Turbo Codes for
Asymmetric Binary Channels

by

Ka Sin Tong

A dissertation submitted to the

Department of Mathematics and Statistics

in conformity with the requirements

for the degree of

Master of Science

in the field of

Mathematics and Engineering

Queen's University

Kingston, Ontario, Canada

December 2003

Abstract

During the past several decades, there has been a great deal of interest in the areas of source and channel coding. Finding an effective coding method to approach the Shannon limit has indeed been the primary concern for researchers and scientists in the communications industry. In 1993, Berrou et al. made a remarkable breakthrough in channel coding by proposing a coding structure known as "Turbo codes", which possesses a superb error-correcting capability with reasonable decoding complexity. Although there has been plenty of research efforts in Turbo codes over the past 10 years, most of the work concentrated on standard channel models, such as binary symmetric channels (BSC) and AWGN channels. Very limited amount of effort has been dedicated for non-symmetric channel models. In this project, we will evaluate the performance of Turbo codes over various types of binary non-symmetric discrete memoryless channels (DMCs).

In this project, we begin with an overview of the Turbo coding structure and principle. We then investigate the application of Turbo codes for the following channels: the Z-channel (ZC) and the binary asymmetric channel (BAC).

The performance of Turbo codes are evaluated through computer simulations. The data obtained from the simulations are then compared with the Shannon limit for various channel settings. We use a sequence or block length of N=65536, two 16-state RSC elementary encoders with encoder generators (37, 21), a pseudo-random

interleaver, 3200 blocks and 20 iterations for each simulation run. For all channels, we investigate the performance of Turbo codes for the following rates: 1/3 and 1/2. For each channel under our investigation, the ratio of the simulated crossover probability to its corrsponding OPTA is only a few percents away from 0.94 at $rate = 1/3$; on the other hand, the ratio becomes significantly less than 0.94 at $rate = 1/2$. Therefore, we observe that the performance of Turbo codes degrades over binary-input binary-output discrete memoryless asymmetric channels such as ZC and BAC as the rate increases.

Acknowledgements

I would like to offer my sincere thanks to my supervisor, Dr. Fady Alajaji, for his excellent supervision and guidance. This project would not have been fulfilled without his precious advice and encouragement. In addition, my thanks go to my mentor, Dr. Guangchong Zhu, for his valuable support. I would also like to thank Mr. Firouz Behnamfar and our Sun workstation system manager, Mr. Andrei Ordine, for their friendly assistance in using the computing equipment.

Personally, I wish to thank Dr. Bob Erdahl, for giving me the opportunity to join the graduate program at the Department of Mathematics and Statistics. I also wish to thank Mrs. Jennifer Read, for her great effort in administrative support.

Last but not the least, I am deeply thankful to all my family members and friends for their continuous encouragement and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Shannon Limit

In 1948, Claude Shannon pioneered a breakthrough in the theory of communications by showing that information transmission can be made arbitrarily reliable if a certain portion of the transmitted signals is judiciously made redundant through channel coding. In his paper "A Mathematical Theory of Communication" [1], he gave two important theorems, which are named the source coding theorem and the channel coding theorem. Generally, source and channel coding could be treated as two separate subjects without any loss of optimality in a communication system. This is known as Shannon's Separation Principle [1]. Nonetheless, Shannon's Separation Principle holds only under the assumption that the block length or information sequence length reaches infinity; so the delay and the system complexity become infinite. In reality, when the block length is finite, joint-source channel coding schemes offer a better performance over separately designed coding systems.

The Separation Principle is also known as the Lossy Information Transmission Theorem or Lossy Joint Source-Channel Coding Theorem [2]. The theorem states that given a binary memoryless or independent and identically distributed (i.i.d.) source and a channel with capacity C, an end-to-end bit error rate (BER) of P_e can be guaranteed through the construction of a source-channel code of rate R_c if

$$R_c \cdot R(P_e) < C, \tag{1.1}$$

where $R(P_e)$ is the rate-distortion function under the Hamming distortion measure, given by

$$R(P_e) = \begin{cases} h_b(p_0) - h_b(P_e) & \text{if } 0 \leqslant P_e \leqslant min\{p_0, 1-p_0\}, \\ 0 & \text{if } P_e > min\{p_0, 1-p_0\}. \end{cases} \tag{1.2}$$

where p_0 is the probability that the binary i.i.d. source generates a 0, and $h_b(\cdot)$ is the binary entropy function defined as

$$h_b(x) = -xlog_2 x - (1-x)log_2(1-x). \tag{1.3}$$

In this thesis, the Shannon limit, also known as the optimal performance theoretically achievable (OPTA), is measured in terms of the crossover probability of a channel. For this reason, the channel capacity can be expressed as a function of the crossover probability of the channel under discussion. By replacing (1.1) with an equality sign, the OPTA can be solved for a given P_e, R_c and p_0.

Over the past 50 years or so, researchers have been trying to design an effective coding method to get close to the Shannon limit. Before the birth of Turbo Codes, the best reported channel coding performance of a binary uniform i.i.d. source over an AWGN channel was more than 2 dB in signal to noise ratio E_b/N_0 away from OPTA [3]. In 1993, C. Berrou, A. Glavieux and P. Thitimajshima introduced Turbo codes [22]. For a BER of 10^{-5} and $R_c = 1/2$, the Turbo codes performance was just 0.5 dB away from OPTA [4]. This superb performance then began to attract a great deal of interest from the telecommunications world.

Over the past decade, plenty of research work has been dedicated to Turbo codes. Most of them focus on standard channel models such as the binary symmetric channel (BSC) and the AWGN channels [5], and very limited amount of effort has been put on non-standard channel models, such as non-symmetric, non-binary and multiuser channels. In [6], McEliece claimed that the superb performance of Turbo codes is not only limited to standard channel models, and that it should also carry over for non-standard channel models.

In this project, we are particularly interested in the performance of Turbo codes over binary non-symmetric discrete memoryless channels (DMCs). In related work, Silverman and Rumsey [7-8] show that for any binary-input binary-output DMC, the ratio of the uniform input capacity (i.e., the channel mutual information evaluated for a uniformly distributed input) to the capacity is at least $(e/2)ln2 = 0.9421$. This result was later generalized for arbitrary binary-input DMCs [9]. Therefore, at a given R_c and P_e for all such channels, one would expect that the ratio of the simulated crossover probability to its corresponding OPTA to be close to 0.94 or 94%. This was indeed claimed recently by McEliece [6] for the entire class of non-standard channels.

1.2 Literature review

In the early days of Turbo codes research, the focal point was solely on how to improve their performance over standard channel models with uniform i.i.d. sources as channel inputs. The possibility of implementing Turbo codes over other types of channels was rarely mentioned. However, a couple years ago, McEliece claimed that Turbo codes should have good potential to perform well over non-standard channels [6]. Due to the non-symmetric nature of most non-standard channels, it is quite likely that the source input that optimizes the performance over such channels is not uniform i.i.d. There has been some significant research in Turbo codes focusing on non-uniform i.i.d. sources in recent years. In [10], Hagenauer mentioned that if the source input is non-uniform i.i.d., the decoder's extrinsic information needs to be modified according to the source input distribution. In [15-19], Zhu et. al. provided an extensive investigation on the issues of designing Turbo codes for non-uniform i.i.d. sources over AWGN and Rayleigh fading channels. The results give us some important insight on how to modify the Turbo encoding and decoding methods in order to cope with non-standard channels.

1.3 Contributions of the project

In this project, we are interested in the performance of Turbo codes over non-standard channel models. In particular, our investigation focuses on binary non-symmetric DMCs, which include the Z-channel (ZC) and the binary asymmetric channel (BAC). We also consider the BSC in order to compare the performance between standard and non-standard channels.

For the non-symmetric channels under our investigation, the optimal source inputs that achieve channel capacity are non-uniform; nevertheless, their distributions are close to uniform [6]. The objective of this project is to verify McEliece's claim about the good performance of Turbo codes over non-standard channels. Specifically, we will investigate the performance of Turbo codes over the non-symmetric binary channels and examine, for each channel, how close to 94% is the ratio of the simulated crossover probability to its corresponding OPTA.

1.4 Overview of the project

In Chapter 2, basic concepts of Turbo encoding are discussed. After introducing recursive systematic convolutional (RSC) encoders, we present the general structures of Turbo encoders.

In Chapter 3, the general structures of Turbo decoders are discussed. The modified BCJR (M-BCJR) decoding algorithm for Turbo codes is introduced, and its derivation is provided. We also reveal the principle of iterative decoding in Turbo codes.

In Chapter 4, we examine the applications of Turbo codes to the following channels: BSC, ZC and BAC. For each channel model, we point out the required adjustments in the Turbo decoder, and the Shannon limit is also calculated.

In Chapter 5, the simulation results are posted and analyzed. The performance of Turbo codes for each channel is evaluated.

In Chapter 6, conclusions of the project are presented. In addition, possible future research work is identified.

Chapter 2

Turbo Encoder

2.1 Convolutional encoders: RSC encoder

To understand Turbo codes, it is important to introduce some fundamental concepts about convolutional codes.

2.1.1 Basic Concepts

There are two kinds of convolutional codes, recursive systematic convolution codes (RSC) and nonsystematic convolution codes (NSC). More specifically, in Turbo codes design, RSC encoders are used.

Convolutional codes map a sequence of k-tuples of information bits to a sequence of n-tuples output bits, which depends on the current k-tuple information bits that enter the encoder and the previous k-tuples input blocks. Encoding can be done by implementing linear finite-state shift registers to the convolutional encoders. The number of shift registers that each encoder has is called the memory, and the code rate, R, is defined by $R = k/n$ [20].

Since convolutional codes are linear codes [16][30], the minimum Hamming distance between any two codewords [20][29] is just the same as the minimum Hamming weight of non-zero codewords. Recall that the Hamming weight of a codeword is defined as the number of non-zero digits. Minimum Hamming distance, also known as free distance, can be calculated by searching for the minimum weight path which makes the convolution encoder return to the all-zero state.

2.1.2 Recursive systematic convolution encoder

Fig. 2.1 is a structural diagram of an RSC encoder. The symbol $\sum$ represents mod 2 summation. There are two summation nodes, which are known as generators $G1$ and $G2$. Shift registers are represented as $R1$, $R2$ and $R3$. $G2$ is a feed-forward generator, and $G1$ is a feedback generator.

Encoder generators:

$$G1 = \{g_{10}, g_{11}, g_{12}, g_{13}\} = \{1, 1, 0, 1\} \tag{2.1}$$

$$G2 = \{g_{20}, g_{21}, g_{22}, g_{23}\} = \{1, 0, 1, 1\} \tag{2.2}$$

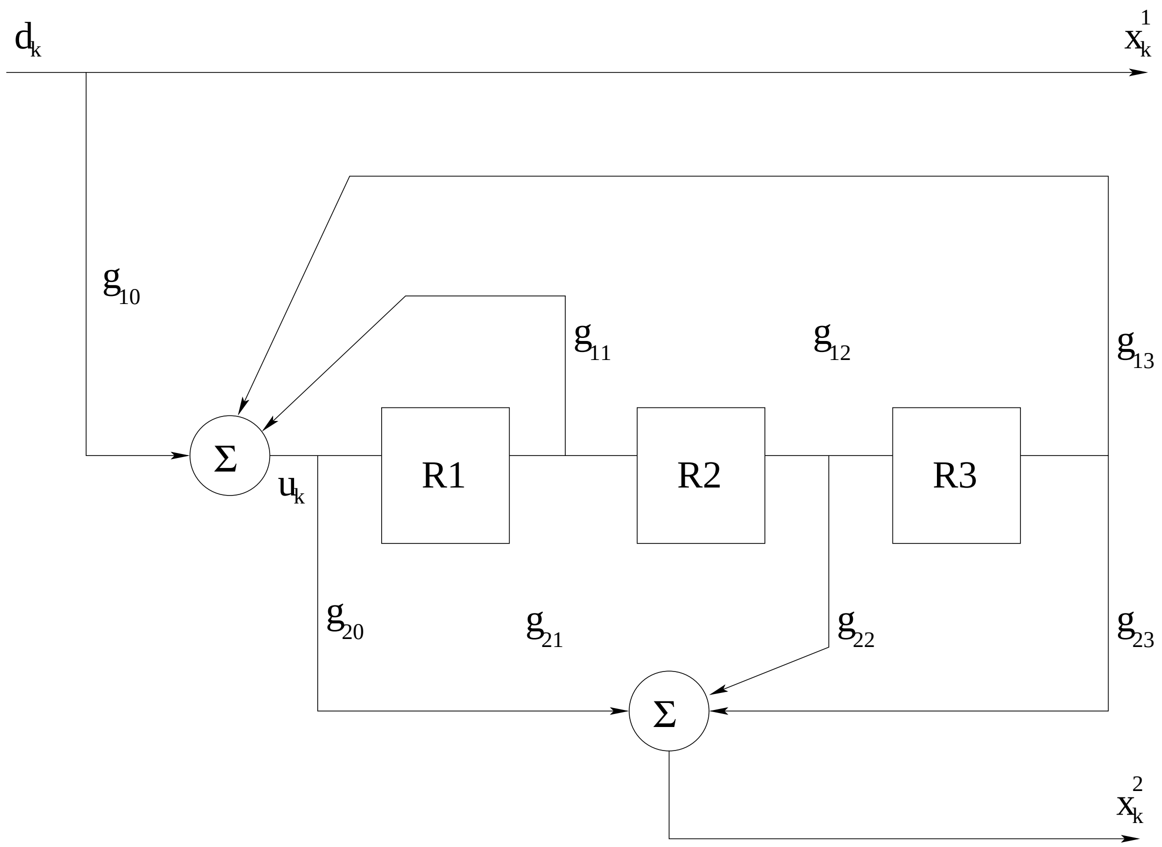

Figure 2.1: Recursive systematic convolution encoder.

Note that g_{10} must always be "1" as it is the direct link between the input bit and $G1$.

Data flow: Since the encoder is systematic, $x_k^1 = d_k$ for all k. x_k^2 is a parity bit which depends on the generators and the contents of the shift registers and input d_k. x_k^2 is then expressed as

$$
\begin{aligned}
u_k &= d_k + \sum_{i=1}^{2} g_{1i} \cdot u_{k-1} \qquad mod2 && (2.3) \\
x_k^2 &= \sum_{i=0}^{v} g_{2i} \cdot u_{k-1} \qquad mod2 && (2.4)
\end{aligned}
$$

State termination: The convention is to begin and end the encoding at the all-zero state. To terminate an RSC encoder with memory size v, we cannot simply insert v consecutive "0"s due to the existence of a feedback generator. Even though termination requires only v bits, each bit must be determined by the current encoder state.

2.2 Structure of Turbo Encoder

Fig. 2.2 is a structural diagram of the Turbo encoder by Berrou et al.used in 1993 [4]. It consists of a parallel concatenation of two elementary (37 ,21) RSC encoders, an interleaver and a puncturing device. The interleaver rearranges the incoming input sequence in a different order. The delay line is just a buffer that stores the incoming bits before going to the interleaver. The puncturing device controls the code rate by

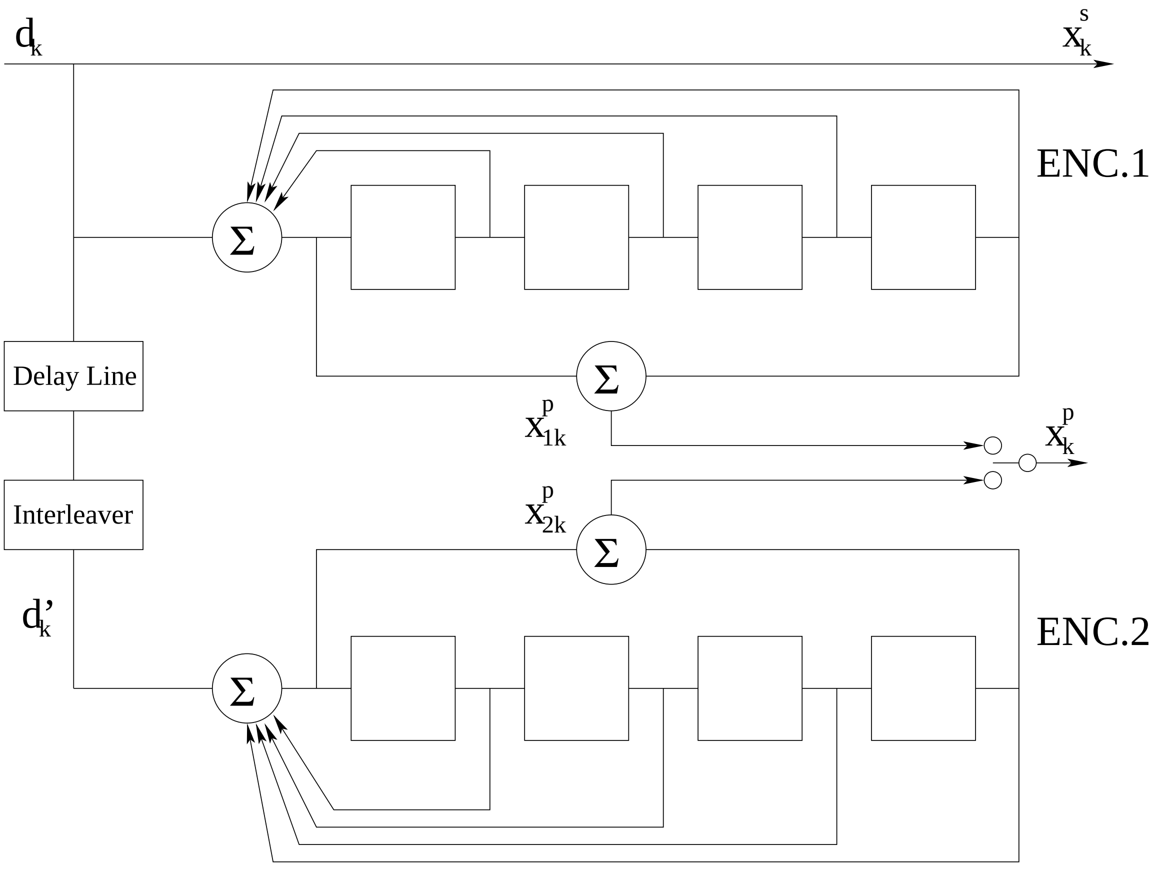

Figure 2.2: Turbo encoder structure.

screening out some parity bits. d_k represents the original sequence. d'_k represents the interleaved sequence. d_k corresponds to the output systematic bit x^s_k and parity bits x^p_{1k} and x^p_{2k}. In general, the number of RSC concatenated encoders can vary, and they are not necessarily identical.

2.3 Interleaving

Interleaving is designed to deal with channel burst errors by placing an interleaver before a channel so that the input sequence is shuffled before transmission. At the decoding end, a de-interleaver is used to rearrange the interleaved sequence back to its original order. The purpose of interleaving is to prevent consecutive bit errors from occurring in the original sequence because consecutive errors are more difficult to detect and correct. In Turbo coding, however, interleaving is used for a different purpose, which is to "randomize" the data before it is encoded by the second RSC encoder.

The uniform and pseudo-random interleavers are the two major types of interleavers. A uniform interleaver is made up of an $N \times N$ matrix and bits are written linewise and read columnwise. A pseudo-random interleaver rearranges the input sequence randomly. It is known that pseudo-random interleavers have a better performance over uniform interleavers on Turbo codes [23].

2.4 Puncturing

If puncturing is not used, the Turbo encoder in Fig 2.3 will have an overall rate of 1/3 because there are 2 parity bits but only one information bit at each time instant. For this particular encoder, if we want to increase the code rate, we should delete some parity bits from ENC.1 and ENC.2. However, there is always a tradeoff between the redundancy and the error correcting capability. Higher rate means lower redundancy and worse error correcting capability. It is up to the users to decide what rates are the best for his applications.

Chapter 3

Turbo Decoding

3.1 Structure of the Turbo decoder

Fig. 3.1 shows the relationship between the bits before and after transmission. All the notations indicated with "x" are before transmission, while the rest are after transmission. x_k^s and y_k^s are information bits. x_k^p and y_k^p are parity bits. x_k^p came from x_{1k}^p and x_{2k}^p by a puncturing device. Similarly, on the receiving end, a puncturing device separate y_k^p into y_{1k}^p and y_{2k}^p. For the sake of convenience, we will call the pair of x_k^s and x_k^p as x_k and the pair of y_k^s and y_k^p as y_k.

Fig. 3.2 is a structural diagram of a Turbo decoder, which consists of two serially concatenated elementary decoders that are implemented with the BCJR algorithm. The interleavers in this diagram are exactly the same as the one from the Turbo

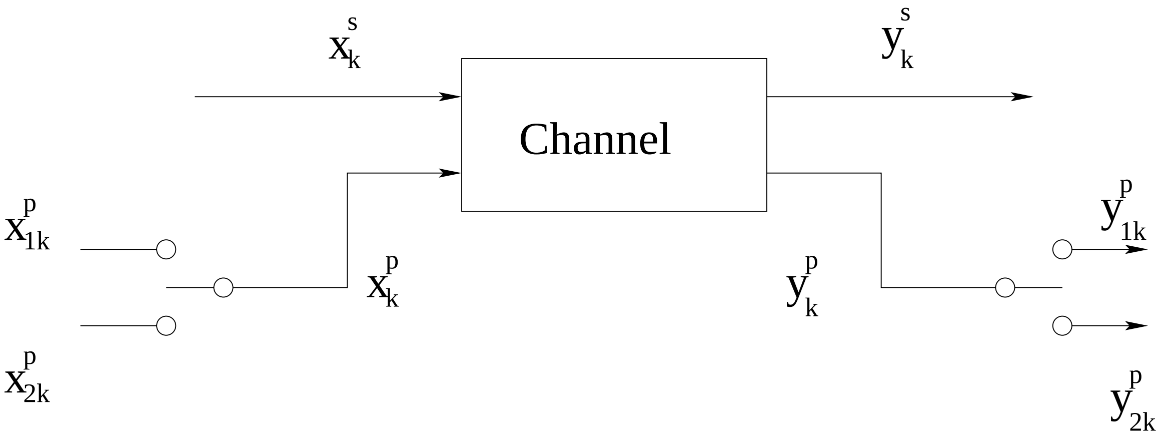

Figure 3.1: Data before and after transmission.

encoder. The de-interleavers also matches the interleavers such that the original bit sequences can be recovered. $y_k^{s'}$, $L_{1e}(d'_k)$, $L_{2e}(d'_k)$ are the interleaved versions of y_k^s, $L_{1e}(d_k)$ and $L_{2e}(d_k)$ respectively. $L_{1e}(d_k)$ and $L_{2e}(d_k)$ are known as extrinsic informations, which are exchanged iteratively between the two elementary decoders. After each iteration, the decoder makes a decision by comparing the log-likelihood ratio (LLR) $\Lambda(d_k)$ such that

$$\hat{d}_k = \begin{cases} 1 & \text{if } \Lambda(d_k) > 0, \\ 0 & \text{if } \Lambda(d_k) \leqslant 0. \end{cases} \tag{3.1}$$

3.2 BCJR algorithm

The Viterbi algorithm (VA) is considered as an optimal decoding algorithm for reducing the probability of sequence error [21]. However, VA is not capable of generating the a priori probability (APP) for each decoded bit. The BCJR algorithm [24], which

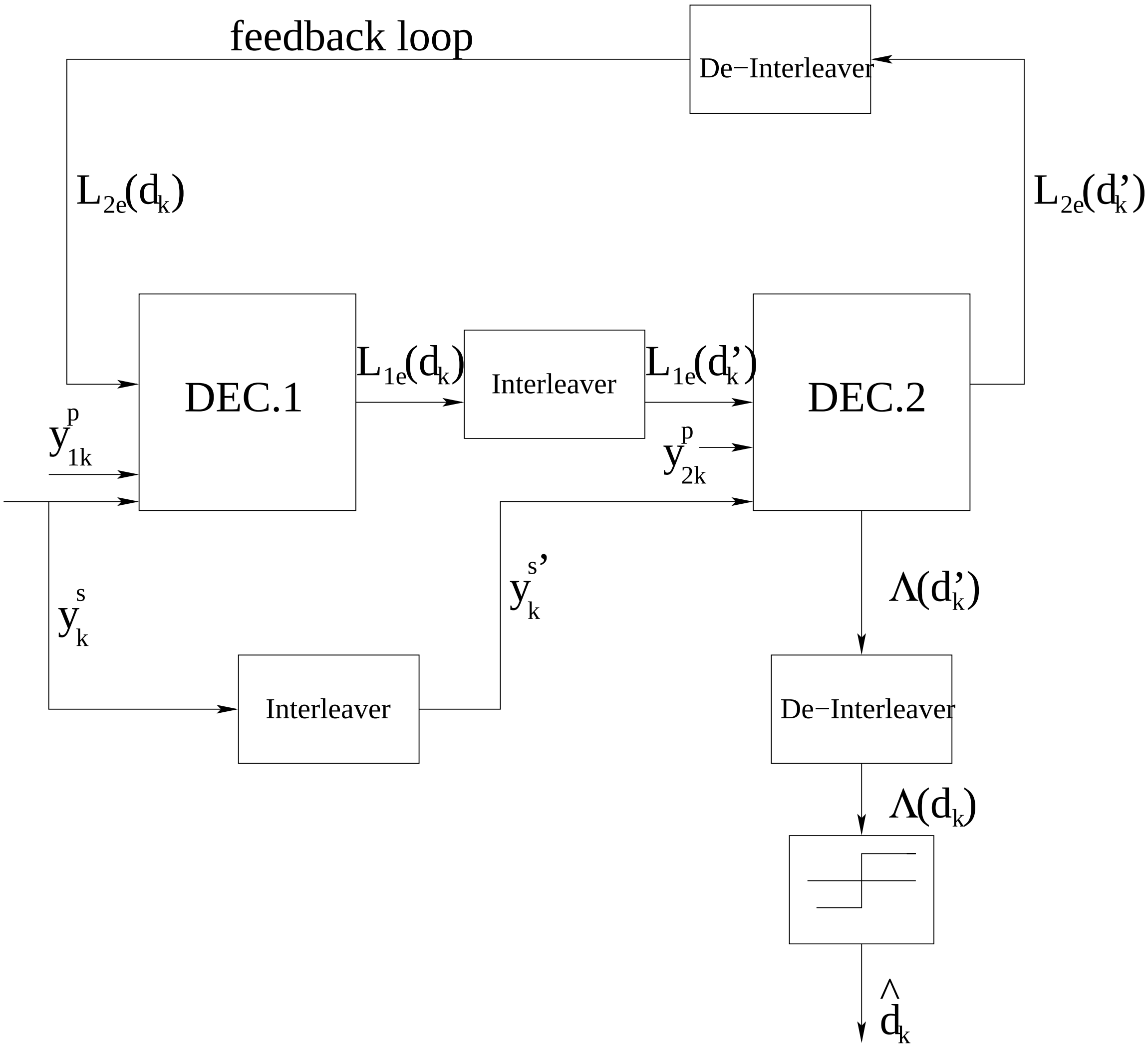

Figure 3.2: Turbo decoder structure.

was proposed by L. Bahl, J. cocke, F. Jelinek, and J. Raviv in 1974, provides this capability. In 1993, Berrou et. al. successfully applied the BCJR algorithm directly for Turbo codes with slight modifications, but their original decoding method required a considerable amount of computation work [22]. Later, a modified version of the BCJR algorithm [17][4], which has much lower computational complexity, was developed. In the next section, the modified BCJR algorithm for Turbo decoding will be derived and explained.

3.3 Modified BCJR algorithm for Turbo decoding

3.3.1 Problem formulation

The goal of the Turbo decoder is to find the original input sequence d_k from the received sequence y_k. We will denote $x_1^N = \{x_1, x_2, \cdots, x_N\}$ and $y_1^N = \{y_1, y_2, \cdots, y_N\}$ as the sequence before and after transmission respectively. We will also denote $\hat{d}_k$ to be the hard decision that the Turbo decoder makes for d_k. Then we have

$$\hat{d}_k = \begin{cases} 1 & \text{if } \Lambda(d_k) > 0, \\ 0 & \text{if } \Lambda(d_k) \leqslant 0, \end{cases} \tag{3.2}$$

where $\Lambda(d_k)$ is the logarithm of likelihood ratio (LLR) [4], which is defined as

$$\Lambda(d_k) = \log \frac{Pr\{d_k = 1 | y_1^N\}}{Pr\{d_k = 0 | y_1^N\}}. \tag{3.3}$$

From (3.3), $Pr\{d_k = i|y_1^N\}$, $i = 0, 1$ is known as the a posteriori probability (APP) of the bit d_k [4], and it can be further expressed as

$$Pr\{d_k = i|y_1^N\} = \sum_{m=0}^{M} \lambda_k^i(m), \tag{3.4}$$

where

$$\lambda_k^i(m) = Pr\{d_k = i, S_k = m|y_1^N\}, \tag{3.5}$$

for $i = 0, 1$ and $k = 1, .., N$, and S_k is the encoder state at time k.

In this subsection, the modified BCJR algorithm (M-BCJR) [26], which can be used to calculate $\Lambda(d_k)$, will be derived and explained.

3.3.2 Derivation of the Modified BCJR algorithm

The M-BCJR algorithm computes each element within a factorized form of $\Lambda(d_k)$ recursively under certain boundary conditions in order to determine the value of $\Lambda(d_k)$.

1) **Factorization of** $\Lambda(d_k)$

First we have to factorize $\lambda_k^i(m)$.

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

$$
\begin{aligned}
&= \frac{Pr\{y_{k+1}^N | S_k = m\} \cdot \sum_{m'} Pr\{d_k = i, S_k = m, S_{k-1} = m', y_1^{k-1}, y_k\}}{Pr\{y_1^N\}} \\
&= \frac{\sum_{m'} Pr\{y_{k+1}^N | S_k = m\} \cdot Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m', y_1^{k-1}\}}{Pr\{y_1^N\}} \\
&\qquad \times Pr\{S_{k-1} = m', y_1^{k-1}\} \\
&= \frac{\sum_{m'} Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m'\} \cdot Pr\{S_{k-1} = m' | y_1^{k-1}\}}{Pr\{y_{k+1}^N | y_1^k\}} \\
&\qquad \times Pr\{y_{k+1}^N | S_k = m\}. \qquad (3.6)
\end{aligned}
$$

For simplicity, we define

$$\alpha_k(m) \triangleq Pr\{S_k = m, y_1^k\}, \tag{3.7}$$

$$\beta_k(m) \triangleq \frac{Pr\{y_{k+1}^N | S_k = m\}}{Pr\{y_{k+1}^N | y_1^k\}}, \tag{3.8}$$

$$\gamma_i(y_k, m', m) \triangleq Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m'\}. \tag{3.9}$$

Now, (3.6) becomes

$$\lambda_k^i(m) = \sum_{m'} \gamma_i(y_k, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m). \tag{3.10}$$

Finally, $\Lambda(d_k)$ can be written as

$$
\begin{aligned}
\Lambda(d_k) &= \log \frac{\sum_m \lambda_k^1(m)}{\sum_m \lambda_k^0(m)} && (3.11) \\
&= \log \frac{\sum_m \sum_{m'} \gamma_1(y_k, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}{\sum_m \sum_{m'} \gamma_0(y_k, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}. && (3.12)
\end{aligned}
$$

2) **Recursive relations and boundary conditions of $\alpha_k(m)$ and $\beta_k(m)$**

Both $\alpha_k(m)$ and $\beta_k(m)$ can be expressed as a factorized form and computed recursively.

$$
\begin{aligned}
\alpha_k(m) &= Pr\{S_k = m | y_1^k\} \\
&= \frac{Pr\{S_k = m, y_k, y_1^{k-1}\}}{Pr\{y_1^k\}} \\
&= \frac{Pr\{S_k = m, y_k | y_1^{k-1}\}}{Pr\{y_k | y_1^{k-1}\}} \\
&= \frac{\sum_{m'} \sum_i Pr\{S_k = m, y_k, S_{k-1} = m', d_k = i | y_1^{k-1}\}}{Pr\{y_k | y_1^{k-1}\}} \\
&= \frac{\sum_{m'} \sum_i Pr\{S_k = m, y_k, d_k = i | S_{k-1} = m', y_1^{k-1}\} \cdot Pr\{S_{k-1} = m' | y_1^{k-1}\}}{Pr\{y_k | y_1^{k-1}\}} \\
&= \frac{\sum_{m'} \sum_i \gamma_i(y_k, m', m) \cdot \alpha_{k-1}(m')}{Pr\{y_k | y_1^{k-1}\}} \\
&= \frac{\sum_{m'} \sum_i \gamma_i(y_k, m', m) \cdot \alpha_{k-1}(m')}{\sum_m Pr\{S_k = m, y_k | y_1^{k-1}\}} \\
&= \frac{\sum_{m'} \sum_i \gamma_i(y_k, m', m) \cdot \alpha_{k-1}(m')}{\sum_m \sum_{m'} \sum_i \gamma_i(y_k, m', m) \cdot \alpha_{k-1}(m')}. \qquad (3.13)
\end{aligned}
$$

$$
\begin{aligned}
\beta_k(m) &= \frac{Pr\{y_{k+1}^N | S_k = m\}}{Pr\{y_{k+1}^N | y_1^k\}} \\
&= \frac{\sum_{m'} \sum_i Pr\{y_{k+1}, y_{k+2}^N, S_k = m, S_{k+1} = m', d_{k+1} = i\}}{Pr\{y_{k+1}^N | y_1^k\} \cdot Pr\{S_k = m\}} \\
&= \frac{\sum_{m'} \sum_i Pr\{y_{k+2}^N | S_{k+1} = m', S_k = m, d_{k+1} = i, y_{k+1}\}}{Pr\{y_{k+1}^N | y_1^k\}} \\
&\qquad \times Pr\{S_{k+1} = m', d_{k+1} = i, y_{k+1} | S_k = m\} \\
&= \frac{\sum_{m'} \sum_i Pr\{y_{k+2}^N | S_{k+1} = m'\} \cdot \gamma_i(y_{k+1}, m, m')}{Pr\{y_{k+1}^N | y_1^k\}} \\
&= \frac{\sum_{m'} \sum_i Pr\{y_{k+2}^N | S_{k+1} = m'\} \cdot \gamma_i(y_{k+1}, m, m')}{Pr\{y_1^N | y_1^k\}} \\
&= \frac{\sum_{m'} \sum_i Pr\{y_{k+2}^N | S_{k+1} = m'\} \cdot \gamma_i(y_{k+1}, m, m')}{Pr\{y_{k+2}^N | y_1^{k+1}\} \cdot Pr\{y_1^{k+1}\} / Pr\{y_1^k\}} \\
&= \frac{\sum_{m'} \sum_i Pr\{y_{k+2}^N | S_{k+1} = m'\} \cdot \gamma_i(y_{k+1}, m, m')}{Pr\{y_{k+2}^N | y_1^{k+1}\} \cdot Pr\{y_{k+1} | y_1^k\}}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{\sum_{m'} \sum_i \beta_{k+1}(m') \cdot \gamma_i(y_{k+1}, m, m')}{Pr\{y_{k+1}|y_1^k\}} \\
&= \frac{\sum_{m'} \sum_i \beta_{k+1}(m') \cdot \gamma_i(y_{k+1}, m, m')}{\sum_m \sum_{m'} \sum_i \gamma_i(y_{k+1}, m, m') \cdot \alpha_k(m)}. \qquad (3.14)
\end{aligned}
$$

From (3.13), we note that $\alpha_k(m)$ depends on $\alpha_{k-1}(m')$, as well as from (3.14), $\beta_k(m)$ depends on $\beta_{k+1}(m')$; therefore, boundary conditions are needed for initialization.

Since the first encoder begins and ends at the all-zero state, the initial boundary conditions for the first decoder become

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

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

The boundary conditions for the second decoder are different from the first decoder because of interleaving. The v-bit tail that is used to terminate the first encoder cannot be used to terminate the second encoder. As a result, the state of the second encoder is unknown when the first encoder is terminated at the all-zero state. The initial boundary conditions for the second decoder become

$$\alpha_0(0) = 1, \qquad \text{and} \qquad \alpha_0(m) = 0 \qquad \text{for } m \neq 0, \tag{3.17}$$

$$\beta_N(m) = 1/M \qquad \text{for } m = 0, 1, ..., M. \tag{3.18}$$

The only term left to be determined is $\gamma_i(y_k, m', m)$. From (3.9), we have

$$\gamma_i(y_k, m', m) = Pr\{d_k = i, S_k = m, y_k | S_{k-1} = m'\} \tag{3.19}$$

$$\begin{aligned} &= p(y_k | d_k = i, S_k = m, S_{k-1} = m') \cdot q(d_k = i | S_k = m, S_{k-1} = m') \\ &\quad \times \pi(S_k = m | S_{k-1} = m'), \end{aligned} \tag{3.20}$$

where $p(.|.)$ is the transition probability of the DMC. $q(.|.)$ is the conditional probability which is based on the current state and the previous state. $\pi(.|.)$ is the state transition probability.

The value of $p(.|.)$ is determined by the channel characteristics. $q(.|.)$ is either 0 or 1 because the elementary encoders are deterministic machines. The calculation mechanism of $\pi(.|.)$ can be explained by iterative decoding, which will be discussed in the next section.

3.4 Iterative decoding

Iterative decoding requires information exchange between two decoders in an iterative manner. Higher number of iterations will result in a better performance [27]. In this section, the mechanism of iterative decoding is explained.

3.4.1 Computation of $\pi(.|.)$

To understand how iteration decoding works, it is essential to know that the $\pi(.|.)$ varies between decoders for every iteration. In particular, if the source distribution is $Pr\{d_k = 0\} = p_0$ and $Pr\{d_k = 1\} = 1 - p_0$, the first decoder in the first iteration will always assume that the information bit d_k can be 0 or 1 according to the source distribution. As a result, we have

$$\pi(S_k = m|S_{k-1} = m') = 1 - p_0$$

$$\text{if } q(d_k = 1|S_k = m, S_{k-1} = m') = 1, \tag{3.21}$$

$$\pi(S_k = m|S_{k-1} = m') = p_0$$

$$\text{if } q(d_k = 0|S_k = m, S_{k-1} = m')\text{=}1. \tag{3.22}$$

For other cases, $\pi(S_k = m|S_{k-1} = m')$ cannot be assumed as in (3.21) and (3.22), but it can be expressed as a general term, which can be written as

$$\pi(S_k = m|S_{k-1} = m') = \frac{e_{L(d_k)}}{1 + e_{L(d_k)}}$$

$$\text{if } q(d_k = 1|S_k = m, S_{k-1} = m') = 1, \tag{3.23}$$

$$\pi(S_k = m|S_{k-1} = m') = 1 - \frac{e_{L(d_k)}}{1 + e_{L(d_k)}}$$

$$\text{if } q(d_k = 0|S_k = m, S_{k-1} = m')\text{=}1. \tag{3.24}$$

where $L(d_k)$ is the LLR of the a priori probability of d_k, and it can be expressed as $L(d_k) = \log[Pr\{d_k = 1\}/Pr\{d_k = 0\}]$.

3.4.2 Information exchange between decoders

For every iteration, each decoder computes a LLR and sends a certain portion of its value towards the next decoder; this particular portion is known as the extrinsic information. For $j = 1, 2$, let $\Lambda_j(d_k)$ be the LLR of DEC.j, and let $L_{je}(d_k)$ be the extrinsic information that corresponds to DEC.j. Therefore, (3.12) can be re-written as

$$
\begin{aligned}
\Lambda_j(d_k) &= \log \frac{\sum_m \sum_{m'} \gamma_1(y_k, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}{\sum_m \sum_{m'} \gamma_0(y_k, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)} && (3.25)\\
&= \log \frac{\sum_m \sum_{m'} p(y_k^s|d_k = 1, S_k = m, S_{k-1} = m') \cdot \gamma_1'(y_{jk}^p, m', m)}{\sum_m \sum_{m'} p(y_k^s|d_k = 0, S_k = m, S_{k-1} = m') \cdot \gamma_0'(y_{jk}^p, m', m)} \\
&\qquad \times \frac{\alpha_{k-1}(m') \cdot \beta_k(m)}{\alpha_{k-1}(m') \cdot \beta_k(m)} + L(d_k) && (3.26)\\
&= \log \frac{\sum_m \sum_{m'} p(y_k^s|d_k = 1) \cdot \gamma_1'(y_{jk}^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}{\sum_m \sum_{m'} p(y_k^s|d_k = 0) \cdot \gamma_0'(y_{jk}^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)} \\
&\qquad + L(d_k) && (3.27)\\
&= \log \frac{p(y_k^s|d_k = 1)}{p(y_k^s|d_k = 0)} + \\
&\quad \log \frac{\sum_m \sum_{m'} \gamma_1'(y_{jk}^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}{\sum_m \sum_{m'} \gamma_0'(y_{jk}^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)} + L(d_k), && (3.28)\\
&= L_c(d_k) + L_{je}(d_k) + L(d_k), && (3.29)
\end{aligned}
$$

where $y_k = (y_k^s, y_{1k}^p, y_{2k}^p)$, which y_k^s represents the received systematic information bit, as well as y_{1k}^p and y_{2k}^p represent the received parity bits. To make $\Lambda_j(d_k)$ a more readable expression, we define $\gamma_i'(y_{jk}^p, m', m)$ such that $\gamma_i'(y_{jk}^p, m', m) = p(y_{jk}^p|d_k = i, S_k = m, S_{k-1} = m')$. Since $x_k^s = d_k, p(y_k^s|d_k = i, S_k = m, S_{k-1} = m') = p(y_k^s|d_k = i)$ as shown in (3.27). $L_c(d_k)$, which is known as the channel measurement [27], is the

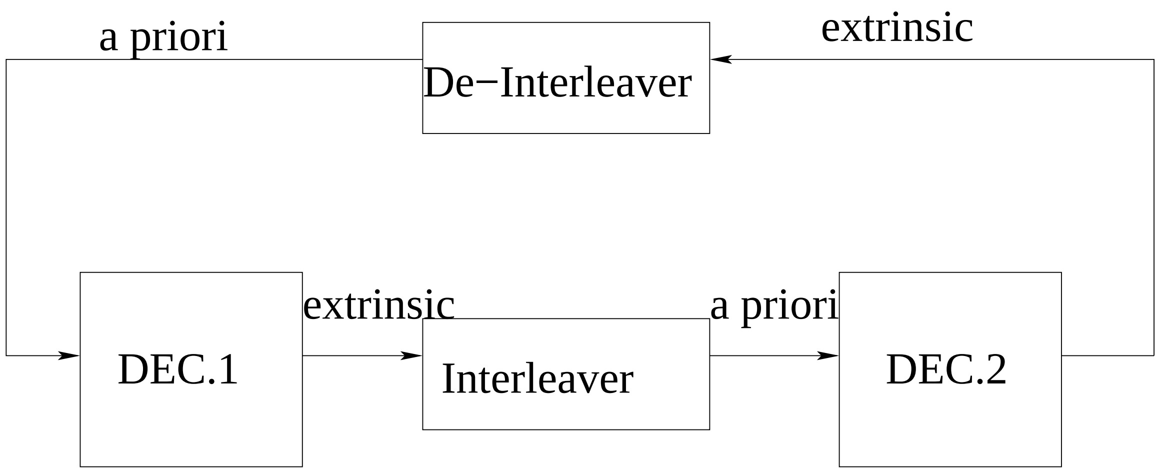

Figure 3.3: Information exchange between the two decoders.

LLR of the channel transition probability.

As iterations go on, both DEC.1 and DEC.2 are responsible for computing and transferring their extrinsic information to each other. The extrinsic information from the current decoder will then become $L(d_k)$, or known as the LLR of the a priori probability for the next decoder, which generates a new extrinsic information as well as a new LLR based on the value of $L(d_k)$. The relationship between the decoders is illustrated in Fig. 3.3.

It is essential to understand why $\Lambda_j(d_k)$ should not pass on to the next decoder as the LLR of the a priori probability. $L_c(d_k)$, the LLR of the channel transition probability, is only based on the statistical characteristics of the channel. Thus, it may not be reliable. Each time a $\Lambda_j(d_k)$ is transmitted, an extra $L_c(d_k)$ will be included as a component of the new $L(d_k)$ for the next decoder. As iterations go on,

$L_c(d_k)$ will accumulate, and the error correcting ability will be diminished. Another reason is that a decoder should not be fed with information that stems from itself [27]. Therefore, only the extrinsic information should be transferred to the next decoder as the LLR of the a priori probability.

Chapter 4

Turbo codes for different channels

In this section, we will investigate the application of Turbo codes to the following discrete memoryless channels (DMCs): the binary symmetric channel (BSC), the Z-channel (ZC) and the binary asymmetric channel (BAC). Since each channel has its unique statistical characteristics, certain adjustments of the M-BCJR algorithm are required at the Turbo decoder according to different channel types. In addition, the Shannon limit will be calculated for each channel so that we can note the ideal performance of Turbo codes.

4.1 Channel models

The models for the BSC, ZC and BAC are similar. They all have discrete inputs and outputs, and the outputs depend only on the inputs at the same discrete time unit.

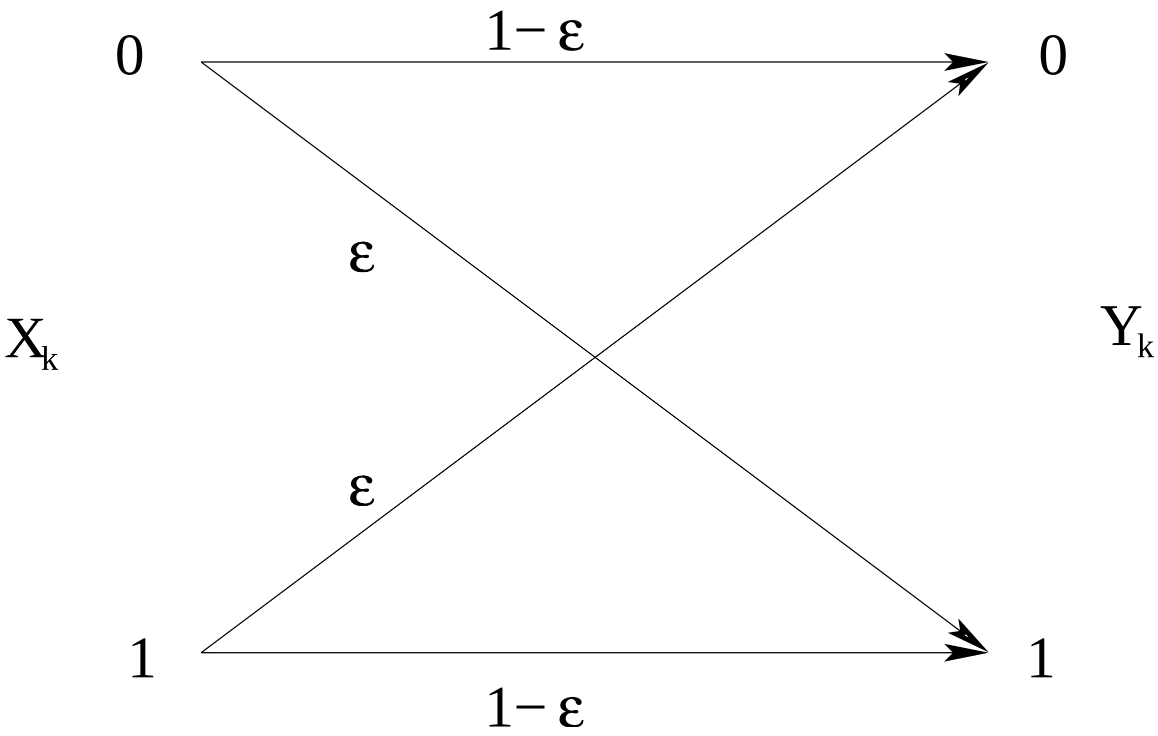

Figure 4.1: Binary symmetric channel.

Fig. 4.1, 4.2 and 4.3 are the channel diagrams for the BSC, ZC and BAC respectively. In each diagram, X_k denotes the channel input, and Y_k denotes the channel output. ϵ, α and β represent the crossover probabilities that either a "0" will be received as a "1" or a "1" will be received as a "0" as indicated in the diagrams.

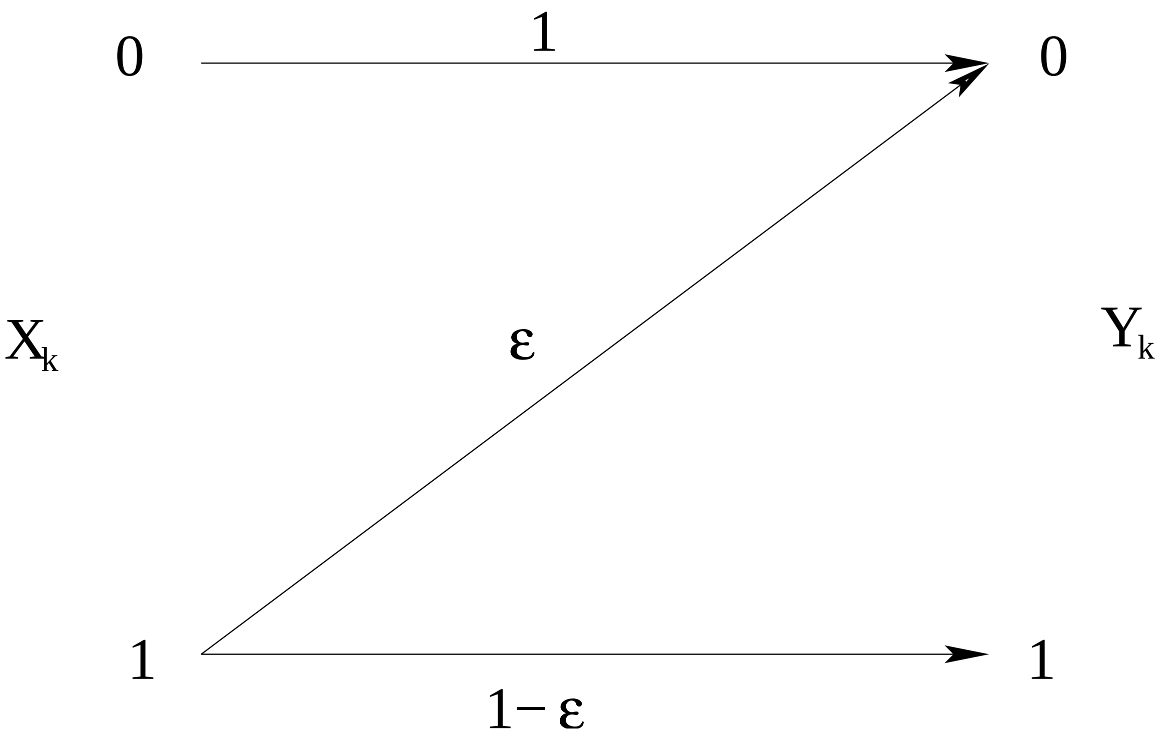

Figure 4.2: Z-channel.

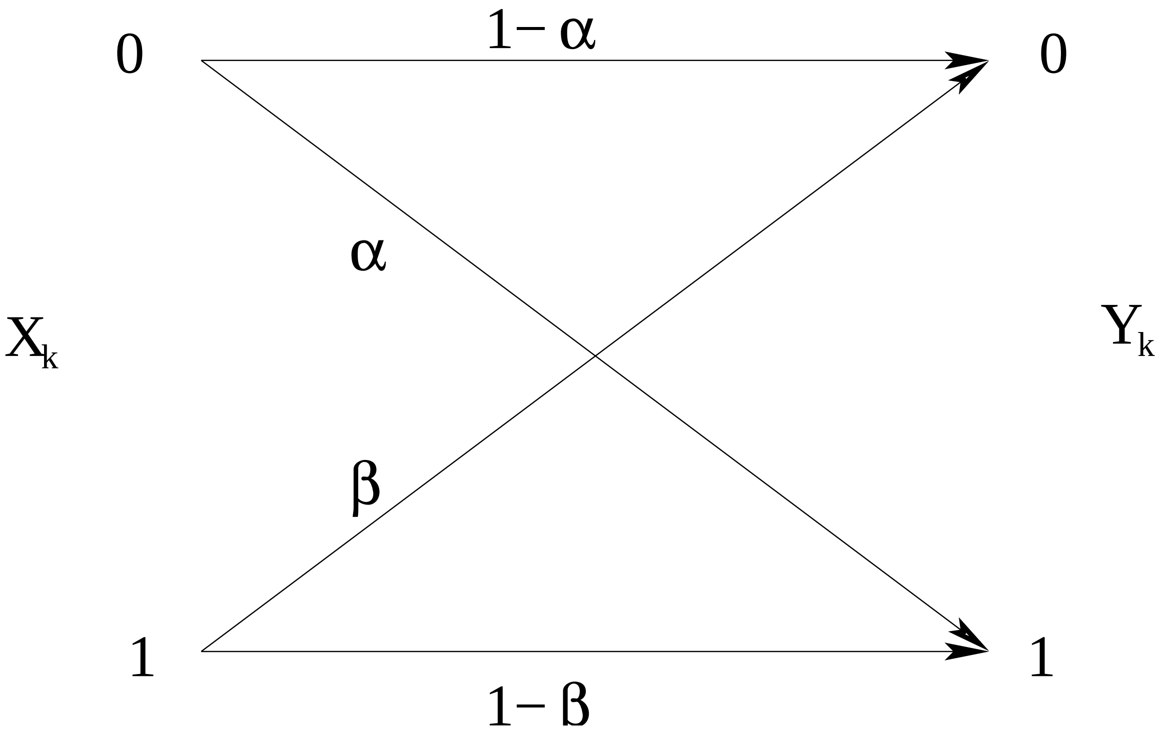

Figure 4.3: Binary asymmetric channel.

4.2 Modifications in the Turbo decoder

From (3.20), we have

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

For the BSC, the channel transition probability of the information bit d_k becomes

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

For the ZC, the channel transition probability of the information bit d_k becomes

$$p(y_k^s|d_k = i) = \begin{cases} 1 & \text{if } y_k^s = 0 \text{ and } i = 0, \\ 0 & \text{if } y_k^s = 1 \text{ and } i = 0; \\ \epsilon & \text{if } y_k^s = 0 \text{ and } i = 1, \\ 1-\epsilon & \text{if } y_k^s = 1 \text{ and } i = 1. \end{cases} \quad (4.3)$$

For the BAC, the channel transition probability of the information bit d_k becomes

$$p(y_k^s|d_k = i) = \begin{cases} 1-\alpha & \text{if } y_k^s = 0 \text{ and } i = 0; \\ \alpha & \text{if } y_k^s = 1 \text{ and } i = 0; \\ \beta & \text{if } y_k^s = 0 \text{ and } i = 1; \\ 1-\beta & \text{if } y_k^s = 1 \text{ and } i = 1. \end{cases} \tag{4.4}$$

For the undeleted parity bit, $p(y_k^p|d_k = i, S_k = m, S_{k-1} = m')$ can be transformed into the channel transition probability as well because the conditions $(d_k = i, S_k = m, S_{k-1} = m')$ are sufficient to determine the parity output of the encoder. Therefore, $p(y_k^p|d_k = i, S_k = m, S_{k-1} = m')$ can be written as

$$p(y_k^p|d_k = i, S_k = m, S_{k-1} = m') = p(y_k^p|x_k^p = j), \tag{4.5}$$

for $j = 0, 1$. The channel transition probabilities of the undeleted parity bits resemble that of the information bit d_k.

Puncturing is used when the rate is above 1/3. Puncturing will cause some parity bits being unavailable at the decoder. To distinguish between an undeleted and a deleted bit, we can assign the value of the deleted bit with another number which must not be "0" or "1". For example, if we set all the deleted parity bits as 3, the conditional probability of a deleted bit becomes

$$p(y_k^p = 3|d_k = i, S_k = m, S_{k-1} = m') = 1. \tag{4.6}$$

4.3 Shannon limits for the different channels

In general, the capacity of a DMC can be written as [28]

$$\begin{aligned} C &= max_{p(x)} I(X;Y) \\ &= max_{p(x)} H(Y) - H(Y|X) \\ &= max_{p(x)} H(Y) - \sum_x p(x) H(Y|X=x), \end{aligned} \tag{4.7}$$

where $p(x)$ is the probability distribution of the channel input.

The capacity of the BSC, the ZC and the BAC can all be easily derived, and they are [7]

$$C_{BSC} = 1 - h_b(\epsilon), \tag{4.8}$$

$$C_{ZC} = \log_2(1 + 2^{-h_b(\epsilon)/(1-\epsilon)}), \tag{4.9}$$

$$C_{BAC} = \log_2[2^{(1-\beta)h_b(\alpha)-\alpha h_b(\beta)/(\beta+\alpha-1)} + 2^{\beta h_b(\alpha)+(1-\alpha)h_b(\beta)/(\beta+\alpha-1)}], \tag{4.10}$$

where $h_b(.)$ is the binary entropy function. For a given code rate R_c, a tolerable BER P_e, and each channel, the Shannon limit (or OPTA), which is the values of ϵ, α and β, can be obtained by solving

$$R_c[1 - h_b(P_e)] = C_{BSC} = 1 - h_b(\epsilon), \tag{4.11}$$

$$R_c[1 - h_b(P_e)] = C_{ZC} = \log_2(1 + 2^{-h_b(\epsilon)/(1-\epsilon)}), \tag{4.12}$$

$$R_c[1-h_b(P_e)] = C_{BAC} = \log_2[2^{(1-\beta)h_b(\alpha)-\alpha h_b(\beta)/(\beta+\alpha-1)} + 2^{\beta h_b(\alpha)+(1-\alpha)h_b(\beta)/(\beta+\alpha-1)}]. \quad (4.13)$$

For simplicity, the OPTA values of ϵ, α and β are denoted as ϵ_{OPTA}, α_{OPTA} and β_{OPTA} respectively. The values of ϵ_{OPTA}, α_{OPTA} and β_{OPTA} for the different channels are shown in Tables 4.1 to 4.5.

4.4 Uniform capacity for different channels

Uniform capacity of a DMC can be achieved when the channel input is uniform. Thus, it can be written as [28]

$$C = I(X;Y)|_{p(x)=1/2} \quad (4.14)$$

$$= H(Y) - H(Y|X)|_{p(x)=1/2} \quad (4.15)$$

$$= H(Y) - \sum_x p(x)H(Y|X=x)|_{p(x)=1/2}. \quad (4.16)$$

Furthermore, the uniform capacity of the BSC, the ZC and the BAC are

$$C_{uniformBSC} = 1 - h_b(\epsilon), \quad (4.17)$$

$$C_{uniformZC} = h_b\left(\frac{1}{2}(1+\epsilon)\right) - \frac{1}{2}h_b(\epsilon), \quad (4.18)$$

$$C_{uniformBAC} = h_b\left(\frac{1}{2}(1-\alpha+\beta)\right) - \frac{1}{2}h_b(\alpha) - \frac{1}{2}h_b(\beta). \quad (4.19)$$

Similarly, for each channel with a uniform source input, a given code rate R_c and a tolerable BER P_e, we obtain the theoretical values of ϵ, α and β that achieve the uniform capacity; such values are denoted as ϵ_{UNI}, α_{UNI} and β_{UNI} respectively. The values of ϵ_{UNI}, α_{UNI} and β_{UNI} for different channels are also shown in Tables 4.1 to 4.5.

The uniform capacity of the BSC is the same as its capacity. This means that the capacity can be achieved by uniform channel input. On the other hand, the optimal channel input distributions for the ZC and the BAC are non-uniform due to the non-symmetric nature of both channels.

The distribution of the source input may not necessarily be the same as the distribution of the channel input. However, there is a way to determine the optimal source input distribution once we know the optimal channel input distribution. It has been shown that the marginal distribution of the parity sequence generated by an RSC encoder is asymptotically uniform under certain constraints if the source input is either uniform i.i.d. or non-uniform i.i.d [5]. It's trivial to see that if the source input is uniform i.i.d., the channel input is also uniform i.i.d.

Table 4.6 shows the channel input distributions that achieve the capacity for different channels. It is clear that none of the channels under our discussion has heavily bias channel input distributions.

	$R_c = 1/3$	$R_c = 1/2$
ϵ_{UNI}	0.1740	0.1101
ϵ_{OPTA}	0.1740	0.1101
$\epsilon_{UNI}/\epsilon_{OPTA}$	1	1
$10\log_{10}\epsilon_{UNI}/\epsilon_{OPTA}$	0	0

Table 4.1: Values of ϵ_{UNI} and ϵ_{OPTA} for different rates when $P_e = 10^{-5}$ (BSC).

	$R_c = 1/3$	$R_c = 1/2$
ϵ_{UNI}	0.4727	0.2939
ϵ_{OPTA}	0.4860	0.3036
$\epsilon_{UNI}/\epsilon_{OPTA}$	0.9728	0.9679
$10\log_{10}\epsilon_{UNI}/\epsilon_{OPTA}$	-0.1198	-0.1415

Table 4.2: Values of ϵ_{UNI} and ϵ_{OPTA} for different rates when $P_e = 10^{-5}$ (ZC).

	$R_c = 1/3$	$R_c = 1/2$
β_{UNI}	0.4325	0.2607
β_{OPTA}	0.4411	0.2666
β_{UNI}/β_{OPTA}	0.9805	0.9780
$10\log_{10}\beta_{UNI}/\beta_{OPTA}$	-0.08539	-0.09696

Table 4.3: Values of β_{UNI} and β_{OPTA} for different rates when $P_e = 10^{-5}$ and $\alpha = 0.01$ (BAC).

	$R_c = 1/3$	$R_c = 1/2$
β_{UNI}	0.3395	0.1836
β_{OPTA}	0.3425	0.1849
β_{UNI}/β_{OPTA}	0.9913	0.9932
$10\log_{10}\beta_{UNI}/\beta_{OPTA}$	-0.03774	-0.02944

Table 4.4: Values of β_{UNI} and β_{OPTA} for different rates when $P_e = 10^{-5}$ and $\alpha = 0.05$ (BAC).

	$R_c = 1/3$	$R_c = 1/2$
β_{UNI}	0.2603	0.1204
β_{OPTA}	0.2611	0.1204
β_{UNI}/β_{OPTA}	0.9971	0.9998
$10\log_{10}\beta_{UNI}/\beta_{OPTA}$	-0.01245	-0.008601

Table 4.5: Values of β_{UNI} and β_{OPTA} for different rates when $P_e = 10^{-5}$ and $\alpha = 0.1$ (BAC).

	$R_c = 1/3$	$R_c = 1/2$
$p^*(x)$ for BSC	0.5	0.5
$p^*(x)$ for ZC	0.5988	0.5795
$p^*(x)$ for BAC @ $\alpha = 0.01$	0.5795	0.5625
$p^*(x)$ for BAC @ $\alpha = 0.05$	0.5474	0.5302
$p^*(x)$ for BAC @ $\alpha = 0.1$	0.5246	0.5045

Table 4.6: Optimal channel input distributions, $p^*(x)$, that achieve the capacity for ϵ_{OPTA} and β_{OPTA} for different channels when $P_e = 10^{-5}$ and $R_c = 1/3, 1/2$.

Chapter 5

Simulation Results

In this section, we present the simulation results on the performance of Turbo codes of different rates for our considered channels.

5.1 Simulation environment

In the simulations, the Turbo encoder consists of two 16-state RSC elementary encoders. Each of them has an encoder generator (37, 21) as well as rate 1/2. The rate of the Turbo encoder is 1/3 without puncturing, but when puncturing is used, higher rates are available. In all of our simulations, only rates 1/3 and 1/2 are considered. Pseudo-random interleaving is implemented. The sequence length N of 65536 is used. The number of blocks and iterations are set to 3200 and 20 respectively. All simulation results are plotted as BER versus crossover probability.

The simulation is done in C. For the BSC, an incorrect transmission occurs with probability ϵ. This can be realized by

$$y = \begin{cases} x & \text{if } drand48() \geqslant \epsilon, \\ (x+1) \quad mod2 & \text{if } drand48() < \epsilon, \end{cases}$$

where x denotes the encoded bit before transmission, and y denotes the received bit after transmission. "$drand48()$" is a function which returns a floating-point value uniformly distributed over the interval [0.0, 1.0].

Similarly, the channel model for the ZC can be realized by

If $x = 0$, then $y = x$;

if $x \neq 0$, then

$$y = \begin{cases} x & \text{if } drand48() \geqslant \epsilon, \\ (x+1) \quad mod2 & \text{if } drand48() < \epsilon. \end{cases}$$

Finally, here is the realization of the channel model for the BAC:

if $x = 0$,

$$y = \begin{cases} x & \text{if } drand48() \geqslant \alpha, \\ (x+1) \quad mod2 & \text{if } drand48() < \alpha. \end{cases}$$

if $x \neq 0$,

$$y = \begin{cases} x & \text{if } drand48() \geqslant \beta, \\ (x+1) \quad mod2 & \text{if } drand48() < \beta. \end{cases}$$

5.2 Turbo codes performance on different channels

In this section, we present our simulation results on the BSC, ZC and BAC. All simulations use $N = 65536$ with 3200 blocks and 20 iterations. Only rates 1/3 and 1/2 are considered. The BER of our interest is at the 10^{-5} level. Source input is chosen to be uniform for all channels. Therefore, the performance evaluation is conducted by comparing the OPTA crossover probabilities with the simulated values of the crossover probabilities at the 10^{-5} level.

For the BSC and the ZC, ϵ is the only parameter that represents the crossover probability in our simulations; on the other hand, for the BAC, there are two crossover probability parameters, denoted as α and β, in our simulations. Since the BAC has two parameters, we evaluate its performance by comparing β_{OPTA} with the simulated β when α is set at 0.01, 0.05 and 0.1.

5.2.1 Turbo codes for the BSC

In the simulations on the BSC, the sample points are chosen by the values of the crossover probabilities, ϵ. According to the OPTA ϵ values in Table 4.1, when the rate is 1/3, we select in our simulation ϵ in the range between 0.147 and 0.159; for rate equals to 1/2, we select ϵ in the range between 0.081 and 0.093.

From Table 5.1, we can see that when the rate increases, the value of ϵ decreases. The result is consistent with what we obtain from the theoretical calculations. At rate 1/3, both ϵ/ϵ_{UNI} and ϵ/ϵ_{OPTA} are 0.8823. At rate 1/2, both ϵ/ϵ_{UNI} and ϵ/ϵ_{OPTA} are 0.7941. Therefore, when the rate is high, the value of ϵ is further away from the Shannon limit.

5.2.2 Turbo codes for the ZC

Like the BSC, in the simulations on the ZC, the sample points are chosen by the values of the crossover probabilities, ϵ. According to the OPTA ϵ values in Table 4.2, when the rate is 1/3, we select in our simulation ϵ in the range between 0.425 and 0.433; for rate equals to 1/2, we select ϵ in the range between 0.253 and 0.269.

From Table 5.2, we can again see that when the rate increases, the value of ϵ decreases. The result is also consistent with what we obtain from the theoretical calculations. At rate 1/3, ϵ/ϵ_{UNI} is 0.9069, and ϵ/ϵ_{OPTA} is 0.8822. At rate 1/2, ϵ/ϵ_{UNI} is 0.8799, and ϵ/ϵ_{OPTA} is 0.8517. As a result, when the rate is high, the value of ϵ is slightly further away from the Shannon limit.

5.2.3 Turbo codes for BAC

At each α level, the samples points of the simulations are chosen by the values of the crossover probabilities, β. At $\alpha = 0.01$, according to the OPTA β values in Table 4.3, when the rate is 1/3, we select β in the range between 0.393 and 0.405; for rate equals to 1/2, we select β in the range between 0.204 and 0.212. At $\alpha = 0.05$, according to the OPTA β values in Table 4.4, when the rate is 1/3, we select β in the range between 0.288 and 0.296; for rate equals to 1/2, we select β in the range between 0.117 and 0.133. At $\alpha = 0.1$, according to the OPTA β values in Table 4.5, when the rate is 1/3, we select β in the range between 0.206 and 0.222; for rate equals to 1/2, we select β in the range between 0.061 and 0.077.

From Table 5.3 to 5.5, we can once again see that when the rate increases, the value of β decreases at different α levels. This relationship is also true for the theoretical values. From Table 5.3, at $\alpha = 0.01$, β/β_{UNI} is 0.9183, and β/β_{OPTA} is 0.9004 when the rate is 1/3. When the rate is 1/2, β/β_{UNI} is 0.8037, and β_{OPTA} is 0.7859. From Table 5.4, at $\alpha = 0.05$, β/β_{UNI} is 0.8624, and β/β_{OPTA} is 0.8549 when the rate is 1/3. When the rate is 1/2, β/β_{UNI} is 0.6644, and β/β_{OPTA} is 0.6599. From Table 5.5, at $\alpha = 0.1$, β/β_{UNI} is 0.8136, and β/β_{OPTA} is 0.8113 when the rate is 1/3. When the rate is 1/2, β/β_{UNI} is 0.5497, and β/β_{OPTA} is 0.5496. Thus, the simulation results show that the performance degrades significantly when the rate increases.

	$R_c = 1/3$	$R_c = 1/2$
ϵ	0.1535	0.0874
ϵ_{UNI}	0.1740	0.1101
ϵ_{OPTA}	0.1740	0.1101
$\epsilon_{UNI}/\epsilon_{OPTA}$	1	1
$10\log_{10}\epsilon_{UNI}/\epsilon_{OPTA}$	0	0
ϵ/ϵ_{UNI}	0.8823	0.7941
$10\log_{10}\epsilon/\epsilon_{UNI}$	-0.5439	-1.0011
ϵ/ϵ_{OPTA}	0.8823	0.7941
$10\log_{10}\epsilon/\epsilon_{OPTA}$	-0.5439	-1.0011

Table 5.1: Comparisons between the simulated ϵ values and the theoretical $\epsilon values$, ϵ_{UNI} and ϵ_{OPTA}, for different rates when $P_e = 10^{-5}$ (BSC).

	$R_c = 1/3$	$R_c = 1/2$
ϵ	0.4287	0.2586
ϵ_{UNI}	0.4727	0.2939
ϵ_{OPTA}	0.4860	0.3036
$\epsilon_{UNI}/\epsilon_{OPTA}$	0.9728	0.9679
$10\log_{10}\epsilon_{UNI}/\epsilon_{OPTA}$	-0.1198	-0.1415
ϵ/ϵ_{UNI}	0.9069	0.8799
$10\log_{10}\epsilon/\epsilon_{UNI}$	-0.4244	-0.5557
ϵ/ϵ_{OPTA}	0.8822	0.8517
$10\log_{10}\epsilon/\epsilon_{OPTA}$	-0.5442	-0.6972

Table 5.2: Comparisons between the simulated ϵ values and the theoretical ϵ values, ϵ_{UNI} and ϵ_{OPTA}, for different rates when $P_e = 10^{-5}$ (ZC).

	$R_c = 1/3$	$R_c = 1/2$
β	0.3972	0.2095
β_{UNI}	0.4325	0.2607
β_{OPTA}	0.4411	0.2666
β_{UNI}/β_{OPTA}	0.9805	0.9780
$10\log_{10}\beta_{UNI}/\beta_{OPTA}$	-0.08539	-0.09696
β/β_{UNI}	0.9183	0.8037
$10\log_{10}\beta/\beta_{UNI}$	-0.3701	-0.9491
β/β_{OPTA}	0.9004	0.7859
$10\log_{10}\beta/\beta_{OPTA}$	-0.4555	-1.0461

Table 5.3: Comparisons between the simulated β values and the theoretical β values, β_{UNI} and β_{OPTA}, for different rates when $P_e = 10^{-5}$ and $\alpha = 0.01$ (BAC).

	$R_c = 1/3$	$R_c = 1/2$
β	0.2928	0.1220
β_{UNI}	0.3395	0.1836
β_{OPTA}	0.3425	0.1849
β_{UNI}/β_{OPTA}	0.9913	0.9932
$10\log_{10}\beta_{UNI}/\beta_{OPTA}$	-0.03774	-0.02944
β/β_{UNI}	0.8624	0.6644
$10\log_{10}\beta/\beta_{UNI}$	-0.6431	-1.7757
β/β_{OPTA}	0.8549	0.6599
$10\log_{10}\beta/\beta_{OPTA}$	-0.6808	-1.8052

Table 5.4: Comparisons between the simulated β values and the theoretical β values, β_{UNI} and β_{OPTA}, for different rates when $P_e = 10^{-5}$ and $\alpha = 0.05$ (BAC).

	$R_c = 1/3$	$R_c = 1/2$
β	0.2118	0.0662
β_{UNI}	0.2603	0.1204
β_{OPTA}	0.2611	0.1204
β_{UNI}/β_{OPTA}	0.9971	0.9998
$10\log_{10}\beta_{UNI}/\beta_{OPTA}$	-0.01245	-0.008601
β/β_{UNI}	0.8136	0.5497
$10\log_{10}\beta/\beta_{UNI}$	-0.8957	-2.5986
β/β_{OPTA}	0.8113	0.5496
$10\log_{10}\beta/\beta_{OPTA}$	-0.9082	-2.5995

Table 5.5: Comparisons between the simulated β values and the theoretical β values, β_{UNI} and β_{OPTA}, for different rates when $P_e = 10^{-5}$ and $\alpha = 0.1$ (BAC).

Chapter 6

Conclusion and Future Work

6.1 Conclusions of the project

Basic concepts of Turbo encoding are described. The basic structures of Turbo decoders are presented together with the derivation of the M-BCJR decoding algorithm. The principle of iterative decoding is discussed.

The application of Turbo codes to different channels is investigated, and our simulation results are shown. The performance of Turbo codes for BSC, ZC and BAC with rates from 1/3 to 1/2 is examined.

For each asymmetric channel under our investigation, the ratio of the simulated crossover probability to its corresponding OPTA is only a few percents away from 0.94 at $rate = 1/3$;on the other hand, the ratio becomes significantly less than 0.94

at $rate = 1/2$. As a result, we can see that as the rate increases, the performance of Turbo codes degrades over binary-input binary-output discrete memoryless asymmetric channels such as ZC and BAC.

6.2 Future Work

Turbo codes remains a great area for future research. There are still many aspects of Turbo codes that need to be explored.

- Turbo codes over channels with memory is an area that requires more attention. Gracia-Frias et al. did some research work on the Gilbert Elliot channel model [11,12]. Alajaji and Fuja considered the binary finite-memory Polya contagion channel model, which contains fewer parameters to work with compared to the Gilbert Elliot channel model [13]. As a result, it will be interesting to study Turbo codes over the binary finite-memory Polya contagion channel model, and the study can be beneficial for the design of powerful coding systems for channels with correlated fading.

- There has been little research for designing Turbo codes over non-binary channels. In [6], McEliece believe that Turbo codes perform effectively over arbitrary q-ary input memoryles channels based on the old results of Gallager [14]. Therefore, we propose designing Turbo codes over q-ary DMCs as well as continuous memoryless channels.

- To optimize the performance of Turbo codes over various source distributions, different encoders must be used. Zhu et al. previously investigated a variety of encoders for different source distributions over AWGN and Rayleigh fading channels [15-19], but encoders design for heavily biased sources remain an open question. Hence, Turbo codes design for such heavily biased sources should be an exciting as well as a challenging area to work on.

Bibliography

[1] C.E. Shannon, "A mathematical theory of communications," *Bell Syst. Tech J.*, vol. 27, pp. 379–423 and pp. 623–656, Jul. and Oct. 1948.

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

[3] C. Heegard and S. B. Wicker, *Turbo Coding*, Kluwer Academic Publishers, 1999.

[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.

[5] G.-C. Zhu, *Joint Source-Channel Coding via Turbo Codes*. Queen's Ph.D. thesis, 2003.

[6] R. J. McEliece, "Are Turbo-like codes effective on nonstandard channels?" *IEEE Inform. Theory Soc. Newsletter*, vol. 51, pp. 1–8, Dec. 2001.

[7] R. Silverman, "On binary channels and their cascades," *IRE Trans. Inform. Theory*, vol. IT-1, pp. 19–27, Dec. 1955.

[8] E. Majani, *A Model for the Study of Very Noisy Channels, and Applications.* Caltech Ph.D. thesis, 1988.

[9] N. Shulman, "Universal encoding–the prior problem," preprint dated June 26, 2001.

[10] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.

[11] J. Garcia-Frias, and J. D. Villasenor, "Turbo decoders for Markov channels," *IEEE Commun. Lett.*, vol. 2, pp. 257–259, Sept. 1998.

[12] J. Garcia-Frias, and J. D. Villasenor, "Turbo decoding of Gilbert-Elliot channels," *IEEE Trans. Commun.*, vol. 50, pp. 357–363, Mar. 2002.

[13] F. Alajaji and T. E. Fuja, "A communication channel modeled on contagion," *IEEE Trans. on Inform. Theory*, vol. 40, pp. 2035–2041, Nov. 1994.

[14] R. G. Gallager, *Information Theory and Reliable Communication.* New York: Wiley, 1968.

[15] G.-C. Zhu, F. Alajaji, J. Bajcsy and P. Mitran, "Transmission of Non-Uniform Memoryless Sources via Non-Systematic Turbo Codes," *IEEE Transactions on Communications*, to appear.

[16] G.-C. Zhu, F. Alajaji, J. Bajcsy and P. Mitran, "Transmission of Non-Uniform Memoryless Sources over Wireless Fading Channels via Non-Systematic Turbo

Codes," *Proceedings of IASTED Wireless Optical Comunication Conference (WOC'2002)*, pp. 119–124, Banff, Alberta, Canada, Jul. 17-19, 2002.

[17] G.-C. Zhu, F. Alajaji, J. Bajcsy and P. Mitran, "Non-Systematic Turbo Codes for non-uniform i.i.d. sources over AWGN channels," *Proceedings of the 2002 Conference on Information Sciences and Systems (CISS)*, pp. 770–774, Princeton, NJ, Mar. 20-22, 2002.

[18] G.-C. Zhu and F. Alajaji, "Turbo Codes for Non-uniform Memoryless Sources over Noisy Channels," *IEEE Commun. Lett.*, vol. 6, pp. 64–66, Feb. 2002.

[19] G.-C. Zhu and F. Alajaji, "Design of Turbo Codes for Non-Equiprobable Memoryless Sources," *Proceedings of the 39th Annual Allerton Conference on Communication, Control and Computing*, pp. 1253–1262, Monticello, IL, Oct. 3-5, 2001.

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

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

[22] 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.

[23] L. C. Perez, J. Seghers and D. J. Costello, Jr., "A distance spectrum interpretation of turbo codes," *IEEE Trans. on Inform. Theory*, vol. 42, No. 6, pp. 1698–1709, Nov. 1996.

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

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

[26] G.-C. Zhu, *Performance Evaluation of Turbo Codes*. Queen's M.Sc. thesis, 1998.

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

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

[29] J. H. van Lint, *Introduction to Coding Theory*, Springer Verlag, 1999.

[30] N. J. A. Sloane and F. J. MacWilliams, *The Theory of Error-Correcting Codes*, North-Holland, 1983.