Performance of Turbo Codes: The Finite Length Case

by

Ali Özgür Yılmaz

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering : Systems)
in The University of Michigan
2003

Doctoral Committee:
Professor Wayne E. Stark, Chair
Assistant Professor Achilleas Anastasopoulos
Associate Professor Satinder Singh Baveja
Associate Professor Kim Winick

To my family.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my advisor, Prof. Wayne E. Stark. The time I spent working with him has been a great opportunity for me to learn about various subjects in communication. His mentorship style sets a good example and inspiration for my future professional life. Prof. Stark's enthusiasm and patience have cleared many of the difficulties I have had over the course of my studies. I also thank him for his invaluable technical and editorial suggestions for this write-up.

I would like to thank Professors Achilleas Anastasopoulos, Satinder Singh Baveja, and Kim Winick for their constructive comments on my thesis. This final form of the thesis wouldn't be possible without their insightful suggestions. Their editorial directions shaped the thesis into a more solid form.

My thanks go to the National Science Foundation (ECS-9979347) and Office of Naval Research (N00014-03-1-0232) for their financial support during my graduate studies. I also thank Prof. Semyon Meerkov with whom I worked as a graduate student instructor for five terms while working on this research. His work ethics has had a big impact on me.

I would also like to thank my colleagues John D. Choi, Roby Gupta, Tingfang Ji, Troy Nolan, Hua Wang, Andrew Worthen, Kar-peo Yar, Ping-Cheng Yeh, Do-Sik Yoo, and Salam Zummo with whom I worked in the Wireless Communications Lab. I have benefited greatly from the nurturing and enjoyable environment we had in our

lab. All the discussions and exchange of ideas have been quite helpful and fruitful. I especially thank Ping-Cheng Yeh for our co-work which resulted in the writing of Chapter 4.

Finally, I would like to thank my parents, my sister, the members of the Özgün and Koca families, and my beloved Bahar for their constant support and encouragement. Nothing would be meaningful, if possible at all, without their love and caring. Many thanks go to my friends in Ann Arbor who made each of the seven years I spent here worthwhile. I am indebted to my friends who have provided continuous support in the last month of my graduate studies.

Thank you all...

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF APPENDICES

APPENDIX

CHAPTER 1

Introduction

The last century witnessed a significant accumulation of knowledge in science and technology. Particularly, the means of communication developed since the late 1800's have had a huge impact on the every day lives of people. Pushed by the demand for rapid exchange of information, future communication systems will be designed to provide substantial improvement over the current ones. Our work aims to analyze in practical settings turbo codes, one of the prospective key components in the future communication technologies.

1.1 The Communication Problem from the Channel Coding Perspective

The overall goal of communication is to communicate information from one point to another. The parameters important to a communication system engineer include signal power, noise power, reliability, bandwidth, delay, complexity, overall network throughput, cost/complexity etc. Our work is particularly related to channel coding, which is the area of communication that is concerned with the effects of the physical medium (channel) on communication. The research in channel coding establishes principles for communication. These principles help develop techniques to facilitate communication with some desired characteristics in certain media.

The basic approach to channel coding is to add structured redundancy in order to

fight against the destructive effects which the channel might impose on the data. A generic communication system model is shown in Fig. 1.1. The information sequence desired to be communicated is encoded by a channel encoder. The channel encoder maps the data sequence u into an encoded sequence c. The encoded sequence is modulated to produce x in order to transfer the coded data into a form that can be transmitted in the physical medium. Redundancy may be added during either the encoding or the modulation operations. The channel may distort the transmitted signal. The central communication problem is to accurately reconstruct the original data by processing the received signal y.

Figure 1.1: A generic communications system.

In his pioneering work, Shannon derived some of the fundamental limits for reliable communication in certain channels. He determined the tradeoff between energy efficiency and bandwidth efficiency for arbitrarily reliable communications without regard to other parameters including delay and complexity. Shannon used a non-constructive method to find the limits to reliable communication. After his formalization, the first step taken was constructing codes of fixed data sequence lengths (block length) with algebraic properties in order to approach the limits. Although other classes of codes, e.g., convolutional codes, were invented later on , most research were on finding codes with some algebraic properties. This emphasis on algebraic structure was prevalent until the paradigm shift that came with the success of turbo codes, which were developed without using diligent mathematical theory.

Turbo codes not only provide excellent error rate performance close to the limits predicted by Shannon but also they achieve such good performance with reasonable complexity. Iterative decoding techniques which is central to the turbo coding concept have been known to the communications research community at least since the 1960's [1]. However, they did not attract much attention possibly due to their substantial complexity in comparison to the state of computing at those times. It can be argued

that the introduction of parallel concatenated codes, a.k.a. turbo codes, has brought the attention iterative techniques now have.

1.2 Motivation and Thesis Overview

Turbo codes attracted much interest in the coding research community due to their close performance to the limits predicted by Shannon. Many turbo coding schemes have been proposed that perform quite close to the Shannon's limits especially in an additive white Gaussian noise (AWGN) channel. Turbo codes are obtained by the parallel concatenation of multiple codes [2]. In a turbo encoder there is a component encoder corresponding to each code. Different versions of the same data sequence are obtained by permuting the data. This permutation operation is called interleaving and done by interleavers. The interleaved data sequences are separately fed into the component encoders.

Optimal decoders for turbo codes may not be practical because of the complexity involved. Thus decoding is performed through a suboptimal iterative decoding algorithm. The suboptimal decoder works as follows. There are multiple component decoders corresponding to each component code. Each component decoder produces and exchanges likelihood information about the data. Ideally, each iteration improves the reliability of the likelihood information. After a number of iterations an estimate of the data can be constructed based on the generated likelihood information.

The good performance achieved by turbo codes without very high complexity was a mystery to the coding research community in the first years following their appearance. This initial period was marked by numerous simulation-based research efforts to understand turbo codes and improve their performance. Further studies revealed that their good performance is owing to the interleaver in the turbo encoder and the iterative decoding algorithm. It was also realized that the existence of the interleaver and the ad hoc nature of iterative decoding make analytical evaluation of turbo code performance extremely difficult. In this thesis we will study how these two features, interleaving and iterative decoding, affect turbo codes' performance.

It was shown that good performance (in the high energy regime) achieved by turbo codes is due to the interleaving operation [3, 4]. It is also observed that some interleavers are better than the others [3]. However, there have not been many studies that attempt to analyze the performance of a turbo code with a specific interleaver. The distance spectrum of a turbo code can be used to analytically evaluate the performance, but interleaving in the turbo encoder makes the task of finding the distance spectrum very hard. Yet, an estimate of the distance spectrum can be obtained by focusing on some special data sequences. As a part of this thesis, we will propose an efficient method to estimate the distance spectrum of a turbo code with a specific interleaver.

Iterative decoding is a suboptimal algorithm which performs quite well for turbo codes. It is difficult to analyze an iterative decoder due to the exchange of information. Analysis of iterative decoding techniques has attracted much attention in the recent years [5–7]. The common approach in iterative decoding analysis is studying the iterative decoding process with a probabilistic model. However, all the attempts to analyze iterative decoding are for infinitely long data sequences. Although the analysis for infinite lengths can provide some design rules to construct turbo codes of large block lengths, use of these codes in practice most often requires small to medium-size block lengths for data sequences. Hence, we will focus on iterative decoding for finite block lengths in this thesis. We will study the properties of decoding modules in an iterative decoder. Through this study, an approximation to a turbo code's finite-length performance will be obtained for the low/medium energy regime.

There are many pending questions associated with iterative decoding. One of them is whether and under what conditions iterative decoding provides optimal solutions. Although some properties of iterative decoding are revealed [8, 9], there is currently no complete answer to this question. Research on this topic is still an open area. Meanwhile, researchers continue to empirically investigate some features of iterative decoding [9–11].

We will propose a classification of a turbo decoder's convergence characteristics. Our classification relates the convergence characteristics to whether an optimal (or

possibly optimal) solution is obtained by a turbo decoder. The information produced by a turbo decoder has different properties when the decoder can obtain an optimal solution and when it can not. So, there are basically two modes of a turbo decoder: optimal solution mode and decoding failures. This bimodal behavior can also be observed from the error rate performance. When the error rate of a turbo code is plotted against the energy allocated for data, mainly two distinct regions of operation are observed. One is the waterfall region where there is a rapid improvement of error rate with increasing energy. The other one is the error floor region where the error rate does not decrease that rapidly with increasing energy. The error floor region corresponds to higher values of signal-to-noise power ratio (SNR), whereas the waterfall region takes place in the low/medium SNR region.

We will study different convergence types and relate them to the two basic modes in Chapter 3. These two modes of operation need different analysis tools since the underlying phenomenon in each case is different. The error floor analysis is based on the properties of the turbo code structure, whereas the analysis for the waterfall region is related with the iterative decoding. The rest of the thesis will be devoted to the development of tools to analyze turbo code performance in each region.

For any given code, optimal decoding rules and corresponding algorithms are defined for a given entity. For example, an optimal decoder minimizes the probability of error of wrong estimation for the whole sequence, or alternatively for the individual bits of a sequence. By their description turbo decoders provide an approximation to a bit-optimal decoder. However, performance of bit-optimal decoders are hard to analyze. There exist efficient techniques that approximate the performance of sequence-optimal decoders [12]. Fortunately, bit-optimal and sequence-optimal decoders are equivalent at high values of energy per data. Hence, techniques for sequence-optimal decoders can be utilized to study the performance of turbo decoders in the error floor region where iterative decoding is conjectured to be optimal in the literature.

One technique used for the analysis of sequence-optimal decoders is the union bound technique [12]. In order to apply this technique, the distance spectrum of a code is needed. The distance spectrum of a code enumerates all the codewords of a

code with their weights. In Chapter 4 we will propose a method to obtain an estimate to the distance spectrum of a turbo code. Our method is based on decreasing the set of codewords to be investigated in order to obtain the most important part of the distance spectrum of a given turbo code. We will also describe the relation between bit-optimal and sequence-optimal decoders, the union bound technique and study some limitations of the proposed method in Chapter 4.

Iterative decoding algorithms have been investigated with probabilistic models [5, 13]. The likelihood information exchanged between modules are regarded as random variables and evolution of their probability densities are observed in order to gain more insight about iterative decoding. The likelihood information passed between the component decoders of a turbo decoder is approximated as a normally distributed random variable [2, 7]. For the case of asymptotically large block lengths, the use of this Gaussian approximation enables to obtain a decoding threshold for a turbo code above which the code can achieve arbitrarily small error rates.

Throughout this thesis we are interested in the performance of turbo codes of practical block lengths for delay reasons. This is why we will investigate the effect of finite block lengths on behavior of component decoders in Chapter 5. By focusing on the information exchanged between the component decoders, we will explore some properties of component decoders. In particular, we will study how the likelihood information is correlated. The behavior of a component decoder will be modelled with a random process. It is found out that the correlation in the likelihood information produced by a component decoder can be used to model the component decoder. These improved models allow us to accurately approximate the performance of turbo codes in the waterfall region in Chapter 6. The approximation is obtained through an extension of a method to find decoding thresholds for infinitely large block lengths [6]. A probabilistic model for the iterative decoding process will be developed to obtain error rate approximations for turbo codes of practical block lengths. We will present numerical results that compare the simulation results to the approximations obtained with the proposed method in Chapter 6. This comparison shows that the error rate performance of an iterative decoder for any given block length can be accurately

estimated by the proposed probabilistic model.

CHAPTER 2

Turbo Codes

2.1 Introduction

Turbo codes were introduced by Berrou et al. [2]. The original designs in [2] provided performance close to the channel capacity and generated much interest in turbo-like codes and iterative decoding in the communications community. Turbo codes are obtained by parallel concatenation of multiple component codes. Hence, there are multiple encoders corresponding to each component code inside a turbo encoder. The component codes can be chosen among any type of the channel codes. However, convolutional codes have been the choice of many researchers mainly due to easy to implement decoding algorithms that are available for them. We also concentrate on convolutional codes in this study but the results are not necessarily limited to turbo codes with convolutional component codes.

As pointed out in [8, 14] the term "turbo" is actually a misnomer in reference to the code itself. "Turbo" refers to the iterative decoding algorithm where it is in analogy with the reutilization of output materials in some mechanical systems. The iterative decoding algorithm of turbo codes, often referred to as turbo decoding, exchanges information between the decoders corresponding to each component code. It is actually the utilization of such an iterative decoding algorithm that enables the good performance of turbo codes while the complexity is kept small.

The original design included two identical recursive systematic convolutional (RSC)

codes as the component codes. The use of RSC codes was justified in the subsequent works [15,16]. We will also base our study on RSC codes. Using different component codes (asymmetric turbo codes) rather than identical have recently been considered in a number of studies [17,18]. In this study we will often concentrate on symmetric turbo codes and pursue the extension to the asymmetric case in a few examples. The system model will be described in Section 2.2. Section 2.3 will be devoted to convolutional codes and some of their properties. In Section 2.4 MAP decoding of convolutional codes will be explained. Encoding and decoding of turbo codes will be described in Sections 2.5 and 2.6 respectively.

2.2 The System Model

This section is devoted to the description of modulation/demodulation and the channel definition used in this study. We will only consider binary codes in this study in which all the symbols in a data or coded sequence come from the set $\{0,1\}$. Whenever a coded sequence c (Fig. 1.1) is available, this sequence is modulated to be sent over the channel. We have used binary shift phase keying (BPSK) modulation throughout this study. For the time duration T during which c_l, the l^{th} bit in sequence c, is transmitted the corresponding modulated signal x_l (in the baseband) to be transmitted over the channel can be written as

$$x_l = \sqrt{E_s}(-1)^{c_l}, \tag{2.1}$$

where E_s is the energy of signal allocated per each coded bit. The sequence x is then passed through the channel. The additive white Gaussian noise channel (AWGN) has been the primary interest in the literature with respect to turbo code performance. Our work is also based on this particular channel. For the purposes of this work, the AWGN channel can be modelled as a discrete system by

$$y_l = x_l + n_l, \tag{2.2}$$

where y_l is the received signal corresponding to x_l and n_l models the destructive effect of the channel as an additive Gaussian noise process with

$$E[n_l] = 0, \tag{2.3}$$

$$E[n_l n_{l+k}] = \frac{N_0}{2}\delta(k), \tag{2.4}$$

for all l and k, and $\delta(\cdot)$ denotes the Kronecker delta function. In this case, the conditional probability density of y_l can be written as

$$f(y_l \mid x_l) = \frac{1}{\sqrt{\pi N_0}} \exp -\frac{||y_l - x_l||^2}{N_0}, \tag{2.5}$$

where $||\cdot||$ denotes the Euclidean distance and defined for two $N \times 1$ vectors x and y by

$$||y - x||^2 = \sum_{k=1}^{N} |y_k - x_k|^2. \tag{2.6}$$

Throughout this study, we will denote probability densities by $f(\cdot)$. The probability of an event will be expressed as $P(\cdot)$ whereas the probability mass function of a random variable will be denoted by $p(\cdot)$. The subscript notation, such as $f_{Y|X}(y \mid x)$, for these functions will be avoided unless necessary. An important parameter for system characterization is the signal-to-noise ratio (SNR) which is defined as the ratio E_s/N_0.

2.3 Convolutional Codes

Convolutional codes provide a channel coding scheme in which redundancy is produced sequentially based on the current data in the sequence and the state which have been formed by all the previous data. Data can be practically encoded by passing the data through a linear finite-state shift register [12]. A convolutional code with n shift registers is called an n-memory code. Equivalently, it is referred as a 2^n-state code since n shift registers can accommodate 2^n states when binary data is stored in the shift registers.

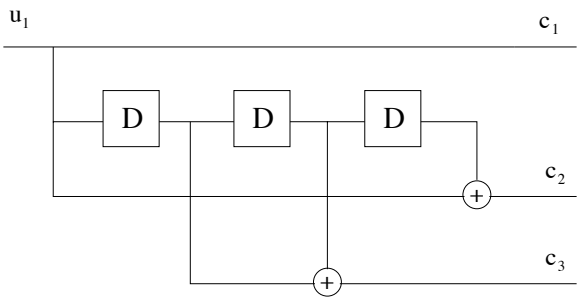


Figure 2.1: A convolutional code with 3 shift registers.

As seen in Fig. 2.1, the output of the encoder is obtained solely based on the current input and the current state of the encoder, i.e., the content of the shift registers. At the same time the state of the encoder is updated as well. This update causes the output sequence to be a function of the whole data sequence rather than individual data.

Systematic convolutional codes are those that include the data sequence directly as part of the encoder output. The term "recursive" refers to the method of updating the encoder's shift registers. In a recursive encoder the next state depends not only the current data but also on the encoder's current state. Component codes of turbo codes have been predominantly chosen to be systematic recursive convolutional codes.

The input to the encoder is called the data sequence and it consists of data bits (only binary codes will be considered in this study) $(u_1^1, u_2^1, \ldots, u_s^1, u_1^2, \ldots, u_s^k, \ldots)$. In general, s data bits $(u_1^k, u_2^k, \ldots, u_s^k)$ are input to the encoder at each run of the encoder and n bits are obtained as the output. If c_l^m denotes the l^{th} coded bit at time m then we can express c_l^m for a nonrecursive convolutional code as

$$c_l^m = \sum_{j=1}^{s} \sum_{i=0}^{n-1} g_{lj}^i u_j^{m-i}, \tag{2.7}$$

where n is the number of shift registers in the encoder and the addition is modulo-2 [14]. The values g_{lj}^i are different for each different code. Although more general versions of (2.7) can be used, (2.7) will be sufficient for the purposes of this work.

The encoder structure can also be represented as a generator polynomial by using

the delay operator D. Multiplication by D is defined in the following way:

$$Du_j^i = u_j^{i-1}. \tag{2.8}$$

Using this notation (2.7) can be written as

$$c_l^m = \sum_{j=1}^{s} [g_{lj}^0 + g_{lj}^1 D + \ldots + g_{lj}^{n-1} D^{n-1}] u_j^m. \tag{2.9}$$

A special form of recursive systematic codes have been extensively used in the turbo code literature. This special form is shown in Fig. 2.2. It is most often represented by the generator matrix

$$[1, \frac{\sum_{i=0}^{n} g_f^i D^i}{1 + \sum_{i=1}^{n} g_b^i D^i}], \tag{2.10}$$

where the first term ('1') refers to the systematic structure, g_f^i and g_b^i's are the coefficients of the forward and backward generator polynomials of the code.

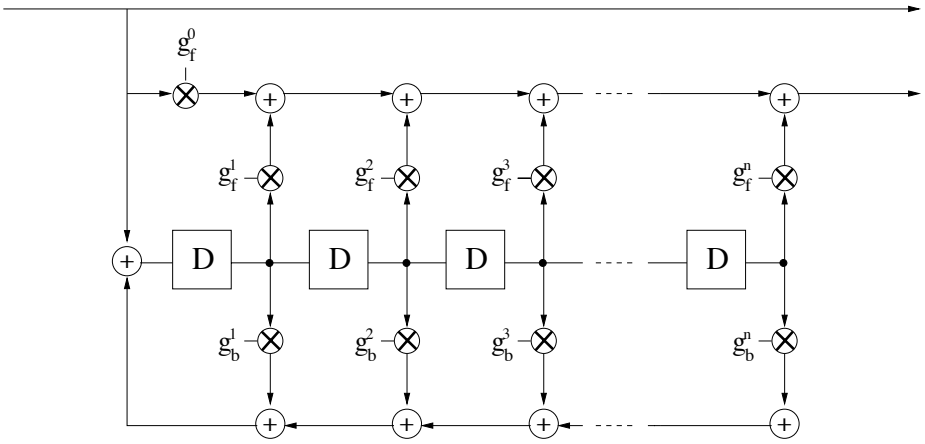


Figure 2.2: The special form of RSC codes used in turbo codes (Both the addition and multiplication operations are binary.)

Another representation of interest is the trellis diagram. Trellis diagram of a convolutional code shows all the possible states, e.g., content of the shift registers, at each time and the possible transitions between these states. This representation helps easier understanding of the decoding algorithms and will be explained in the

next section.

2.4 Decoding of Convolutional Codes

Inside a turbo decoder bit or symbol information is exchanged since passing information on each data sequence from one decoder to the other is not practical. Thus, we will concentrate on symbolwise/bitwise maximum *a posteriori* (MAP) decoding of convolutional codes. We are more interested in the *a posteriori* probability of data symbols/bits rather than their most likely values since there will be more information loss in the iterative decoding process otherwise.

We will call the content of the shift registers the state and will represent the encoder with the state diagram of a finite state machine (FSM). An example of such a state diagram with its corresponding encoder is given in Fig. 2.3. The information attached to each transition reveals the corresponding input and outputs.

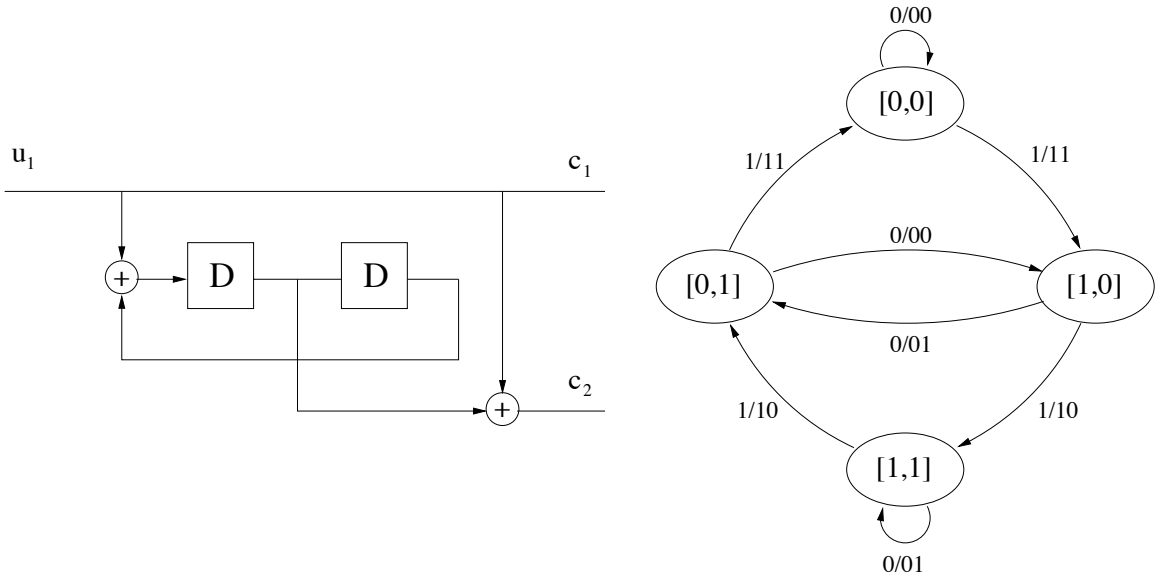


Figure 2.3: A recursive systematic encoder with 2 shift registers and the corresponding state diagram.

The state diagram can be extended to a "trellis diagram" showing all the possible states at each time. Trellis diagram of the encoder in Fig. 2.3 is given in Fig. 2.4 where S_i stands for the state at time i . All the possible paths of a data sequence can

be obtained from the trellis diagram.

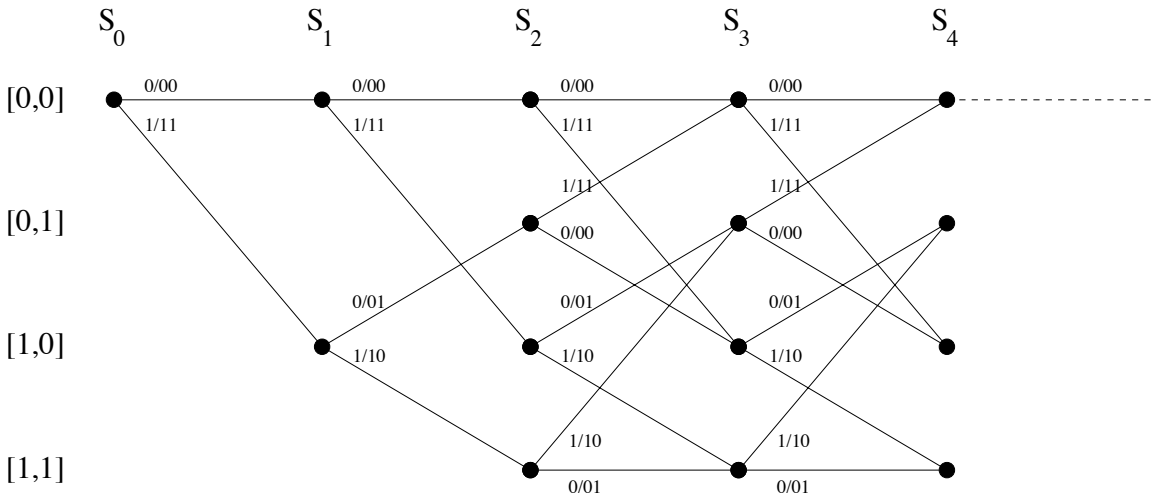


Figure 2.4: A trellis diagram.

A posteriori probability of a data symbol $P(u_l = i \mid y)$ [1] can be efficiently obtained by employing the BCJR algorithm [19]. The symbolwise MAP decoding is provided in Appendix A. The basic idea is marginalization over all the possible paths that would be possible with $u_l = i$ (Fig. 2.5). Consider all paths $P(m', m)$ that would have a particular transition (m', m) at time l (m' and m are states of the encoder). BCJR algorithm proposes an efficient way to find the sum of likelihoods of all the paths in $P(m', m)$. As a summary of Appendix A, we will provide the functions that are used to obtain $P(d_l = i \mid y)$.

$$\gamma_l(m', m) = p(u_l) f(y_l \mid m', m) \tag{2.11}$$

$$\alpha_l(m) = \sum_{m'} \alpha_{l-1}(m') \gamma_l(m', m) \tag{2.12}$$

$$\beta_l(m) = \sum_{m'} \beta_{l+1}(m') \gamma_{l+1}(m, m'). \tag{2.13}$$

$$P(u_l = i \mid y) = \sum_{(m', m): u_l = i} \alpha_{l-1}(m') \gamma_l(m', m) \beta_l(m). \tag{2.14}$$

[1] We will define $P(X \epsilon E \mid Y = y) \triangleq \frac{f_{Y|X}(y|X \epsilon E) P(X \epsilon E)}{f_Y(y)}$ for two random variables X, Y, and an event E of nonzero probability when Y is a continuous random variable.

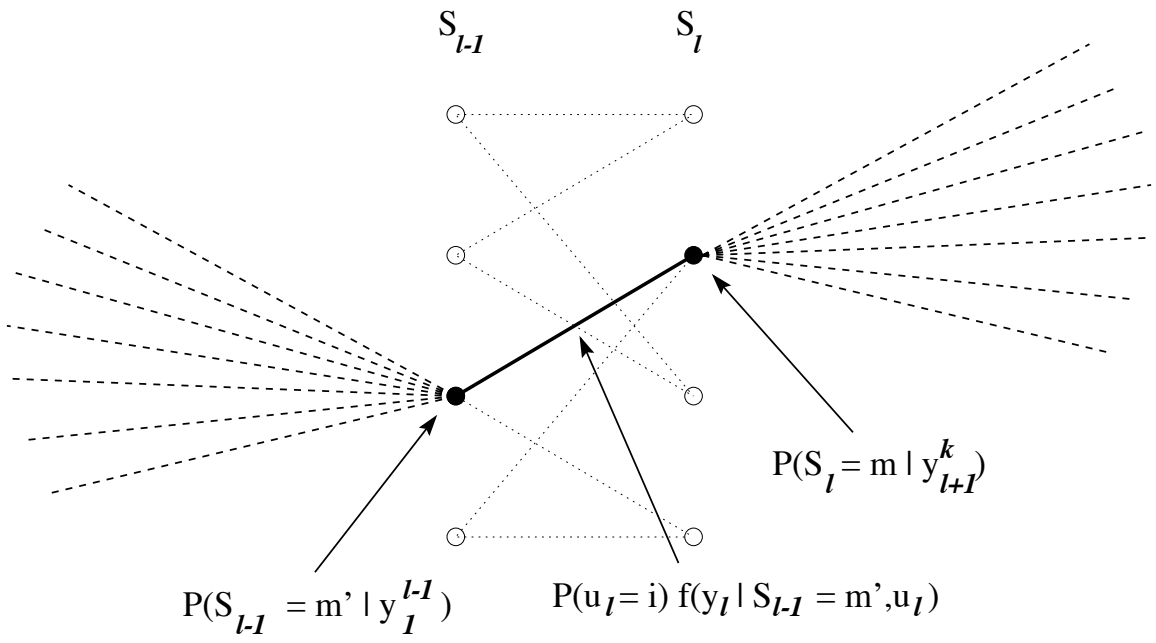


Figure 2.5: BCJR Algorithm.

2.5 Turbo Codes

Turbo codes were introduced by Berrou et al. [2]. The original design is depicted in Fig. 2.6. As is shown in the figure, there are multiple encoders in a turbo code. The encoders are most often chosen to be convolutional codes. There are multiple sequences of bits comprising the output of a turbo encoder.

When the turbo code is systematic, the first sequence in the output sequence is the data sequence itself. The data sequence is encoded by the first encoder and the encoded bits are added to the output sequence. Then the data sequence is interleaved, i.e., permuted, and encoded by the second encoder. The encoded bits from the second encoder is also added to the output sequence. More encoders and interleavers can be added. Various concatenation combinations can be employed. We will base our study on turbo codes with two component codes as this structure is the most studied in literature.

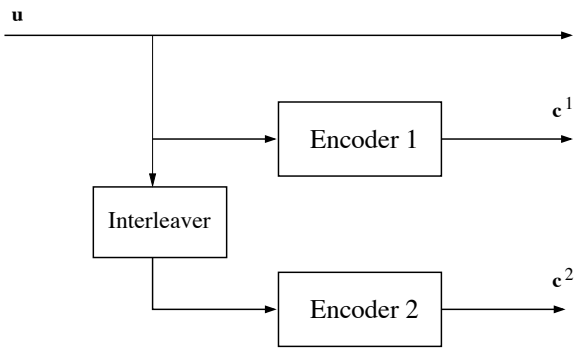


Figure 2.6: The original turbo code scheme.

2.6 Turbo Decoding

A turbo decoder consists of decoding modules corresponding to each component code. Decoding modules for each component convolutional code are explained in the previous section. This section will describe the information exchange between these soft-input soft-output (SISO) modules. This name is given bearing in mind that both the inputs and outputs are likelihoods for the decoding module. As seen in formulae in Section 2.4, the main function to obtain the *a posteriori* probability of a data symbol is $\gamma_l(m', m)$. This function needs the *a priori* information $p(u_l)$ and the observation from the channel so that the branch metric be calculated. These two data comprise the soft inputs and the soft output is the $P(u_l = i \mid y)$.

Berrou et al. introduced the idea of extrinsic information [2]. While the information is exchanged between the decoding modules, positive feedback due to information that is already present in the decoder should be avoided. Let's consider the case of a systematic turbo code and BPSK modulation. In this case the data will be transmitted over the channel along with the coded bits. Denote the observation of u as y^u and the observation of coded bits corresponding to the first encoder as y_1^c. In that

case,

$$P(u_l = i \mid y) = \sum_{(m',m):u_l=i} \alpha_{l-1}(m')\gamma_l(m',m)\beta_l(m) \quad (2.15)$$

$$= \sum_{(m',m):u_l=i} \alpha_{l-1}(m')p(u_l)f(y_l^u \mid m',m)f(y_l^{c_1} \mid m',m)\beta_l(m) \quad (2.16)$$

$$= p(u_l)f(y_l^u \mid m',m) \sum_{(m',m):u_l=i} \alpha_{l-1}(m')f(y_l^{c_1} \mid m',m)\beta_l(m) \quad (2.17)$$

The last equality suggests that the input is directly seen as a factor in the output. This causes positive feedback problems in the subsequent iterations. By omitting the information that is repeated in the output this problem is resolved. The summation term in (2.17) is called the extrinsic information for the l^{th} bit. Only the extrinsic information should be passed to other decoding modules and in reality natural logarithm of the extrinsic likelihood ratios ($\ln \frac{P(u_l=0)}{P(u_l=1)}$) is forwarded. Then, the extrinsic information corresponding to the l^{th} bit in the i^{th} iteration can be written as

$$e_l^i = \ln \frac{\sum_{(m',m):u_l=0} \alpha_{l-1}(m')f(y_l^{c_1} \mid m',m)\beta_l(m)}{\sum_{(m',m):u_l=1} \alpha_{l-1}(m')f(y_l^{c_1} \mid m',m)\beta_l(m)} \quad (2.18)$$

Fig. 2.7 provides an illustration of the overall turbo decoding scheme in detail. In the figure, π and π^{-1} are the interleaver and the deinterleaver. The observation of systematic part of the encoded bits is denoted by y^u, where y^{c_1} and y^{c_2} are the observation of the encoded bits for the first and second encoders respectively. In a turbo decoder the actual inputs to the decoders are likelihoods (or the ratios in the log domain) and observations are directly written only for illustrative purposes. Subtraction operation takes place since all the likelihoods are in the log-domain.

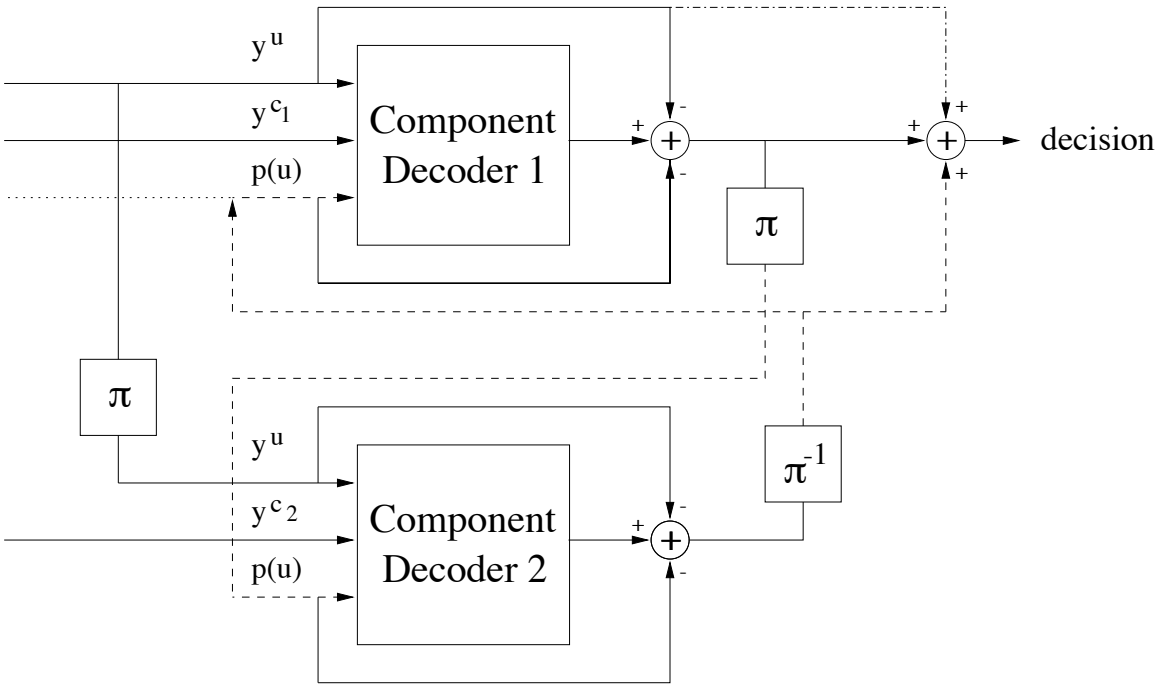


Figure 2.7: Turbo decoder structure.

CHAPTER 3

Convergence Characteristics and Classification

Iterative decoding algorithms attracted much attention after the introduction of turbo codes. Much research has been devoted to understanding the convergence of iterative decoding. In this chapter, we will provide a classification of the turbo decoding convergence characteristics based on simulation observations and the recent literature on the subject.

3.1 Introduction

When the turbo codes were first introduced in 1993, such good performance achieved by them was a mystery to the communications and coding community. Immediately following, much research have been directed toward the study of the turbo code structure, improving the performance with some design changes and applying the turbo decoding idea to different problems such as joint demodulation and decoding.

Recently a number of researchers showed interest in understanding the iterative decoding. Richardson posed the turbo decoding algorithm as a discrete dynamical system in [8]. He presented a geometric argument to show the existence of fixed points of the iterative decoder. Fixed points are points of operation at which the output (extrinsic information) from the turbo decoder is the same as the input. Therefore, when the turbo decoder hits such a point no further improvement of the extrinsic informa-

tion is possible. The existence of fixed points was also proven independently by Duan and Rimoldi [20]. Moreover, conditions for uniqueness and stability were given in [8]. Properties of fixed points were investigated through bifurcation theory by Agrawal and Vardy in [9]. However, none of these studies are completely conclusive. There are many questions regarding the uniqueness and conditions for convergence. This is the reason why simulation-based convergence characterizations are still beneficial.

Turbo decoding does not always terminate at a fixed point. Especially in the "waterfall" region, where rapid improvement of error performance is noted, oscillating error patterns are observed. The extrinsic information produced by the turbo decoder might oscillate seemingly randomly without convergence. Such points of operation will be referred to as oscillating points in this work.

Both fixed and oscillating points will be explained and a classification based on their properties will be provided in Section 3.2. Based on the classification, a stopping rule will be established and its performance will be evaluated in Section 3.4.

3.2 Classification of Convergence Characteristics

In this section, a classification of turbo decoding convergence characteristics will be presented. Extensive simulations were conducted to identify the various forms of convergence behavior with the goal that a categorization of these forms would provide a clearer understanding of the iterative decoding. In each simulation 50 iterations were performed and at least 50 packet errors were obtained. Simulation results are compared to other similar works [10, 11]. The results will also be explained with and compared to studies on fixed points [8, 9]. This classification will also mark the motivation for two different performance analysis tools which will be investigated in Chapters 4 and 6.

MAP decoders are capable of producing likelihoods for parity bits along with data bit likelihoods. This actually doesn't add substantial complexity since the parity bit likelihoods can be obtained using the following expression as in equation (2.14) of

Chapter 2:

$$P(c_l = i \mid y) = \sum_{(m',m):c_l=i} \alpha_{l-1}(m')\gamma_l(m',m)\beta_l(m). \tag{3.1}$$

The only change is the summation over the paths with the given parity bit rather than the given data bit. Combining the data and parity bit likelihoods, the iterative decoder's estimate of transmitted codeword can be obtained at any iteration by hard decision, i.e., making a decision for what the bit is, 0 or 1, based on its likelihood. We will relate the behavior of iterative decoder to the form this estimate takes.

We categorize the forms of turbo decoder's convergence behavior in Table 3.1. These categories will be explained in more detail in the next section. The categorization is based on convergence to a codeword in the code space and quantity of erroneous data bits. This categorization is a comprehensive summary of our extensive simulations and the classifications offered in [10, 11].

Table 3.1: Categories of Iterative Decoders' Behavior

1. The codeword estimate converges to a codeword with very few or no data bit errors.
2. The codeword estimate converges to a vector that is not in the code space of the turbo code with many data bit errors.
3. The codeword estimate oscillates without convergence with many data bit errors.
4. The codeword estimate oscillates without convergence with a few bit errors.
5. The codeword estimate converges to a vector which is not in the code space with a few errors.

Figure 3.1: A generic communications system.

To understand this classification better, let's consider the optimal decoder for the communication system given in Fig. 3.1. For equally likely bits, the decoder that is optimal on sequence detection is the maximum likelihood sequence detector (MLSD)

which can be written as

$$\hat{u} = \arg\max_{i} P(u = i \mid y), \tag{3.2}$$

where $\hat{u}$ is the MLSD's estimate of the transmitted data sequence. There is a one-to-one correspondence between the set of data sequences and the set of coded sequences for the encoders that are of interest in communication. The MLSD first chooses the most likely codeword (coded sequence) given the channel observation y. It then provides the corresponding data sequence as its estimate of the transmitted data sequence. However, a turbo decoder actually estimates individual bits/symbols rather than the whole data sequence. It can be shown that the optimal bit/symbol detector is identical to the optimal sequence detector at asymptotically high signal-to-noise ratio (SNR) values (see Section 4.3). This implies that if the turbo decoder approximates the optimal bit/symbol detector it also approximates the optimal sequence detector at high SNR values. Since it approximates the optimal sequence detector then the codeword estimate (meaning the data bits and parity bits combined) should, with high likelihood, be a vector in the code space of the codeword and thus a valid codeword. This fact has been used in Table 3.1 and the categories of the table will be investigated in the next section. The relation between a turbo decoder and MLSD will be explained in more detail in Chapter 4.

3.3 Categories

With the bit-optimal solution, referred to as the ML solution hereafter, there would, with high probability, be either a correct decoding or a decoding error with convergence to a codeword closest to the received vector. The received vector is nearby the transmitted codeword with high likelihood. Thus the codeword closest to the received vector should usually have small number of errors with respect to the transmitted codeword. If the turbo decoder actually approximated an optimal decoder, category 1 would always be the most likely category. The frequencies that the other categories are observed suggest that categories other than 1 are due to the suboptimality of the iterative decoder.

Categories 4 and 5 correspond to pathological cases which seldom occur. These categories will be explained briefly and then ignored in the remainder of the chapter.

3.3.1 Category 1

Category 1 is observed over all values of SNR and it is the desired operation of the turbo decoder. All the correct decoding cases are in this category. The incorrect decoding cases in this category, referred to as ML-decoding errors, are observed in the high SNR region. The ratio of ML decoding errors to all packet errors is shown in Table 3.2 for a rate-1/3 turbo code with two identical RSC component codes having the generator matrix

$$[1, \frac{1+D+D^3+D^4}{1+D+D^2+D^3+D^4}]. \tag{3.3}$$

The data sequences to the turbo code are of block length 2000 in this case. It can be concluded from the table that ML-decoding errors occur dominantly at high SNR. A turbo decoder converges to fixed points in this category. The fixed points observed with Category 1 are referred to as unequivocal fixed points in [9]. Unequivocal fixed points are points such that the bit/symbol likelihoods obtained from the decoder are extreme, e.g., very close to 0 or 1. This category is the combination of modes 1 and 2 in [11] and of types 1 and 2 in [10].

Table 3.2: Ratio of ML decoding errors to all packet errors

E_b/N_0(dB)	0.1	0.3	0.5	0.8	1.0
Ratio	5/50	21/50	49/50	50/50	154/154

We have observed for Category 1 that the codeword estimate is in the code space of the turbo code. This characteristic was noted in [9] as well. Since the optimal decoder would have the same property for the codeword estimate (with high probability), we will regard this category as providing the ML solution all the time. This claim can be tested by comparing the Euclidean distance between the received vector and the transmitted vector to the Euclidean distance between the received vector and the codeword estimate as in Section 4.3. In almost all cases, the distance between the

received vector and the codeword estimate is smaller which is an indication that the turbo decoder really provides the ML solution if the solution is in this category.

3.3.2 Category 2

Category 2 is dominant in the low SNR region. A solution in this category has error performance close to the uncoded performance of the channel. The number of bit errors are very high and the codeword estimate is not in the code space of the turbo code. Category 2 is associated with another type of fixed point called indecisive fixed point in [9]. The likelihoods produced by the turbo decoder is rather ambiguous, distributed around 0.5, for an indecisive fixed point. This category is considered as mode 1 in [11] and as type 4 in [10].

3.3.3 Category 3

As is well known, a turbo decoder doesn't always converge to a solution. In that case, a turbo decoder produces different solutions at every iteration. The solutions appear to be randomly changing. Then, it is said that the turbo decoder's solution corresponds to an oscillating point [9]. The oscillating point associated with this category seldom occur at low SNR [9]. This category also produces a lot of erroneous data estimates. Convergence to a codeword naturally doesn't occur. All the oscillating solutions are given in mode 3 in [11]. In [10] this category is called type 5.

3.3.4 Category 4

Category 4 is another case of an oscillating point for the turbo decoder. The number of errors are low similar to the number of errors in ML decoding errors. However, this category seldom occurs in practice and it won't be considered in the remainder of this report. This category corresponds to type 3 in [10].

3.3.5 Category 5

This category is one of the other pathological cases. The solution converges to a small number of bit errors. The codeword estimate is not in the code space of the turbo code. This category provides an unequivocal fixed point.

3.4 A Stopping Criterion

In this section, a new stopping criterion for turbo decoding and a corresponding retransmission scheme will be explained based on the classification offered in this chapter. In Section 3.2, it was pointed out that different types of decoding errors can be easily distinguished by checking the iterative decoder's estimate of the transmitted codeword. If the estimate is a codeword, then either correct decoding occurs or there is a ML-decoding error which has a small number of erroneous data bits. On the other hand, when the estimate is not a codeword the error occurs because the iterative decoder fails to provide the optimal solution. We will call this error type *decoding failure*. A decoding failure causes a large number of erroneous bits.

These observations naturally lead us to the idea that using this codeword criterion, i.e., whether the estimate is in the code space or not, the decoder can be stopped. In case the estimate is a codeword, ML solution is achieved and estimation of the data can be done by hard decision based on the data likelihoods.. In the other case, either convergence has not happened yet or the packet will never converge to the ML solution and ultimately cause a decoding failure. Specifying a maximum allowable number of iterations, the following rule is acquired:

- ***S**topping rule:* Whenever an estimate is a codeword, stop. Otherwise continue decoding as long as estimate is not a codeword up to a prespecified number of iterations.

This rule obviously allows ML decoding errors. ML decoding errors cause, with high probability, small number of erroneous data bits and correcting them might not be of interest in many settings although doing so is possible by using an outer code

[10,21]. However, this stopping rule sometimes halts the decoder before convergence. This might occur in cases where the iterative decoder's convergence slips into a slow progress after a quick improvement phase. There is an obvious trade-off between complexity (maximum number of iterations) and error performance in the light of this fact, however this issue won't be pursued in this study.

We will neglect categories 4 and 5, since they are less frequent than the other categories. We will mark the categories 2 and 3 as decoding failures. Decoding failures cause a large number of errors and asking for a retransmission for this case makes more sense. We propose the following automatic repeat request (ARQ) scheme: In case the decoder doesn't stop before a specified number of iterations, request a retransmission of the packet. This scheme almost insures that only ML decoding errors occur. This approach simply uses the intrinsic properties of iterative decoding and can be further refined using the idea of utilization of an outer code as in [10, 11, 21].

Two different codes have been used for the evaluation of this stopping rule and ARQ performances in this study. The codes are rate-1/3 turbo codes of data sequence length 1024 with two identical RSC component codes having the generator matrices

$$[1, \frac{1 + D + D^3 + D^4}{1 + D^3 + D^4}] \tag{3.4}$$

and

$$[1, \frac{1 + D^2}{1 + D + D^3}] \tag{3.5}$$

respectively. The interleaver used for both codes is an S-random interleaver with $S = 15$ (see Section 4.1 for the definition of S-random interleavers).

In Figures 3.2 and 3.4 bit error rates (BER) and packet error rates (PER) of the aforementioned codes are evaluated. The solid lines are the error performances of iterative decoding with 50 full iterations. A full iteration corresponds to one iteration for both component decoders of the turbo decoder. The dashed lines show the performance of the proposed stopping rule with at most 50 iterations. Based on these figures, it can be concluded that the stopping rule slightly degrades the error performance as expected. In both cases the degradation is within 0.1dB of the regular

scheme with fixed number of iterations.

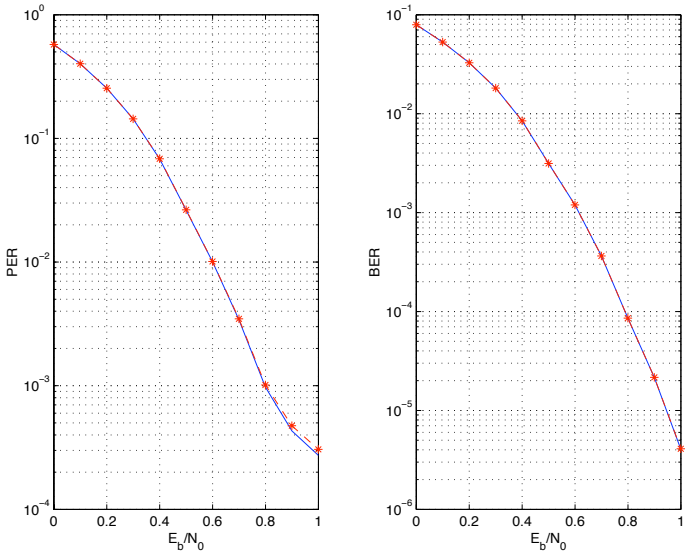


Figure 3.2: BER and PER comparisons of the stopping rule to the case of fixed number of iterations for a symmetric turbo code of the memory-4 component code with the generator matrix in (3.4).

Figures 3.3 and 3.5 show the average number of iterations with the stopping rule. The solid line is with the best possible stopping rule and the dashed line is with the proposed one. With the best possible stopping rule the decoder stops whenever what is transmitted is the same as what the estimate is and stops after 50 iterations even if errors still exist. This is unrealistic, of course, since the knowledge of what is transmitted is necessary. As seen in both figures, the average number of iterations is a monotically decreasing function of E_b/N_0. The proposed stopping rule is within 0.5 iterations.

The proposed ARQ algorithm performance is depicted in Figure 3.6. The solid lines are for the regular turbo coding scheme. The dashed lines represent the performance of the ARQ scheme proposed in this section. The throughput values are written next to the simulation points on the PER plot. Throughput is defined as the ratio of the number of transmitted packets over the number of total transmissions. As seen in the figure, the BER performance changes more significantly than the PER performance. This is due to the fact that decoding failures, which cause a large number of errors, are avoided by the ARQ scheme. Only packets with small

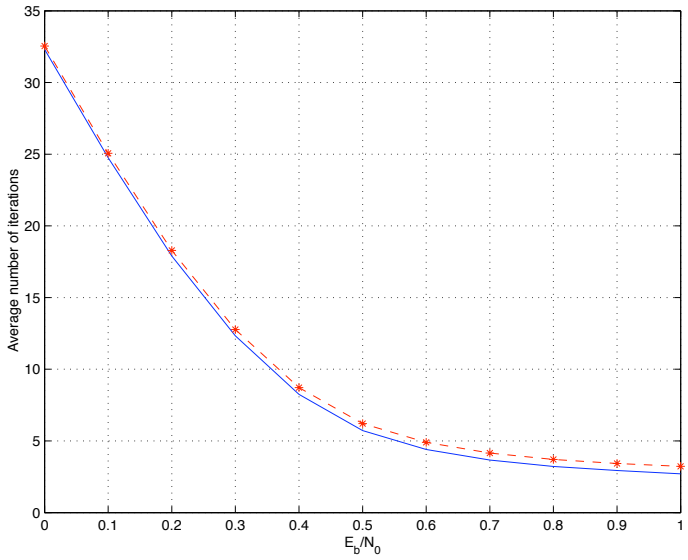


Figure 3.3: Average number of iterations compared to the best possible stopping rule for a symmetric turbo code of the memory-4 component code with the generator matrix in (3.4).

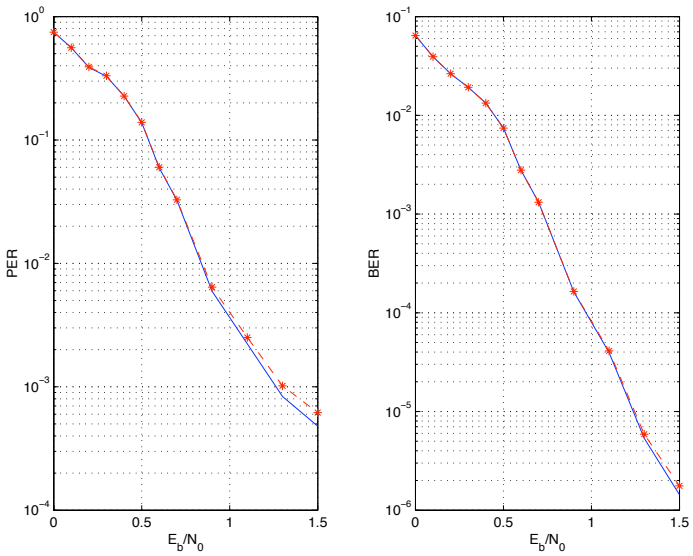


Figure 3.4: BER and PER comparisons of the stopping rule to the case of fixed number of iterations for a symmetric turbo code of the memory-3 component code with the generator matrix in (3.5).

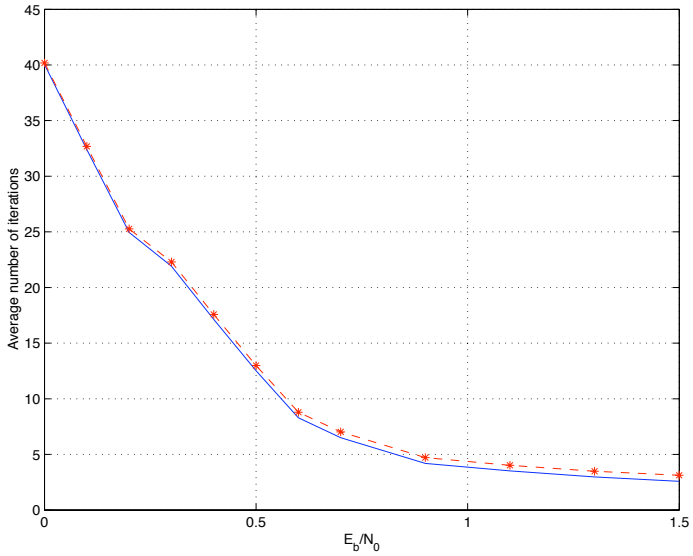


Figure 3.5: Average number of iterations compared to the best possible stopping rule for a symmetric turbo code of the memory-3 component code with the generator matrix in (3.5).

number of errors are allowed. Even if the same PER was to persist, the BER would be considerably smaller compared to the regular scheme up to the SNR value beyond which ML decoding errors dominate. In essence, the proposed ARQ scheme enables performance close to the performance of ML decoding at earlier SNR values.

This stopping rule is an alternative to the previously proposed rules such as sign-ratio-change and hard-decision-aided rules of [22] and average likelihood stopping rule of [11]. These different choices of rules should be compared in terms of computational complexity and memory requirements in a practical system.

3.5 Conclusions

Based on the classification presented in this chapter we can now differentiate between two modes of incorrect decoding: ML decoding errors (category 1) and decoding failures (the rest). The former mode draws on small number of errors whereas the latter causes a large number of errors. This statement is parallel to other researchers' observations that iterative decoders produce either no error or a large number of errors in the low/medium SNR regions. Moreover, ML decoding errors fail with a codeword

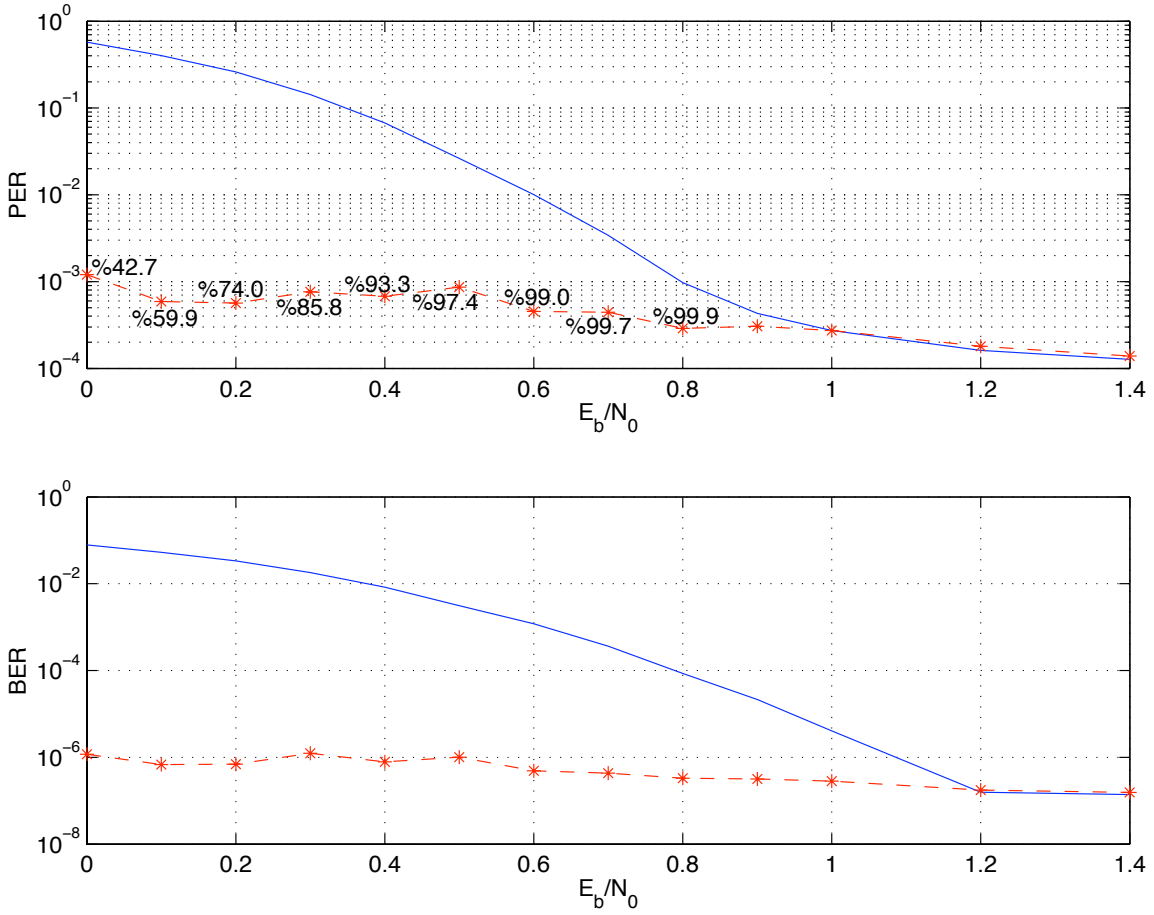


Figure 3.6: Performance of the ARQ scheme compared to the regular turbo coding scheme for a symmetric turbo code of the memory-4 component code with the generator matrix in (3.4) (Throughput values are not written after $E_b/N_0 = 0.8$dB since the throughput becomes very close to 1 afterwards).

estimate in the code space whereas decoding failure cases provide codeword estimates not in the code space. Moreover, these two different modes are prevalent in different SNR regions. ML decoding errors are dominant in the high SNR regime, whereas decoding failure events occur in the low/medium SNR regions. The distinction of these two different modes is the basis for two different performance analysis tools where each one is effective in different SNR regions. These tools will be explained in the next chapters.

CHAPTER 4

Error Floor Performance of Turbo Codes

In this chapter we will provide a method to find the dominant terms of a turbo code's distance spectrum for a given interleaver. This estimate of the dominant terms will enable us to obtain approximations to the performance of a given turbo code in the error floor region.

4.1 Introduction

Different modes of turbo decoding errors are distinguished in Chapter 3. When a turbo decoder operates at high SNR, very few bits of an incorrectly decoded packet are in error. In addition, the codeword estimate of the turbo decoder is in the code space of the turbo code. A maximum likelihood sequence detector (MLSD) finds the closes codeword to a received vector and then outputs the data corresponding to the codeword as its data estimate. So, MLSD always has a codeword estimate in the code space. This fact leads us to the belief that a turbo decoder acts like a MLSD at high SNR. Some results that imply the validity of this conjecture will be provided in Section 4.3. The component decoders of a turbo decoder produces bit likelihoods. By exchanging the likelihoods produced in each component decoder, a turbo decoder approximates the maximum likelihood decoder for each bit, rather than the sequence. Since MLSD and maximum likelihood bit decoder are equivalent at high SNR, the observation that a turbo decoder approximates MLSD at high SNR can be explained.

The related optimal decoders will be defined and their equivalence at sufficiently high SNR values will be proven in Section 4.2.

The error probability of MLSD can be bounded by using the well-established union-bound techniques [12]. The weight distribution of data sequences (input) and codewords (output) of the code are necessary to evaluate the union bound. Obtaining the input-output codeword weight distribution, referred to as distance spectrum, is not a simple task for the case of turbo codes because of the interleaver(s). Benedetto et al. provided a bound that averages the performance over all interleavers using the *uniform interleaving* assumption in [15]. The performance of a code is dominated by the minimum distance between all the codewords of a code at high SNR. Recursive codes have been the component code of choice in almost all turbo codes. The use of recursive codes help obtain approximations to the distance spectrum of a turbo code. By searching only for input sequences of weight 2, an upper bound to the minimum distance of a turbo code were obtained for particular interleavers in [23] and for all the possible interleavers in [24]. A method to find a lower bound to the minimum distance of a turbo code for a particular interleaver is introduced in [25]. As opposed to emphasis on minimum distance, we will focus on the dominant terms of the distance spectra of turbo codes. Daneshgaran et al. proposed a method to obtain the distance spectrum in [26]. The complexity of the method proposed in [26] is very high.

In this chapter, S-random interleavers are used to permute the data sequence. In the vast majority of literature, S-random interleavers are the choice of interleavers due to their good performance. An S-random interleaver permutes the data sequence in such a way that bits within a distance S are separated by more than S in the interleaved sequence. This can be expressed in the following by

$$|\pi(i) - \pi(i-j)| > S, \quad \forall 0 < j < S, \tag{4.1}$$

where $\pi(\cdot)$ is the permuting function.

We will propose a method to find an estimate of the distance spectrum of a turbo

code with a particular interleaver. The method will be justified by studying the properties of recursive encoders. An approximation based on their properties will be presented. The method proposed in this work is based on an approximation stated and explained in Section 4.5. We will also provide the limitations of the proposed algorithm in Section 4.7.

4.2 Optimal Decoding at High SNR

We will define two optimal decoders and establish their equivalence at high SNR. Consider the generic communication system in Fig. 3.1. Without loss of generality, we will take u to be a $K \times 1$ vector and c to be a $N \times 1$ vector. Let's also define c_u as the codeword corresponding to data u. Throughout this study the data sequence u is taken to consist of bits equally likely to be 0 or 1. Maximum likelihood *bit* decoding (MLBD) can be defined by the following rule:

$$MLBD : \hat{u}_{m,MLBD} = \arg\max_{i} P(u_m = i \mid y), \tag{4.2}$$

where u_m is the m^{th} bit in a sequence u, $\hat{u}_m$ is MLBD's estimate of u_m and i is in the alphabet of the code, e.g., in $\{0, 1\}$ if the code is binary.

MLSD can be defined with the same notation as follows:

$$MLSD : \hat{u}_{MLSD} = \arg\max_{v \epsilon D_u} P(u = v \mid y), \tag{4.3}$$

where v is a sequence from the set D_u of all data sequences. Define the Euclidean distance $||\cdot||$ between two $N \times 1$ vectors x and y by

$$||y - x||^2 = \sum_{l=1}^{N} |y_l - x_l|^2. \tag{4.4}$$

The Euclidean distance is used when the conditional probability density of an observation is evaluated both in MLBD and MLSD. We will prove that these two decoders behave identically at high enough SNR. A similar proof is given in [27].

Claim 1 *MLBD and MLSD provide the same solutions in the additive white Gaussian noise (AWGN) channel at asymptotically high SNR so that*

$$\lim_{SNR\to\infty} \hat{u}_{k,MLSD} = \hat{u}_{k,MLBD}. \tag{4.5}$$

Proof: In Section 2.2 the likelihood of receiving an observation y_l based on x_l is given as

$$f(y_l \mid c_l) = \frac{1}{\sqrt{\pi N_0}} \exp -\frac{||y_l - x_l||^2}{N_0}, \tag{4.6}$$

where x_l is in the set $\{-\sqrt{E_s}, +\sqrt{E_s}\}$ and y_l is corrupted by AWGN noise with one-sided power spectral density N_0. Equivalently, it can also be written as

$$f(y_l \mid c_l) = \frac{1}{\sqrt{\pi \frac{N_0}{E_s}}} \exp -\frac{||y_l - x_l||^2}{\frac{N_0}{E_s}}, \tag{4.7}$$

where x_l is in the set $\{-1, +1\}$ and y_l is corrupted by AWGN noise with one-sided power spectral density $\frac{N_0}{E_s}$. We will make use of this representation for the likelihood of receiving a sequence y for a transmitted sequence x:

$$f(y \mid c) = \frac{1}{(\pi \frac{N_0}{E_s})^{N/2}} \exp -\frac{||y - x||^2}{\frac{N_0}{E_s}}, \tag{4.8}$$

where N is the length of both sequences. MLSD chooses the codeword with maximum a posteriori likelihood. By (4.8) this also corresponds to choosing the closest codeword to the received sequence with respect to the Euclidean distance. Assume the codeword x_{u^1} is the closest one to y and x_{u^2}, is the second closest. Then,

$$||y - x_{u^1}|| = ||y - x_{u^2}|| - \epsilon, \epsilon > 0. \tag{4.9}$$

MLBD's rule can also be written in the following way:

$$\hat{u}_{m,MLBD} = \arg\max_i \sum_{v:v_m=i} P(u = v \mid y), \tag{4.10}$$

where v_m stands for the m^{th} bit in a sequence v. Let M denote the total number of sequences u. If there exists a sequence u^1 whose likelihood is more than the sum of likelihoods of all other sequences for small enough SNR ($\frac{E_s}{N_0}$), then

$$\begin{aligned}
P(u = u^1 \mid y) &> \sum_{j=2}^{M} P(u = u^j \mid y) \\
P(u = u^1 \mid y) + \sum_{j \neq 1, j: u_m^j = u_m^1} P(u = u^j \mid y) &> \sum_{j \neq 1, j: u_m^j \neq u_m^1} P(u = u^j \mid y) \\
\sum_{j: u_m^j = u_m^1} P(u = u^j \mid y) &> \sum_{j: u_m^j \neq u_m^1} P(u = u^j \mid y)
\end{aligned}$$

which implies that

$$\hat{u}_{m,MLBD} = u_m^1.$$

Let's now prove the existence of such a sequence at high enough SNR values when $\frac{E_s}{N_0} > \frac{\ln(M-1)}{\epsilon}$. For an equally likely source of data u

$$\begin{aligned}
\frac{N_0}{E_s} &< \frac{\epsilon}{\ln(M-1)} \\
\epsilon &> \frac{N_0}{E_s} \ln(M-1) \\
\|y - x_{u^1}\|^2 &< \|y - x_{u^1}\|^2 + \epsilon - \frac{N_0}{E_s} \ln(M-1) \\
\|y - x_{u^1}\|^2 &< \|y - x_{u^2}\|^2 - \frac{N_0}{E_s} \ln(M-1) \\
\frac{1}{(\pi \frac{N_0}{E_s})^{N/2}} \exp - \frac{\|y - x_{u^1}\|^2}{\frac{N_0}{E_s}} &> (M-1) \frac{1}{(\pi \frac{N_0}{E_s})^{N/2}} \exp - \frac{\|y - x_{u^2}\|^2}{\frac{N_0}{E_s}} \\
f(y \mid x = x_{u^1}) \frac{P(x = x_{u^1})}{f(y)} &> (M-1) f(y \mid x = x_{u^2}) \frac{P(x = x_{u^2})}{f(y)} \\
P(x = x_{u^1} \mid y) &> (M-1) P(x = x_{u^2} \mid y) \\
P(x = x_{u^1} \mid y) &> \sum_{j \neq 1} P(x = x_{u^j} \mid y).
\end{aligned}$$

Since $P(x = x_v \mid y) = P(u = v \mid y)$, the proof is complete. ■

In practice, the SNR does not have to be as small as predicted by the proof above. One reason is that not all of the rest of sequences will be at the second closest distance.

Another reason is that the distance difference ϵ is not likely to be very small. Hence, MLSD and MLBD behavior starts converging at reasonable SNR values.

4.3 Turbo Decoding at High SNR

As explained prior to this section in this paper, turbo decoding provides an approximation to MLBD with an efficient algorithm. However, there has not been any study to this day that actually proves this property of turbo codes. The iterative algorithm makes showing this analytically very hard. By the equivalence of MLBD and MLSD at high SNR proven in the previous section, this conjecture will be strengthened if turbo decoding approximates MLSD at high SNR. We will present some numerical results strongly suggesting that turbo decoding indeed approximates MLSD.

The experiment considered in this section is based on calculating distances between the observation and different codewords. Let us denote the data by $\tilde{u}$, the corresponding codeword by $c_{\tilde{u}}$ and its corresponding transmitted signal by $x_{\tilde{u}}$. Note that the set of codewords and the set of transmitted signals have a one-to-one correspondence by (2.1). Let x_u^1 be the closest signal to the received vector, that is

$$||y - x_{u^1}||^2 \leq ||y - x_v||^2, \tag{4.11}$$

for any sequence v. In particular,

$$||y - x_{u^1}||^2 \leq ||y - x_{\tilde{u}}||^2. \tag{4.12}$$

Two different codes have been used for the plots in this section. The codes are turbo codes of data sequence length 1024 with two identical RSC component codes having the generator matrices

$$[1, \frac{1 + D + D^3 + D^4}{1 + D + D^2 + D^3 + D^4}]. \tag{4.13}$$

and

$$[1, \frac{1+D+D^3+D^4}{1+D^3+D^4}]. \tag{4.14}$$

The interleaver used for both codes is an S-random interleaver with $S = 15$. We will denote the data estimate of the turbo decoder by $\hat{u}$ and its codeword as $c_{\hat{u}}$. Let ϵ be the difference between $||y - x_{\tilde{u}}||^2$ and $||y - x_{\hat{u}}||^2$. If $\epsilon > 0$ this indicates that $x_{\hat{u}}$ is closer to observation than $x_{\tilde{u}}$. We will show the difference ϵ for each packet by using bars in our figures. Not all of the error packets in each simulation is depicted in the figures due to the fact that there also exist decoding failure events in turbo decoding.

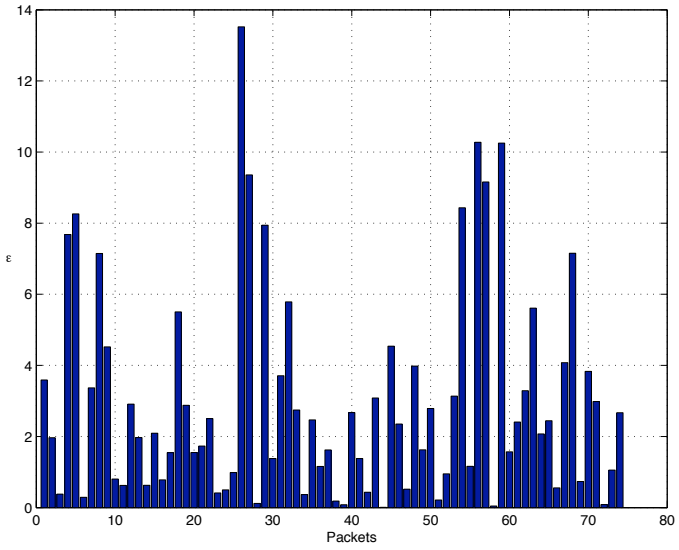


Figure 4.1: ϵ for 74 ML-decoding errors for the turbo code with component code generator matrix given in (4.13) at $\frac{E_b}{N_0} = 1.6$dB where overall PER equals $\frac{80}{84{,}960}$.

As seen in Figures 4.1-4.4, the decoded codeword is almost always closer to the observation than the transmitted codeword. The signal x_{u^1} always has this property. This suggests that the turbo decoder estimate $x_{\hat{u}}$ is likely to be the MLSD's estimate x_{u^1}. However, this approach does not rule out the possibility that some other codeword is even closer to the observation than the turbo decoder's estimate. Due to this shortcoming, the results presented in this section can only indicate strongly that the turbo decoder approximates MLSD at high SNR values because it approximates MLBD. This statement is the basis of utilizing the union bounding techniques for the error performance evaluation of turbo codes.

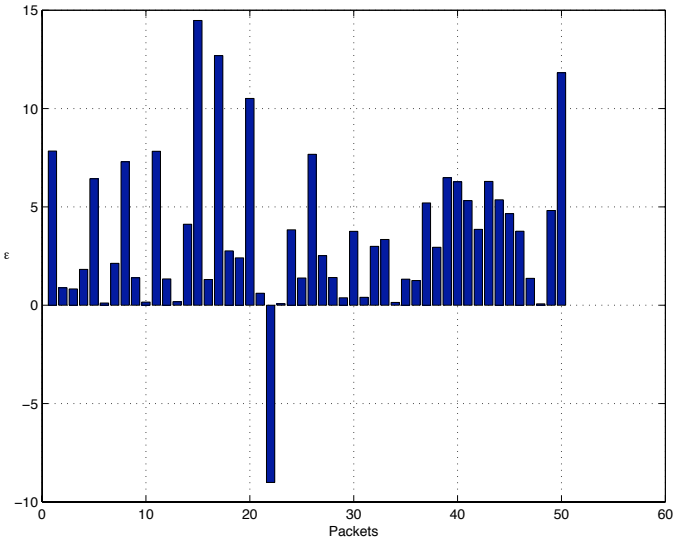


Figure 4.2: ϵ for 50 ML-decoding errors for the turbo code with component code generator matrix given in (4.14) at $\frac{E_b}{N_0} = 0.8$dB where overall PER equals $\frac{166}{147{,}385}$.

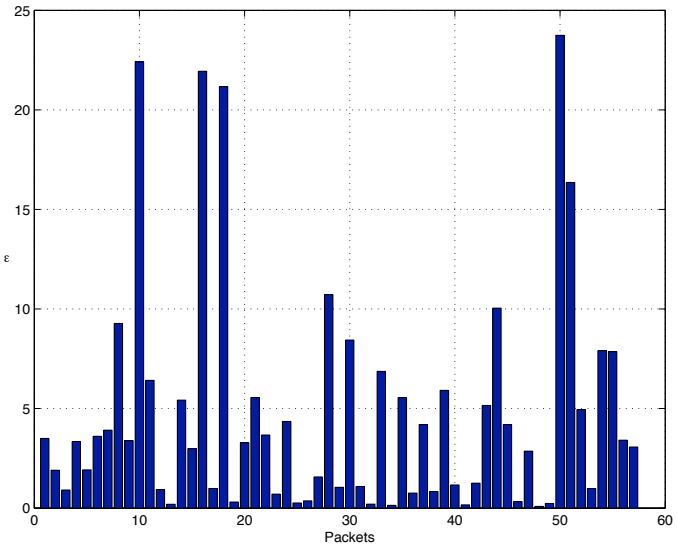


Figure 4.3: ϵ for 57 ML-decoding errors for the turbo code with component code generator matrix given in (4.14) at $\frac{E_b}{N_0} = 0.9$dB where overall PER equals $\frac{86}{187{,}305}$.

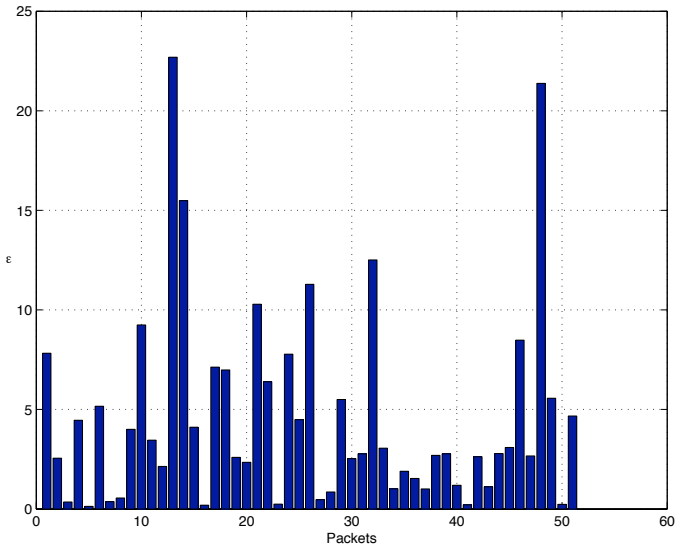


Figure 4.4: ϵ for 51 ML-decoding errors for the turbo code with component code generator matrix given in (4.14) at $\frac{E_b}{N_0} = 1.0$dB where overall PER equals $\frac{58}{176{,}842}$.

4.4 Error Rate Performance of Codes

For linear codes, the performance of a code can be analyzed by simply transmitting the all-zero codeword without loss of generality [28]. This fact suggest that the input-output codeword weight distribution, can be used to predict the error rate performance of a MLSD in AWGN channel [12]. Hamming weight of a binary sequence is defined as the number of nonzero elements in it. The term "weight" will refer to the Hamming weight hereafter. Consider that the function $A(D, P)$ represents the distance spectrum with data length K excluding the all-zero codeword. That is,

$$A(D, P) = \sum_{d>0} \sum_{p\geq 0} a_{d,p} D^d P^p, \tag{4.15}$$

where $a_{d,p}$ is the number of codewords with input Hamming weight d and output Hamming weight p. For the case of systematic codes, the packet error rate (PER) for MLSD can be upper bounded by using the union bound technique by [12]

$$PER \leq \sum_{d>0} \sum_{p\geq 0} a_{d,p} Q(\sqrt{2\frac{E_s}{N_0}(d+p)}), \tag{4.16}$$

where $\frac{E_s}{N_0}$ is the SNR per channel symbol (x_l in (2.1)) and $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp(-\frac{t^2}{2})\, dt$. Similarly, bit error rate for MLSD can be upper bounded as in

$$BER \leq \sum_{d>0} \sum_{p\geq 0} \frac{d}{K} a_{d,p} Q(\sqrt{2\frac{E_s}{N_0}(d+p)}). \tag{4.17}$$

Due to the fact that $Q(x)$ decreases quite sharply with increasing x, only the terms with small $d+p$ sum contribute to the error rates significantly. This lays the ground for focus on the dominant terms (low codeword weights) of $A(D, P)$.

4.5 Dominant Terms of Turbo Code Distance Spectrum

4.5.1 Error Events and Basic Input Fragments

As stated in 4.4, we can obtain an approximation to the error rate of a MLSD by simply considering the transmission of the all-zero codeword. We will refer to the path of the all-zero codeword in a given code's trellis (see Section 2.4) as the correct path. The correct path traverses the zero-state on the trellis all the time. The term $a_{d,p}$ in (4.15) is equivalently the number of incorrect paths corresponding to input sequences of Hamming weight d that have parity (the output other than the systematic part) Hamming weight p. An incorrect path is one that diverges from the correct path and remerges to it at some later time. Multiple divergence is possible. Each path corresponds to a different input sequence. Thus, we can equivalently investigate the codewords corresponding to incorrect paths.

Consider a fragment of an incorrect path which starts with a divergence from the zero-codeword path and ends after remerging to it for the first time. We will call such a fragment *basic input fragment* (BIF) and will refer to the case that a decoder favors this fragment instead of looping in the zero-state as an *error event*. An error event happens when the Euclidean distance between the received vector (observation) and the incorrect codeword (i.e. its corresponding transmitted signal) is smaller than

the Euclidean distance between the received vector and the all-zero codeword. As an example, consider the code in Fig. 2.3. One BIF of this code is 1 0 1. The parity output corresponding to this BIF is 1 1 1. If zeros are added to the front and/or back of a BIF the error event does not change. For instance, if 0 0 0 1 0 1 0 0 is an input to the encoder, the parity output is 0 0 0 1 1 1 0 0. The same error event is observed when zero-padded sequences of a BIF is fed into an encoder.

4.5.2 An Approximation for Low Weight Codewords

Consider the original turbo code structure in Fig. 2.6. The following arguments will also hold with slight changes for other settings. The weight of a turbo codeword $c_u = (u, c_u^1, c_u^2)$, where c_u^i corresponds to the parity (the output other than the systematic part) sequence from the i^{th} encoder, is defined as

$$w(c_u) = w(u) + w(c_u^1) + w(c_u^2). \tag{4.18}$$

The length of the data sequence u is K. There is currently no analytic way to obtain the distance spectrum of a turbo code due to the existence of the interleaver π before the second encoder. The only way to obtain the actual distance spectrum and extract its dominant terms is through exhaustive search over all 2^K turbo codewords, which is not feasible. However, we can avoid the exhaustive search and still obtain a good estimate of the dominant terms by an approximation.

In the case that $w(c_u)$ is small, both $w(u) + w(c_u^1)$ and $w(u) + w(c_u^2)$ should be small. This means both component codes should provide small codeword weights. Note that component codes of a turbo code are typically recursive codes. In most cases when a recursive encoder diverges from the all-zero state for a long time, the resulting output weight will be large[1] . A recursive encoder which doesn't loop in the

[1]This statement is true unless there exists a loop in the state diagram of the code with zero-input and zero-output. This sort of a code should be avoided in practice since it brings out a very large number of small weight codewords. Such a code can be easily detected by using the state transition matrix method described in Section 4.5.3. When raised to the L^{th} power, this matrix is the L-step transition matrix and its diagonal reveals the information whether such a loop exists or not. No such codes are used in this study and all the statements in this work should be interpreted accordingly.

zero-state continues to produce nonzero parity bits even though 0's are continuously fed into the encoder. Once the encoder leaves the zero-state, only specific sequences (BIFs) rather than consecutive 0's can force the encoder to go back to the zero state, unlike nonrecursive codes.

For a recursive code, a sequence u with small $w(u) + w(c_u^1)$ is most likely to be an input sequence that causes a single error event. Otherwise it will accumulate a large parity weight as discussed in the last paragraph. Each error event correspond to a BIF. A BIF is zero-padded in the front and/or back in order to obtain an input sequence of length K. Zero-padding a BIF does not change the error event corresponding to the BIF. Thus, any zero-padded sequence obtained for a BIF has the same component parity weight. All these input sequences contribute equally to the error rate of a code in (4.16) and (4.17). All of them should be considered when evaluating the error rates if their contribution is significant, i.e., the parity weight due to their BIF is not very large.

Since from Section 4.4 small weight terms are of interest, the number of basic input fragments to investigate can be limited. The longer a BIF is, the more likely its corresponding error event will cause a larger parity weight. Similarly, larger weight input sequences are more likely to cause larger parity weights. Therefore, we will only investigate BIFs up to a maximum length (T_L), a maximum input weight (T_D), and a maximum parity weight (T_P).

In the remainder of this subsection we will relate small weight codewords to the BIFs explained above. Define a set B_1 which contains all the codewords having 1^{st} through n^{th} smallest codeweights of the first encoder. The set B_1 can be written as

$$B_1 = \bigcup_{i=1}^{n} \arg_u \min_i \{w(u) + w(c_u^1)\}, \tag{4.19}$$

where

$$\arg_u \min_i \{f(x)\} \triangleq \{u \mid f(u) \text{ is the } i^{th} \text{ minimum of } f(x)\}, \tag{4.20}$$

and n is some threshold. The set B_1 consists of the input sequences which contain one BIF and have the $1^{st}, 2^{nd}, \ldots, n^{th}$ smallest codeweights. By symmetry, all the

arguments hold for the second encoder and a set B_2 can be defined as follows:

$$B_2 = \bigcup_{i=1}^{n} \arg_u \min_i \{w(u) + w(c_u^2)\}. \tag{4.21}$$

In the case that there is no termination, i.e., the encoder is not forced to a known state after all the data is encoded, another type of event arises which is likely to cause a small component parity weight. If all the 1's in u happen to be located very close to the end, the component codeword weight is upper bounded by $2(K - \mathcal{I}_u)$ where $\mathcal{I}_u$ is the index of the first 1 in u. In order to take this effect into consideration we will also define the following sets for the first and second codes respectively:

$$B_3 \triangleq \{u \mid w(u) < T_1 \text{ and } \mathcal{I}_u > K - T_2\}, \tag{4.22}$$

$$B_4 \triangleq \{u \mid w(u) < T_1 \text{ and } \mathcal{I}_{\pi(u)} > K - T_2\}, \tag{4.23}$$

with thresholds T_1 and T_2.

The approximation that is used to obtain the dominant terms of the distance spectrum is stated below.

Approximation: If $w(u) + w(c_u^1) + w(c_u^2)$ is small, then $u \in \bigcup_{i=1}^{4} B_i$ in case no termination scheme is used and $u \in \bigcup_{i=1}^{2} B_i$ otherwise.

Only single error events are considered with this approximation. The number of input sequences to be investigated for small turbo codeweights thus decreases substantially. Once B_i's, $i = 1, \ldots, 4$, are obtained, all the sequences in these sets can be turbo-encoded using a given interleaver. All the low weight codewords are very likely to be in this set of turbo-encoded codewords. The remaining problem is how to find the basic input fragments of a convolutional code. A method to realize this will be explained in the next subsection.

4.5.3 Obtaining Basic Input Fragments

Some properties of a convolutional code can be obtained using its state diagram. This can be done by representing the information in the state diagram in a format

that is analytically tractable. One such representation can be obtained by transferring the state diagram into a state transition matrix. The state diagram of Fig. 2.3 can be represented by the state transition matrix $\tilde{S}$

$$\tilde{S} = \begin{bmatrix} 1 & 0 & DP & 0 \\ DP & 0 & 1 & 0 \\ 0 & P & 0 & D \\ 0 & D & 0 & P \end{bmatrix}, \tag{4.24}$$

where the (i, j) entry in $\tilde{S}$ is denoted by $\tilde{S}_{i,j}$ and it represents the link from state i to state j (states are numbered by their corresponding decimal representations, e.g., $i = 2$ for the state $[1, 0]$.) By $\tilde{S}_{i,j} = a_{d,p} D^d P^p$, we mean there are $a_{d,p}$ different input sequences that start in state i and finish in state j with input weight d and parity weight p.

This matrix can be defined in many ways and some other information can also be put into its structure. Since our focus is only on the basic input fragments (they leave the zero state only once), we will use a modified version of this matrix. The zero-state will be split into two different states to account for the one and only divergence from the zero-state. We will denote the starting state by $[0, 0]_s$. All the links (inputs) leaving the starting state end up in another state. The ending zero-state will be denoted by $[0, 0]_e$. All the links from it loop back to itself while producing 0 as its parity. The state $[0, 0]_e$ will be added as a 5^{th} state to the modified state transition matrix S as follows

$$S = \begin{bmatrix} 0 & 0 & DP & 0 & 1 \\ 0 & 0 & 1 & 0 & DP \\ 0 & P & 0 & D & 0 \\ 0 & D & 0 & P & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.25}$$

The matrix S is the state transition matrix for an input length of 1. The L^{th} power of this matrix, S^L, is the transition matrix for the input length of L. Truncation, i.e.,

neglecting some of the terms in each polynomial $S_{i,j}$, can be employed due to the fact that only terms with small exponents (d, p) are of interest. In addition, truncation also helps avoid the computational burden due to the great diversity among the terms of the polynomial $S_{i,j}$ with increasing L. We will thus ignore all the terms with input weight higher than T_D and output weight higher than T_P. The basic input sequences with these thresholds can be found by examining the polynomial in $S^L_{1,5}$. We will use the algorithm below to find the basic input sequences.

1. Initialize L equal to 1 and construct S for the given component code.
2. Increment L by 1 and find S^L.
3. Obtain $\Delta = S^L - S^{L-1}$. If $\Delta_{1,5}$ has nonzero terms, then these terms are due to the basic input sequences with input length L. Record the nonzero terms along with the length L.
4. Continue with step 2 until $L = T_L$ for some threshold T_L.
5. For each recorded term, find the actual sequence by checking all the possible sequences that fit in the description. Record these basic input fragments.

With the steps $1 - 4$, the input and parity weights of basic input sequences of lengths up to T_L are obtained. Using these weights and the lengths,the last step finds each basic input fragment. For example, assume a term with $L = 10$, $d = 4$ and $p = 3$ is recorded in step 3. Each of the $\binom{L-2}{d-2} = \binom{8}{2}$ (since it is known that the first and the last bits should be 1) input sequences of weight 4 is checked to determine if it finishes at the zero-state with parity weight $p = 3$. The complexity of all these steps are low due to the fact that the dominant terms can be obtained by checking up to small values of T_L, T_D and T_P.

4.6 Numerical Results

In this section we will compare the error rates obtained by simulation to the error rate approximations explained in the previous sections. We have employed 3 different

codes for this study, each with different state complexities. All three codes are turbo codes with two identical RSC component codes having the generator matrices

$$[1, \frac{1+D^2}{1+D+D^2}], \tag{4.26}$$

$$[1, \frac{1+D^2}{1+D+D^3}], \tag{4.27}$$

and

$$[1, \frac{1+D+D^3+D^4}{1+D^3+D^4}]. \tag{4.28}$$

For each packet 50 full iterations are run. A full iteration refers to an iteration for each component decoders with two iterations in total. T_L threshold is set at 20 where $T_D = 5$ and $T_P = 8$.

In order to show the effectiveness of the proposed approximation with respect to block length K, the simulated error rates and the approximations are plotted in Figures 4.5 and 4.6 for various block lengths. Figure 4.5 depicts the packet error rates and Figure 4.6 shows the bit error rates. Four different block lengths (128, 256, 512, and 1024) are investigated where the interleaver parameters S are 7, 8, 10, and 15 respectively. The solid lines with the marker '+' show the simulated error rates whereas the approximations are plotted by the dashed lines. Error rate approximations are obtained by using equations (4.16) and (4.17). Naturally, only a partial sum is evaluated for the error rates. There are multiple dashed lines corresponding to different degrees of truncation, i.e. different number of terms in the error rate summations. In the lowest bound only the smallest distance with the smallest number of bit errors is depicted with its corresponding multiplicity. The higher approximations have the next smallest distances and next smallest number of bit errors. None of the approximations shown in this section has more than 10 terms in the summation.

We will provide the same comparisons for the turbo codes with the memory-2 and memory-4 component codes in Figures 4.7 and 4.8. As seen in all the figures, the prediction of the dominant terms by the stated approximation captures the error rate performance of turbo codes in the high SNR region for different block lengths

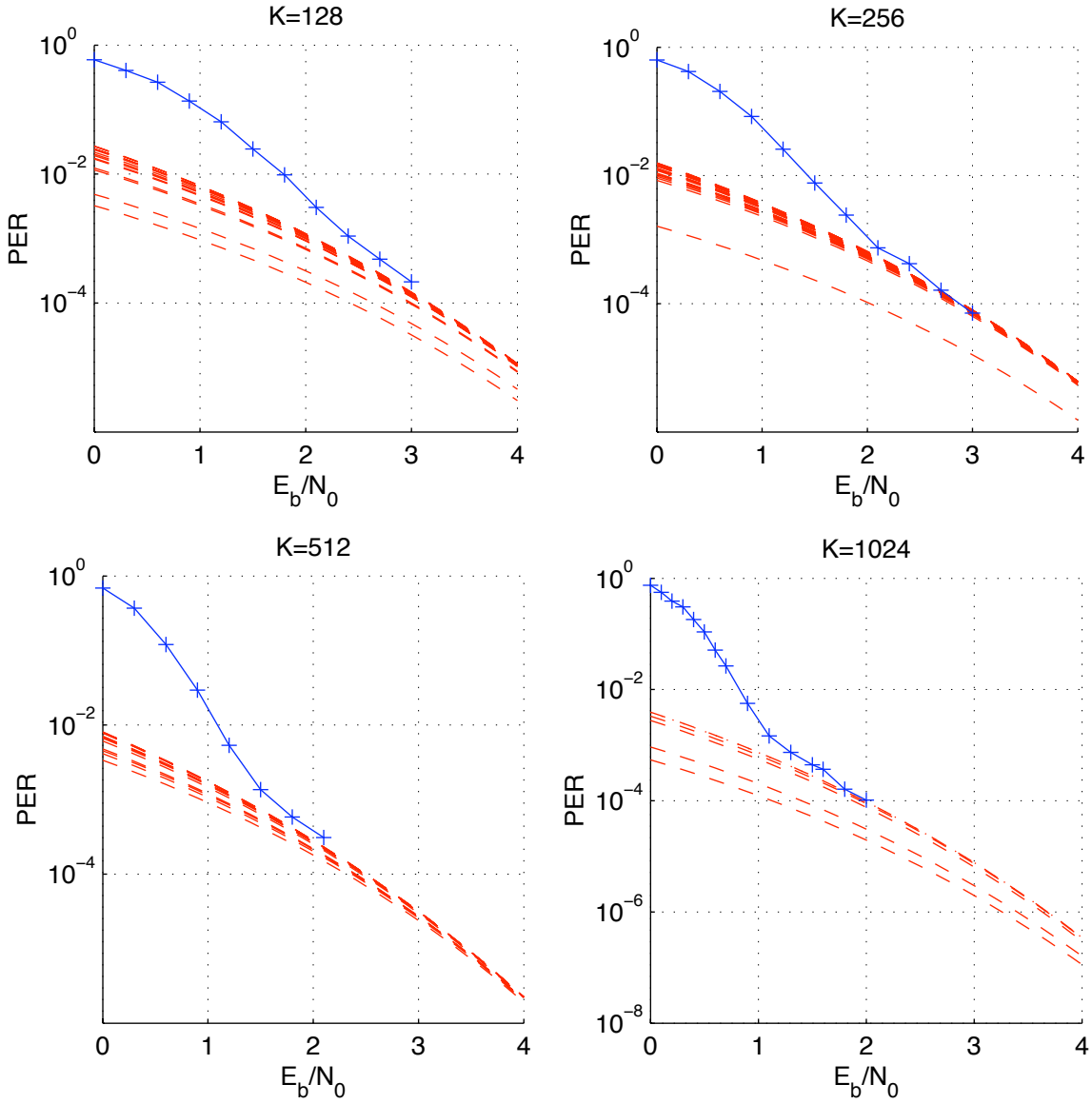


Figure 4.5: Comparison of the packet error probability approximation and simulated packet error rates with different block length turbo codes with a memory-3 component code.

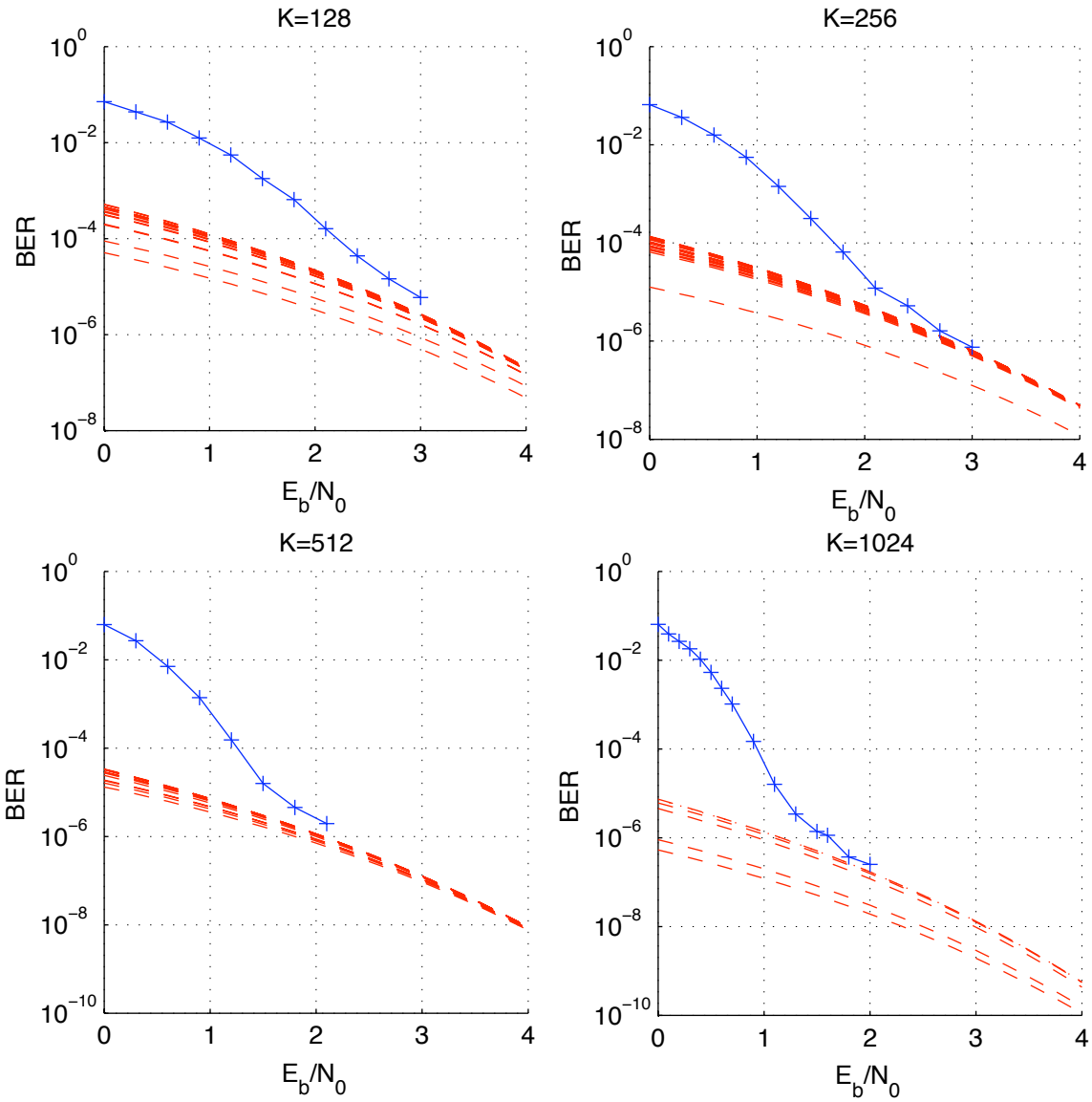


Figure 4.6: Comparison of the bit error probability approximation and simulated bit error rates with different block length turbo codes with a memory-3 component code.

and different component codes. The PER approximation is actually closer to the simulation results than the BER approximation. This difference can be explained based on the classification proposed in Chapter 3. When decoding failures occur a large number of bit errors are made by the decoder. Even if there is only one error event of that type, the bit error rate can be dominated by that event. Whereas, for the case of packet errors one decoding failure can not dominate the performance.

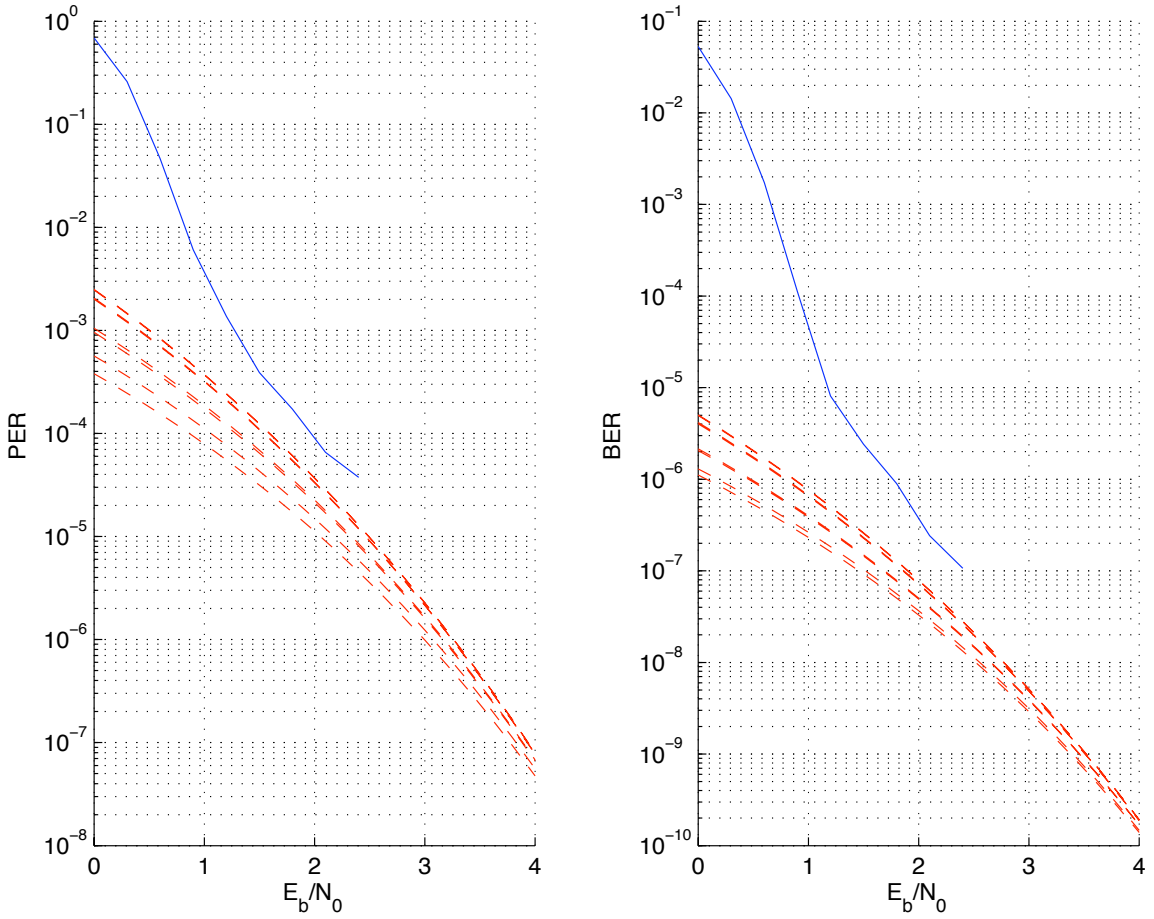


Figure 4.7: Comparison of the error rate approximation and simulated error rates of the turbo code with a memory-2 component code ($K = 1024$).

4.7 Limitations of the Proposed Algorithm

In this section, we will investigate the effect of multiple error events on the distance spectrum. Only those sequences which have not been considered by single error event

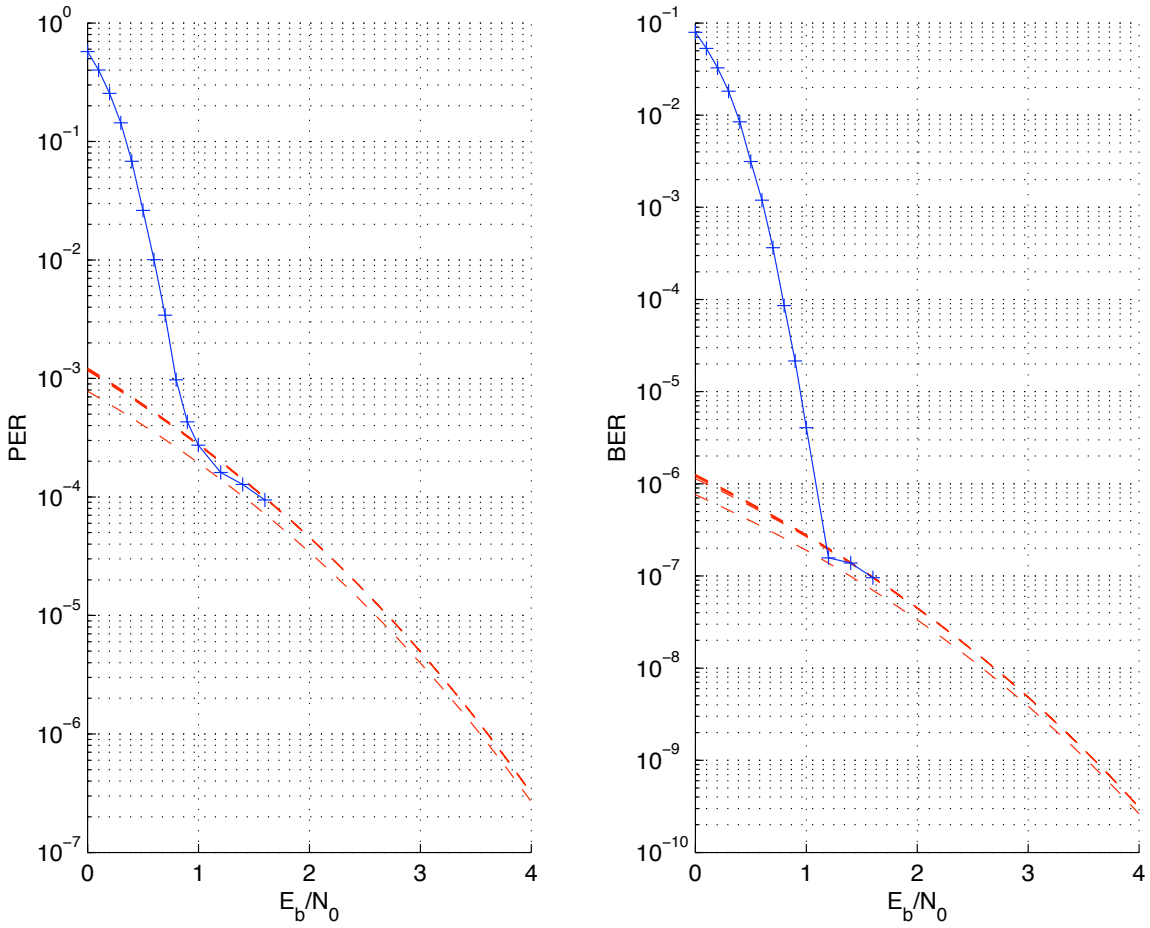


Figure 4.8: Comparison of the error rate approximation and simulated error rates of the turbo code with a memory-4 component code ($K = 1024$).

methodology will be explored. The average number of sequences causing multiple error events will be investigated. The basic idea is conveyed through presenting a few cases of sequences as examples. Then we will make use of a lemma by Perez et al. [4]. The average number of sequences will be derived for each case under the uniform interleaving assumption. The uniform interleaving argument states that all sequences are equally likely after interleaving. For example, if an input sequence of weight-W is interleaved when the sequence length is K, any of the $\binom{K}{W}$ output sequences are equally likely with likelihood $\frac{1}{\binom{K}{W}}$ after interleaving. The block length of an input sequence will be denoted by K. The thresholds T_L, T_D, T_P as defined in 4.5.2 determine the number of basic input fragments of interest for any given code. The number of basic input fragments of weight t for the i^{th} encoder will be denoted by $N_{i,t}$ and is a fixed number determined by the given thresholds that do not depend on K. Without loss of generality we will concentrate on the input sequence to the first interleaver.

We will define the following $\Theta(\cdot)$ notation for easier enumeration of the number of sequences.

Definition 1 *Let f and g be functions from the set of integers to the set of real numbers. $f(x)$ is said to be $\Theta(g(x))$ if and only if there exist c_1, c_2 and x_0 such that*

$$c_1 g(x) \leq f(x) \leq c_2 g(x),$$

for any $x \geq x_0$.

It is a simple exercise to show that if $a(x) = \Theta(f(x))$ and $b(x) = \Theta(g(x))$, then $a(x)b(x) = \Theta(f(x)g(x))$ and $\frac{a(x)}{b(x)} = \Theta(\frac{f(x)}{g(x)})$. We will use these properties in the following.

Each of the below cases have multiple error events. We will obtain an average number of input sequences in each case using the uniform interleaving assumption and the $\Theta(\cdot)$ notation defined above.

- Case 1

  $w(u) = 3$. In this case we have one error event caused by a BIF of weight-2

followed by a 1 near the end of the input sequence. This single 1 should be close the end of the sequence since it will cause the first encoder's parity sequence to have a large weight otherwise. After interleaving the same case will hold (or there will be an error event caused by a weight-3 BIF which is considered within the proposed algorithm thus we won't consider here). When the first encoder is considered, the number of such sequences is $N_{1,2}\Theta(K)$. This is due to the fact that the single 1 is in some area close to the end of the sequence and the other two 1's can make up any of the weight-2 basic input fragments anywhere in the sequence. When interleaved, the likelihood of the sequence having the same structure is $\frac{N_{2,2}\Theta(K)}{\binom{K}{3}}$. So the average number of sequences in Case 1 is

$$N_{1,2}\Theta(K)\frac{N_{2,2}\Theta(K)}{\binom{K}{3}} = \frac{\Theta(K^2)}{\Theta(K^3)} = \Theta(K^{-1}).$$

Therefore, in average we will have $\Theta(K^{-1})$ codewords.

- Case 2

  $w(u) = 4$. Two separate error events both with BIFs of weight-2 in both encoders. Using the same approach in Case 1 and noticing that these 2 error events can occur anywhere in the sequence, the average number of sequences in Case 2 is

$$N_{1,2}^2\Theta(\binom{K}{2})\frac{N_{2,2}^2\Theta(\binom{K}{2})}{\binom{K}{4}} = \frac{\Theta(K^4)}{\Theta(K^4)} = \Theta(1).$$

  In each turbo code there exist, on average, $\Theta(1)$ codewords.

- Case 3

  $w(u) = 4$. One error event by a weight-3 BIF and a single 1 in the end of the sequence in both encoders. The number of input sequences that are in this case is

$$N_{1,3}\Theta(K)\frac{N_{2,3}\Theta(K)}{\binom{K}{4}} = \frac{\Theta(K^2)}{\Theta(K^4)} = \Theta(K^{-2}).$$

- Case 4

  $w(u) = 4$. An input sequence in Case 2 and its interleaved version in Case 3

(or vice versa). The average number of such codewords is

$$N_{1,2}^2\Theta(\binom{K}{2})\frac{N_{2,3}^2\Theta(K)}{\binom{K}{4}} = \frac{\Theta(K^3)}{\Theta(K^4)} = \Theta(K^{-1}).$$

- Case 5

  $w(u) = 5$. 2 error events with BIFs with weights of 2 and 3 in both encoders. The average number of such codewords is

$$N_{1,2}N_{1,3}\Theta(\binom{K}{2})\frac{N_{2,2}N_{2,3}\Theta(\binom{K}{2})}{\binom{K}{5}} = \frac{\Theta(K^4)}{\Theta(K^5)} = \Theta(K^{-1}).$$

These examples help clarify the meaning of the following lemma from [4]. In the lemma S_1 and S_2 refer to the paths taken in the first and second encoder of a turbo code respectively.

Lemma 1 *Consider a turbo code based on two RSC encoders and a random interleaver of length K. Assume that the first encoder is forced back to the all-zero state. The contribution to the distance spectrum of the turbo code of two incorrect paths S_1 and S_2 with the same information sequence weight, averaged over all random interleavers of length K, converges to a nonzero constant as $K \to \infty$, if and only if*

1. *S_2 leaves the second encoder in the all-zero state;*
2. *S_1 and S_2 contain the same number of error events;*
3. *each error event in S_1 and S_2 is caused by a weight-2 BIF.*

In all other cases, the contribution goes to zero as $K \to \infty$.

What the lemma states is that only sequences with the same number of error events in both encoders caused by weight-2 basic input fragments will exist in almost every turbo code. The minimum turbo codeword weight in such a case has to be larger than or equal to

$$w_{min,s} = s(w_{min,2}^1 + w_{min,2}^2 + 2), \tag{4.29}$$

where $w_{min,s}$ denotes the minimum codeweight that can occur when S_1 and S_2 both have s error events and $w^i_{min,2}$ stands for the minimum parity weight produced by a BIF of weight-2 in the i^{th} encoder. The method proposed in this study cannot detect these sequences. The more error events occur, the larger the minimum codeweight is. So, only events with $s = 2$ is of interest. Therefore, one way to tackle this problem is checking the superpositions of any two weight-2 input sequences containing a BIF. This is an operation of complexity $\Theta(K^2)$ and might not be preferred to simulations in most settings. Another approach is to use better component codes with higher $w_{min,s}$ values. We will discuss the limitations of the proposed algorithm as it stands in the rest of this section.

In the case that weights of the dominant terms in the weight enumerator estimate from the proposed algorithm are below $w_{min,2}$, the error rate estimate is not affected much. Otherwise it is quite likely that the algorithm misses some of the dominant terms. This may affect only the multiplicities in some cases and both the multiplicities and the distance values in others. Therefore this is a limitation for the use of the proposed method. We will provide two examples both with comparably large block lengths ($K = 4096$). For smaller block lengths there is not much room for the interleavers to avoid small weight codewords stemming from single error events and thus the proposed method performs quite well.

The simulation results and error floor estimations for two different symmetric turbo codes with specific S-random interleavers are plotted in Figures 4.9 and 4.10. The memory-2 and memory-4 codes have the generator matrices $[1, \frac{1+D^2}{1+D+D^2}]$ and $[1, \frac{1+D+D^3+D^4}{1+D+D^2+D^3+D^4}]$ respectively. The turbo code minimum distance estimate is 20 for the memory-2 code case. This code has $w_{min,2} = 20$. As clearly seen in Fig. 4.9, the proposed method underestimates the dominant terms in the case of the 2-memory code. As observed in Fig. 4.10, the proposed method estimates the minimum distance for the turbo code as 16 where the memory-4 code has $w_{min,2} = 20$. The missed codewords do not affect the error floor estimate significantly as argued herein.

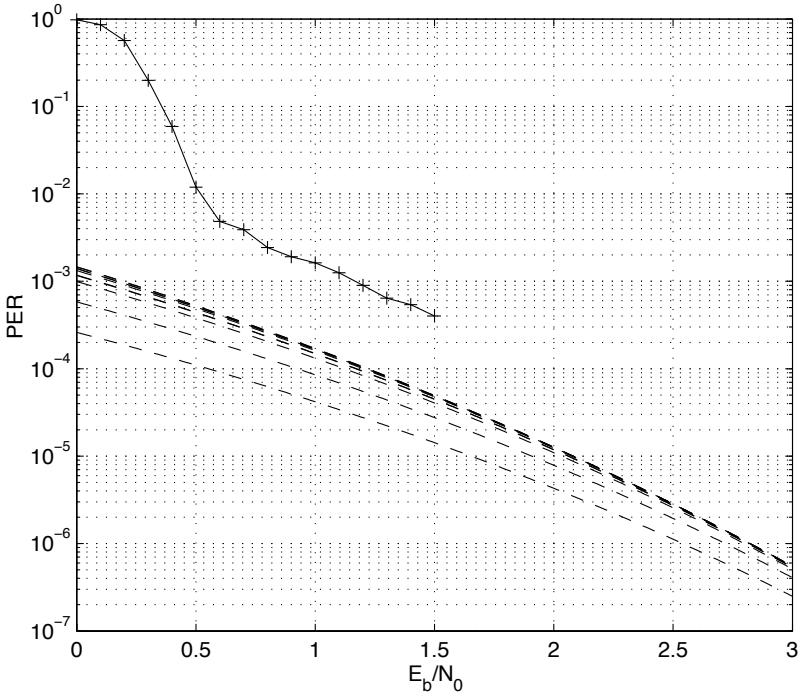


Figure 4.9: Comparison of the error rate approximation and simulated error rates of a symmetric turbo code of the memory-2 component code with generator matrix $[1, \frac{1+D^2}{1+D+D^2}]$ ($K = 4096$.)

4.8 Conclusions

Based on the convergence characteristics classification of turbo codes, we have provided an approximation to the bit error probability and packet error probability in the error floor region. The proposed method obtains an approximation to the dominant terms of the distance spectrum of a turbo code with a given interleaver. The error rate approximations are very close to the simulation results as presented in Section 4.6. We also presented some results showing the limitations of the proposed algorithm. Special attention should be paid especially when working on larger block lengths. The approach of this method can be applied to any generalized form of turbo codes.

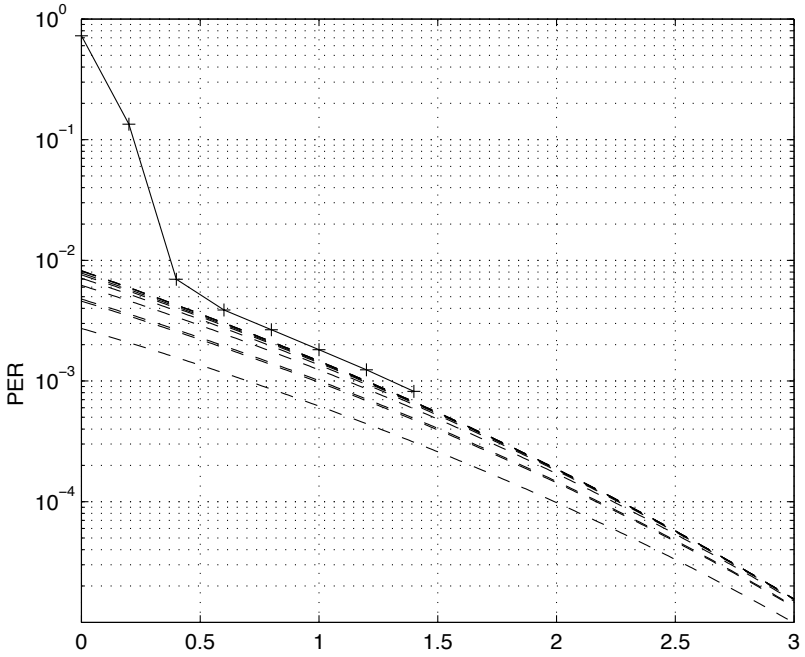


Figure 4.10: Comparison of the error rate approximation and simulated error rates of a symmetric turbo code of the memory-2 component code with generator matrix $[1, \frac{1+D+D^3+D^4}{1+D+D^2+D^3+D^4}]$ ($K = 4096$.)

CHAPTER 5

Modelling the Component Decoders

5.1 Introduction

Iterative decoding is a relatively easy to implement suboptimal solution to MAP decoding for turbo codes as explained in Chapter 2. Multiple component decoders are used inside an iterative decoder. In recent years, there has been considerable research concentrated on behavior of component decoders of a turbo decoder. When investigated independently from other component decoders, the isolated behavior of a component decoder help model the turbo decoder based on the component decoders [6,7]. In this chapter, we will investigate some properties of component decoders that allow us to understand how the component codes work together. Convolutional codes and MAP decoding will be considered for each component code. Extension to other decoders is possible.

The density evolution method was proposed in [5] to understand and predict the large block length behavior of iterative decoding of low-density parity-check (LDPC) codes. LDPC codes and iterative decoding techniques were first introduced by Gallager [1]. However, these codes were largely neglected until iterative techniques were popular. Density evolution provides a tool to find a SNR decoding threshold for an infinitely long LDPC code. For SNRs larger than the decoding threshold, arbitrarily small error probabilities are possible with the code when the block length becomes larger and larger. Density evolution is based on tracking the probability densities of

the soft information passed between modules of the iterative decoder. In [29], the method was greatly simplified using a Gaussian approximation to the soft information.

The Gaussian approximation to extrinsic information was employed to analyze turbo decoding for large block lengths in [7]. The existence of a convergence threshold for infinitely long block lengths was proven by regarding the component decoders as SNR transformers. The same approach was taken by ten Brink in [6] where he proposed a method to find convergence thresholds using a different measure to characterize the information generated by the decoders.

Current literature on analysis of iterative decoding concentrates on asymptotic behavior, i.e.,very large block lengths. However, a wide range of applications mandate the use of small to medium-size block lengths in practice. In this chapter, we will first describe methods to characterize the component decoders used in an iterative decoder. We will concentrate on the extrinsic information produced in a component decoder and investigate some of its properties. We will present a probabilistic model for the component decoders in the case of small to medium-size block lengths. Finite block lengths cause random variation in the properties of component decoders. The probabilistic model accurately characterizes this variation in component decoder behavior. The probabilistic model of component decoders developed in this chapter will later be used to examine the iterative decoding process in Chapter 6.

5.2 Decoding Thresholds at Infinite Block Lengths

5.2.1 Density Evolution Method

LDPC codes were introduced by Gallager in [1]. He states in [1] that the aim of employing LDPC codes is to provide an approach to simple decoding for long constraint codes. The definition below is almost the same as the original. Extensions of this definition (irregular LDPC) have been pursued in [13] and references therein.

Definition 2 *A regular (n, j, k) low-density parity-check code is a code of block length*

n with a sparse parity-check matrix, where each column contains a small fixed number j of 1*'s and each row contains a small fixed number k of* 1*'s.*

This definition suggests that LDPC codes are defined based on their parity-check matrices rather than generator matrices as is the case for many other classes of codes. This construction from the parity-check matrix actually results in a potentially more difficult encoding structure. We will not consider the encoding problem since it is not related to this study.

As there are very few 1's in the parity-check matrix of a LDPC code, parity checks can be performed with very low computational complexity. This fact enables the possibility of finding easy-to-implement optimal/suboptimal decoding strategies. Decoding algorithms for LDPC codes are easy to conceptualize with the graph representation of parity-check matrices. Let's consider the following parity-check matrix H.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{5.1}$$

Each row corresponds to a parity check equation and each parity check equation can be visualized as a node of a graph. The variables of the code will be matched with the parity check nodes based on H. Fig. 5.1 shows the graph representation for the H matrix in (5.1). The nodes on the left are called variable (v) nodes ($v_k, k = 1, \ldots, n$) and they represent the bits of the codeword. The nodes on the right represent the parity check equations and are called check (c) nodes.

Without loss of generality, we will assume each variable node is associated with a random variable $v_k, k = 1, \ldots, n$ that is either 0 or 1. We will often refer to a node when we are actually speaking of its corresponding random variable. A check node's random variable is 0 if its corresponding parity check equation is satisfied and 1 otherwise. A channel signal $x_k = \sqrt{E_s}(-1)^{v_k}$ is transmitted for each variable node v_k as in our general setting. The received signal corresponding to x_k is

$$y_k = x_k + n_k, k = 1, \ldots, n \tag{5.2}$$

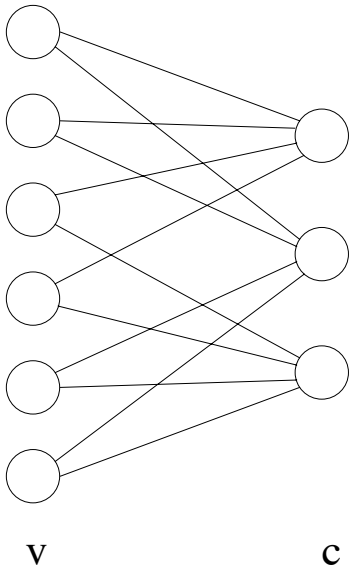


Figure 5.1: Graph representation of a $(6, 2, 4)$ LDPC code.

where n_k is the AWGN noise term (see Section 2.2). The decoding problem is estimating what the transmitted $v_k, k = 1, \ldots, n$ values are based on all the observation $(y_k, k = 1, \ldots, n.)$

We will now provide a decoding algorithm, called belief propagation, which exchanges soft information between the variable and check nodes. There is a good discussion of this algorithm's assumptions and general setting and other decoding algorithms in [5]. In belief propagation, a continuous message alphabet is used for the soft information that is passed between the nodes. This information is roughly the *a posteriori* probability for the node's random variable from which it is originating. The message Λ_a^b passed from node a to node b is the natural logarithm of $\frac{P(a=0)}{P(a=1)}$, where $P(a = 0)$ has been used liberally to denote the likelihood of a at a given iteration. The new likelihood values are obtained based on the observation and the information produced in the previous iteration of the decoding algorithm. The message passed from a variable node v_k to a check node c_l, to which v_k is connected, is obtained by

$$\Lambda_{v_k}^{c_l} = \Lambda_{y_k} + \sum_{c \epsilon S_c(v_k), c \neq c_l} \Lambda_c^{v_k}, \tag{5.3}$$

where $S_c(v_k)$ is the set of check nodes to which v_k is connected and Λ_{y_k} is $\log \frac{P(y_k|v_k=0)}{P(y_k|v_k=1)}$. In the very first update Λ_c^v has the initial value 0 for every c and v.

The message from a check node c_k to a linked variable node v_l is later updated by

$$\tanh \frac{\Lambda_{c_k}^{v_l}}{2} = \prod_{v \epsilon S_v(c_k), v \neq v_l} \tanh \frac{\Lambda_v^{c_k}}{2}, \tag{5.4}$$

where $S_v(c_k)$ is the set of variables nodes to which c_k is connected. These updates are performed repetitively until some criterion is met.

Richardson and Urbanke presented a general method called "density evolution" to determine the decoding threshold of a LDPC code in [5]. Density evolution method regards each message between the nodes as a random variable. It keeps track of the probability densities of these messages. From (5.3) and (5.4), the probability densities of the random variables can be kept track of analytically. The only input to the message passing algorithm is Λ_{y_k} whose statistics change with SNR. The minimum SNR value that enables decoding without any error is declared as the decoding threshold.

5.2.2 A Gaussian Approximation for Belief Propagation

As explained in the last subsection, the density evolution method provides a decoding threshold above which reliable communication is possible. However, the method of finding the threshold is not very practical since the probability densities of the information sent between the nodes should be recorded all the time. A Gaussian approximation offers a simple solution to representing the probability density since a Gaussian random variable can be characterized completely by its mean and variance.

It is empirically found that message densities resemble Gaussian densities [29,30]. It is further assumed in [29] that the ratio $\frac{\text{var}(\Lambda_{y_k})}{\text{mean}(\Lambda_{y_k})} = 2$ is preserved for all the exchanged messages. This property will be explained in Section 5.2.4. This effectively converts an infinite dimensional problem (keeping track of the probability densities) to a one-dimensional problem (keeping track of only the mean). Decoding thresholds obtained with this approximation are reportedly very close to the exact thresholds obtained by the density evolution method.

5.2.3 A Gaussian Approximation for Turbo Decoding

The density evolution methodology can also be applied to turbo decoding since a turbo decoder is also an iterative decoder. In most LDPC algorithms there are simple analytical expressions, such as (5.3) and (5.4), through which the probability densities of the messages between nodes can be obtained. However, the MAP decoder for convolutional codes (as described in Chapter 2) does not have such simple analytical expressions to represent the extrinsic information it produces. This mandates a focus on individual MAP decoders to analyze the iterative decoding.

The density evolution method was applied to small memory convolutional codes [16]. In general, direct application of density evolution without the Gaussian approximation is difficult. The Gaussian approximation to the extrinsic information has been widely applied [6,7,16]. It has been observed early that extrinsic information in the log-domain has a Gaussian-like distribution [2]. The validity of Gaussian approximation was demonstrated empirically in [6,7]. We will simply use this approximation and related assumptions in the literature without much elaboration.

From this point on we will denote the sequence of observations by $\{y_i, i = 1, \ldots, N\}$ and the *a priori* (or extrinsic, depending on whether the input or output of a decoder is considered) information for any iteration i by $\{e_k^i, k = 1, \ldots, K\}$, where N is the number of coded bits and K is the number of data bits. From (2.18) the extrinsic information at the i^{th} iteration in the log-domain is

$$e_k^i = \ln \frac{P(u_k = 0 \mid y_1, y_2, \ldots, y_N, e_1^{i-1}, e_2^{i-1}, \ldots, e_K^{i-1})}{P(u_k = 1 \mid y_1, y_2, \ldots, y_N, e_1^{i-1}, e_2^{i-1}, \ldots, e_K^{i-1})} - \ln \frac{f(y^{u_k} \mid u_k = 0)}{f(y^{u_k} \mid u_k = 1)} - e_k^{i-1}, \quad (5.5)$$

where u_k is the k^{th} data bit and y^{u_k} is the observation corresponding to u_k. In fact, it is of interest to represent how well the extrinsic information predicts the transmitted data. Define a new sequence related to e_k^i by

$$z_k^i = (-1)^{u_k} e_k^i. \quad (5.6)$$

We say u_k is correctly estimated if u_k is 0 and e_k^i is positive or u_k is 1 and e_k^i is

negative. The new variable z_k^i is defined in order to account for the likelihood of a correct decoding of u_k. We will refer to z_k^i's as extrinsic information from this point on.

The Gaussian approximation for the case of turbo codes can be summarized as follows.

1. An extrinsic information sequence $\{z_k^i, k = 1, \ldots, K\}$ for any iteration i is Gaussian, wide-sense stationary (WSS) and ergodic.

2. $\{y_i, i = 1, \ldots, N\}, \{z_k^1, k = 1, \ldots, K\}, \{z_k^2, k = 1, \ldots, K\}, \ldots$ are jointly Gaussian and uncorrelated in the sense that any finite collection of y_i and z_k^j are jointly Gaussian and uncorrelated.

The first assumption can be justified by noting that the extrinsic information is the sum of many terms as seen in (2.14) and by the central limit theorem its distribution should approach that of a Gaussian random variable. If the bits close to the beginning and the end of a block are disregarded, it can be said that the decoder neither favors nor disfavors any of the data bits. All the observations and *a priori* information around any bit have the same statistics. We provide Fig. 5.2 as an evidence for the wide-sense stationarity of extrinsic information sequences. Seen in the figure are estimates of mean and covariance statistics over 5000 extrinsic information sequences. Block length of each sequence was chosen to be 600. The code is the memory-2 RSC code with the generator matrix $[1, \frac{1+D^2}{1+D+D^2}]$. The SNR is $E_s/N_0 = -4.27$dB and the extrinsic info sequences are all zeros which correspond to the first iteration of a turbo decoder. In Fig. 5.2a the sample mean $\bar{Z}_k$ (corresponding to the expected value of Z_k which is the random variable for realizations of z_k) over 5000 sequences is shown for each k. The 95% confidence intervals are obtained by assuming the samples are normally distributed with unknown variance [31]. The mean over all the 600×5000 samples are also drawn with a solid line and the confidence intervals are plotted around it. Excluding around 10 bits both from the beginning and end of a block, we almost have the same mean within the confidence intervals for any k.

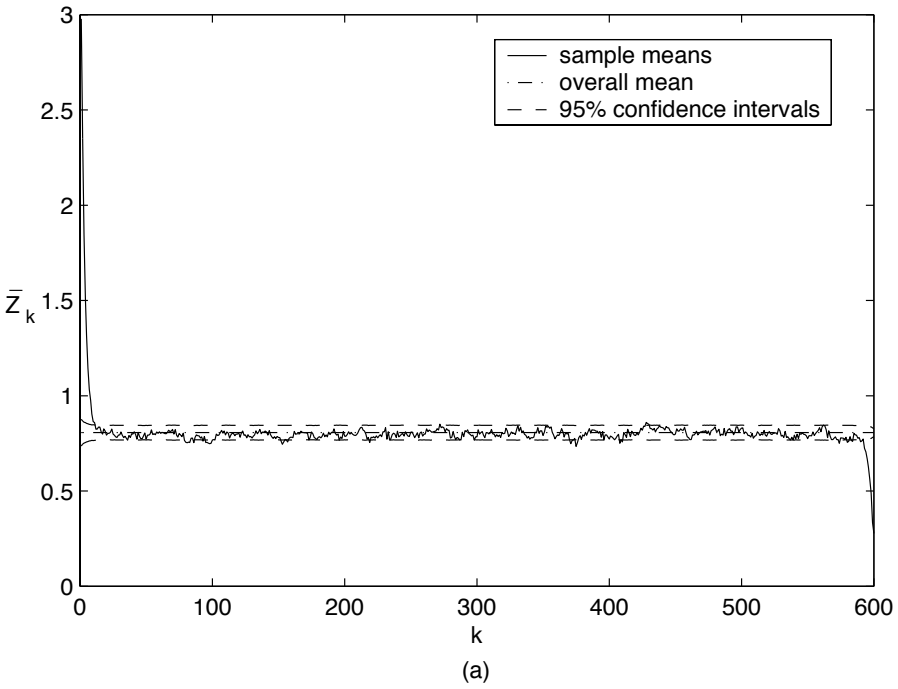
To estimate the covariance, an unbiased estimate of the covariance is obtained by

$$\overline{\mathrm{Cov}(Z_k, Z_m)} = \frac{1}{4999}\sum_{i=1}^{5000}(z_{i,k}z_{i,m} - \bar{Z}_k\bar{Z}_m), \tag{5.7}$$

where $z_{i,j}$ is the extrinsic information corresponding to the j^{th} bit in the i^{th} sequence. The contour map of the covariance estimate is plotted in Fig. 5.2b. The covariance estimates are nonzero when $|k-m|$ is smaller than around 20 and essentially zero otherwise. As seen in the covariance plot, the covariance between Z_k and Z_{k+l} is nearly independent of k. Hence, the extrinsic information sequence can be closely approximated by a WSS random process. Although the plot is for the extrinsic information corresponding to the first iteration of a turbo decoder, similar results hold for *a priori* extrinsic information sequences generated at later iterations.

The second assumption can partly be derived from the first one since extrinsic information of one decoder will be fed as the *a priori* information of the other after interleaving or deinterleaving. The extrinsic information is obtained by omitting the channel observation and *a priori* information in (5.5). The uncorrelatedness in the second assumption follows from this property. The use of almost any randomly generated interleaver ensures that the uncorrelatedness holds. This is the prime reason why any realization of a random interleaver exhibits essentially the same performance as any other realization in the waterfall region [32]. Thus, all the results obtained in this chapter are independent of the specific interleaver used in the turbo code construction. Again the ratio of the variance and the mean of the extrinsic information (in log domain) will be assumed to be 2 since the distribution of a log-likelihood based on Gaussian distributions should have this property [6].

The MAP decoder in the turbo decoding algorithm can be regarded as a device which produces extrinsic information from the given observation and *a priori* information. It transforms the statistics of the inputs to that of the output. Various parameters can be used to keep track of the iterative process [6,7,16]. In [6] the mutual information between the extrinsic information and transmitted data was shown to provide better results with respect to obtaining decoding thresholds compared to



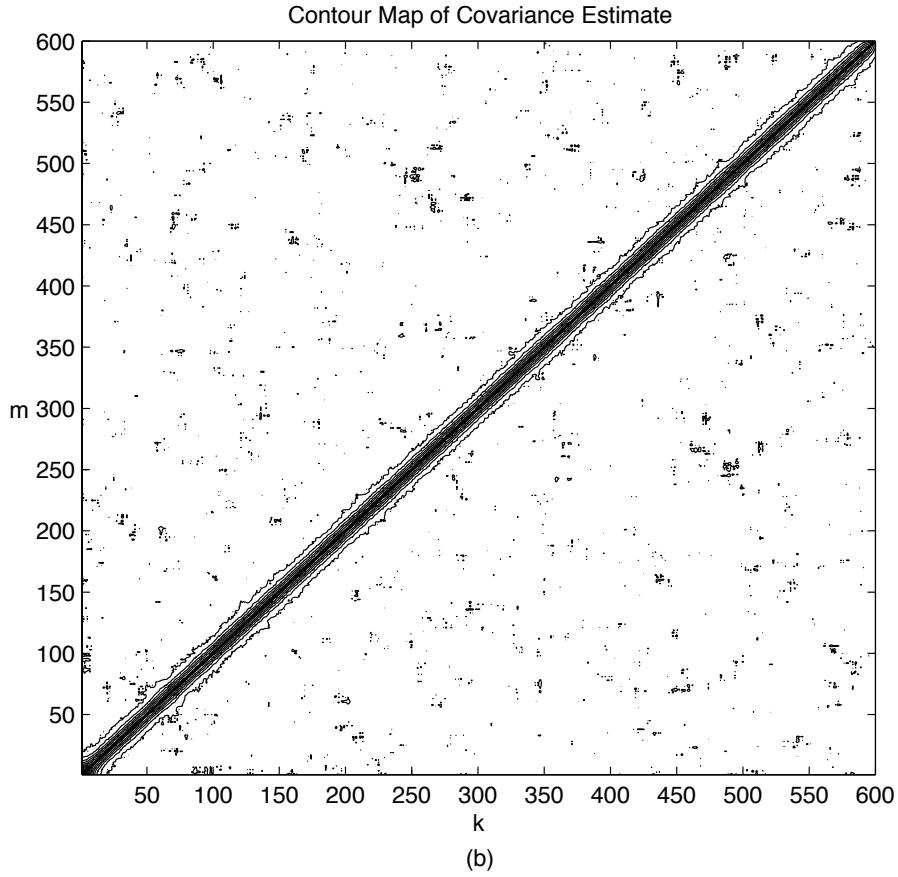


Figure 5.2: Mean and covariance estimates for extrinsic information.

other considered parameters. We will also base our study on mutual information.

5.2.4 Characterization of Component Decoders

Based on the discrete-time signal model explained in Section 2.2, the log-likelihood ratio for any observation is a random variable as defined in the following sense:

$$L_y = \ln \frac{f(y|c=0)}{f(y|c=1)}. \tag{5.8}$$

L_y can be simplified to

$$L_y = \frac{4\sqrt{E_s}}{N_0} y = \frac{4\sqrt{E_s}}{N_0}(\sqrt{E_s}(-1)^c + n). \tag{5.9}$$

In that case,

$$E[L_y|c] = \frac{4E_s}{N_0}(-1)^c, \tag{5.10}$$

$$\mathrm{Var}(L_y|c) = \frac{8E_s}{N_0}. \tag{5.11}$$

Then, L_y has the following property:

$$\frac{\mathrm{Var}(L_y|c)}{E[L_y|c](-1)^c} = \frac{\mathrm{Var}(L_y|c)}{E[(-1)^c L_y|c]} = 2. \tag{5.12}$$

It is assumed that extrinsic information also has this property and it is preserved throughout the iterative process.

We regard an extrinsic information sequence $\{e_k^i, k = 1, \ldots, K\}$ as a realization of a random sequence $\{E_k^i, k = 1, \ldots, K\}$. We will define the mutual information between the extrinsic information E_k^i and the transmitted bit U_k as in [6]:

$$I(E_k^i, U_k) = \sum_{u \in \{0,1\}} \int_{-\infty}^{\infty} f_{E_k^i, U_k}(u, e) \log_2 \frac{f_{E_k^i|U_k}(e|u)}{\sum_{u' \in \{0,1\}} f_{E_k^i, U_k}(u', e)} de, \tag{5.13}$$

for any bit location k. The data U_k is equally likely to be 0 or 1. The correspond-

ing extrinsic information E_k^i is Gaussian when conditioned on U_k by the Gaussian approximation explained in Section 5.2.3. In that case, $I(E_k^i, U_k)$ can be written in the following way by using the definition $\{Z_k^i = (-1)^{u_k} E_k^i\}$ and the variance-to-mean ratio assumption explained in the previous paragraph:

$$I(E_k^i, U_k) = 1 - \int_{-\infty}^{\infty} f_{Z_k^i}(z) \log_2(1 + e^{-z}) dz. \tag{5.14}$$

This result suggests that the random variable Z_k^i can be used instead of E_k^i to obtain the mutual information between the produced extrinsic information and the transmitted data. The same result holds for *a priori* information. In the remainder of the paper we will refer to the sequence $\{Z_k^i, k = 1, \ldots, K\}$ and its realizations as the extrinsic information with an abuse of notation.

There is currently no analytical formula to derive the probability density function $f_{Z_k^i}(z)$ of the extrinsic information at an iteration i for any decoder of a convolutional code. However, we can run Monte Carlo simulations of a decoder with appropriate inputs and investigate the produced extrinsic information sequences. The superscript i from the extrinsic information random sequences will be dropped since we will focus on conditional statistics of the extrinsic information given an input rather than unconditional statistics at a given iteration i. In other words, we examine the input-output relationship of a given decoder.

The definition below will be useful in representing the mutual information.

Definition 3 *The information content function $I(\alpha)$ for a given sequence $\{\alpha_k\}$ is defined as*

$$I(\alpha) = \frac{1}{K} \sum_{k=1}^{K} [1 - \log_2(1 + \exp(-\alpha_k))]. \tag{5.15}$$

In the context of extrinsic information this function can be considered as quantifying the reliability of a sequence or the information content of a sequence. For example, if $\alpha_k = 0$ for all k, then there is no information in the sequence $\{\alpha_k\}$ and $I(\alpha)$ equals 0. In the next section, we will show using the properties in Appendix B that if $Z = \{Z_k\}$ then $I(Z)$ approaches a normally distributed random variable as $K \to \infty$.

Furthermore, since the sequence $\{Z_k\}$ is assumed to be ergodic, $I(Z)$ converges to $I(E_k, U_k)$ (for any k since $\{Z_k\}$ is stationary) in the mean square sense [33] such that

$$\lim_{K\to\infty} E|I(Z) - I(E_k, U_k)|^2 = 0. \tag{5.16}$$

In essence, this function captures the information content in a sequence. Hence, we will use the parameter $I(Z)$ to track the progress of the iterative process. Note that interleaving a sequence does not change the information inherent in a sequence. This property is also reflected with the information content formulation.

When the information content is used as the parameter to track the extrinsic information progress, a decoder transforms an *a priori* information sequence $\{Z_k^a\}$ with $I(Z^a) = I_{in}$ into an extrinsic information sequence $\{Z_k^e\}$ with $I(Z^e) = I_{out}$. For shorter notation, we will denote $I(Z^a)$ of the *a priori* information sequence by $I(A)$ and $I(Z^e)$ of the extrinsic information sequence by $I(E)$. This characterization of a component decoder can be effectively done by using as the input the *a priori* information with different $I(A)$ and channel observations at different SNR values. Based on this, an input/output (IO) relation as in Fig. 5.3 can be obtained.

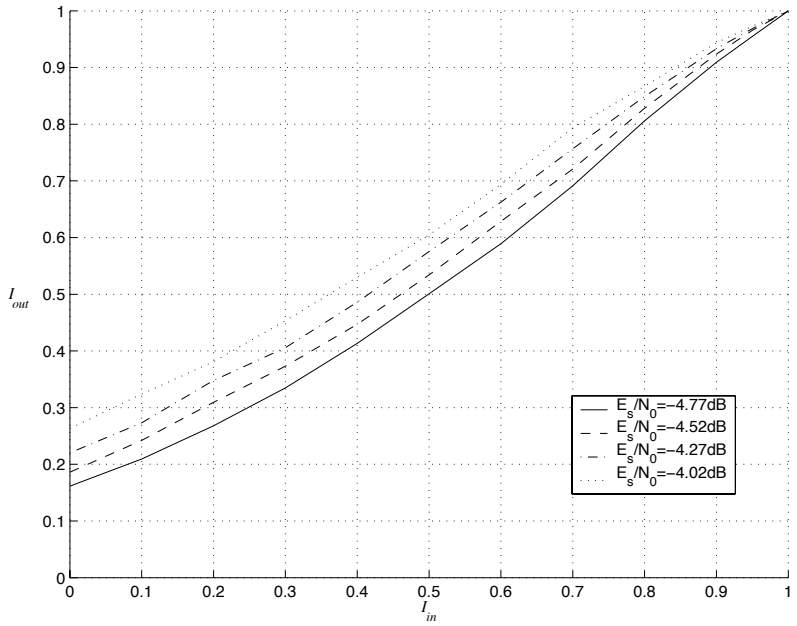


Figure 5.3: Input/output relation of a MAP decoder at different SNR values. I_{in} and I_{out} are $I(A)$ and $I(E)$ respectively.

The IO relation provides the I_{out} value conditioned on an input I_{in} value. Different component decoders have different IO relations. Note that increasing SNR shifts the IO relation in the upward direction as decoders produce more reliable information at higher SNRs. When the progress of a turbo decoder is investigated, the extrinsic information transfer (EXIT) chart method [6] is an effective tool. This method will be explained in detail in Chapter 6. The IO relations of both decoders are plotted on an EXIT chart. The first decoder's IO relation is plotted with x and y axes holding the $I_{in,1}$ and $I_{out,1}$ respectively. The second decoder's IO relation is plotted with y and x axes holding the $I_{in,2}$ and $I_{out,2}$ respectively. This enables a nice visualization of the extrinsic information progress in a turbo decoder. Fig. 5.4 gives an example of an EXIT chart and progress of extrinsic information.

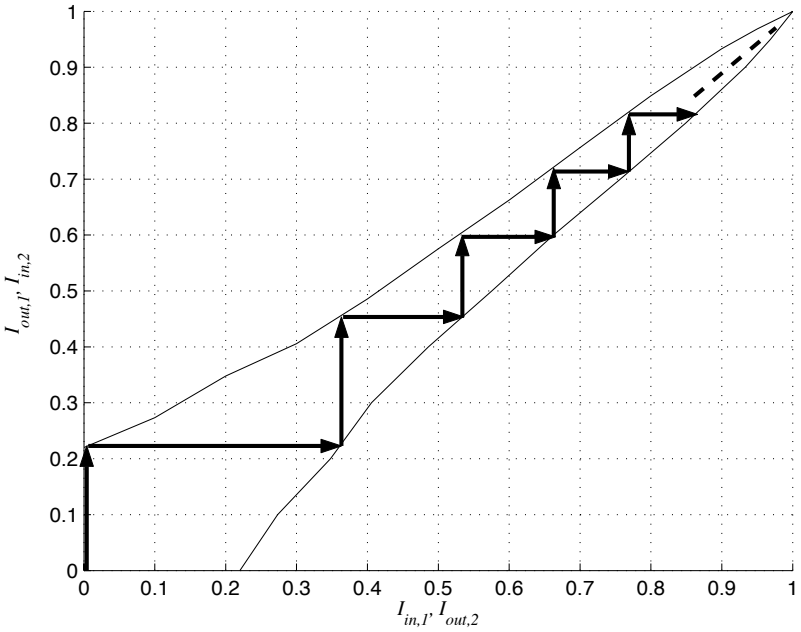


Figure 5.4: An EXIT chart and progress of extrinsic information.

In Fig. 5.4, since the component decoders used are the same, the IO relations are symmetric. The two curves do not intersect before $I_{in} = I_{out} = 1$ at this SNR value for which the IO relation is shown. In this case, the progress of extrinsic information continues until almost all the bits (excluding the errors due to ML-decoding errors) are decoded correctly ($I_{out} = 1$). These two curves might intersect before $I_{in} = I_{out} = 1$ at small SNR values for which the extrinsic information cannot progress beyond some

point. The lowest SNR value at which these two curves do not intersect marks the decoding threshold below which the turbo decoder cannot perform well. An extension of this idea will be made use of in the next chapter in order to obtain an approximation to the probability of decoding failure for iterative decoders.

5.3 The Nature of Extrinsic Information Sequence

As pointed out before, the extrinsic information sequence $\{z_k\}$ can be visualized as a length-K realization from a Gaussian source. We will denote by Z and Z_i the corresponding random vector and variables respectively. We will make use of the WSS assumption and obtain some properties for the sequence of variables in Z.

We will define the covariance function for this WSS sequence by

$$K_Z(m) = E[Z_i Z_{i+m}] - E[Z_i]E[Z_{i+m}]. \tag{5.17}$$

In order to find covariance function of the extrinsic information from a component decoder, we simulate large packets ($K = 100,000$) and obtain an estimate of $K_Z(m)$ by the unbiased auto-covariance estimator [33]

$$\hat{K}_Z(m) = \begin{cases} \frac{1}{K-|m|-1} \sum_{n=1}^{K-|m|} \left(z_{n+m} - \frac{1}{K}\sum_{i=1}^{K} z_i \right) \left(z_n - \frac{1}{K}\sum_{i=1}^{K} z_i \right) & , m \geq 0 \\ \hat{K}_z(-m) & , m < 0 \end{cases} \tag{5.18}$$

The choice of code is a memory-3 code with generator matrix

$$G = [1, \frac{D^2 + D^3}{1 + D^2 + D^3}]. \tag{5.19}$$

We evaluate $\hat{K}_Z(m)$ for this code's MAP decoder at an arbitrary SNR value with two arbitrary I_{in} values.

In Figures 5.5 and 5.6, the covariance function estimates from 4 different packets are plotted with two different input information values. The same properties are observed in both figures. First, no matter what the channel observation and *a priori*

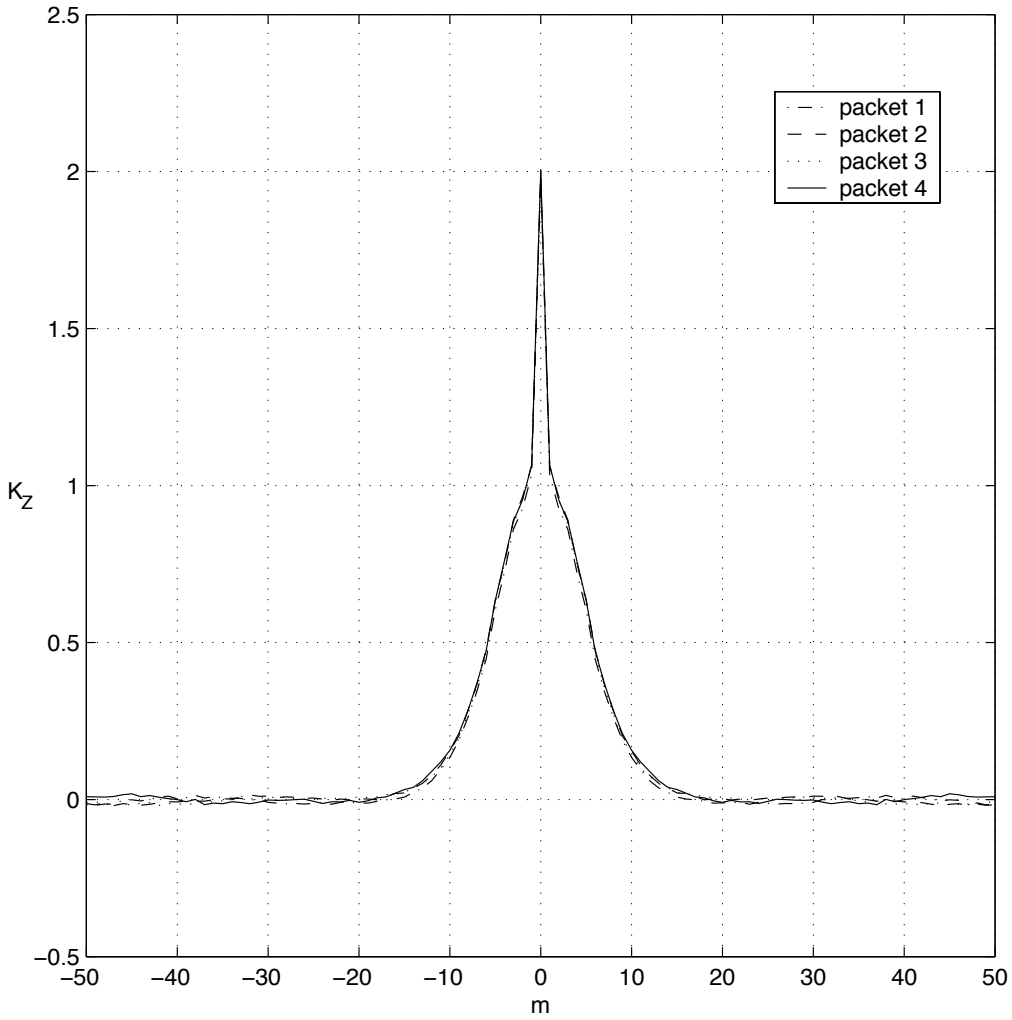


Figure 5.5: Estimates of covariance function $K_Z(m)$ for a memory-3 RSC code when $E_s/N_0 = -4.17$dB and $I_{in} = 0$.

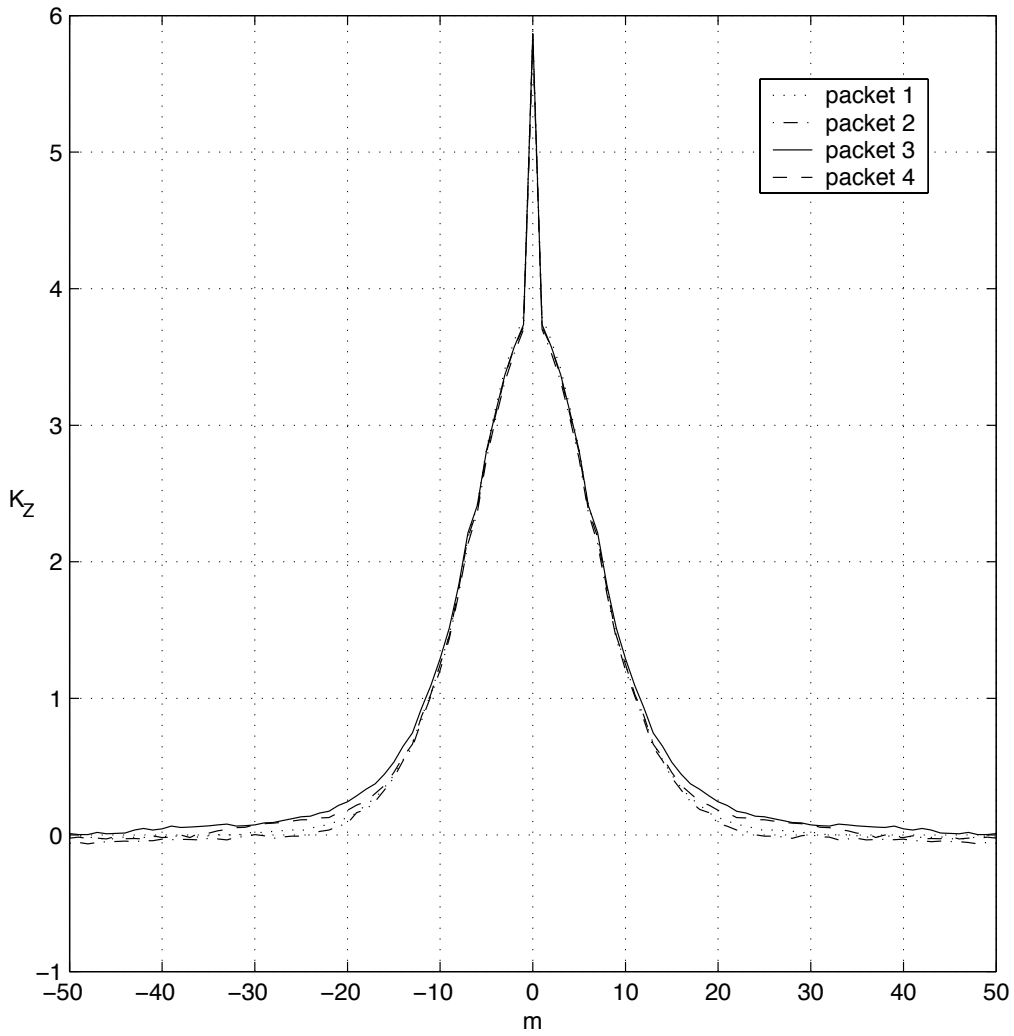


Figure 5.6: Estimates of covariance function $K_Z(m)$ for a memory-3 RSC code when $E_s/N_0 = -4.17$dB and $I_{in} = 0.5$.

information realizations are the covariance estimate is almost the same. Second, the shape of the covariance function is the same for both input information values: it is nonzero in some region and then zero for the rest. The size of the region where the covariance function is zero does not depend on the block length K. In the case of finite number of samples, a statistic obtained from one realization may be quite different than the statistic obtained from another one. Thus, a sufficiently large block length should be picked in order to suppress the effects of finite number of samples. A block length of $K = 100,000$ is sufficient in estimating the covariance function in this case.

By testing many other codes, SNR and I_{in} values we have observed that this structure of the covariance function is typical for MAP decoders. This is actually not a surprise since the MAP decoder obtains the likelihood of a bit by averaging (see Section 2.4 and Appendix A). Averaging makes the extrinsic information of the bits close to each other correlated. The structure of the covariance function implies that the correlation between the extrinsic information of two bits dies off quickly when the two bits are separated further from each other. When the covariance between the extrinsic information of two bits is zero, then the extrinsic information corresponding to these bits become independent under the Gaussian approximation. The properties of information content function will be investigated based on these findings in the next section.

5.4 Information Content Function

The information content function for the extrinsic information sequence $\{Z_k\}$ is written as:

$$I(Z) = \frac{1}{K}\sum_{k=1}^{K}[1 - \log_2(1 + \exp(-Z_k))]. \tag{5.20}$$

Define a new random variable

$$T_k = 1 - \log_2(1 + \exp(-Z_k)) = g(Z_k). \tag{5.21}$$

The extrinsic information sequence $\{Z_k\}$ is m-dependent and stationary as defined and proved in Appendix B. Since $g(x) = 1 - \log_2(1 + \exp(-x))$ is a strictly monotonically increasing function, $\{T_k\}$ is also stationary and m-dependent from Lemmas 4 and 5 in Appendix B. We will assume that $\{T_k\}$ is also ergodic. By definition, $I(Z)$ is the sample mean of the sequence $\{T_k\}$. By showing that the following properties of $\{T_k\}$ hold, we will prove that $I(Z)$ is normally distributed following Hoeffding-Robbins Theorem in Appendix B.

Since T_k is a function of Z_k, the mean and covariance of $\{T_k\}$ can be obtained from $\{Z_k\}$. But, when finding the mean and variance of $I(Z)$, we will directly use $\{T_k\}$ in practice. Thus, we will turn to empirical results in order to show that the mean and variance of the sample mean requirements in Hoeffding-Robbins Theorem hold.

- Each random variable in the sequence has to have the same finite mean. As $\{T_k\}$ is stationary, $E[T_i]$'s are all equal for all i. By the ergodicity assumption, we can find $E[T_i]$ for a very large packet. Below are the mean values found by sample averaging for 5 different packets of length $1,000,000$. The same code as in the previous section is used. The SNR is -4.17dB and $I_{in} = 0$. The averages are almost the same. This result is typical for any MAP decoder, SNR, and I_{in} value and thus suggests that $E[T_i] = c$ for any i and some $c < \infty$.

packet #	sample average
1	0.244297
2	0.246143
3	0.246585
4	0.245607
5	0.245837

- The covariance function will be found using the same procedure employed in the previous section. For an easier comparison, the plotted covariance functions are for the exact same cases as Figures 5.5 and 5.5. As clearly seen in the Figures

5.7 and 5.8, the covariance function has the dying off property again. That is,

$$K_T(m) = \begin{cases} \sigma_0^2 & , s = 0 \\ 0 & , |s| > m. \end{cases} \tag{5.22}$$

For such a covariance function, $\sum_{i=1}^{\infty} K_T(i)$ is convergent. By Lemma 2 in Appendix B, the expected value of the sample mean requirement of Hoeffding-Robbins Theorem is satisfied.

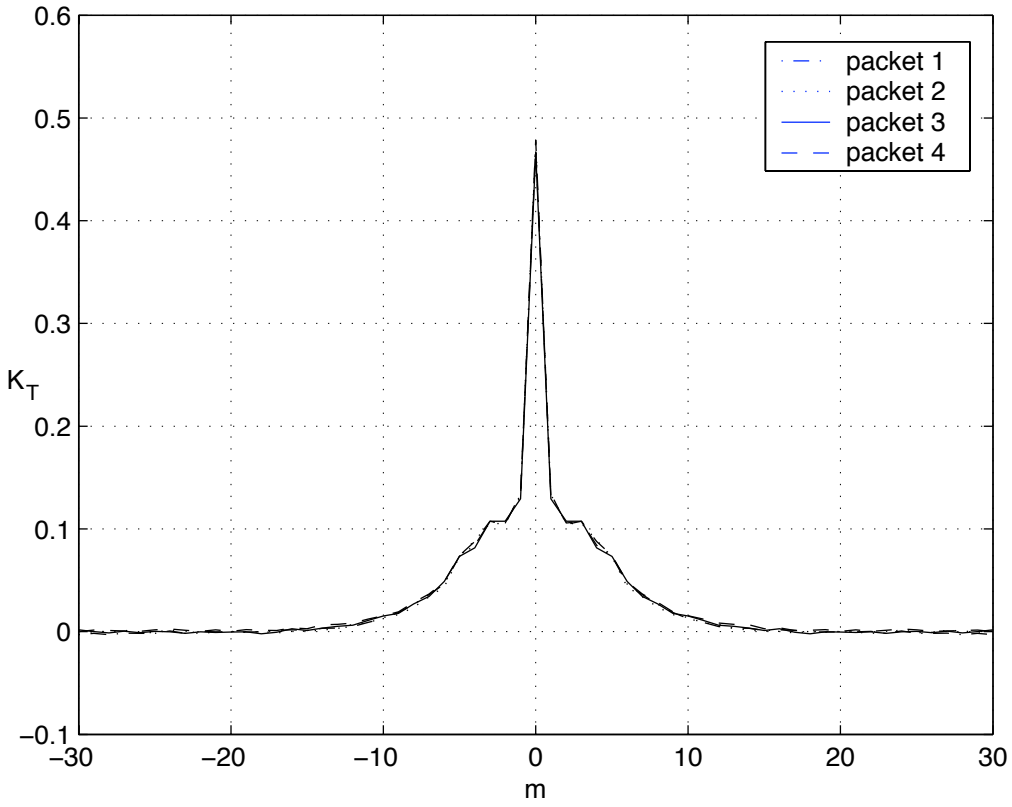


Figure 5.7: Estimates of covariance function $K_T(m)$ for a memory-3 RSC code when $E_s/N_0 = -4.17$dB and $I_{in} = 0$.

- The third requirement is that $E|T_i|^{2+\delta}$ is bounded.

$$\begin{aligned} E|T_i|^k &= \int_{-\infty}^{\infty} |1 - \log_2(1 + e^{-x})|^k f_{Z_i}(x) dx \qquad (5.23) \\ &= \int_{0}^{\infty} (1 - \log_2(1 + e^{-x}))^k f_{Z_i}(x) dx + \int_{-\infty}^{0} (\log_2(1 + e^{-x}) - 1)^k f_{Z_i}(x) dx. \end{aligned}$$

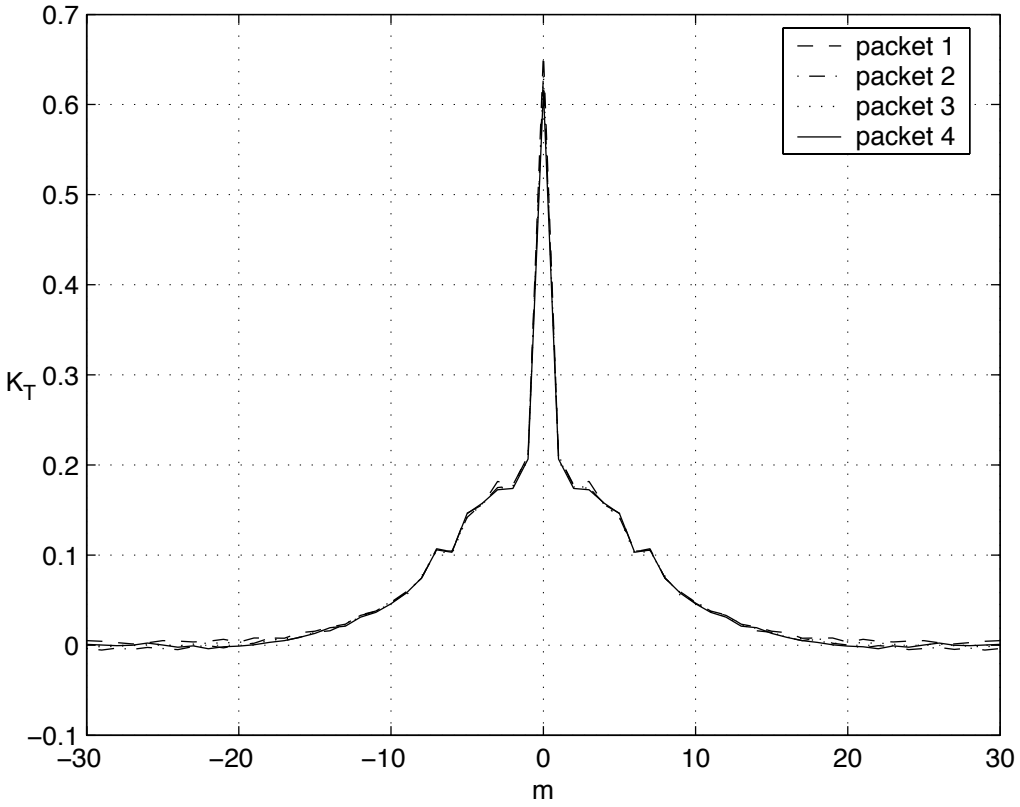


Figure 5.8: Estimates of covariance function $K_T(m)$ for a memory-3 RSC code when $E_s/N_0 = -4.17$dB and $I_{in} = 0.5$.

For $x > 0$, $g(x) = 1 - \log_2(1 + e^{-x}) \leq 1$. Also, $\log_2(2e^{-x}) \geq \log_2(1 + e^{-x})$ for $x \leq 0$. Then

$$\begin{aligned} E|T_i|^k &\leq 1 + \int_{-\infty}^{0} (\log_2(2e^{-x}) - 1)^k f_{Z_i}(x)dx \\ &\leq 1 + \int_{-\infty 0}^{0} (\frac{\ln(2e^{-x})}{\ln 2} - 1)^k f_{Z_i}(x)dx \\ &= 1 + \int_{-\infty}^{0} (\frac{-x}{\ln 2})^k f_{Z_i}(x)dx \\ &= 1 + \int_{-\infty}^{0} \frac{(-x)^k}{(\ln 2)^k} f_{Z_i}(x)dx \\ &\leq 1 + \int_{-\infty}^{\infty} \frac{|x|^k}{(\ln 2)^k} f_{Z_i}(x)dx \\ &\leq M', \end{aligned} \tag{5.24}$$

since $E|Z_i|^k \leq M$ for any i.

Based on the assumption that $\{Z_k\}$ is Gaussian and wide sense stationary, $\{T_k\}$ is an m-dependent sequence and has the above properties. Then, the sample mean of $\{T_k\}$ $(I(Z))$ is normally distributed with a fixed mean and variance by Hoeffding-Robbins Theorem. This will be the basis of probabilistic modelling of the MAP decoder in the next section.

5.5 Probabilistic Model of the MAP Decoder

In this section we will concentrate on the behavior of individual MAP decoders that exist within a turbo decoder. We will make use of the properties obtained in previous sections. Only convolutional codes and their MAP decoders will be considered and the results can be extended to other types of codes and decoders. First, we will provide experimental results for the finite length case. By using the aforementioned properties of information content function, we will estimate the finite-length behavior. At roughly the same time this study was being conducted, Lee and Blahut independently articulated the importance of MAP decoder's extrinsic information in the case of finite length turbo decoding in [34]. Both approaches are strikingly similar

in regard to the focus on the extrinsic information generated in the MAP decoder. We will comment on the similarities and differences of our work to [34] as appropriate.

5.5.1 Behavior of the MAP Decoder in the Finite Length Case

As mentioned in Section 5.2, the MAP decoder is a device that transforms the inputs (channel observation and *a priori* information) to the extrinsic information as its output. This device has been modelled in [6, 7] as a probabilistic one which transforms the statistics of the inputs. All the previous research on iterative decoding are aimed at infinite block lengths which is deemed as asymptotic performance. This approach naturally leads to the analysis of the individual MAP decoders with quite large block lengths. In the asymptotic case, the produced extrinsic information is regarded as a random vector with fixed parameters. We choose the information content as developed in the last section as the choice of parameter to investigate, but similar arguments hold for other parameters. This fixed parameter argument translates to the following statement: if an *a priori* information sequence with $I(A) = I_{in}$ is fed into the decoder, the generated extrinsic information always has a fixed $I(E) = I_{out}$ for fixed SNR no matter what the actual realizations of *a priori* information, noise or data realizations happen to be. In this case, the IO relation of the decoder is said to be deterministic.

In case of small-to-medium-size block lengths, the decoder's IO relation does not maintain the deterministic property. Even if the same statistics of the observation, data and *a priori* information are supplied to a MAP decoder, I_{out} differs by varying the realizations. We will observe this phenomenon by checking the I_{out} for the extrinsic information with the same input statistics but with differing realizations using the memory-2 component code with generator matrix $[1, \frac{1+D^2}{1+D+D^2}]$. In order to generate an *a priori* information sequence with a given mutual information I_{in}, a Gaussian sequence $\{z_k, k = 1, \ldots, K\}$ with some mean μ_Z corresponding to I_{in} has

to be generated through the following rewrite of (5.14)

$$I_{in} = 1 - \int_{-\infty}^{\infty} \frac{\exp(-(z-\mu_Z)^2/4\mu_Z)}{\sqrt{4\pi\mu_Z}} \log_2(1+e^{-z})dz \tag{5.25}$$

with the assumption that Z is normally distributed with mean μ_Z and variance $2\mu_Z$. With the Monte Carlo simulation principle, the above equation corresponds to

$$I_{in} = 1 - \frac{1}{K}\sum_{k=1}^{K} \log_2(1+e^{-z_k}), \tag{5.26}$$

where $\{z_k, k = 1, \ldots, K\}$ are samples from an uncorrelated Gaussian source with mean μ_Z and variance $2\mu_Z$. The equations (5.25) and (5.26) are equivalent when K is large. Using a large value for K the relation between μ_Z and I_{in} can be found. For a desired value of I_{in}, the sequence $\{z_k, k = 1, \ldots, K\}$ is obtained from the Gaussian source using the corresponding value of μ_Z. The sequence $\{e_k, k = 1, \ldots, K\}$ can be obtained from $\{z_k\}$ as specified before and input to the decoder[1].

Fig. 5.9 depicts the histogram of I_{out} for a large number of packets. 50000 different packets of data block length 512 were decoded wherein the data, channel and *a priori* information realizations are randomized. The data is assumed to be equally likely. The channel realizations are for a fixed SNR. The *a priori* information is generated from an uncorrelated Gaussian source with information content mean I_{in}. The histogram of the extrinsic information's I_{out} is plotted for the input sequences with mutual information close to the intended I_{in}. The simulation results of two different I_{in} values are plotted in the figure. In the first iteration of a turbo decoder, the a priori information is a sequence of zeros which corresponds to $I_{in} = 0$. For that case, all 50000 packets have been shown in the histogram. For nonzero values of I_{in}, we only show the histogram for the packets with I_{in} close to the intended mutual information. The number of packets for I_{in} within 0.01 for the case of intended mutual information of 0.6 is around $11,000$ in our experiment. Fig. 5.10 is the same snapshot

[1]As shown in the Section 5.4, I_{in} is also a random variable and not all the sequences obtained in this way will have mutual information very close to I_{in}. We will only consider the sequences that have mutual information within a small neighborhood of the desired I_{in} in our plots.

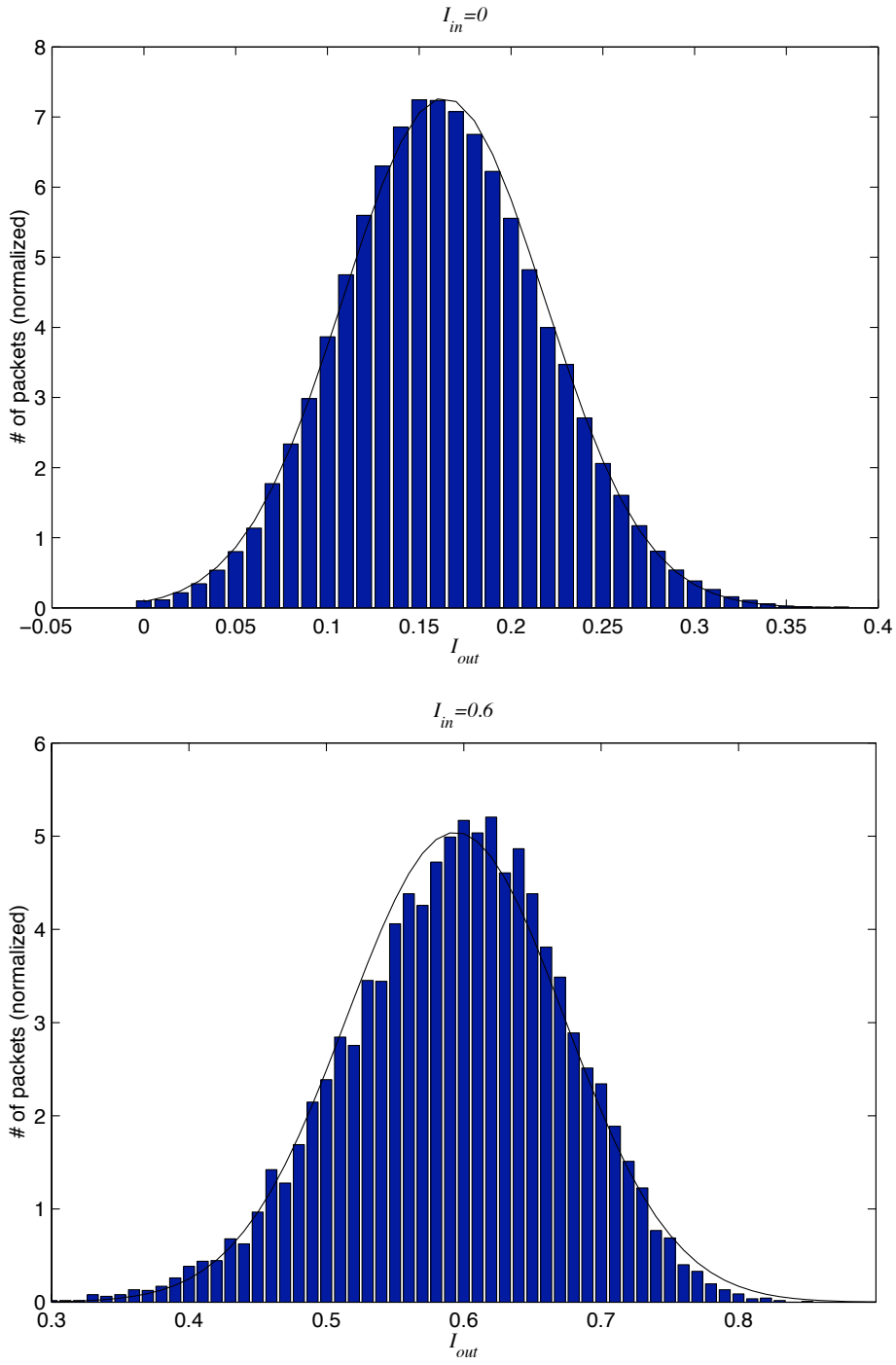


Figure 5.9: Mutual information I_{out} for each packet and the corresponding normalized histogram at $E_s/N_0 = -4.77$dB which corresponds to $E_b/N_0 = 0$dB if the code has a 1/3 overall rate (The solid line is the probability density function of a Gaussian random variable normalized with the mean, variance and the total number of packets).

at a different SNR value.

As seen in the figures, $I(E)$ for the extrinsic information produced by the MAP decoder is not fixed for given input statistics. Instead, $I(E)$ has a distribution which is itself Gaussian-like as predicted in Section 5.4. Similar results have been obtained in [34] with the aim of providing qualitative explanation of turbo decoding at finite block lengths. We will assume in the remainder of this study that $I(E)$ is normally distributed. Although this approximation does not quite hold when the mean of $I(E)$ is close to 1, it is still a good approximation in most cases.

5.5.2 Estimating $I(E)$

We showed in previuos sections that $I(E)$ has a normal distribution when the $\{Z_k\}$ sequence is assumed to be a WSS Gaussian sequence. The experimental data presented in Section 5.5 supports the normal distribution claim. In this section we will explain how to obtain estimates of the mean and variance of $I(E)$.

In Section 5.3 it is pointed out that a packet with large block length can provide good estimates for both the mean and variance of $I(E)$. An arbitrarily chosen memory-2 code with generator matrix

$$[1, \frac{1+D+D^2}{1+D^2}] \tag{5.27}$$

will be employed to show that these estimates can be obtained even with not so large block lengths.

The $\{t_k\}$ sequence of the output of a decoder is obtained from the extrinsic information sequence $\{z_k\}$ by $t_k = 1 - \log_2(1+e^{-z_k})$. The sample mean of the $\{t_k\}$ sequence for a large block length is an estimate of the expected value of $I(E)$. We will denote this expected value by μ_T and

$$\mu_T = \frac{1}{K}\sum_{i=1}^{K} t_i. \tag{5.28}$$

In addition, an estimate $\hat{K_T}(i)$ of the covariance function $K_T(i)$ can be obtained from

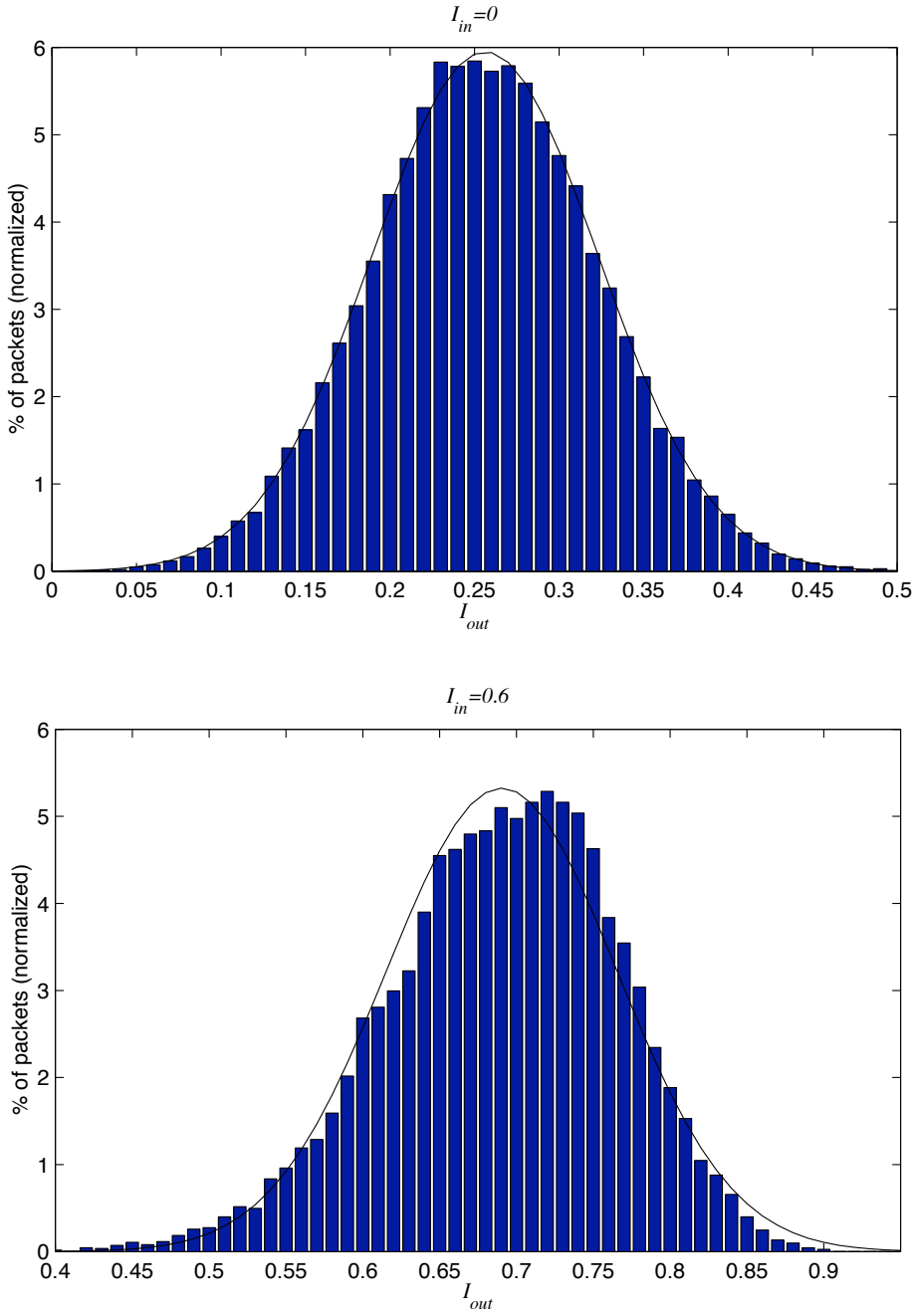


Figure 5.10: Mutual information I_{out} for each packet and the corresponding normalized histogram at $E_s/N_0 = -4.07$dB which corresponds to $E_b/N_0 = 0.7$dB if the code has a 1/3 overall rate(The solid line is the probability density function of a Gaussian random variable normalized with the mean, variance and the total number of packets).

the same packet by the unbiased estimator defined in Section 5.3. The variance of $I(E)$ will be denoted as σ_T^2 and is found by

$$\sigma_T^2 = \frac{1}{K} \sum_{i=-s}^{s} \hat{K_T}(i), \tag{5.29}$$

where s is the maximum value where $K_T(s)$ is nonzero. Again, $I(E)$ for the extrinsic information produced by the MAP decoder is not fixed for given input statistics. Rather, $I(E)$ is a random variable normally distributed with mean μ_T and variance σ_T^2. For infinitely large block lengths, σ_T^2 goes to zero. This is the reason why there is no variation in the information content of infinitely long extrinsic information sequences as previously studied in [6,7,16].

In Fig. 5.11 there are two plots: one for the mean and one for the standard deviation of $I(E)$. The lines are obtained by running decoders with many different packets of $K = 512$ at selected I_{in} values. The solid line is the estimate of the parameter corresponding to the plot. The dashed lines are the 95% confidence intervals (obtained for unknown variance [31]) for the estimates. The '+' markers are estimates based on one long packet of length 100,000. Fig. 5.12 shows the same plots at another SNR value. The one-packet estimates are usually within 95% intervals in this case for both the mean and standard deviation values. Equivalent results hold for other codes and SNR values as well. If one needs better estimates, using larger block lengths is always an option. Based on this approach, we arrive at an IO relation represented in the error bar notation as in Fig. 5.13. With the error bar notation, the mean of the output $I(E)$ conditioned on the input $I(A) = I_{in}$ values is plotted with a line. The standard deviation of $I(E)$ conditioned on $I(A) = I_{in}$ is shown with the bars around the mean value.

5.6 Conclusions

We have studied the input-output relation of component codes of an iterative decoder by focusing on the properties of extrinsic information. We developed a method

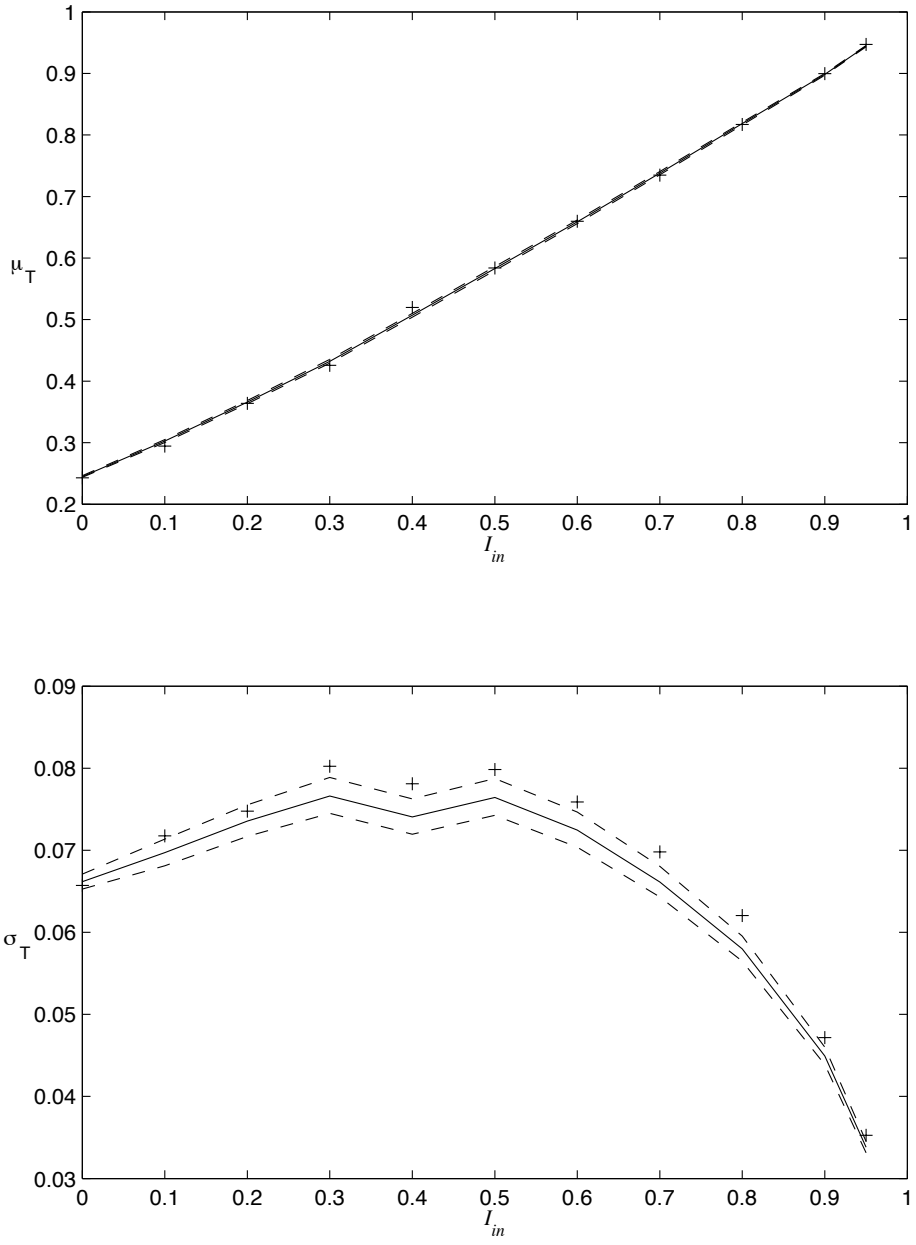


Figure 5.11: Estimates of μ_T and σ_T at $E_s/N_0 = -4.27$dB.

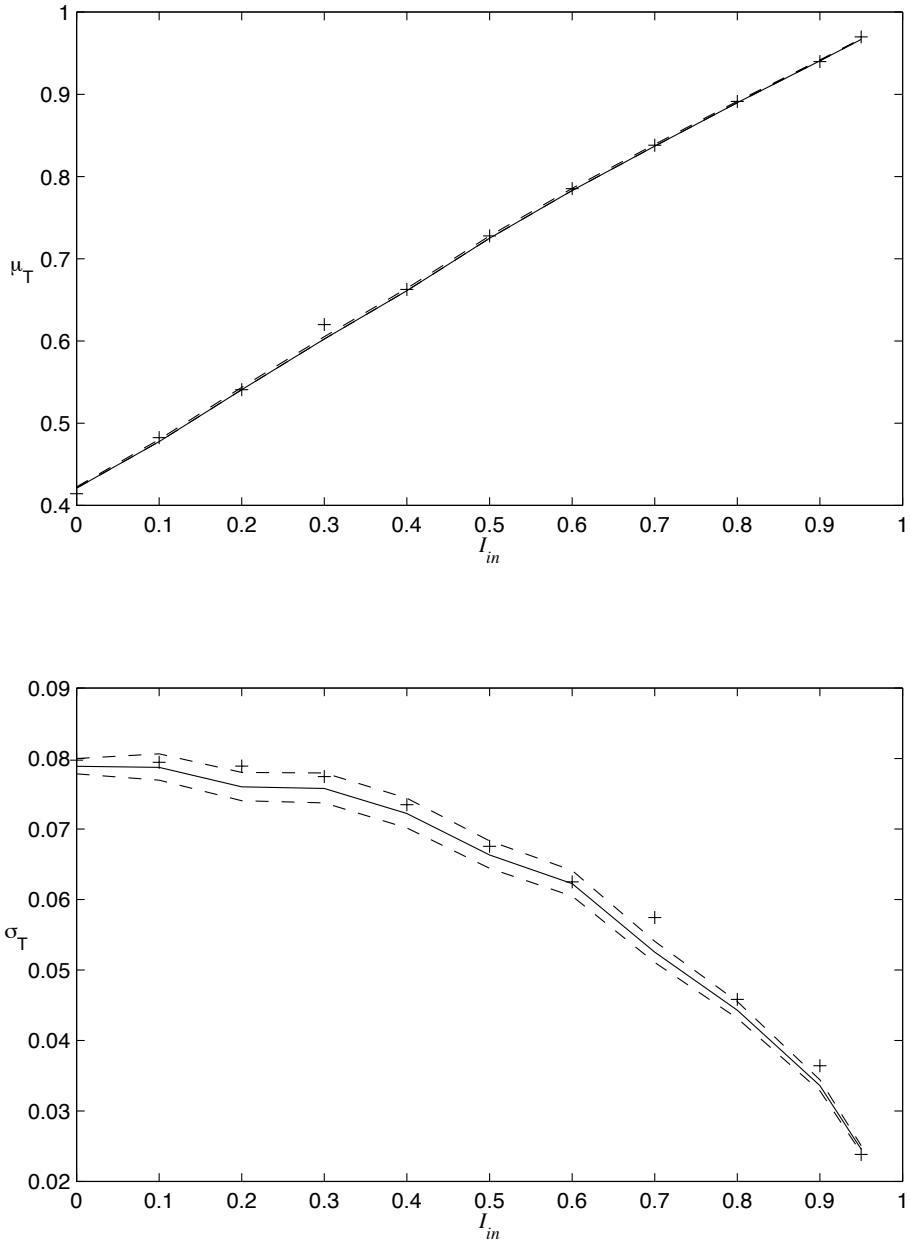


Figure 5.12: Estimates of μ_T and σ_T at $E_s/N_0 = -3.27$dB.

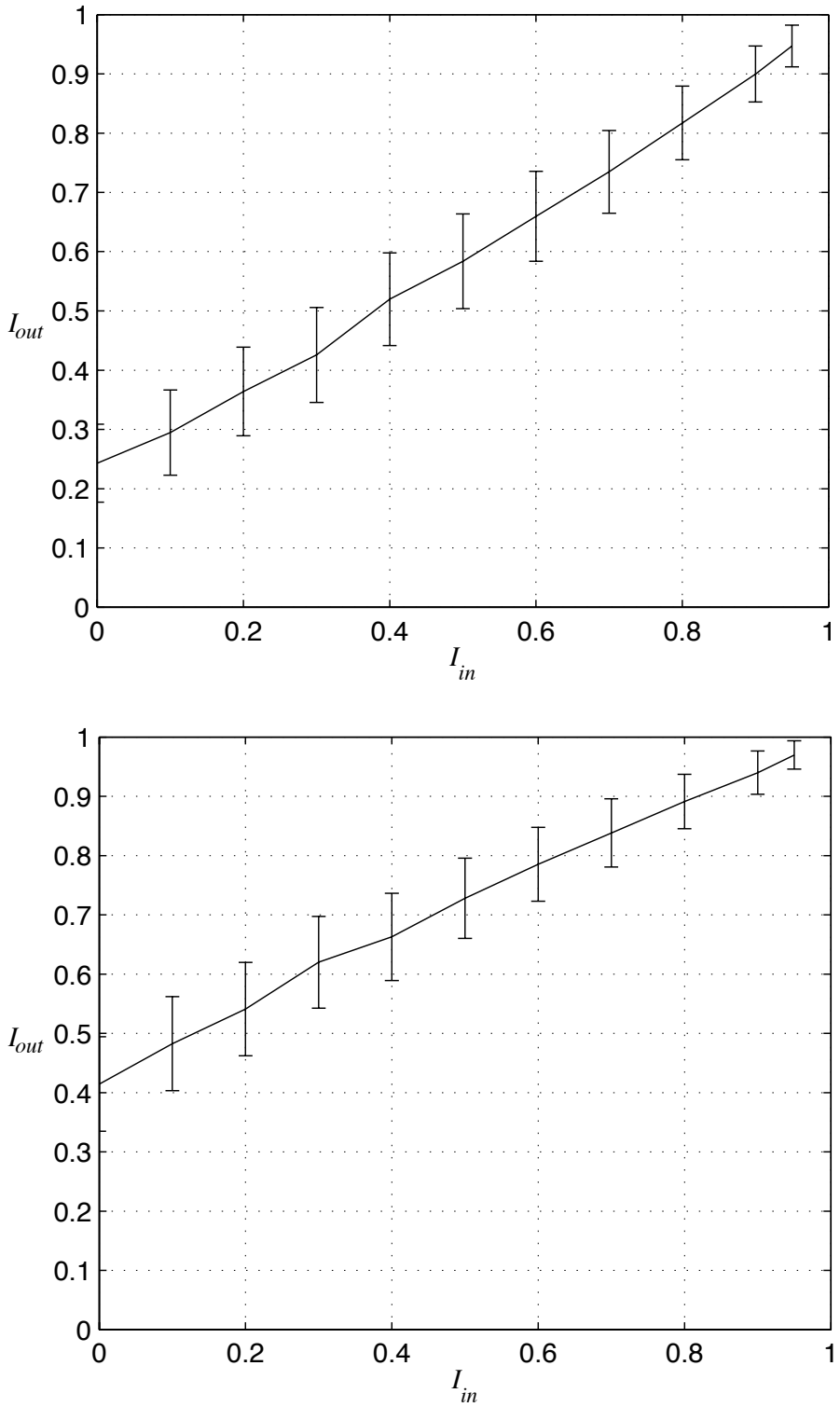


Figure 5.13: IO relation for a memory-2 code at $E_s/N_0 = -4.27$db (top) and $E_s/N_0 = -3.27$dB (bottom).

to model the behavior of a component MAP decoder in the finite block length case. Any decoder and concatenation scheme can be studied in a similar way. This characterization of the finite-length behavior will be utilized in the next chapter to predict the performance of a turbo decoder in the region where decoding failures dominate.

CHAPTER 6

EXIT Chart Method to Approximate Probability of Decoding Failure

Properties of component decoders of an iterative decoder were investigated in the previous chapter. This chapter will be devoted to the application of this knowledge to the study of decoding failure events in an iterative decoder.

6.1 Introduction

In Chapter 3, decoding failure is defined to be the event that an iterative decoder does not yield the optimal solution. Decoding failures are the prevalent cause of incorrect decoding in the waterfall region where the performance of a turbo code improves rapidly with a small increase in SNR. Performance of turbo codes in the waterfall region are not affected much by a particular choice of interleaver and the distance spectrum of the code corresponding to the interleaver [32]. Since the errors in the waterfall region is dominantly due to decoding failures and the waterfall region corresponds to low/medium SNR, an approximation to the probability of decoding failure becomes an approximation to the packet error rate in the low/medium SNR region.

The extrinsic information transfer (EXIT) chart method was proposed to predict the performance of turbo codes for the case of infinitely large interleavers in [6]. It gives a capacity-like decoding threshold for a given code above which zero error rate is

possible. It is reported that the use of mutual information in the EXIT chart results in decoding thresholds that can closely predict the behavior of the iterative decoder.

The probabilistic model of component decoders will be used in EXIT charts to approximate the probability of decoding failure for the case of finite interleaver sizes. The iterative decoding has the Markov property that the future (extrinsic information to be produced) is independent of the past (previous *a priori* information inputs) given the present (current *a priori* information input). A Markov process model will be defined and employed to obtain an approximation to the probability of decoding failure.

In Section 6.2 the EXIT chart method will be explained. We will extend the EXIT chart methodology to the study of turbo decoding for finite block lengths in Section 6.3. Based on the EXIT chart method for finite block lengths, we will develop a Markov process model to estimate the probability of decoding failure in Section 6.4. The chapter will be concluded by numerical results in Section 6.5.

6.2 The EXIT Chart Method for Infinitely Large Block Lengths

As briefly discussed in Chapter 5, the input-output (IO) relations of the component decoders are instrumental in obtaining decoding thresholds using the EXIT chart methodology. The IO relation of a decoder reveals the information content of extrinsic information (output) conditioned on a given value of information content of *a priori* information (input). The IO relations of both component decoders have to be considered while constructing the chart. The IO relation of the first code's decoder (C_1) is plotted on the $x-y$ axes. The IO relation of the second code's decoder (C_2) is plotted on the $y-x$ axes (swapped coordinates). This enables a stepwise illustration of the iterative decoding process. The iterations corresponding to C_1 and C_2 are displayed in the vertical and horizontal directions respectively.

In Fig. 6.1, trajectories of 4 arbitrarily chosen packets are shown. A fixed number

of 30 iterations were run in each simulation presented in this section. The code used in the figure is a symmetric turbo code of a memory-2 code with generator matrix $[1, \frac{1+D^2}{1+D+D^2}]$. The block length is taken to be $K = 100,000$ to mimic the case of an infinitely large block length. The IO relations of the two component decoders intersect in the EXIT chart for the specific SNR value of the figure. This implies that the information $I(E)$ in the extrinsic information sequence should not progress after the intersection point. That is exactly what is observed in the figure. Fig. 6.2 provides a more detailed picture of the region where the IO relations of the component decoders intersect. Since K is quite large in this case, the variation in the IO relations is small. Yet, the randomness predicted for the iterative process in the last chapter can be observed even at this considerably large value of K. Small differences add up and thus the trajectories are different.

Fig. 6.3 depicts the trajectories of 4 arbitrary packets of $K = 100,000$ of the same code at a higher SNR value. The IO relations of the component decoders do not intersect at the SNR value of the figure. The progress of $I(E)$ continues until $I(E)$ is approximately 1 for all packets in that case. Almost all the bits, excluding the ML-decoding errors, are decoded correctly when $I(E) \approx 1$. Small differences between trajectories are present but do not cause any decoding failure.

In order to obtain a rule to determine the decoding threshold, a more general scenario than the symmetric case has to be taken into account. The IO relations of two component decoders are identical for symmetric turbo codes. In asymmetric turbo codes, different component codes are used for turbo code construction. The component decoders do not necessarily have the same IO relations. Consider the IO relations of two different component codes C_1 ($[1, \frac{D}{1+D^2}]$) and C_2 ($[1, \frac{1+D+D^3+D^4}{1+D^3+D^4}]$) as in Fig. 6.4. As predicted by the EXIT chart method, the iterative decoder of this asymmetric code is able to decode all the packets correctly at the SNR value of the figure. The trajectory of an arbitrarily chosen packet is shown in the figure.

All the results obtained in this section imply that if two IO relation curves do not intersect on the EXIT chart then correct decoding occurs [6, 7]. This statement

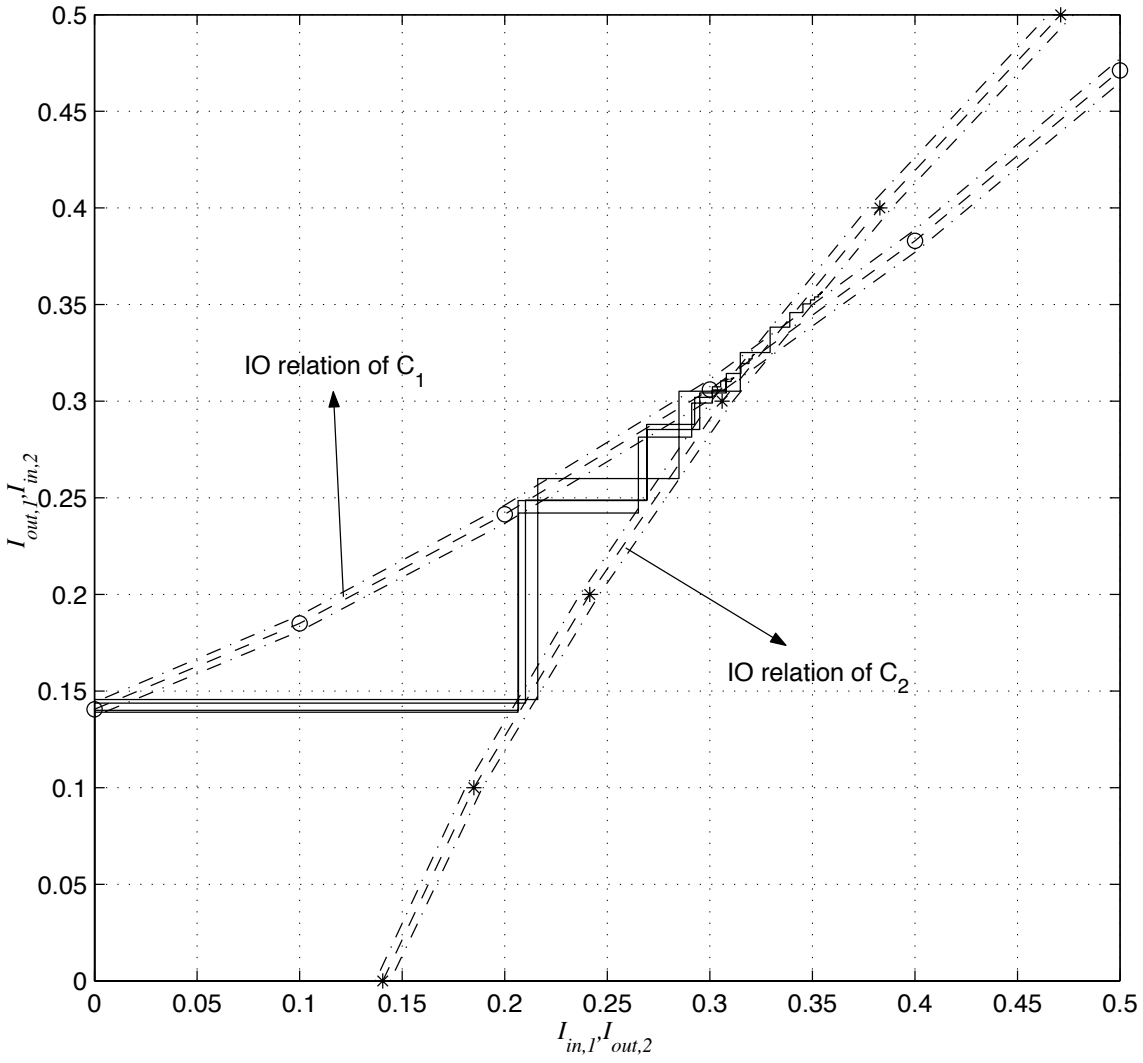


Figure 6.1: Trajectories of an iterative decoder at $E_b/N_0 = -0.2$dB. The lines marked with 'o' and '*' correspond to the IO relations of C_1 and C_2 respectively. The dash-dotted line shows the σ-bands of the IO relations for $K = 100,000$.

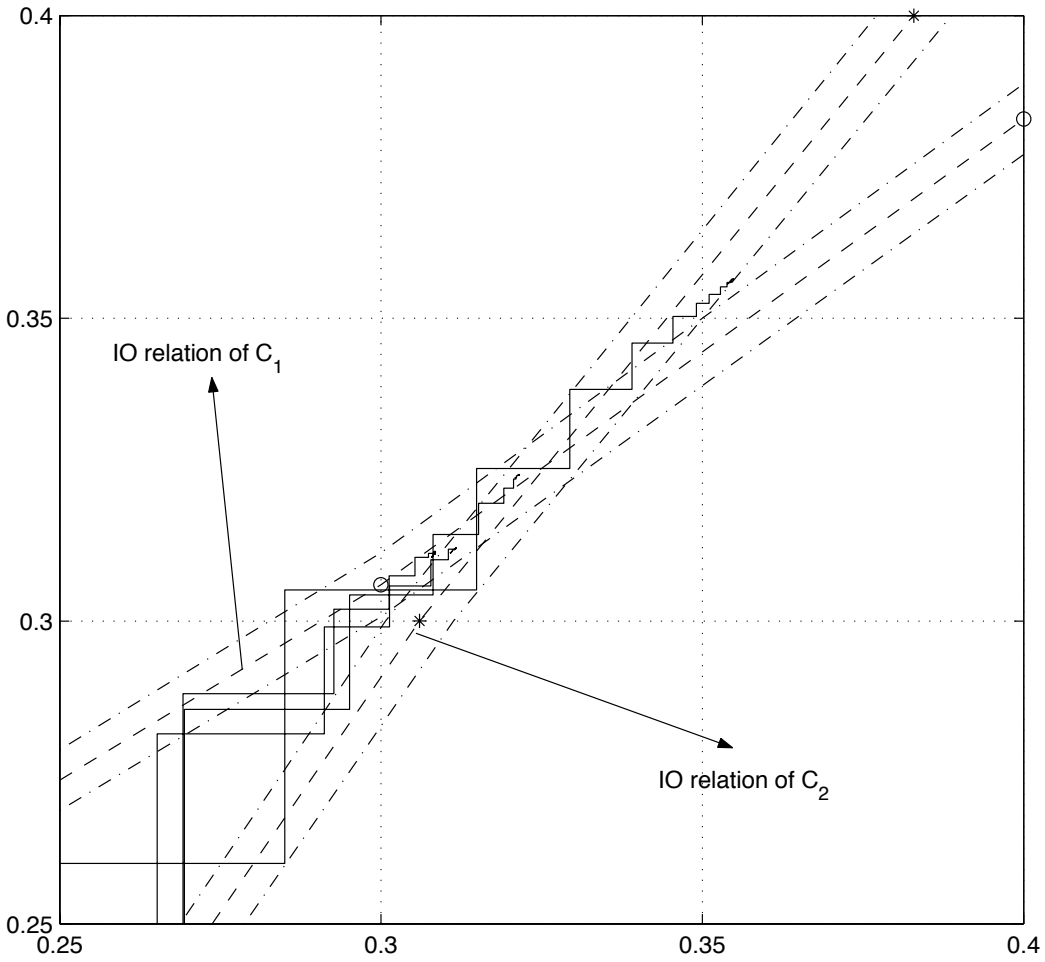


Figure 6.2: Trajectories of an iterative decoder at $E_b/N_0 = -0.2$dB. The lines marked with 'o' and '*' correspond to the IO relations of C_1 and C_2 respectively. The dash-dotted line shows the σ-bands of the IO relations for $K = 100,000$.

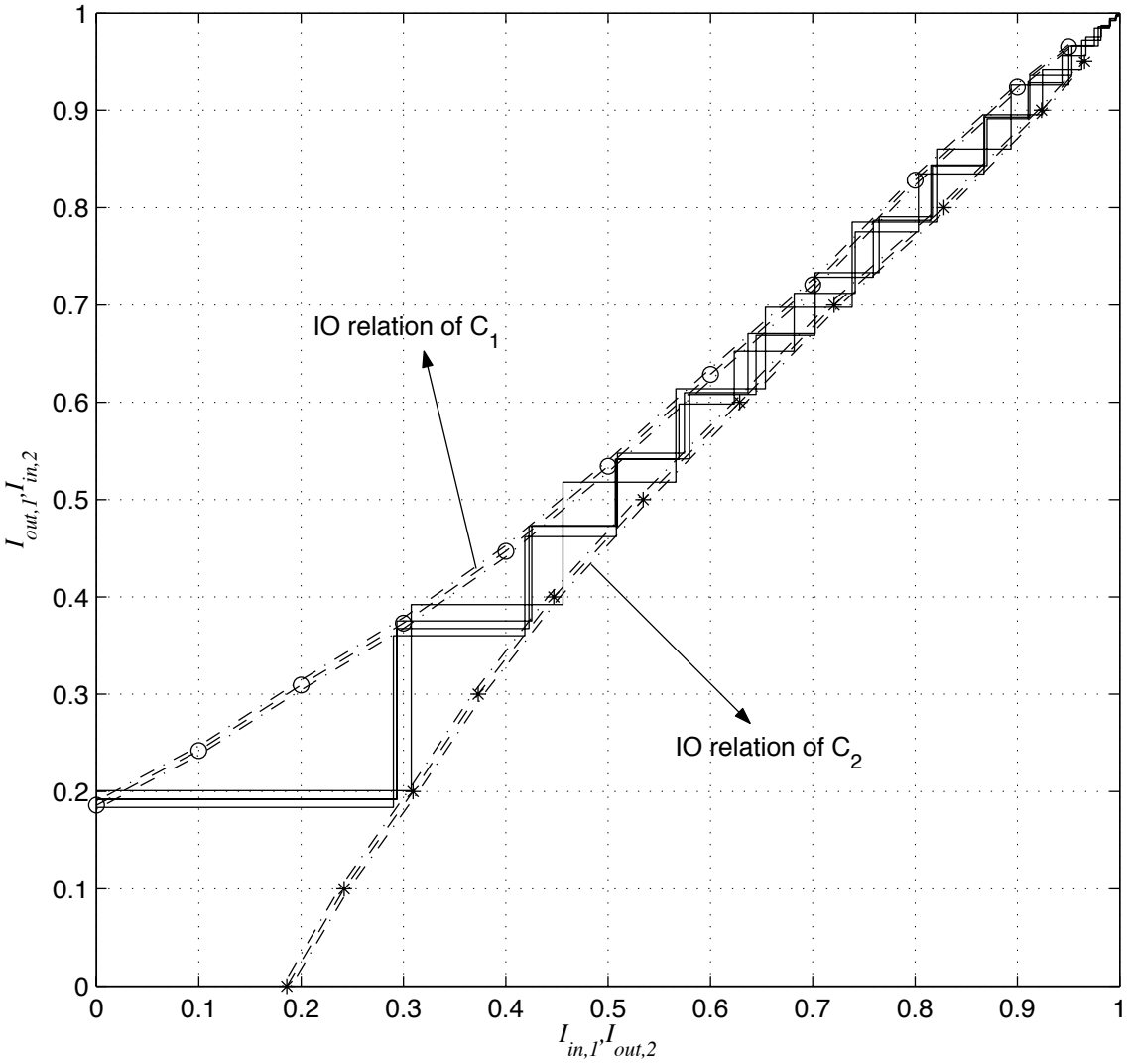


Figure 6.3: Trajectories at $E_b/N_0 = 0.25$dB on the same EXIT chart. The lines marked with 'o' and '*' correspond to the IO relations of C_1 and C_2 respectively. The dash-dotted lines show the σ-bands of the IO relations for $K = 100,000$.

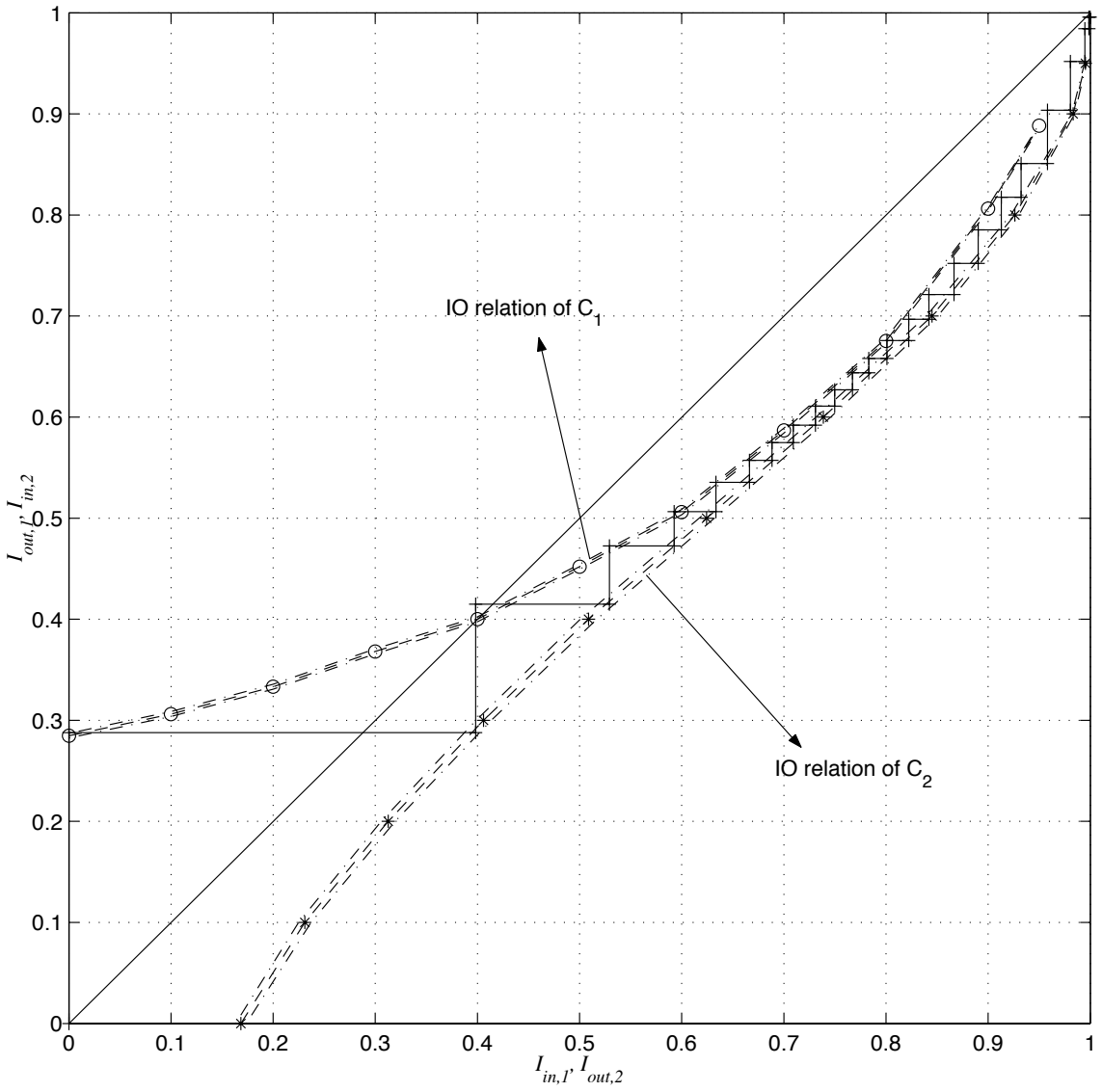


Figure 6.4: Trajectory of a packet at $E_b/N_0 = 0.5$dB for the asymmetric case. The lines marked with 'o' and '*' correspond to the IO relations of C_1 and C_2 respectively.

corresponds to

$$IO_{C_1}(x) > IO_{C_2}^{-1}(x) \tag{6.1}$$

for any x (except $x = 1$), where $IO_{C_i}(\cdot)$ is the IO relation of the i^{th} component decoder inside the iterative decoder. Let I_k denote the extrinsic information content I_{out} at the k^{th} iteration (I_1 is I_{out} from the first decoder, I_2 is I_{out} from the second decoder, I_3 is I_{out} again from the first decoder and so on.) Without loss of generality, assume $I_i = x$ for some i and $x \in [0, 1]$. Since the IO relation does not have any variance in the case of infinitely large block lengths, $I_{i+1} = IO_{C_1}(x)$ for the next iteration. For the previous iteration, $I_{i-1} = IO_{C_2}^{-1}(x)$. In case of correct decoding, $IO_{C_1}(x) > IO_{C_2}^{-1}(x)$ for all x. Then, $I_{i+1} > I_{i-1}$. Hence, the intersection statement can be translated into: "Iterative decoding does not have a decoding failure if

$$I_{i+2} > I_i$$

for all i" [6]. Although the intersection argument can be equally helpful for the case of infinitely large block lengths, it does not mean much for the finite length case. In the finite length case, there is a variation of the IO relation. The rule "$I_{i+2} > I_i$" is very important in that respect.

6.3 EXIT Chart Method for Finite Block Lengths

As it was discussed in detail in Chapter 5, the IO relation of a code is probabilistic. Although the EXIT chart method cannot be of direct help for finite block lengths, the basic idea can still be applied. The basic idea is to extract some property from the IO relations of component decoders that can be used to predict the behavior of iterative decoding. For the infinite length case, this property is stated as the intersection rule or "$I_{i+2} > I_i$". Through experimentation, we will try to decide on such a rule to be used for finite block lengths that is both practical and has good prediction capability.

In Fig. 6.5, 4 correctly decoded packets which obey the "$I_{i+2} > I_i$" rule are shown. An asymmetric turbo code with two different component codes having generator

matrices $[1, \frac{1+D^2}{1+D+D^2}]$ and $[1, \frac{1+D+D^3+D^4}{1+D^3+D^4}]$ are used. Two different SNR values are considered. Most of the correctly decoded packets obey the rule as the packets in this figure. Fig. 6.6 depicts 4 correctly decoded packets which do not obey the mentioned rule. In each of the packets' trajectories there is a tangling point where there is a decrease in $I(E)$. However, the component decoders recover after a few iterations and the progress resumes. This case is observed less often compared to the former case. So, some packets do not obey the rule, yet they are still decoded correctly.

In order to approximate the probability of decoding failure in some way, a rule similar to "$I_{i+2} > I_i$" is necessary. In the case of incorrect decoding (with regard to decoding failure, not the ML-decoding errors), the iterative process cannot go beyond some value of $I(E)$ away from $I(E) \approx 1$. It is either stuck at a fixed point or oscillates around some point. We discussed properties of different convergence categories with respect to fixed and oscillating points in Chapter 3. Since $I(E)$ is stuck around a point, any rule, be it "$I_{i+1} > I_i$", "$I_{i+2} > I_i$" etc., can detect decoding failures. Then, the accuracy of a rule is contingent on the frequency that the rule can predict the correct decoding of a packet. For this purpose, we ran simulations to follow trajectories and evaluate how frequently some given rules can predict correct decoding.

In Fig. 6.7 the percentage of correctly packets which obey a particular rule is plotted as a function of SNR. The rules are k-step rules such that decoding is deemed correct whenever $I(E)$ keeps on increasing in every k iterations. In general, it should be expected that rules that can allow more steps permit more failures to recover and thus should perform better. This is almost what is observed in Fig. 6.7. The 1-step rule is quite bad. The 2-step rule has a considerable performance improvement in comparison to the 1-step rule. The 3-step rule performs worse than the 2-step. (The EXIT chart in Fig. 6.4 clearly shows the case where the 3-step rule will not work properly.) The 4-step rule is better than the 2-step. However, the difference is not as striking as the difference between the 1-step rule and the 2-step.

As seen in Fig. 6.7 all the rules perform better as the SNR increases. The 2-step rule "$I_{i+2} > I_i$" is the rule by which the decoding threshold for infinitely large block

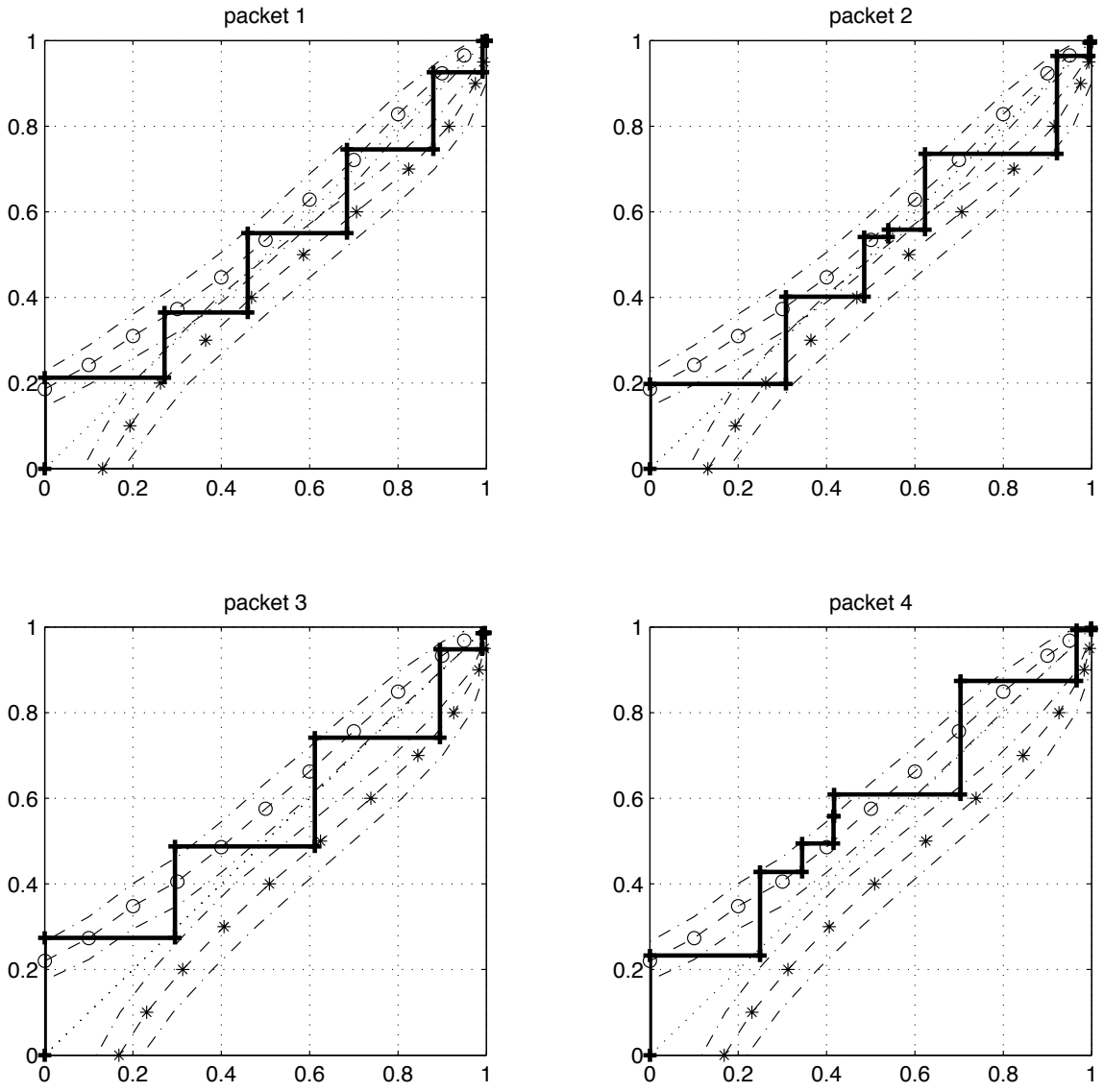


Figure 6.5: Trajectories of 4 packets of $K = 1024$ at $E_b/N_0 = 0$dB (packets 1 and 2) and 0.5dB (packets 3 and 4). The lines marked with 'o' and '*' correspond to the IO relations of C_1 and C_2 respectively. The dash-dotted lines show the σ-bands of the IO relations for $K = 1024$.

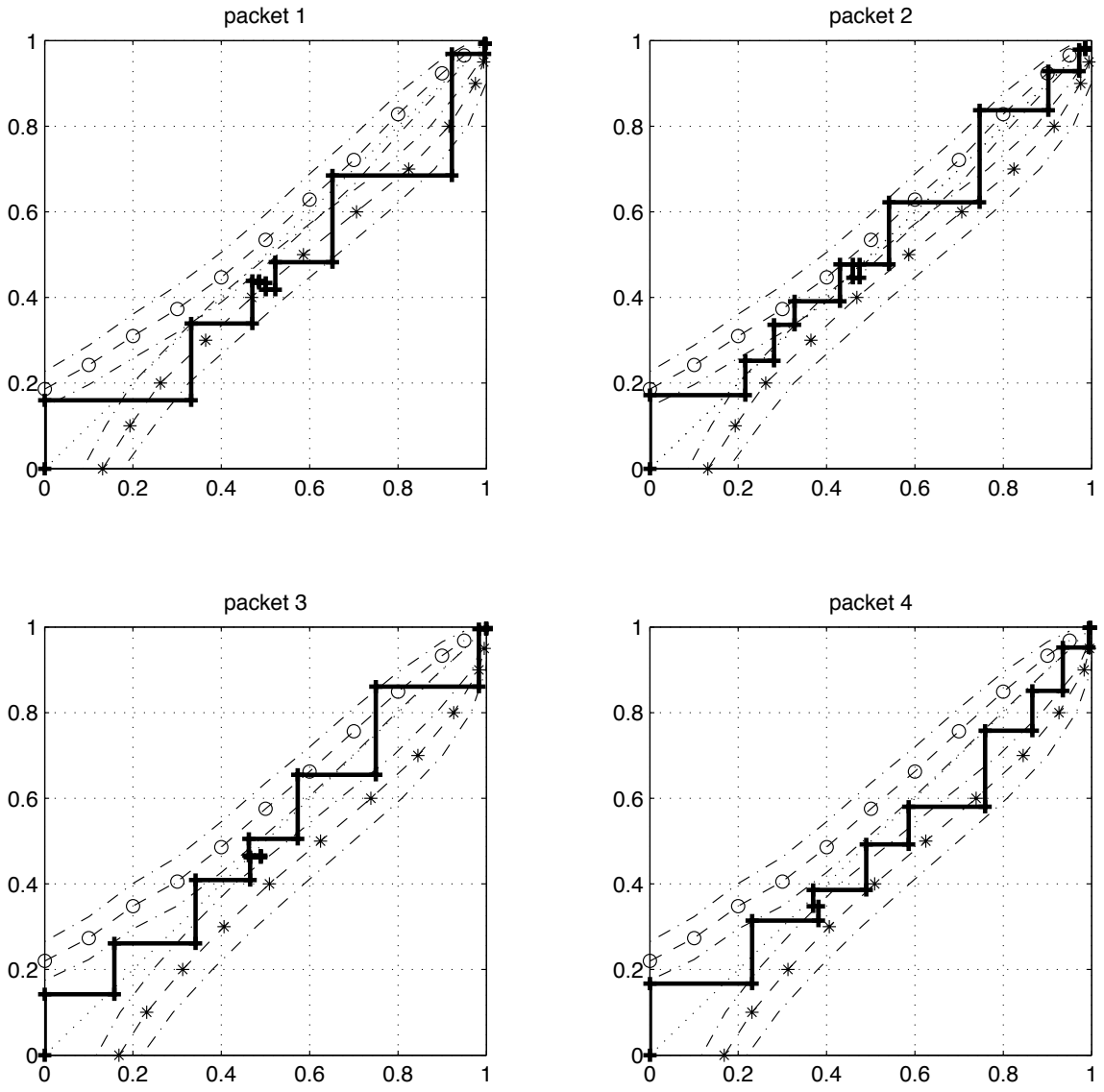


Figure 6.6: Trajectories of 4 packets of $K = 1024$ at $E_b/N_0 = 0$dB (packets 1 and 2) and 0.5dB (packets 3 and 4). The lines marked with 'o' and '*' correspond to the IO relations of C_1 and C_2 respectively. The dash-dotted lines show the σ-bands of the IO relations for $K = 1024$.

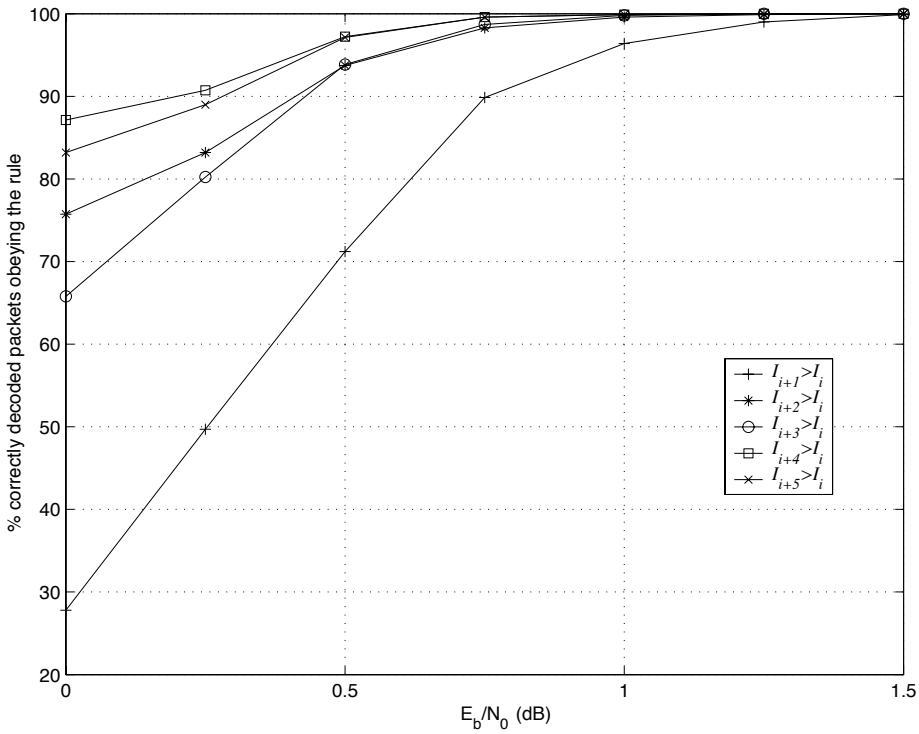


Figure 6.7: The percentage of correctly decoded packets that obey a certain rule.

lengths can be determined. Above the decoding threshold arbitrarily small error rates are possible. The 2-step rule also performs quite well in the finite length case. The simplicity of this rule compared to higher order rules should be kept in mind as well. Based on all these arguments, we choose this rule to use for evaluating the probability of iterative decoding failure.

6.4 A Markov Model of Iterative Decoding

The extrinsic information produced by one decoder is fed as the *a priori* information to the other decoder in the next iteration. The Gaussian assumption, as explained in Chapter 5, asserts that *a priori* information sequences that are fed into a component decoder at different iterations are independent. This suggests that the extrinsic information produced by a component decoder at any iteration depends only on the observation and the *a priori* information at that iteration. It does not depend on the *a priori* information at earlier iterations. Consequently, the Gaussian

assumption implies a memoryless property in iterative decoding so that past *a priori* information can be ignored. This further implies that I_{k+1}, as defined in the last section, only depends on I_k when conditioned on a given observation. We will make use of this Markov property in order to approximate the probability of decoding failure.

Definition 4 *Let $\{X_k, k = 1, 2, 3, \ldots\}$ be a discrete-time stochastic process with state space S. The conditional density function of X_{t_n} is denoted by $f_{X_{t_n}|X_{t_{n-1}},\ldots,X_{t_1}}(\cdot|\cdot,\ldots,\cdot)$. $\{X_k, k = 1, 2, 3, \ldots\}$ is said to be a Markov process when, for any collection of $t_1 < t_2 < \ldots < t_n \in T$ and $x_1, x_2, \ldots, x_n \in S$,*

$$f_{X_{t_n}|X_{t_{n-1}},\ldots,X_{t_1}}(x_n|x_{n-1},\ldots,x_1) = f_{X_{t_n}|X_{t_{n-1}}}(x_n|x_{n-1}),$$

provided that both probability density functions exist.

By the above definition of a Markov process, the iterative decoding process represented by the $\{I_k\}$ sequence forms a Markov process. The transition probability function of this chain ($f_{I_{k+1}|I_k}(\cdot|\cdot)$) is time-invariant for a given SNR by the probabilistic model developed in Chapter 5. The state space S of this process is the range of mutual information function: $[0, 1]$. S is a continuous state space and thus hard to work with. A Markov chain is a Markov process with finite state space and easier to work with. It is shown in [35] that continuous state space Markov processes can be discretized into Markov chains. We will simply use this result without much elaboration as it is common practice in engineering applications.

The state space S will be discretized and taken to be a finite set for the remainder of the chapter. The effect of discretization levels will be investigated later. The following form of the partitions of the interval $[0, 1]$ will be considered,

$$S = \bigcup_{i=1}^{n} q_i, \tag{6.2}$$

where q_i's are continuous intervals and have the property that

$$x \in q_i, y \in q_{i+1} \Rightarrow x < y, \tag{6.3}$$

for all x, y and i. The set q_i will be referred to as the quantization level q_i from this point on. We have the following approximation to the transition probability from level q_j to q_i

$$p_{I_{k+1}|I_k}(q_i|q_j) = \begin{cases} \int_{-\infty}^{\sup_{x\in q_i} x} f_{I_{k+1}|I_k}(x|\bar{q_j})dx, & i = 1 \\ \int_{\inf_{x\in q_i} x}^{\sup_{x\in q_i} x} f_{I_{k+1}|I_k}(x|\bar{q_j})dx, & 1 < i < n \\ \int_{\inf_{x\in q_i} x}^{\infty} f_{I_{k+1}|I_k}(x|\bar{q_j})dx, & i = n \end{cases} \tag{6.4}$$

for any k, q_i and q_j where $\bar{q_j} = \frac{\inf_{x\in q_j} x + \sup_{x\in q_j} x}{2}$. If the time-invariant nature of the chain is considered, then we can simplify notation as

$$p_{ji} = p_{I_{k+1}|I_k}(q_i|q_j) = p(q_i|q_j). \tag{6.5}$$

6.4.1 Decoding Failure at a Given Iteration

We will evaluate the likelihood of first decoding failure at the i^{th} iteration $P_{df}(i)$. By the "$I_{i+2} > I_i$" rule, a decoding failure occurs at the i^{th} iteration if $I_{i+2} < I_i$. Define an indicator function that determines whether any decoding failure occurred up to iteration i by

$$N_{df}(i) = \begin{cases} 1, & \text{if } \exists j : I_j < I_{j-2}, j \leq i \\ 0, & \text{otherwise .} \end{cases} \tag{6.6}$$

We will slightly change the rule so that it will be assumed convergence to correct decoding occurs when $I_k = q_n$, i.e., I_k is large, and $N_{df}(k-1) = 0$ for some k. By this change, q_n becomes an absorbing state of the Markov chain, i.e., $p_{nn} = 1$, which means that the chain stays indefinitely in state q_n whenever the process enters it. By

the above definitions and explanations,

$$
\begin{aligned}
P_{df}(i) &= P(I_i < I_{i-2}, N_{df}(i-1) = 0) && (6.7)\\
&= \sum_{q^{i-2}=q_1}^{q_n} P(I_i < I_{i-2}, I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) && (6.8)\\
&= \sum_{q^{i-2}=q_1}^{q_n} P(I_i < I_{i-2} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) \\
&\qquad P(I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) && (6.9)
\end{aligned}
$$

If both terms in the last summation can be found, then $P_{df}(i)$ can be determined. First, consider the decoding failure term.

$$
\begin{aligned}
P(I_i &< I_{i-2} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) \\
&= \sum_{q^{i-1}, q^i = q_1}^{q_n} P(I_i < q^{i-2}, I_i = q^i, I_{i-1} = q^{i-1} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) && (6.10)\\
&= \sum_{q^i < q^{i-2}} \sum_{q^{i-1}=q_1}^{q_n} P(I_i = q^i, I_{i-1} = q^{i-1} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) && (6.11)\\
&= \sum_{q^i < q^{i-2}} \sum_{q^{i-1}=q_1}^{q_n} P(I_i = q^i | I_{i-1} = q^{i-1}, I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) \cdot && (6.12)\\
&\qquad P(I_{i-1} = q^{i-1} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) \\
&= \sum_{q^i < q^{i-2}} \sum_{q^{i-1}=q_1}^{q_n} p(q^i | q^{i-1}) p(q^{i-1} | q^{i-2}). && (6.13)
\end{aligned}
$$

Hence, $P(I_i < I_{i-2} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0)$ is a function of q^{i-2}, a dummy variable, and does not depend on the iteration number i. It will be denoted by $Q_{df}(x)$ and can be obtained from the IO relations for any value of x. Similarly, the

$P(I_{i-2} = q^{i-2}, N_{df}(i-1) = 0)$ term of (6.9) can be evaluated as follows.

$$
\begin{aligned}
P(\quad I_{i-1} \quad &= q^{i-1}, N_{df}(i) = 0) \\
&= \sum_{q^{i-2}=q_1}^{q_n} P(I_{i-2} = q^{i-2}, I_{i-1} = q^{i-1}, N_{df}(i) = 0) && (6.14) \\
&= \sum_{q^{i-2}=q_1}^{q_n} P(I_{i-2} = q^{i-2}, I_{i-1} = q^{i-1}, I_i \geq I_{i-2}, N_{df}(i-1) = 0) && (6.15) \\
&= \sum_{q^{i-2}=q_1}^{q_n} P(I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) \cdot \\
&\qquad P(I_{i-1} = q^{i-1}, I_i \geq I_{i-2} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) && (6.16) \\
&= \sum_{q^{i-2}=q_1}^{q_n} P(I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) \cdot \\
&\quad P(I_i \geq I_{i-2} | I_{i-2} = q^{i-2}, N_{df}(i-1) = 0) P(I_{i-1} = q^{i-1} | I_{i-2} = q^{i-2}) && (6.17) \\
&= \sum_{q^{i-2}=q_1}^{q_n} P(I_{i-2} = q^{i-2}, N_{df}(i-1) = 0)(1 - Q_{df}(q^{i-2}))p(q^{i-1}|q^{i-2}) && (6.18)
\end{aligned}
$$

The last set of equations implies that only $P(I_i = q^i, N_{df}(i+1) = 0)$ at each iteration has to be recorded in order to find the probability of decoding failure. In the next subsection, we will explain how to obtain these probabilities using matrices.

6.4.2 Numeric Evaluation of P_{df}

Denote $P(I_{i-1} = q_k, N_{df}(i) = 0)$ by $\pi_{k,i}$. Consider a matrix P_t such that

$$[P_t]_{i,j} = p_{ij} = P(I_k = q_j | I_{k-1} = q_i), \tag{6.19}$$

where p_{ij} is the transition probability function of the Markov chain as defined in (6.5). In consequence, (6.13) can be written as

$$Q_{df}(q_k) = \sum_{j=1}^{k-1} [P_t^2]_{k,j}, \tag{6.20}$$

for $k \geq 2$ with $Q_{df}(q_1) = 0$.

Using the description of $\pi_{k,i}$ above, (6.18) becomes

$$\pi_{k,i} = \sum_{j=1}^{n} \pi_{j,i-1}(1 - Q_{df}(q_j))[P_t]_{j,k}. \tag{6.21}$$

For easier representation, the likelihood of decoding failure up until the i^{th} iteration will be denoted by $\pi_{n+1,i}$ with an abuse of notation. Then,

$$\pi_{n+1,i} = \pi_{n+1,i-1} + \sum_{j=1}^{n} \pi_{j,i-1} Q_{df}(q_j). \tag{6.22}$$

Let's define a matrix T such that

$$[T]_{k,j} = \begin{cases} (1 - Q_{df}(q_j))[P_t]_{j,k}, & 1 \leq j \leq n, 1 \leq k \leq n \\ Q_{df}(q_j), & 1 \leq j \leq n, k = n+1 \\ 1, & j = k = n+1. \end{cases} \tag{6.23}$$

By this definition,

$$\pi_i = \begin{bmatrix} \pi_{1,i} \\ \pi_{2,i} \\ \vdots \\ \pi_{n,i} \\ \pi_{n+1,i} \end{bmatrix} = T\pi_{i-1}. \tag{6.24}$$

In the case of iterative decoding of equally likely bits, the mutual information in the beginning of decoding is 0. As a result,

$$\pi_i = T^i \pi_0, \pi_0 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{6.25}$$

The last equation can be considered as the transition matrix of yet another finite state Markov chain. Since a unique stationary distribution of π_i exists in the case of

finite state space [36], $\pi_s = \lim_{i\to\infty} T^i \pi_0$ can be found by using a sufficiently large i. In that case, the probability of decoding failure can be obtained from π_s by

$$P_{df} = [\pi_s]_{1,n+1}. \tag{6.26}$$

6.5 Numerical Results

For all the results that will be presented in this section, the properties of extrinsic information sequences are obtained at various SNR values with various values of I_{in} as discussed in Chapter 5. Based on the properties of the extrinsic information, the IO relation for each SNR and I_{in} value was found. In most cases, the I_{in} values in steps of 0.1 have been used and the IO relation have been obtained by interpolation. Since the IO relation is smooth, this interpolation does not pose any threat and it effectively reduces the complexity of the method. Once the IO relation for a particular component decoder is obtained, the transition probability matrix to be used in last section's methodology can be found.

6.5.1 Quantization

We will first explain the quantization method employed and then present results on the effect of quantization. For any quantization step q_{step} we will have the following quantization levels:

$$\begin{aligned}
q_1 &= [0, \frac{q_{step}}{2}) \\
q_2 &= [\frac{q_{step}}{2}, \frac{3}{2}q_{step}) \\
q_3 &= [\frac{3}{2}q_{step}, \frac{5}{2}q_{step}) \\
&\dots \\
q_{n-1} &= [I_{max} - \frac{3}{2}q_{step}, I_{max} - \frac{q_{step}}{2}) \\
q_n &= [I_{max} - \frac{q_{step}}{2}, 1].
\end{aligned}$$

We will set I_{max} value to 0.95 in this study. This value of I_{max} was determined based on the simulation results for determining the correct decoding rule. It was observed that when a packet reaches $I(E) \approx 0.95$ it is almost always correctly decoded. It follows from the description of quantization that the smaller the number q_{step} is, the finer the quantization gets.

The effect of quantization is shown in Fig. 6.8. A symmetric turbo code with component code generator matrix $[1, \frac{1+D^2}{1+D+D^2}]$ is used. The three plots correspond to the P_{df} estimates for three different block lengths $K = 256, 1024$, and 4096. As seen in all the plots, finer quantization increases the P_{df} estimate since more states will be deemed as failed in that case. However, the estimate differences between different values of q_{step} are not very significant. The differences are also decreasing so that there is the sign of a convergent behavior. We will choose $q_{step} = 0.005$ to be used for the rest of the chapter. This value of q_{step} seems to be sufficient since the error rates of interest for the codes we investigate is up to packet error rate $PER = 10^{-5}$.

6.5.2 The Symmetric Case

When the component decoders are the same, the method explained as in the last section is directly applied. In all the simulations in this section, the turbo codes were run for 100 iterations (each decoder is run 50 times). At least 200 packet errors have been collected in all cases.

In Fig. 6.9, the packet error rate performance of a symmetric turbo code with component code generator matrix $[1, \frac{1+D+D^3+D^4}{1+D^3+D^4}]$. In each packet there are 1024 data bits ($K = 1024$). Four different lines are shown. The solid line is the PER obtained from simulation. The dashed line with '*' is the P_{df} estimate obtained from the EXIT chart method. The dashed line with '+' is the error floor estimate studied in Chapter 4. Since the sum of P_{df} and error floor estimates is the total PER estimate, that sum is plotted with the bold dashed line in the figure. A symmetric turbo code with component code generator matrix $[1, \frac{1+D^2}{1+D+D^3}]$ is employed in Fig. 6.10. In order to test the effectiveness of the P_{df} estimation method with respect to block length, the

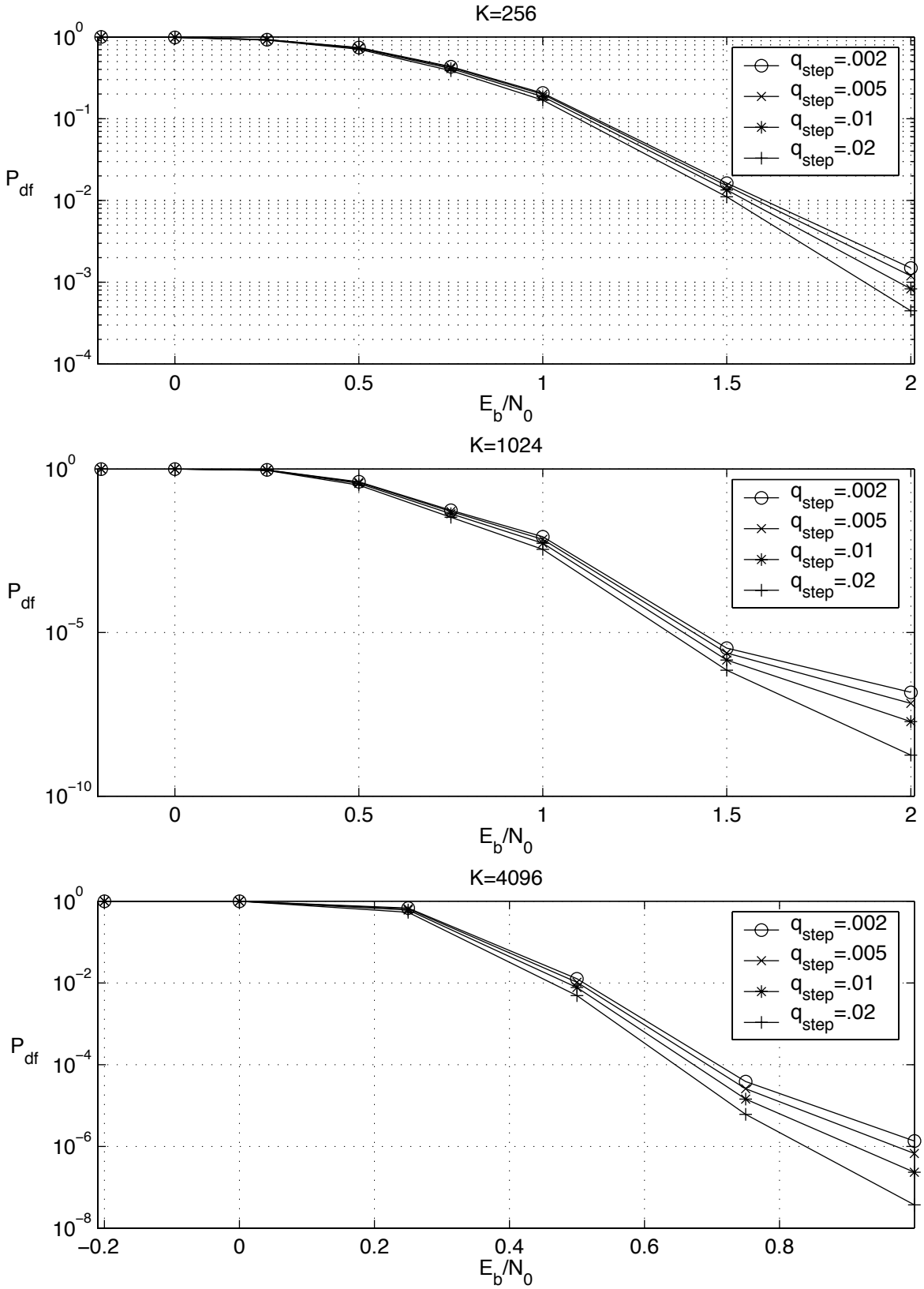


Figure 6.8: The effect of quantization on P_{df} estimation.

estimates and simulation results are displayed for four different values of block length K in the figure.

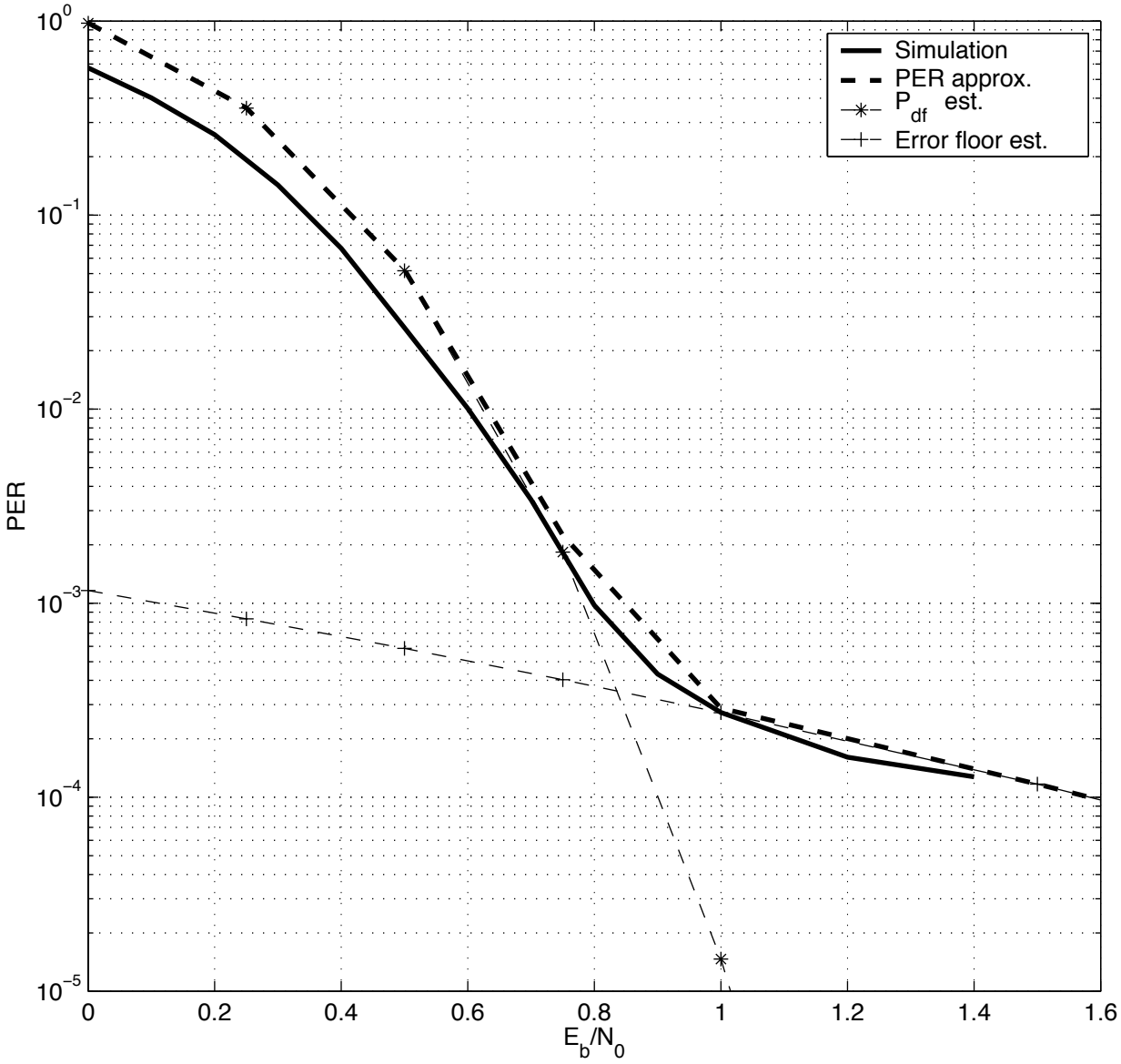


Figure 6.9: Packet error rate estimates and simulation results of a turbo code ($K = 1024$).

As observed in these figures, the P_{df} estimate behaves as an upper bound to the actual P_{df}. In the model we defined, it was said that the rule "$I_{i+2} > I_i$" does not perfectly predict the correct decoding of a packet. Thus, the developed model used with this rule offers a worst-case scenario so that its estimate behaves like an upper bound. The PER estimate, which is the sum of P_{df} and error floor estimate, is within

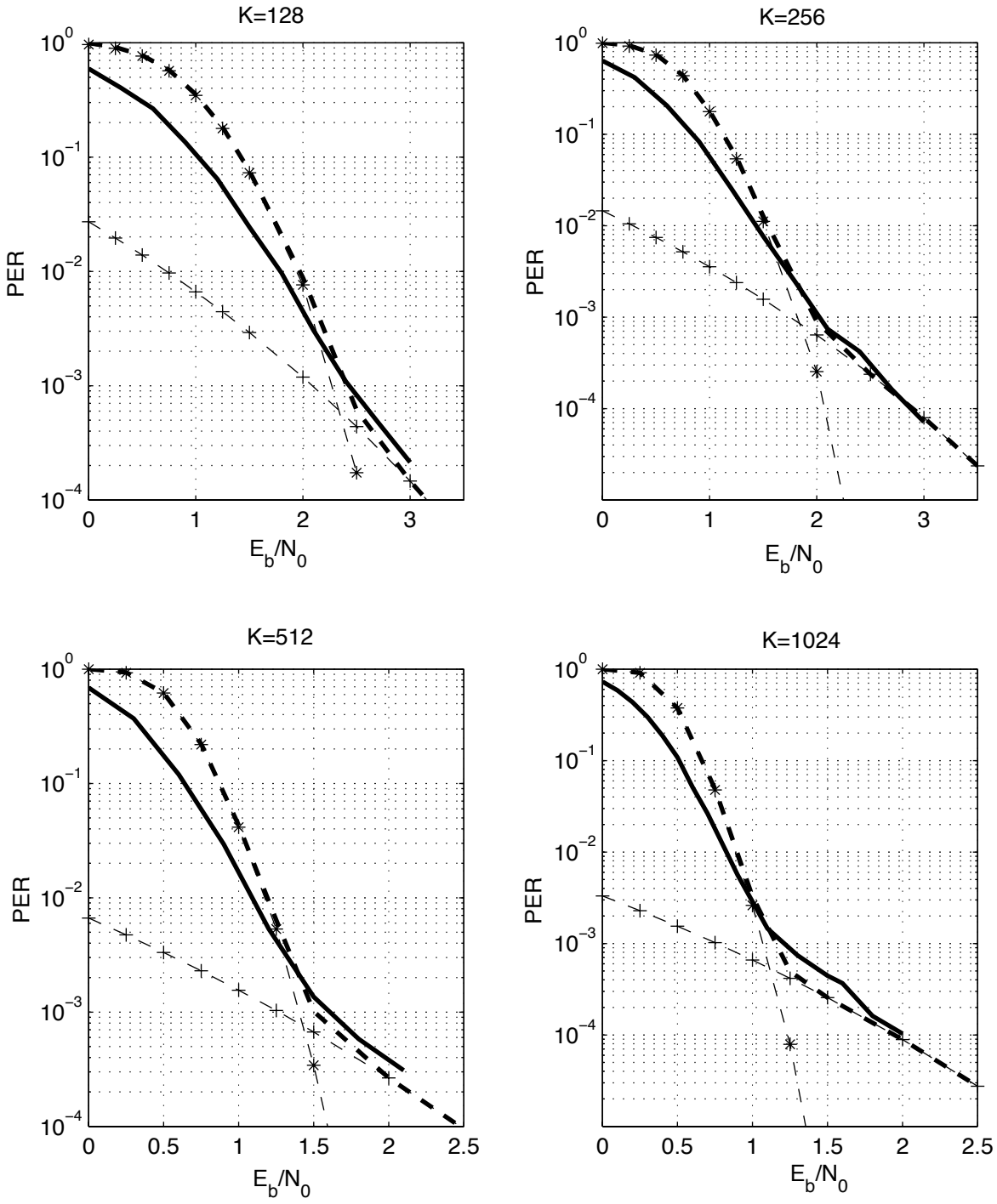


Figure 6.10: Packet error rate estimates and simulation results of a turbo code at $K = 128, 256, 512$, and 1024('-':simulation, '+':error floor estimate, '*':P_{df} estimate, '- -':combined PER estimate).

a small neighborhood $(0.1 - 0.3\text{dB})$ of SNR values of operation in all of the cases.

6.5.3 The Asymmetric Case

In Section 6.4, the methodology for obtaining P_{df} estimate of an iterative decoder is explained for the case of two identical component decoders inside an iterative decoder. All the arguments will hold when the component decoders are not the same. The only difference exists in the order that the transition probability functions are used. The derivations for the asymmetric case won't be pursued in order to avoid redundancy.

In Fig. 6.11, the simulation results and error rate estimates are displayed for an asymmetric turbo code. This turbo code consists of two codes with generator matrices $[1, \frac{1+D^2}{1+D+D^2}]$ and $[1, \frac{1+D+D^3+D^4}{1+D^3+D^4}]$. The block length of the code is $K = 256$. Both error floor and P_{df} estimates are shown. The combined PER estimate is also seen in the figure and within 0.1dB of the simulation results for PER$< 10^{-1}$.

As a last example, we will consider the serial concatenation of two convolutional codes. Although we haven't discussed explicitly, it is obvious from the mechanics of the EXIT chart that the EXIT chart method can be applied to any system that has component decoders which accepts soft information (as in the form of likelihoods) and produces soft information. The serial concatenation algorithm and the application of EXIT chart method have been discussed in [16]. We will directly apply our method following the studies for asymptotically long serially concatenated codes. The serially concatenated code of Fig. 6.12 is constructed by using the same memory-2 code with generator matrix $[1, \frac{1+D^2}{1+D+D^2}]$. 128 data bits are fed into the first encoder and 256 coded bits are generated. These 256 bits are fed into the second decoder to generate 512 bits in total. The overall rate of the code is 1/4. No error floor is observed until $PER = 10^{-4}$ for this code. The P_{df} estimate is about 0.15dB from the simulation curve for PER less than 10^{-1}.

In both cases of asymmetry, the P_{df} estimate predicts the simulation results closely. The estimates are within $0.05 - 0.3$dB of the simulation results depending on the

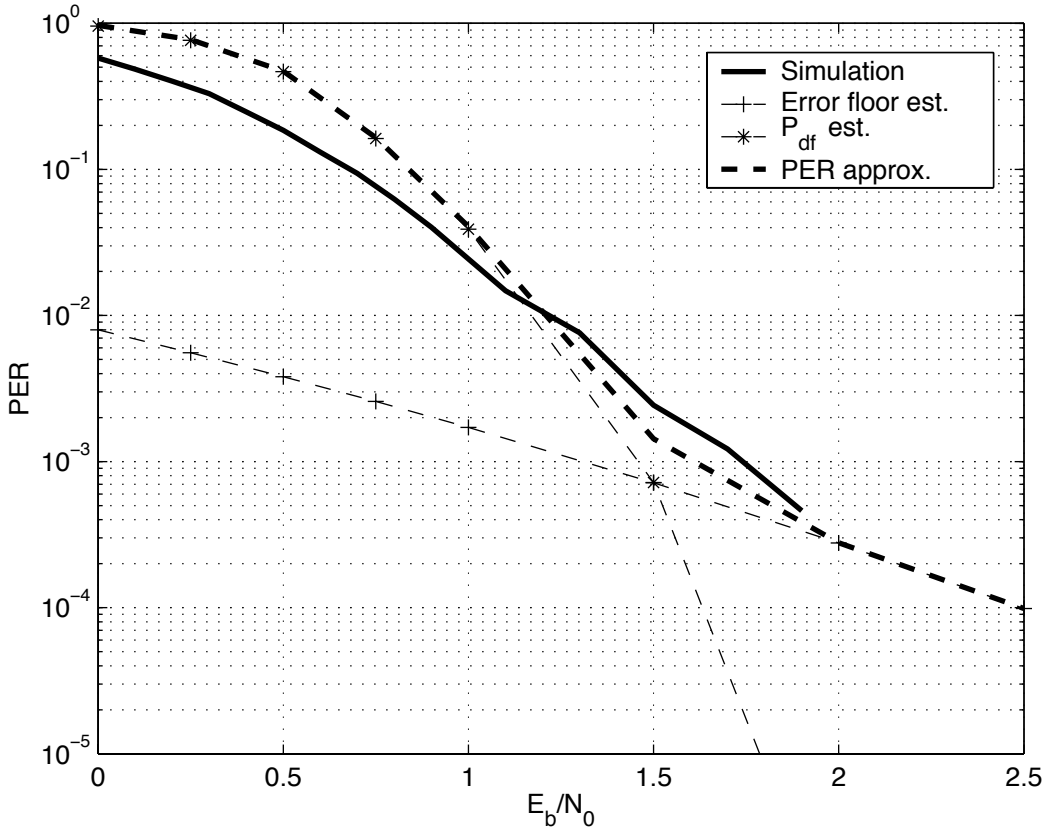


Figure 6.11: Packet error rate estimates and simulation results of an asymmetric turbo code ($K = 256$).

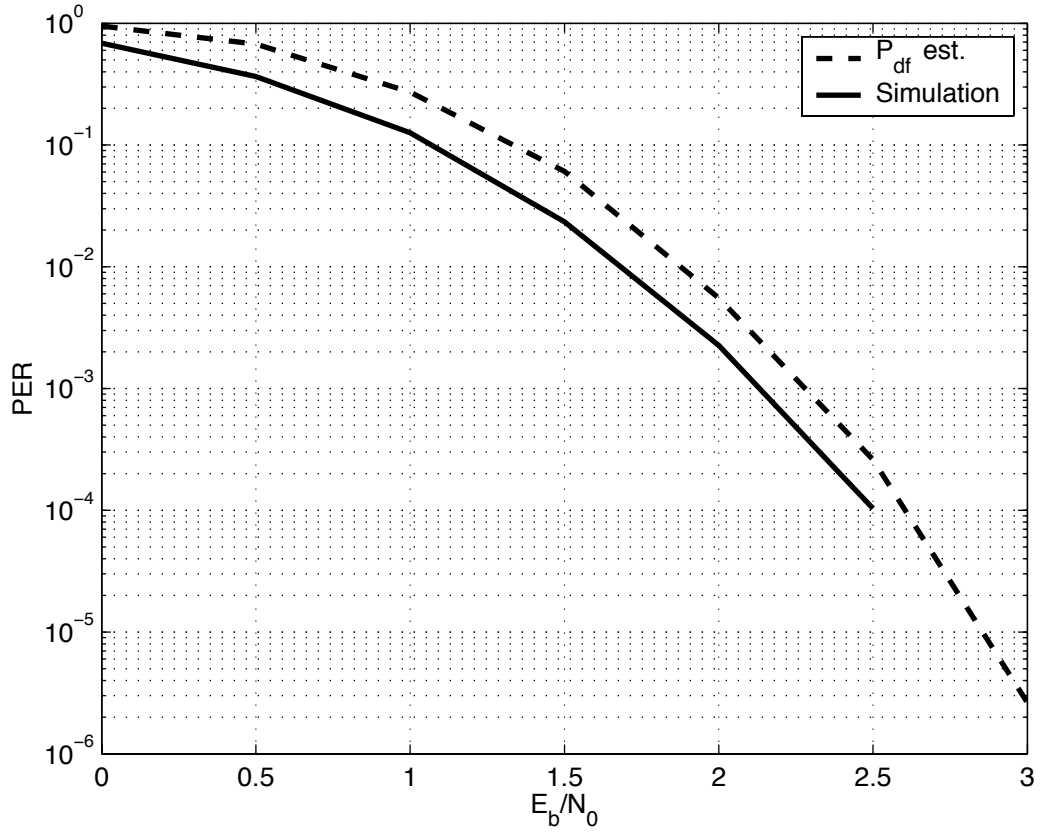


Figure 6.12: Packet error rate estimates and simulation results of a serially concatenated code ($K = 128$).

operation SNR of the code. There are methods to predict the error floor of serially concatenated codes [25]. However, use of almost any random interleaver in a serially concatenated code results in a very low error floor which makes the error floor and its estimate irrelevant for most practical situations. In this respect, the result with the serially concatenated code is especially important.

6.6 Conclusions

The EXIT chart method has been used to predict the behavior of iterative decoders at asymptotically large block lengths. In this chapter we proposed a method to extend the use of the EXIT chart method to finite block lengths. As supported by numerical results, this method can predict the performance of iterative decoding quite accurately in the low/medium SNR region where decoding failures dominate the performance. The proposed methodology can be utilized to study any iterative system with components that exchange soft information.

CHAPTER 7

Conclusions

7.1 Contributions

We studied turbo codes of practical block lengths. The main contribution of this thesis is distinguishing two distinct modes of a turbo decoder's operation and analyzing them. Different types of convergence behavior were identified and classified. The classification illustrates the existence of bimodal behavior of a turbo decoder.

An approximation to the error floor performance of a turbo code was obtained by estimating the distance spectrum of the code. The estimation method is based on an approximation through which the number of codewords that have to be searched for small code weights can be decreased effectively and substantially. The proposed method accurately predicts the error rate of a turbo code at high values of SNR at which the error floor is observed.

The waterfall region analysis is done through the use of a probabilistic model for the iterative decoding process. Since the emphasis of this thesis is on the turbo codes of practical block lengths, the effect of finite block lengths on the component decoders was investigated. It was found out that the correlation in the information produced by a component decoder can be used to model the component decoder. This probabilistic model was then embedded into a Markov process to obtain error rate approximations to the performance in the waterfall region. This approximation combined with the approximation for the error floor region accurately predicts the

performance of a turbo code over both regions.

7.2 Future Work

The study on the information produced by a component decoder does not assume a specific form of decoder as long as a few assumptions hold. Therefore the methodology developed for the MAP decoder can be easily extended to any other decoding module that can be considered as an independent soft-input soft-output module. The immediate applications will be to the study of max-log-MAP decoder, soft output Viterbi algorithm and other decoding algorithms for convolutional codes. The method will be extended to modulation schemes such as M-PSK, continuous phase modulation etc. We will also study the applicability of this method in fading channels. The proposed methodology is not only restricted to the study of decoders. For example, a demodulator which performs per-survivor processing to account for the fading process in a fading environment can also be investigated with our methodology as long as it stands as an independent soft-input soft-output block. Thus, we will apply our technique to study turbo equalization schemes. We will also apply the EXIT chart methodology to obtain bit error rate approximations.

The distance spectrum estimation method will be applied to different flavors of turbo codes such as punctured turbo codes, asymmetric turbo codes, turbo codes with more than two component codes etc. with different modulation schemes. The extension of the estimation method will be sought for the case of parallel concatenated block codes, a.k.a. product codes. The estimation method also motivates more research on how to obtain turbo codes with low error floors. This issue will be elaborated in subsequent studies.

These two methods offer practical tools to compare codes of different rate, structure and complexity. Based on these comparisons design optimizations will be carried out to obtain the best systems that can meet given criteria in a given scenario in order to accommodate quality of service requirements.

APPENDICES

APPENDIX A

Symbolwise MAP Decoding of Convolutional Codes

A data symbol u_k is the collection of all the s data bits $(u_1^k, u_2^k, \ldots, u_s^k)$. We will derive the algorithm based on data symbols and bit probabilities can easily be obtained by summing over the data symbols. As mentioned before, the output and the next state of a convolutional code is determined by the current state and the data. Then,

$$P(\, u_k = i \mid y) = \sum_{(m',m)\,:\,u_k = i} P(\, S_{k-1} = m', S_k = m \mid y), \tag{A.1}$$

where S_k is the encoder's state at discrete time index k, y is all the channel observation, i is from the alphabet of data symbols and summation is over all the transitions between states m' and m where $u_k = i$. Let's define a function $M_k(m', m)$ such that

$$M_k(m', m) = P(\, S_{k-1} = m', S_k = m \mid y). \tag{A.2}$$

By this definition, we can rewrite (A.1) as

$$P(\, u_k = i \mid y) = \sum_{(m',m)\,:\,u_k = i} M_k(m', m). \tag{A.3}$$

Now consider all the paths that has the transition $S_{k-1} = m' \rightarrow S_k = m$. $M_k(m', m)$ can be written as

$$M_k(m', m) = \sum_{u \epsilon D_k(m', m)} p(\, u \mid y), \tag{A.4}$$

where u is a hypothesized input sequence to the encoder, $D_k(m', m)$ is the set of all input sequences which traverse the code's trellis through state m' at time $k-1$ and state m at time k.

Using Bayes' theorem, we can write the following expression:

$$p(\, u \mid y) = \frac{f(\, y \mid u) p(\, u)}{f(\, y)}. \tag{A.5}$$

In (A.5), the denominator is a constant for all k, m' and m so we can disregard it. Then,

$$M_k(m', m) \doteq \sum_{u \epsilon D_k(m', m)} f(\, y \mid u) p(\, u)^3, \tag{A.6}$$

It is assumed that the symbols in the data sequence are independent. In that case,

$$p(\, u) = \prod_{l=1}^{N} p(u_l). \tag{A.7}$$

From the structure of the encoder, it is known that there is 1-1 correspondence between the domain of u and the domain of the encoder output sequence c. Using this fact,

$$f(\, y \mid u) = f(\, y \mid c_u), \tag{A.8}$$

where c_u is the encoded sequence corresponding to u.

By defining a state variable for the past data and the hypothesized output sequence c_u, we can express $f(\, y \mid c_u)$ as a product of N terms. Let's define

$$\sigma_l(c_u) = [y_1, \ldots, y_l; c_u],$$

[3] $\doteq$ stands for equality with a constant.

where y_k denotes the channel signal for the k^{th} symbol interval and $\sigma_l(c_u)$ is the state for the l^{th} symbol in codeword c_u ($\sigma_0(c_u)$ holds the starting state of the inner encoder.) If we apply the chain rule, we obtain

$$f(\, y \mid c_u) = \prod_{l=1}^{N} f(y_l \mid \sigma_{l-1}(c_u)). \tag{A.9}$$

By substituting (A.7),(A.8) and (A.9) into (A.6), we arrive at the following expression for $M_k(m', m)$:

$$M_k(m', m) \doteq \sum_{u \epsilon D_k(m', m)} \prod_{l=1}^{N} p(u_l) f(y_l \mid \sigma_{l-1}(c_u)). \tag{A.10}$$

In case memoryless channels are considered and the convolutional code's encoder is a FSM,

$$\begin{aligned} f(y_l \mid \sigma_{l-1}(c_u)) &= f(y_l \mid y_1, \ldots, y_{l-1}; c_u) \\ &= f(y_l \mid c_u) \\ &= f(y_l \mid S_{l-1}(u), S_l(u)), \end{aligned}$$

where $S_l(u)$ is the state of the encoder at time l when the input to the encoder is u. Then, (A.10) can be simplified to

$$M_k(m', m) \doteq \sum_{u \epsilon D_k(m', m)} \prod_{l=1}^{N} \gamma_l(S_{l-1}(u), S_l(u)), \tag{A.11}$$

where

$$\gamma_l(m', m) = p(u_l) f(y_l \mid m', m).$$

Computation of (A.11) is a basic problem which can be solved by the BCJR algorithm [19]. This algorithm has been studied quite extensively in the literature in the discussion of turbo codes and their derivatives. BCJR algorithm proposes a recursive solution to the problem of summing the metrics of all paths passing through

a fixed transition in a given trellis. Two functions $\alpha_k(m)$ and $\beta_k(m)$ are defined, The former is for the forward direction and the latter is for the backward. $\alpha_k(m)$ can be regarded as the sum of all path metrics ending up at state m at time k. In symmetry, $\beta_k(m)$ can be thought of as the sum of all path metrics starting at state m at time k. The following recursions are utilized to find $\alpha_k(m)$ and $\beta_k(m)$:

$$\begin{aligned} \alpha_k(m) &= \sum_{m'} \alpha_{k-1}(m')\gamma_k(m', m) \\ \beta_k(m) &= \sum_{m'} \beta_{k+1}(m')\gamma_{k+1}(m, m'). \end{aligned}$$

Initial values depend on the additional information that is available to the encoder. The encoder is preferably initialized to zero state. The last state of the encoder is not known unless termination sequences are deployed. In the case no termination strategy is used, the last state for the encoder can be any of the states with equal likelihood. These recursions are run once for the whole sequence and $M_k(m', m)$ can be written in terms of these functions the following way:

$$M_k(m', m) \doteq \alpha_{k-1}(m')\gamma_k(m', m)\beta_k(m). \tag{A.12}$$

In this case, (A.4) can be easily obtained by

$$P(u_k = i \mid y) \doteq \sum_{(m', m):u_k=i} \alpha_{k-1}(m')\gamma_k(m', m)\beta_k(m).$$

APPENDIX B

Some Properties of Random Sequences

Definition 5 *A random sequence* $\{X_i\}$ *is an* m*-dependent sequence if* $\{X_{a-r}, X_{a-r+1}, \ldots, X_a\}$ *and* $\{X_b, X_{b+1}, \ldots, X_{b+s}\}$ *are independent sets of variables for any* $r, s > 0$ *if* $b - a > m$ *[37].*

Theorem 1 *(Hoeffding-Robbins). If* $\{X_i\}$ *is an* m*-dependent sequence satistfying*

1. $E[X_i] = 0$,
2. $\lim_{n\to\infty} E[S_a^2] = A^2$ *for all* a $(A^2 > 0)$,
3. $E|X_i|^{2+\delta} \leq M$ *(for some* $\delta > 0$ *and* $M < \infty$*),*

where $S_a = \frac{1}{\sqrt{n}} \sum_{i=a+1}^{a+n} X_i$, *then* $S = S_0$ *is asymptotically normal such that it is normally distibuted with mean* 0 *and variance* A^2 *(Theorem* 1.1 *in [37]).*

Corollary 1 *Theorem 1 also holds with finite mean* $E[X_i] = c < \infty$ *for all* i *and* $S_a = \frac{1}{\sqrt{n}} \sum_{i=a+1}^{a+n} (X_i - E[X_i])$. *In that case,* $S = S_0$ *is asymptotically normal such that it is normally distributed with mean* c *and variance* A^2.

Lemma 2 *When* X_i*'s have common mean, the covariances satisfy* $Cov(X_a, X_{a+k}) = r_k$, *a function depending only on* k, *and* $\sum_{k=1}^{\infty} r_k$ *converges, then* S_a *satisfies the second property in Theorem 1 with* $A^2 = r_0 + 2\sum_{k=1}^{\infty} r_k$ *(Lemma* 5.1 *in [37]).*

Property 1 *A WSS Gaussian sequence with finite mean and autocovariance function*

$$K_X(s) = \begin{cases} \sigma^2 & , s = 0 \\ 0 & , |s| > m \end{cases} \tag{B.1}$$

is an m-dependent sequence.

Proof: Given $K_X(s)$, X_i and X_{i+s} are uncorrelated for $|s| > m$. Since X_i and X_{i+s} are jointly Gaussian and uncorrelated, they are independent. Take any block of r random variables before X_i and any block of k random variables after X_{i+s}. $\{X_{i-r}, X_{i-r+1}, \ldots, X_i\}$ and $\{X_{i+s}, X_{i+s+1}, \ldots, X_{i+s+k}\}$ are independent sets for any i. Then $\{X_i\}$ is an m-dependent sequence. ■

Lemma 3 *The sample mean of a WSS Gaussian sequence with finite mean c and autocovariance function given as in (B.1), $S = \frac{1}{n}\sum_{i=1}^{n} X_i$, is asymptotically normal with mean 0 and variance $\frac{1}{n}\sum_{s=-\infty}^{\infty} K_X(s)$.*

Proof:

- $E[X_i] = c$ for any $i > 0$, by the WSS property. Without loss of generality, we will consider X_i as a zero-mean sequence for easier reading. Nonzero mean will only add some constants to the derived expressions.

- For the second property of Theorem 1,

$$\begin{aligned} E[S_a^2] &= E[\frac{1}{n}\sum_{i=a+1}^{a+n}\sum_{j=a+1}^{a+n} X_i X_j] && \text{(B.2)} \\ &= \frac{1}{n}\sum_{i=a+1}^{a+n}\sum_{j=a+1}^{a+n} E[X_i X_j] && \text{(B.3)} \\ &= \frac{1}{n}\sum_{i=a+1}^{a+n}\sum_{j=a+1}^{a+n} K_X(i-j) && \text{(B.4)} \\ &\approx \frac{1}{n}\sum_{i=a+1}^{a+n}\sum_{j=i-s}^{i+s} K_X(i-j) && \text{(B.5)} \\ &= \frac{1}{n}\sum_{i=a+1}^{a+n}\sum_{j=-s}^{s} K_X(j) && \text{(B.6)} \\ &= \sum_{j=-s}^{s} K_X(j). && \text{(B.7)} \end{aligned}$$

  The last summation is convergent. Then the second assumption also holds.

- It is well known that any moment of a Gaussian random variable can be obtained from its mean and variance. The assumption holds directly from this fact for the even powers. For the odd powers, ($f_{X_i}(\cdot)$ being the probability density function of a Gaussian random variable with mean zero and variance σ^2)

$$\begin{aligned} E|X_i|^n &= 2\int_0^\infty x^n f_{X_i}(x)dx && \text{(B.8)} \\ &= \frac{n!}{\sqrt{2\pi}} 2^n \sigma^{2n+1}. && \text{(B.9)} \end{aligned}$$

  So, $E|X_i|^n < M$ for any $n > 1$ and some $M < \infty$.

Since $\{X_i\}$ is also an m-dependent sequence, the statement of the lemma holds. ■

Lemma 4 *Let $\{X_i\}$ be a stationary sequence. Define $\{Y_i\}$ sequence such that $Y_i = g(X_i)$ for some strictly monotonically increasing function g defined in $\Re$. Then, $\{Y_i\}$ is also a stationary sequence.*

Proof:

$$
\begin{aligned}
&P(Y_{i+k} \leq y_0, Y_{i+k+1} \leq y_1, \ldots, Y_{i+k+n} \leq y_n) \\
&\quad = P(X_{i+k} \leq g^{-1}(y_0), X_{i+k+1} \leq g^{-1}(y_1), \ldots, X_{i+k+n} \leq g^{-1}(y_n)) \\
&\quad = P(X_i \leq g^{-1}(y_0), X_{i+1} \leq g^{-1}(y_1), \ldots, X_{i+n} \leq g^{-1}(y_n)) \\
&\quad = P(Y_i \leq y_0, Y_{i+1} \leq y_1, \ldots, Y_{i+n} \leq y_n),
\end{aligned}
$$

for any i, k and n. Hence, $\{Y_i\}$ is also stationary. ■

Lemma 5 *Let $\{X_i\}$ be an m-dependent sequence. Define a sequence $\{Y_i\}$ such that $Y_i = g(X_i)$ for some strictly monotonically increasing function g defined in $\Re$. Then, $\{Y_i\}$ is also m-dependent .*

Proof: $\{X_{a-r}, X_{a-r+1}, \ldots, X_a\}$ and $\{X_b, X_{b+1}, \ldots, X_{b+s}\}$ are independent sets of variables for any $r, s > 0$ if $b - a > m$. Functions of independent random variables are also independent random variables. In that case, $\{g(X_{a-r}), g(X_{a-r+1}), \ldots, g(X_a)\}$ and $\{g(X_b), g(X_{b+1}), \ldots, g(X_{b+s})\}$ are independent sets. Then, $\{Y_{a-r}, Y_{a-r+1}, \ldots, Y_a\}$ and $\{Y_b, Y_{b+1}, \ldots, Y_{b+s}\}$ are independent sets for any $r, s > 0$ if $b - a > m$. ■

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] R. G. Gallager, *Low-Density Parity-Check Codes*, Ph.D. thesis, MIT.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo Codes," *Proc. IEEE International Conference on Communications*, pp. 1064–1070, May 1993.

[3] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *TDA Progress Report 42-122, Jet Propulsion Lab.*, August 1995.

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

[5] T. Richardson and R. Urbanke, "The capacity of low density parity check codes under message passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 619–637, February 2001.

[6] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, October 2001.

[7] H. El Gamal and A. R. Hammons, "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 671–686, February 2001.

[8] T. J. Richardson, "The geometry of turbo-decoding dynamics," *IEEE Transactions on Information Theory*, vol. 46, pp. 9–23, January 2000.

[9] D. Agrawal and A. Vardy, "The turbo decoding algorithm and its phase trajectories," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 699–722, February 2001.

[10] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, "On the frame-error rate of concatenated turbo codes," *IEEE Transactions on Communications*, vol. 49, no. 4, pp. 602–608, April 2001.

[11] A. C. Reid, T. A. Gulliver, and D. P. Taylor, "Convergence and errors in turbo-decoding," *IEEE Transactions on Communications*, vol. 49, no. 12, pp. 2045–2051, December 2001.

[12] J. G. Proakis, *Digital Communications*, McGraw-Hill, 1995.

[13] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, February 2001.

[14] A. Burr, *Modulation and Coding for Wireless Communications*, Prentice Hall, 2001.

[15] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 409–428, March 1996.

[16] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on density evolution," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 891–907, May 2001.

[17] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, "A note on asymmetric turbo codes," *IEEE Commun. Lett.*, vol. 3, pp. 69–71, March 1999.

[18] P. C. Massey and Jr. D. J. Costello, "New low-complexity turbo-like codes," *Proc. of Information Theory Workshop*, pp. 70–72, Sept. 2001.

[19] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, March 1974.

[20] L. Duan and B. Rimoldi, "The iterative turbo decoding algorithm has fixed points," *IEEE Transactions on Information Theory*, pp. 2993–2995, November 2001.

[21] K. R. Narayanan and G. L. Stüber, "A novel ARQ technique using the turbo principle," *IEEE Commun. Lett.*, vol. 1, pp. 49–51, February 1997.

[22] R. Shao, S. Lin, and M. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1117–1120, August 1999.

[23] W. Blackert, E. Hall, and S.Wilson, "An upper bound on turbo code free distance," *Proc. IEEE International Conference on Communications*, pp. 957–961, 1996.

[24] M. Breiling and J. B. Huber, "Upper bound on the minimum distance of turbo codes," *IEEE Transactions on Communications*, vol. 49, no. 5, pp. 808–815, May 2001.

[25] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers: Algorithms and applications," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 800–812, May 2001.

[26] F. Daneshgaran and M. Mondin, "An efficient algorithm for obtaining the distance spectrum of turbo codes," *International Symposium on Turbo Codes*, pp. 251–254, 1997.

[27] P. Ödling, H. B. Eriksson, T. Koski, and P. O. Börjesson, "When are the MLSD respectively the matched filter receiver optimal with respect to the BER?," *Proc. IEEE International Symposium on Information Theory*, p. 331, 1995.

[28] C. Lee, "Convolutional coding fundamentals and applications," 1997.

[29] S. Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, February 2001.

[30] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. thesis, Univ. Linköping, 1996.

[31] A. M. Mood, F. A. Graybill, and D. C. Boes, "Introduction to the theory of statistics," 1963.

[32] P. Le Bars, C. Le Dantec, and P. Piret, "Bolt interleaver for turbo codes," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 391–400, February 2003.

[33] H. Stark and J. W. Woods, *Probability, Random Processes, and Estimation Theory for Engineers*, Prentice Hall, 1994.

[34] J. W. Lee and R. E. Blahut, "Analysis of the extrinsic values in the finite length turbo decoding," *Conference on Information Sciences and Systems, Princeton University*, March 2002.

[35] C. Guihenneuc-Jouyaux and C. P. Robert, "Discretization of continuous Markov chains and Markov chain Monte Carlo convergence assessment," *Journal of American Statistical Association*, vol. 93, no. 443, pp. 1055–1067, September 1998.

[36] P. G. Hoel, S. C. Port, and C. J. Stone, *Introduction to Stochastics Processes*, Waveland Press, 1987.

[37] R. J. Serfling, "Contributions to central limit theory for dependent variables," *Annals of Mathematical Statistics*, vol. 39, no. 4, pp. 1158–1175, August 1968.

ABSTRACT

Performance of Turbo Codes: The Finite Length Case

by

Ali Özgür Yılmaz

Chair : Wayne E. Stark

Parallel concatenated convolutional codes, a.k.a. turbo codes, have attracted much attention since their appearance in 1993. Turbo codes are decoded by iterative (turbo) decoders in which a component decoder for each component code exists. Turbo decoders perform quite well with respect to bit and packet error rates. The code structure coupled with the iterative decoding scheme provides performance close to the fundamental communication limits predicted by Shannon with reasonable complexity. In this thesis we investigate the performance of finite length turbo codes.

Due to the *ad hoc* nature of turbo decoding, error rate performance analysis is difficult. In the first part of this thesis, we classify the different characteristics of turbo decoding based on empirical results and analyses available in the literature. This classification helps distinguish two modes of turbo decoding behavior: optimal mode and decoding failures.

The performance in the optimal mode is affected by the code structure of a turbo

code. The interleaver in turbo code construction makes it difficult to find the code structure for a particular interleaver. In order to overcome this difficulty, we concentrate on special error events of the component convolutional codes. We propose an efficient method to estimate the code structure. By this estimate, an approximation to the error rate performance can be obtained for the optimal mode.

Decoding failures occur due to nonoptimal operation of a turbo decoder. Therefore, analysis of iterative decoding is of interest with regard to decoding failures. Iterative decoding analysis methods are available for infinite length turbo codes. By investigating the properties of the component decoders, a probabilistic model is developed for finite lengths. Using this model, we find approximations to probability of decoding failure.

Combining the analysis methods for the two modes of a turbo decoder yields an accurate overall approximation to the error rate performance. Numerical results are presented to verify the accuracy of the approximation.