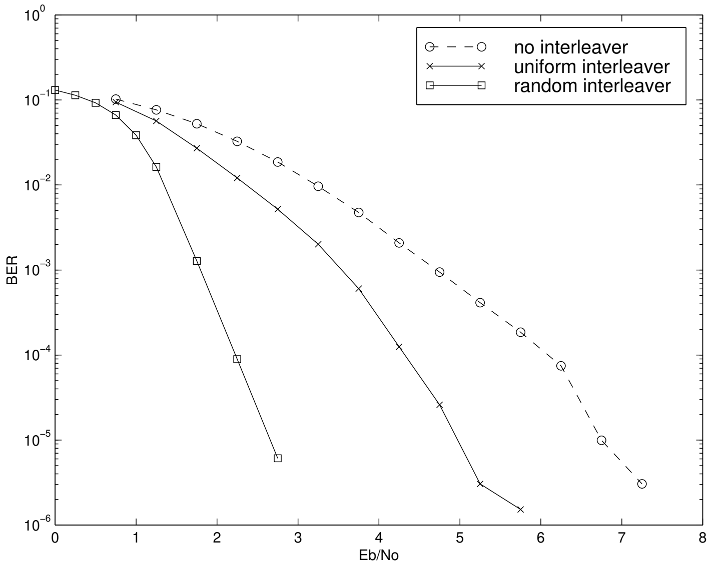

Figure 5.2: Effect of interleaving on the second iteration, (37, 21) encoder, N=65536, 20 blocks, rate=1/2, AWGN channel.

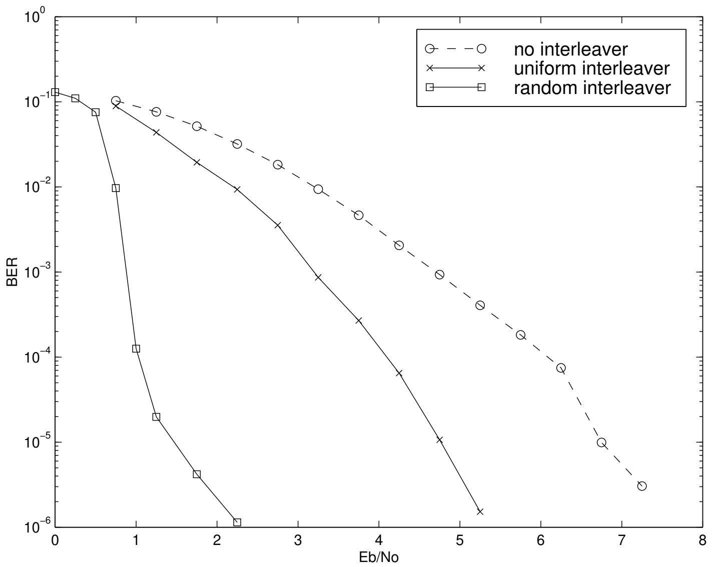

Figure 5.3: Effect of interleaving on the sixth iteration, (37, 21) encoder, N=65536, 20 blocks, rate=1/2, AWGN channel.

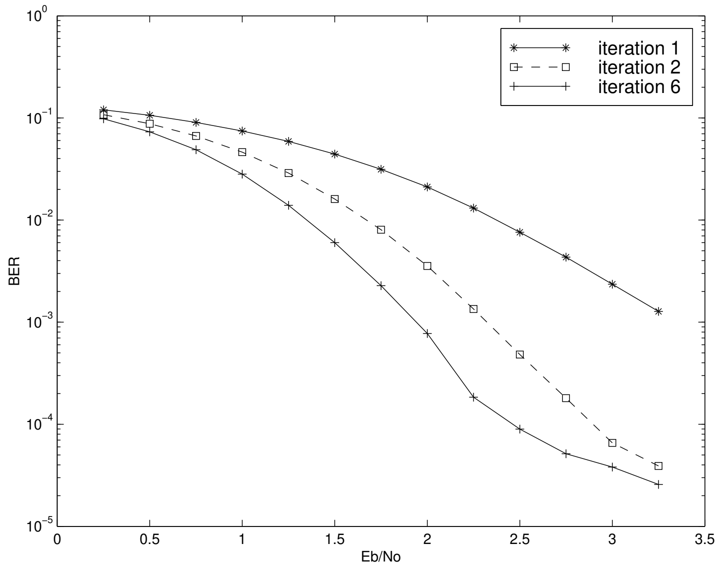

Figure 5.4: Sequence length $N = 16 \times 16 = 256$, 4096 blocks, (37, 21) encoder, rate=1/2, random interleaver, AWGN channel.

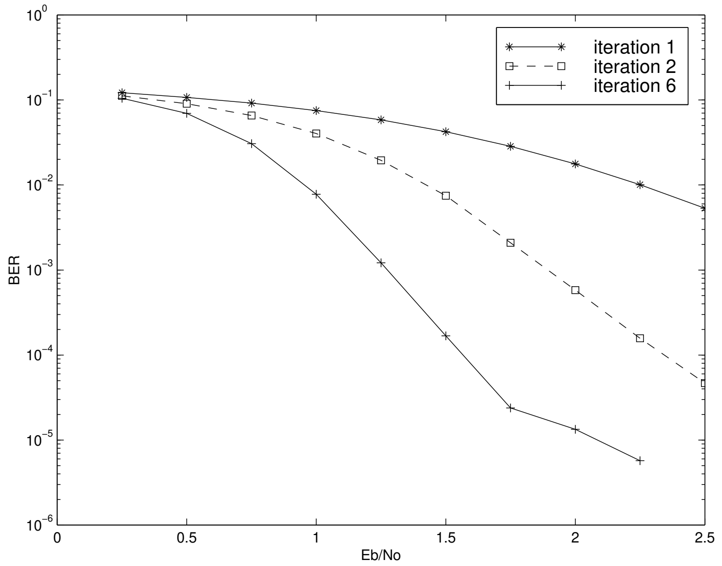

Figure 5.5: Sequence length $N = 32 \times 32 = 1024$, 1024 blocks, (37, 21) encoder, rate=1/2, random interleaver, AWGN channel.

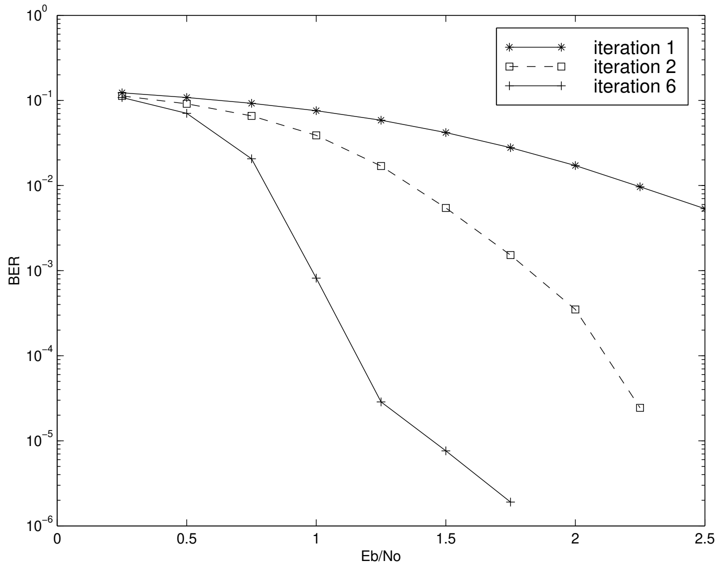

Figure 5.6: Sequence length $N = 64 \times 64 = 4096$, 256 blocks, (37, 21) encoder, rate=1/2, random interleaver, AWGN channel.

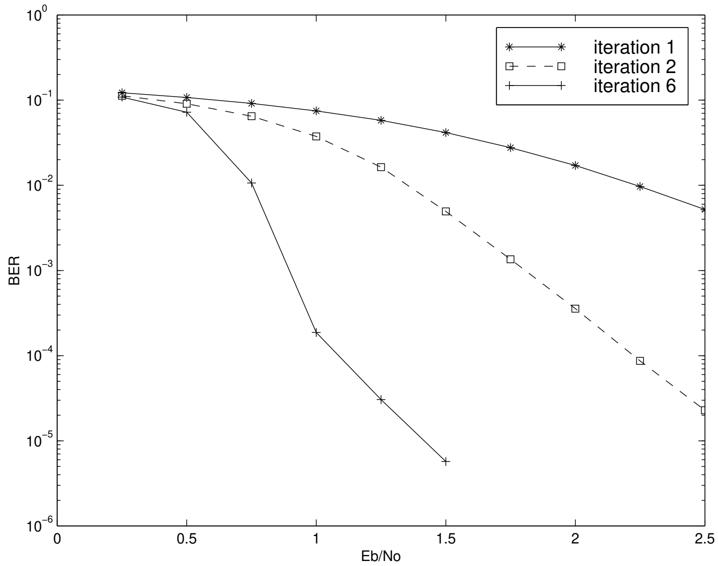

Figure 5.7: Sequence length $N = 128 \times 128 = 16384$, 64 blocks, (37, 21) encoder, rate=1/2, random interleaver, AWGN channel.

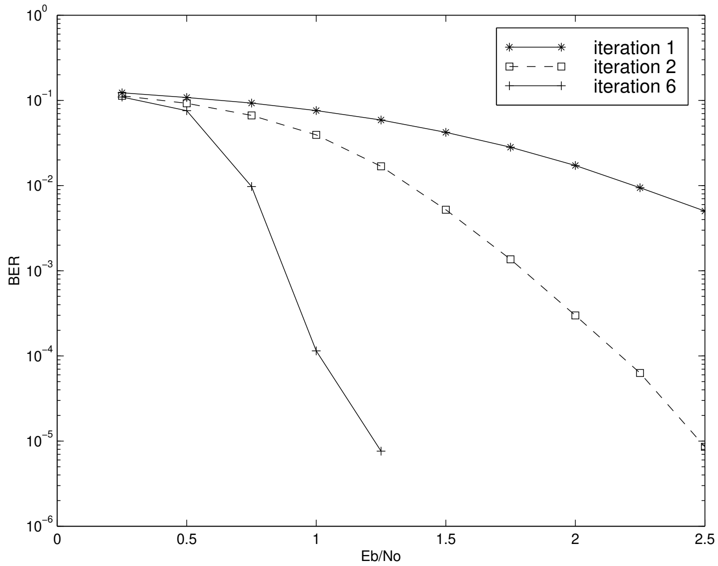

Figure 5.8: Sequence length $N = 256 \times 256 = 65536$, 16 blocks, (37, 21) encoder, rate=1/2, random interleaver, AWGN channel.

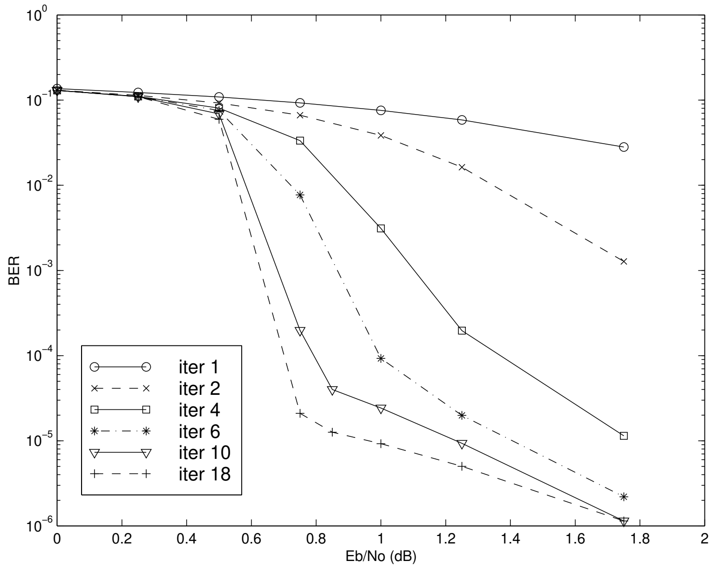

Figure 5.9: Effect of iterations, (37, 21), random interleaver, rate=1/2, N=65536, 80 blocks, AWGN channel.

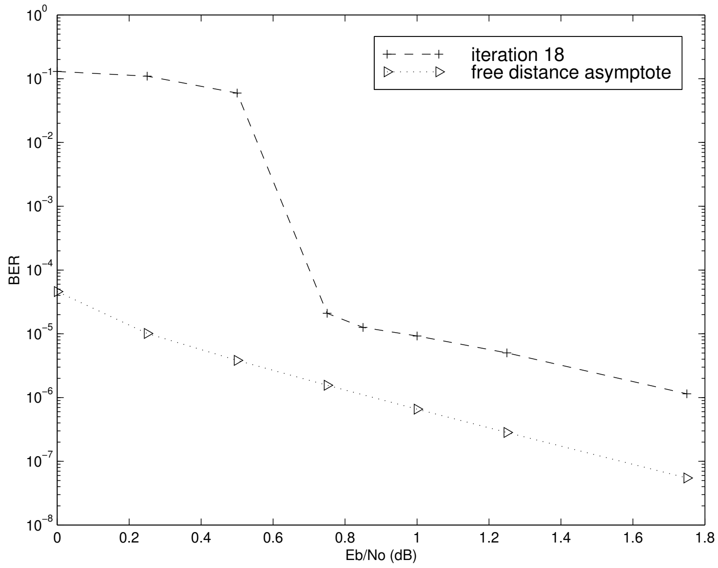

Figure 5.10: Free-distance asymptote, (37, 21), random interleaver, rate=1/2, N=65536, 80 blocks, AWGN channel.

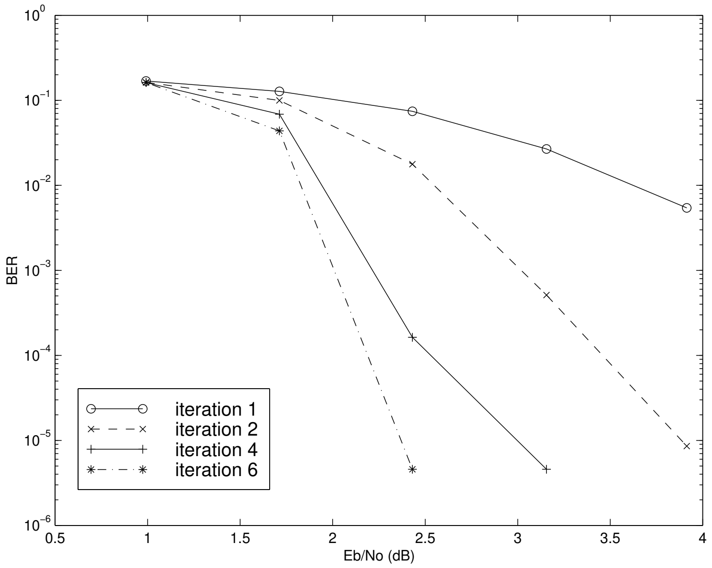

Figure 5.11: Turbo codes for BSC, (37, 21), random interleaver, rate=1/3, N=65536, 20 blocks.

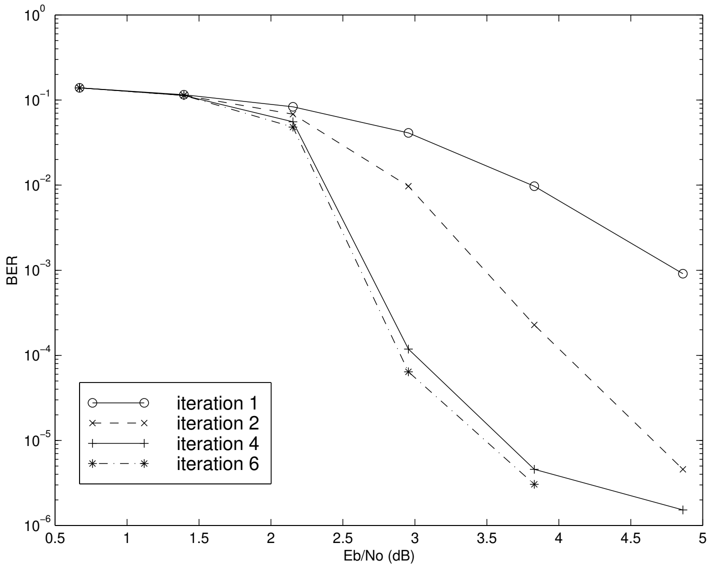

Figure 5.12: Turbo codes for BSC, (37, 21), random interleaver, rate=1/2, N=65536, 20 blocks.

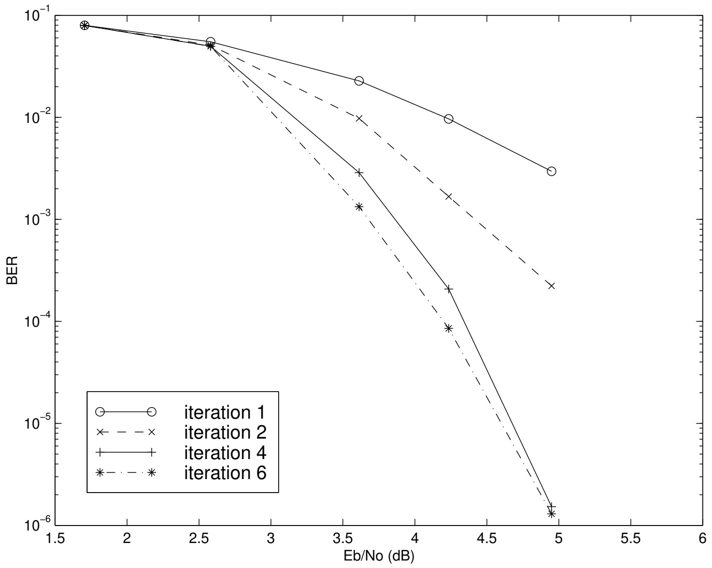

Figure 5.13: Turbo codes for BSC, (37, 21), random interleaver, rate=2/3, N=65536, 20 blocks.

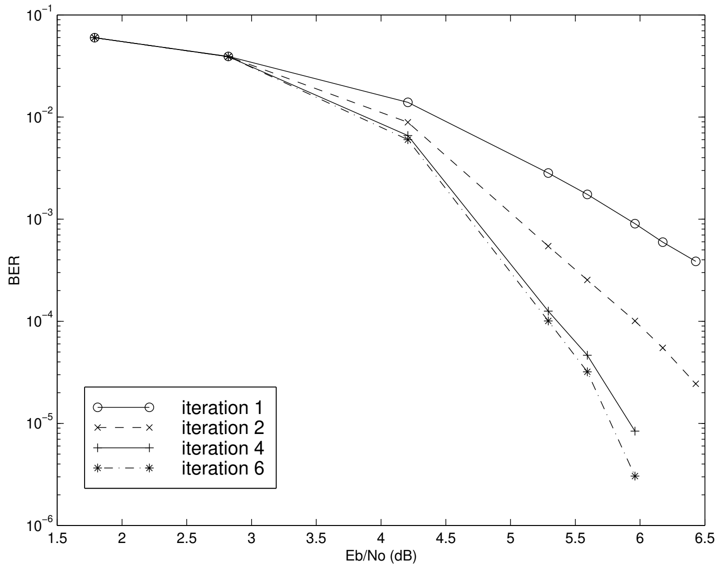

Figure 5.14: Turbo codes for BSC, (37, 21), random interleaver, rate=4/5, N=65536, 20 blocks.

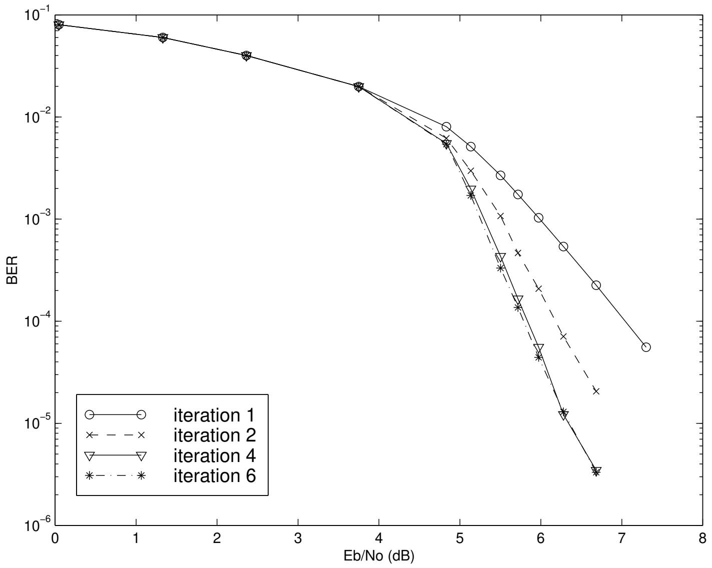

Figure 5.15: Turbo codes for BSC, (37, 21), random interleaver, rate=8/9, N=65536, 20 blocks.

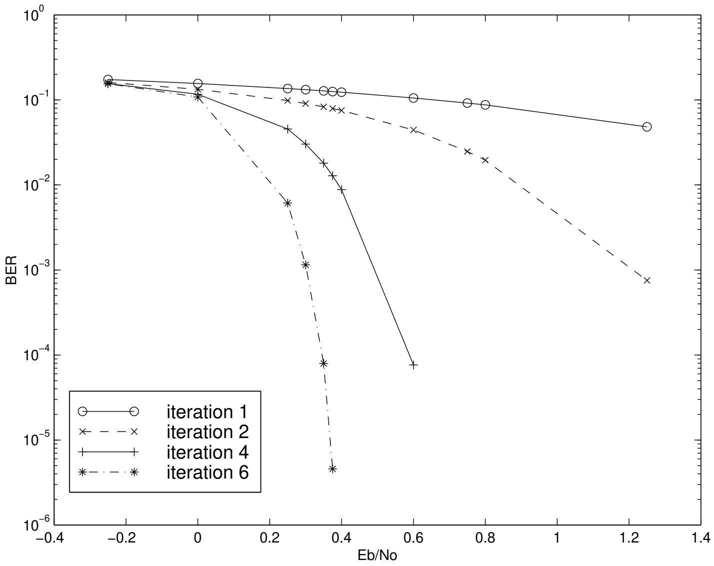

Figure 5.16: Turbo codes for AWGN channel, (37, 21), random interleaver, rate=1/3, N=65536, 20 blocks.

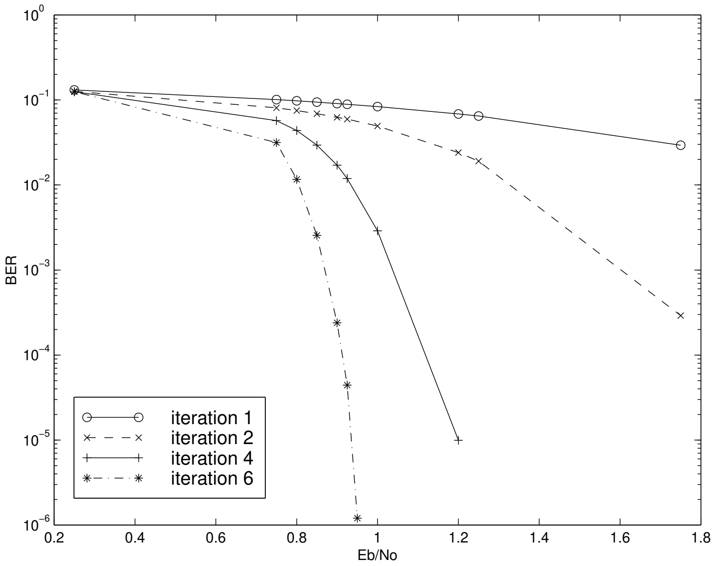

Figure 5.17: Turbo codes for AWGN channel, (37, 21), random interleaver, rate=1/2, N=65536, 20 blocks.

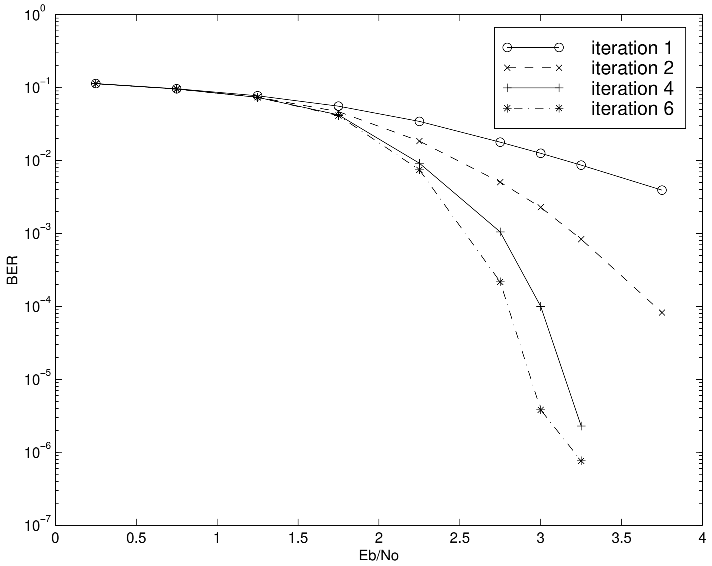

Figure 5.18: Turbo codes for AWGN channel, (37, 21), random interleaver, rate=2/3, N=65536, 20 blocks.

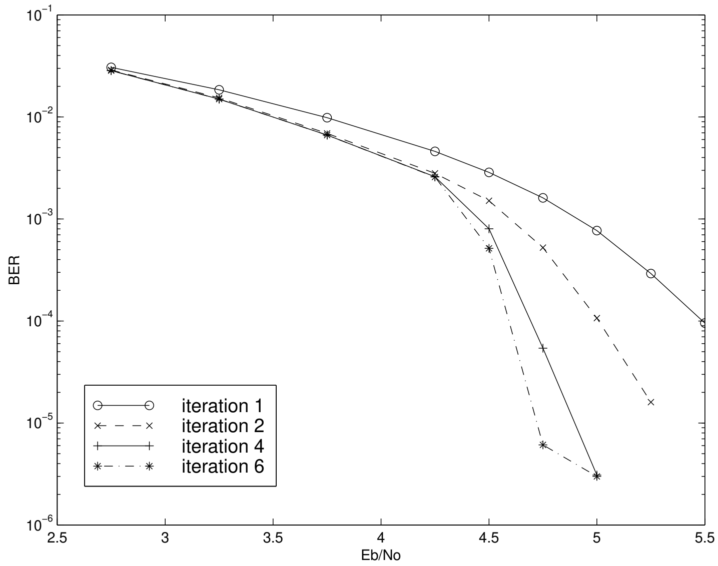

Figure 5.19: Turbo codes for AWGN channel, (37, 21), random interleaver, rate=4/5, N=65536, 20 blocks.

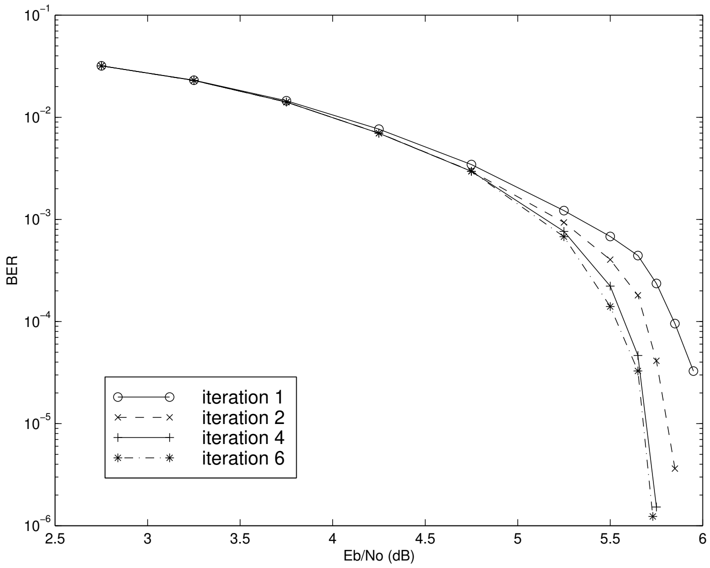

Figure 5.20: Turbo codes for AWGN channel, (37, 21), random interleaver, rate=8/9, N=65536, 20 blocks.

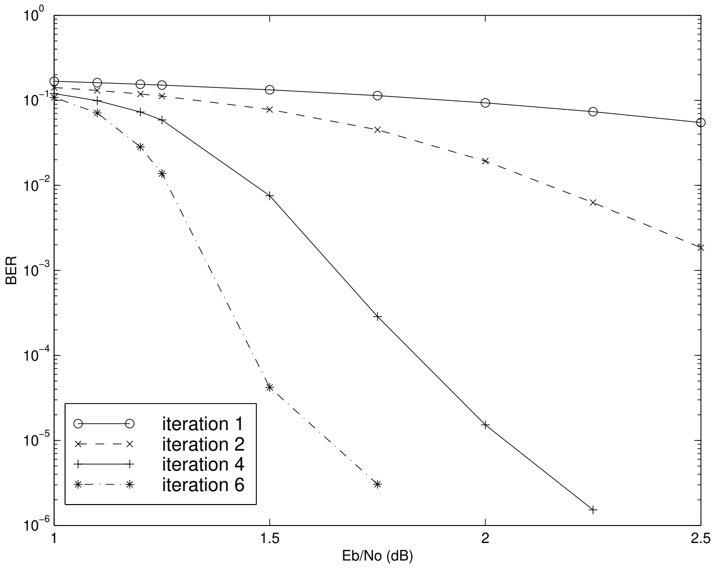

Figure 5.21: Turbo codes for Rayleigh fading channel (SI), (37, 21), random interleaver, rate=1/3, N=65536, 20 blocks.

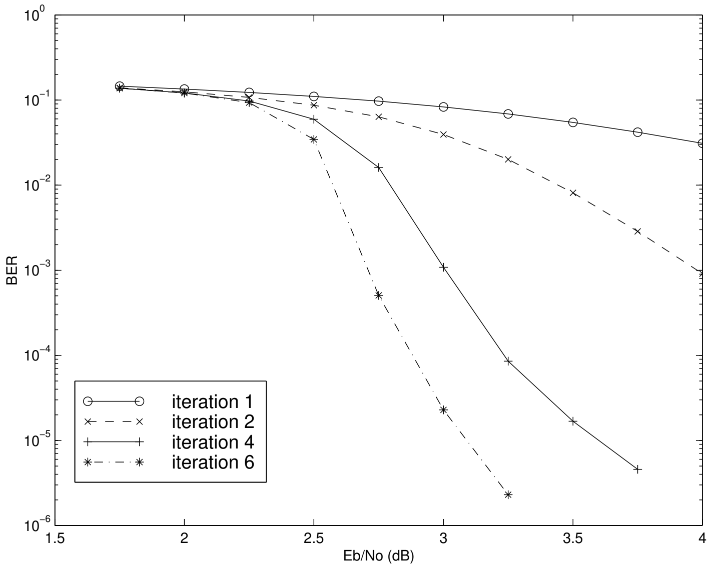

Figure 5.22: Turbo codes for Rayleigh fading channel (SI), (37, 21), random interleaver, rate=1/2, N=65536, 20 blocks.

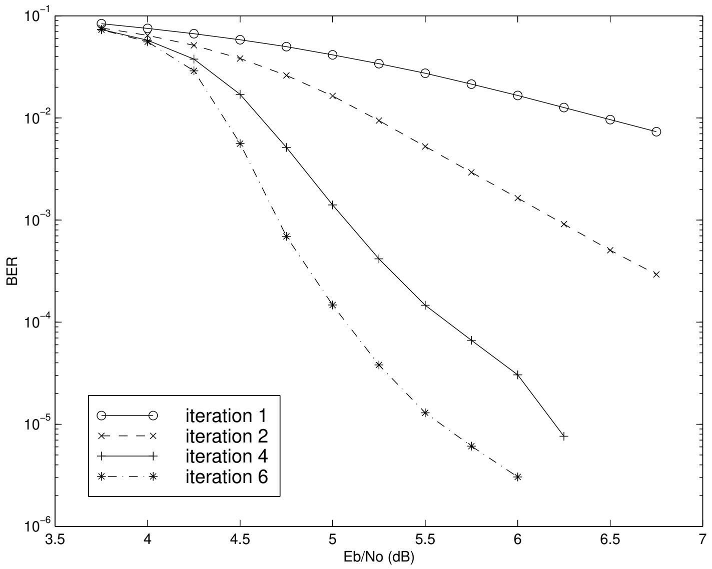

Figure 5.23: Turbo codes for Rayleigh fading channel (SI), (37, 21), random interleaver, rate=2/3, N=65536, 20 blocks.

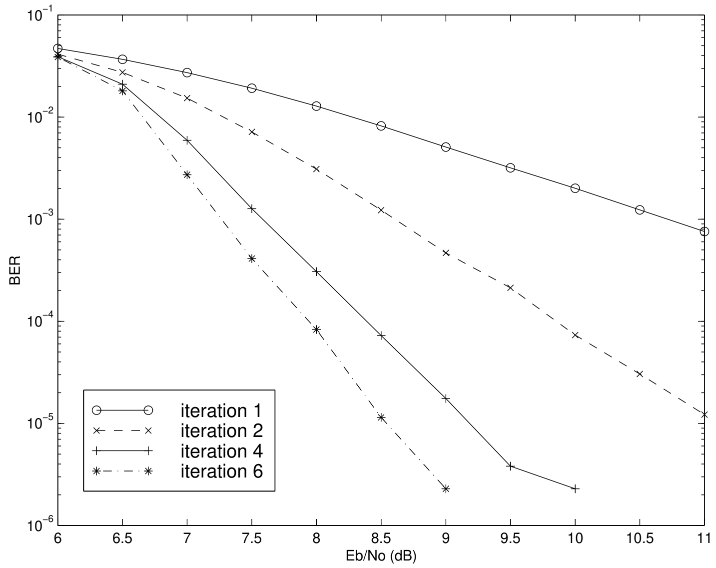

Figure 5.24: Turbo codes for Rayleigh fading channel (SI), (37, 21), random interleaver, rate=4/5, N=65536, 20 blocks.

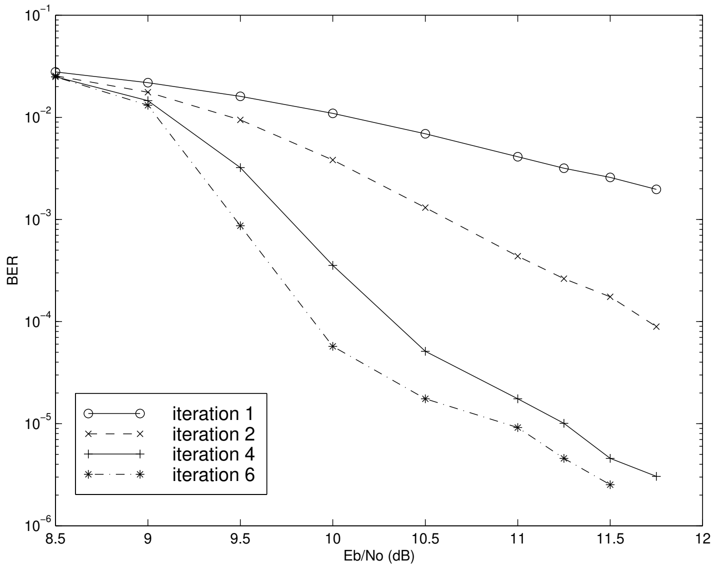

Figure 5.25: Turbo codes for Rayleigh fading channel (SI), (37, 21), random interleaver, rate=8/9, N=65536, 20 blocks.

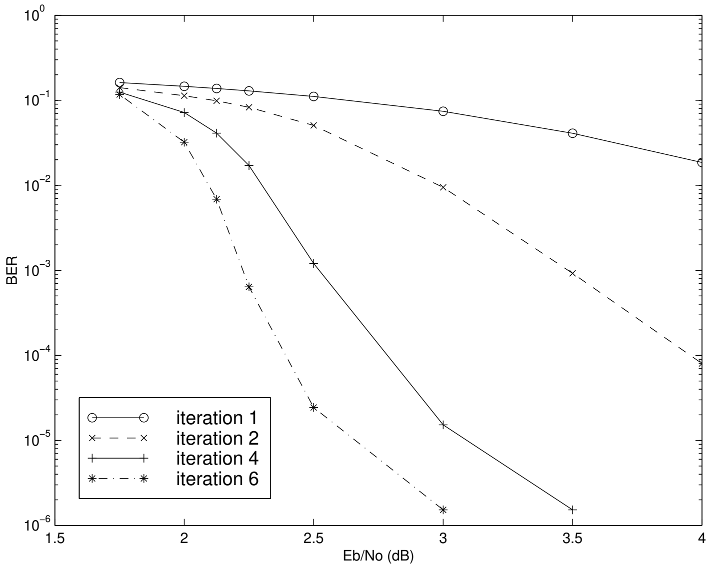

Figure 5.26: Turbo codes for Rayleigh fading channel (NSI), (37, 21), random interleaver, rate=1/3, N=65536, 20 blocks.

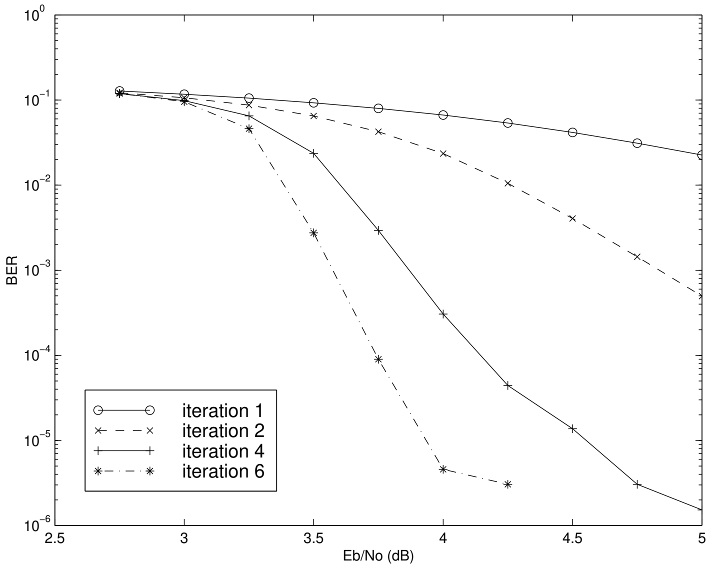

Figure 5.27: Turbo codes for Rayleigh fading channel (NSI), (37, 21), random interleaver, rate=1/2, N=65536, 20 blocks.

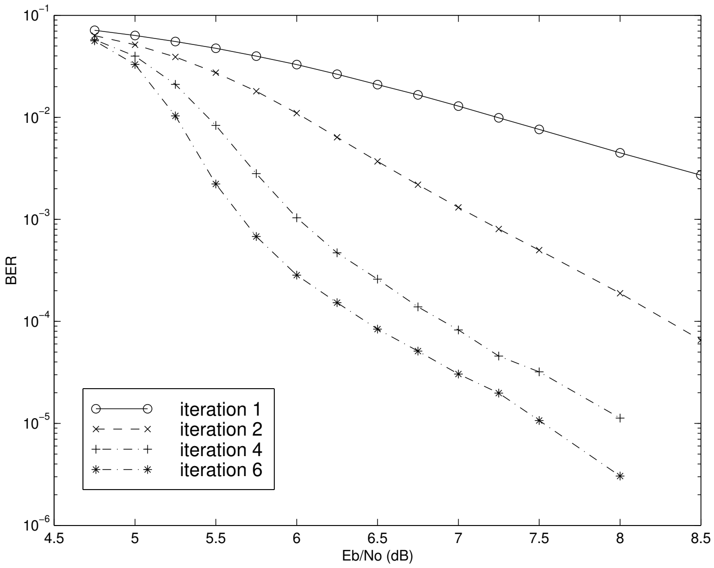

Figure 5.28: Turbo codes for Rayleigh fading channel (NSI), (37, 21), random interleaver, rate=2/3, N=65536, 20 blocks.

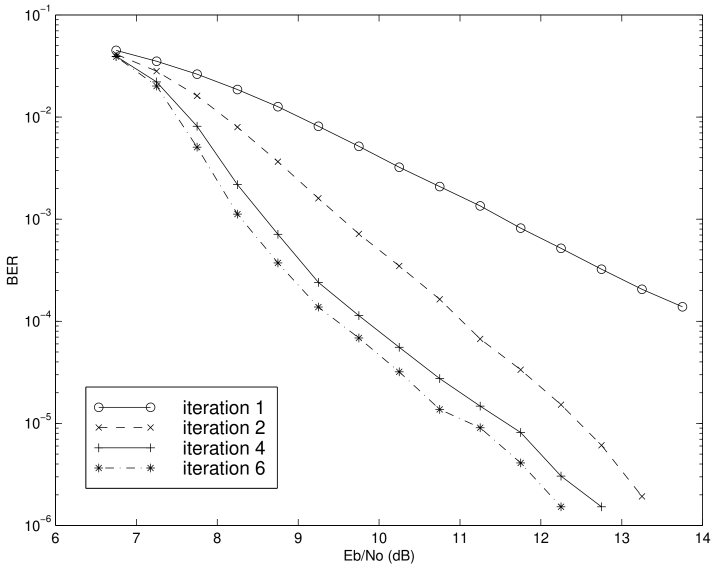

Figure 5.29: Turbo codes for Rayleigh fading channel (NSI), (37, 21), random interleaver, rate=4/5, N=65536, 20 blocks.

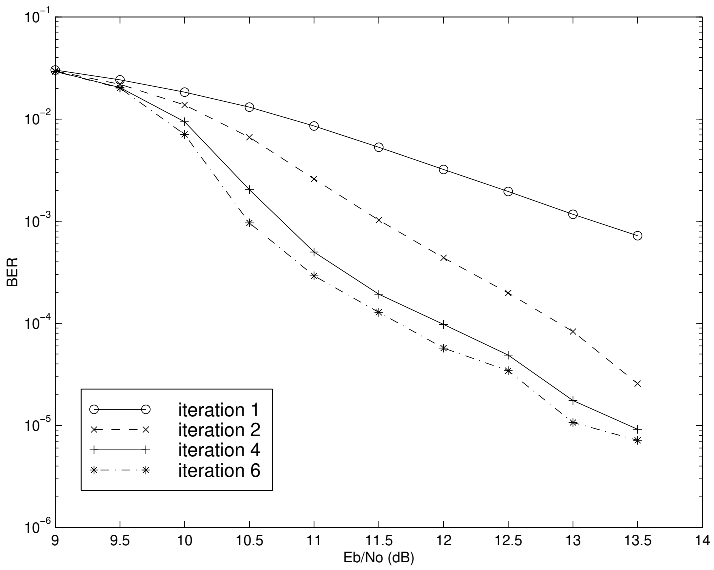

Figure 5.30: Turbo codes for Rayleigh fading channel (NSI), (37, 21), random interleaver, rate=8/9, N=65536, 20 blocks.

Chapter 6

Conclusions and future work

6.1 Conclusions of the project

The structure of the Turbo encoder is described along with a discussion on a performance bound in Chapter 2. In Chapter 3, the structure of the Turbo decoder and the principle of iterative decoding are studied. The modified BCJR algorithm for Turbo decoding is analyzed in comparison with the original BCJR algorithm.

After the discussions on the Turbo encoding and decoding, as well as the application for different channels in Chapter 4, our simulation results are presented in Chapter 5. The effects of the sequence length, interleaving, and the number of iterations are examined on the AWGN channel with rate 1/2. For each channel (BSC, AWGN channel, Rayleigh fading channel), the performance of Turbo codes with rates from

1/3 up to 8/9 is investigated, Different degradation phenomena are observed when the rate increases.

In general, a higher E_b/N_0 away from the Shannon limit is needed to achieve a desired BER level as the rate increases. This is common for all the three channels.

For the BSC and the AWGN channel, the most significant degradation is the decrease of the coding gain improved by iterations when the rate increases. For the Rayleigh fading channel, the most significant degradation is the decrease of the slope of the performance curve. The lack of side information exacerbates the degradation.

6.2 Future work

As an active area, there are many aspects on Turbo codes awaiting for further research, but we would like to consider the following topics in the future.

1) In this project, we found that when the rate increases, the performance becomes worse in terms of the distance to the Shannon limit. We believe the reason lies in the fact that the redundancy becomes inadequate for correcting errors when the puncturing rate is too high. Alternatively, we are interested in re-designing the Turbo encoder using high-rate constituent encoders and discard the puncturing device. We would like to perform simulations with such a Turbo encoder and investigate the performance.

2) Up to now, there are no rigorous theories to explain the extraordinary performance of Turbo codes. Since Turbo codes are generated by a linear system [15], one approach we are interested in is: using the theory of linear systems to analyze Turbo codes. This could provide a new angle to look at the Turbo codes.

3) While theoretical approaches are attempted, Turbo codes seem to have a promising future for various applications. We would like to investigate the application of Turbo codes for the transmission of compressed images.

Bibliography

[1] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory,* Vol. IT-20, pp. 248-287, Mar. 1974.

[2] S. Benedetto, and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, No. 2, pp. 409–428, Mar. 1996.

[3] S. Benedetto, and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol.44, No.5, pp. 591–600, May 1996.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes(1)," *Proc. 1993 IEEE Int. Conf. on Communication* Geneva, Switzerland, pp. 1064-1070, 1993.

[5] C. Berrou, and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Transactions on Communications*, Vol. 44, No. 10,

October 1996.

[6] C. Berrou, and A. Glavieux, "Reflections on the prize paper: 'Near optimum error-correcting coding and decoding: turbo codes'," *IEEE Information Theory Society Newsletter,* Vol.48, No.2, pp. 23–31, June 1998.

[7] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley series in Telecommunications, 1991.

[8] T. M. Duman, and M. Salehi, "New performance bounds for turbo codes," *IEEE Trans. on Communications,* vol. 46, No. 6, pp. 717–723. June 1998.

[9] J. Hagenauer, "Viterbi decoding of convolutional codes for fading and burst channels," *Proc. Int. Zurich Seminar,* 1980.

[10] J. Hagenauer, P. Robertson, and L. Papke, "Iterative (turbo) decoding of systematic convolutional codes with MAP and SOVA algorithms," *Proc. ITG Conf. on "Source and Channel Coding,"* Frankfurt, Germany, pp. 1-9, Oct. 1994.

[11] E.K. Hall, and S.G. Wilson, "Design and analysis of turbo codes on Rayleigh fading channels," *IEEE Journal on Selected Areas in Communications*, vol.16, No.2, pp. 160-174, February 1998.

[12] S. Lin and D. J. Costello, Jr. "Error Control Coding: Fundamentals and Applications," Prentice-Hall, 1983.

[13] R. J. McEliece, "The Theory of Information and Coding," Addison-Wesley Publishing Company, 1977.

[14] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1991.

[15] L.C.Perez, J.Seghers, and D.J. Costello, Jr., "A distance spectrum interpretation of turbo codes," *IEEE Transactions on Information Theory,* vol.42, No.6, pp. 1698-1709, November 1996.

[16] J.G. Proakis, *Digital Communications*, McGraw-Hill, 1983.

[17] P. Robertson, "Illuminating the structure of parallel concatenated recursive systematic (turbo) codes," *Proc. GLOBECOM'94* San Francisco, CA, vol. 3, pp. 1298-1303, November 1994.

[18] J. Seghers, "On the free distance of TURBO codes and related product codes," Final Rep., Diploma Project SS 1995, no. 6613, Swiss Federal Institute of Technology, Zurich, Switzerland, Aug. 1995.

[19] C.E. Shannon, "A mathematical theory of communications," *Bell Syst. Tech. J.*, Vol. 27, pp. 379–423, 1948.

[20] B. Sklar, "A primer on turbo code concepts," *IEEE Communication Magazine*, December 1997.

[21] K. Tepe, and J. Anderson, "Turbo codes for binary symmetric and binary erasure channels," *Proceedings of ISIT 1998, Cambridge, MA, USA.* pp. 59, Aug. 1998.

[22] A. J. Viterbi, and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979.

[23] A.M. Viterbi, and A. J. Viterbi, "Improved union bound on linear codes for the input-binary AWGN channel, with application to turbo codes," *Proceedings of ISIT 1998, Cambridge, MA, USA.* pp. 29, Aug. 1998.