University of Alberta

Analysis and Decoding of LDPC Codes under Some Practical Considerations

by

Raman Yazdani ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2007

Your file Votre référence
ISBN: 978-0-494-33378-5
Our file Notre référence
ISBN: 978-0-494-33378-5

One thing I have learned in a long life: that all our science, measured against reality, is primitive and childlike and yet it is the most precious thing we have.

– Albert Einstein.

Abstract

This dissertation presents new methods for the analysis, design and decoding of low-density parity-check (LDPC) codes. First, we propose an efficient method for analyzing finite-length LDPC codes on symmetric channels. This method is based on the threshold behavior of LDPC codes and studying the atypical behavior of the channel when observed during a finite block length. Different channel parameters can be used to model the channel behavior and predict the performance. We investigate and compare the results obtained by considering different channel parameters for predicting the performance. Second, we consider iterative decoding on uncorrelated fading channels where the channel fading gain and/or the noise power is not known at the receiver. We propose a linear LLR approximation method which is optimum in the sense of maximum achievable transmission rate on the channel. This method is also applicable to other iteratively decoded codes.

To My Parents...

Acknowledgements

First, I would like to express my sincere and deeply grateful thanks to my supervisor Dr. Masoud Ardakani. I am certain that this work would not be possible without his valuable support, advice, kindness, and encouragement. He spent uncountable hours of discussion with me even about the smallest details of my research. I really cannot imagine a supervisor more caring and supportive than him. It is immeasurable how much I have benefited from my supervisor during my master's program. I have learnt a lot from him regarding conducting research, technical writing, writing papers, presentation skills, etc. I have also learnt much from his genial personality. I would also like to thank him for his tireless efforts in editing this dissertation and my work in general.

I would also like to thank my friend Ali Sanaei in our research group. We had lots of discussion about our research and his suggestions and solutions helped me a lot improving this work.

My sincere thanks goes to my friends here in Edmonton who made my life enjoyable. In particular, I would like to thank Ali Hendi, Navid Paydavosi, Pirooz Chubak, Mohammad Behnam, and Mahdi Hajiaghayi for their endless help and friendship. I would also like to deeply thank my friends back home who supported me and did not forget me.

Finally, I would like to express my eternal gratefulness to my parents for their everlasting support in all aspects of my life even though I left them to study thousands of miles away.

Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	**Description**	**First use**
LDPC	Low-density parity-check	1
EXIT	Extrinsic information transfer	3
OFDM	Orthogonal frequency division multiplexing	4
BICM	Bit-interleaved coded modulation	4
MIMO	Multiple-input multiple-output	4
RA	Repeat accumulate	4
LT	Luby transform	4
DVB	Digital video broadcasting	4
BEC	Binary erasure channel	5
LLR	Log-likelihood ratio	5
BSC	Binary symmteric channel	10
BPSK	Binary phase-shift keying	11
BIAWGN	Binary-input additive white Gaussian noise	10
pdf	Probability density function	11
BIMS	Binary-input memoryless symmetric	11
ML	Maximum likelihood	23
pmf	probability mass function	28
SNR	Signal-to-noise ratio	31
URF	Uncorrelated Rayleigh fading	45
SI	Side information	48
cdf	Cumulative density function	49

MCLA	Maximum-capacity linear approximation	73
BER	Bit error rate	74

List of Symbols

Symbol	Description	First use
$\mathrm{P}(E)$	Probability of the event E	8
ϵ	Erasure probability in BEC and crossover probability in BSC	10
z	The additive white Gaussian noise	11
σ_z	Standard deviation of the additive white Gaussian noise	11
r	Channel fading gain and the number of checks in an LDPC code	11
C	Capacity	11
$h(\cdot)$	The binary entropy function	11
k	Dimension of a linear block code	12
n	Block length of a linear block code	12
$\mathbf{G}$	Generator matrix of a linear block code	12
$\mathbf{H}$	Parity-check matrix of a linear block code	12
v	A variable node	15
c	A check node	15
E	Number of edges in the graph	16
d_{v}	Maximum degree of the variable nodes	16
d_{c}	Maximum degree of the check nodes	16
R	Code rate	17
$\lambda(x)$	Generating polynomial of the variable degree distribution	17
$\rho(x)$	Generating polynomial of the check degree distribution	17
$\mathcal{C}^n(\lambda(x),\rho(x))$	Ensemble of LDPC codes of length n and degree distributions $\lambda(x)$ and $\rho(x)$	17
$p_e^{(\ell)}$	Extrinsic messages error rate in ℓth iteration	26

p_{out}	Error probability of the outgoing extrinsic messages	31
p_{in}	Error probability of the incoming extrinsic messages	31
p_0	Error probability of the intrinsic messages	31
$\mathrm{P_B}$	Block error probability	36
$Q(\cdot)$	Q-function	36
ϵ^*	Decoding threshold	36
P_{obs}	Error probability observed from the codewords perspective	43
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2	43
C_{obs}	Channel capacity observed from the codewords perspective	44
L, l	Log-likelihood ratio	45
$\mathfrak{C}(\sigma)$	A binary-input memoryless symmetric channel parameterized by σ	48
$\mathrm{P_b}$	Bit error probability	53
$\Lambda(x)$	Node-perspective variable degree distribution generating polynomial	53
$\hat{l}$	Linear log-likelihood ratio	71
$\hat{\sigma}_z$	Estimate of σ_z	71
$\hat{r}$	Estimate of r	71
$\hat{C}$	A measure of capacity	72
σ_z^*	Decoding threshold	73

Chapter 1

Introduction

This work is concerned with design and analysis of a powerful class of error-correcting codes called low-density parity-check (LDPC) codes and iterative decoding in general. It has been shown that LDPC codes can perform near the Shannon capacity in many channels and yet this great performance is achieved with a practical decoding complexity. Therefore, these codes are extremely efficient from a practical point of view.

In this chapter, we briefly introduce the field of study and review the historical backgrounds of this research area. Also, we discuss the recent active topics and propose interesting problems to be tackled in this thesis.

1.1 Iterative Decoding and Codes Defined on Graphs

In a modern society, exchange of information in an efficient, reliable and secure manner is of fundamental importance. Any communication is, to some extent, affected by noise and interference. In order to detect or correct the errors that occur, using error-correcting codes, the information is coded to include some redundancy. Physical solutions for error-reduction (such as improving the communication channel) are far more expensive and less effective. Therefore, the quality of data communication systems totally depends on efficient error-correction coding.

In 1948, Claude E. Shannon published a remarkable paper on the limits of reliable data transmission over unreliable channels and he introduced the concept of information [1]. Shannon derived bounds on the maximum amount of information that can be transmitted over unreliable communication channels. He showed that there exists a quantity, called the capacity of the channel, such that reliable data transmission is only possible for data rates below this capacity. Shannon proved that there exist codes of rates arbitrarily close to capacity such that as their block length goes to infinity the probability of error of their detection goes to zero. Nevertheless, he did not propose a practical solution for finding such codes. In fact, for the proof of the channel capacity theorem, he used random codes. The problem with random codes is their impractical decoding complexity with grows exponentially with the block length.

In order to measure the power efficiency of a coding solution, its performance gap is measured from the Shannon limit. The Shannon limit is defined as the minimum transmission power required to have reliable transmission for a given data rate. This is an alternative way to measure how close the coding solution is to the capacity. Since the introduction of capacity and the Shannon limit, there have been many attempts to design coding methods which can achieve the capacity and can be encoded and decoded with practical complexity.

Until the discovery of Turbo codes [2], no proposed coding scheme with a practical encoding or decoding complexity could approach the Shannon limit with a small gap. The near Shannon limit performance and practical decoding complexity of Turbo codes was made possible by introducing *iterative message-passing* decoding algorithms. This class of low-complexity decoding algorithms are applicable to the codes defined on graphs. The performance achieved using these iterative message-passing algorithms is very close to the optimal decoding algorithm. Turbo codes drew a lot of attention and a great deal of research to the field of iterative decoding and graphical codes.

Iterative decoding allows the use of long codewords with reasonable decoding complexity. This is due to the fact that the complexity of iterative decoding algorithms grows linearly with the length of the code and thus complexity per information bit is independent of the code length. This property of iterative decoding and their near optimal performance allow us to design codes that can approach the Shannon limit. Therefore, iterative decoding and graphical codes have received much attention in the past decade due to their exemplary performance and there have been many advances in this area. However, there are still many open problems in this field.

1.2 A Brief Historical Review

The field of graphical codes started from the graphical representation of linear codes, called Tanner graphs [3]. After the discovery that the turbo decoder can be represented graphically [4], it was shown in [5] and [6] that the turbo decoding algorithm based on the graphical representation of the code is a special case of belief propagation on Bayesian networks [7].

The research on turbo codes and graphical codes attracted attention to a class of codes, called LDPC codes which were first proposed by Gallager in 1963 [8] and were long neglected. With the advances in iterative message-passing algorithms and graphical codes, it was shown that LDPC codes can have extremely good performance [9,10].

LDPC codes have sparse parity-check matrices. Therefore, their graphical representation is simple and they can be decoded by iterative message-passing decoders with low complexity. These properties of LDPC codes drew a lot of attention and LDPC codes became one of the most active research topics in coding theory. In 2001, Luby *et al.* showed a very important result that the LDPC codes performance can be extremely improved by using irregular graphs [11]. It was shown in [12] that carefully designed LDPC codes can perform a few thousandth of a dB away from the Shannon limit on the additive white Gaussian noise channel.

An asymptotic analysis method called *density evolution* was developed in [13] by Richardson and Urbanke, which enabled designing capacity approaching irregular LDPC codes for different symmetric channels. This method tracks the probability density of the messages in each iteration.

Due to the computational complexity of density evolution, some approximations were proposed later. Chung *et al.* proposed the Gaussian approximation to density evolution which assumes that all the messages are Gaussian random variables [14]. Later, a more refined approximation was also proposed in [15].

Another method to analyze the behavior of iterative decoders is called extrinsic information transfer (EXIT) chart analysis which is first introduced by ten Brink in [16]. EXIT chart tracks the evolution of a single parameter representing the density of the messages passed in each iteration and is a good method to visualize the convergence behavior of iterative decoders. EXIT chart analysis allows designing irregular codes with desired convergence behavior [17].

For highly optimized LDPC codes, which can perform extremely close to the Shannon limit, the decoder may need to perform too many iterations. Although their decoding complexity is low for each iteration, the overall decoding can be complex and slow. Therefore, using an EXIT chart analysis, complexity-optimized LDPC codes have been proposed [18].

Since LDPC codes were able to approach the Shannon limit on many channels, they have also been proposed for fading channels [19] and also for different modulation schemes such as orthogonal frequency division multiplexing (OFDM) [20], multi-level coding and bit-interleaved coded modulation (BICM) [21] and also for multiple-input multiple-output (MIMO) fading channels [22].

The analysis methods described here are all asymptotic methods. In other words, it is assumed that the code length is infinite. In practice, however, finite-length LDPC codes are used and the asymptotic analysis gives inaccurate results. Therefore, a finite-length analysis method is presented in [23] for the binary erasure channel. Also, it is shown in [24] that the performance of finite-length LDPC codes in the waterfall region follows a scaling law on the binary erasure channel.

LDPC codes have also played an important role in the discovery of many other graphical codes, e.g., Repeat-accumulate (RA) codes [25], Luby transform (LT) codes [26], and Raptor codes [27]. LDPC codes have been used in some communication standards such as digital video broadcasting (DVB) ETSI EN 302 307 [28] and in IEEE 802.16 WiMAX standard [29].

Although there have been many research activities and advances in the area of graphical codes, there are still many open problems. The field of graphical codes and especially LDPC codes is still one of the most active research areas in coding theory today.

1.3 Overview

As stated before, there are still many open problems in the field of LDPC codes. In this thesis, we address some of these problems and we raise new questions.

In most of the LDPC code design and analysis methods, it is assumed that the code's block length is infinite. However, in practice, finite-length codes are used. The accuracy of these asymptotic methods degrades as the block length gets smaller. Therefore, there is a vital need for a method capable of analyzing and designing finite-length LDPC codes.

The performance degradation of finite-length LDPC codes is usually blamed on the cycles, stopping sets [23], trapping sets [30], and other weaknesses in the graphical structure of the code. For the case of binary erasure channel (BEC) the effect of cycles and stopping sets can be identified and quantified based on a combinatorial analysis [23]. Therefore, the average performance of LDPC codes can be estimated on the BEC. However, this combinatorial analysis is computationally complex and even for practical block lengths only very simple LDPC ensembles can be analyzed.

In another work, it has been proved that the finite-length performance of LDPC codes follows a *scaling law* on the BEC in the waterfall region [24]. Unfortunately, this method is also complex and it cannot be applied to other channels.

There is still a need for an accurate and efficient method to analyze the behavior of finite-length LDPC codes on different channels. An accurate finite-length analysis of LDPC codes may be used for designing powerful finite-length codes. In this thesis, we address these problems and we come up with a solution. In particular, we show that most of the performance degradation of finite-length LDPC codes can be described by the channel atypical behavior when observed over a finite block length. To this end, we analyze the behavior of the channel and two of its parameters as random variables. We then use them in the analysis of the finite-length LDPC codes. Moreover, we use this analysis to provide design guidelines for finite-length LDPC codes.

One of the most important applications of error-correcting codes is on wireless fading channels. In fact, carefully designed LDPC codes have very good performance on fading channels [19]. However, when the channel fading gain is not known at the receiver, the computation of the channel log-likelihood ratio (LLR) is too complicated. Furthermore, when the channel noise power is not known at the receiver, the decoder is not able to calculate the LLR correctly and the performance degrades. LLR calculation in fading channels when no information about the channel is available at the receiver is complex and is a problem that needs to be solved.

In this thesis, we propose a linear method of LLR calculation for LDPC codes and in general iterative decoding on fading channels when the channel parameters are not known at the receiver. This method maximizes the achievable transmission rate on the channel. The complexity of our method is low and its performance is extremely close to the optimum achievable performance obtained by true LLR calculation. This method is applicable to other codes decoded by iterative message-passing decoders such as turbo codes.

1.4 Organization of the Thesis

In the next chapter we briefly review the necessary background material on LDPC codes and iterative message-passing algorithms. In particular, we provide the required background on graphical codes and LDPC codes' structure, different decoding algorithms, different analysis methods such as density evolution and its approximations and EXIT charts.

In Chapter 3, an efficient method for finite-length LDPC code analysis on binary-input memoryless symmetric channels is proposed. This method is based on studying the variations of the channel quality around its expected value when observed during a finite-length codeword. We propose modeling these variations with a single parameter. This parameter is then viewed as a random variable and its probability density function is obtained. Assuming that a decoding failure is the result of an observed channel worse than the code's decoding threshold, the block error probability of finite-length LDPC codes is estimated. Using an EXIT chart analysis, bit error probability is obtained from the block error probability. The effects of using different channel parameters for modeling these variations are also studied. This method can closely predict the performance of LDPC codes of a few thousand bits or longer in the waterfall region[1].

In Chapter 4, we investigate uncorrelated fading channels with no channel state information at the receiver where calculating true LLRs is difficult. Existing work assume that the power of the additive noise is known and use the expected value of the fading gain in a linear function of the channel output to find approximate LLRs. In this chapter, we first assume that the power of the additive noise is known and we find the optimum linear approximation of LLRs in the sense of maximum

[1]The results of this chapter have been accepted for publication in the Proceedings of IEEE International Conference on Communications (ICC), Glasgow, Scotland, 2007 [31] and have also been submitted for publication in IEEE Transactions on Communications [32].

achievable transmission rate on the channel. The maximum achievable rate under this linear LLR calculation is almost equal to the maximum achievable rate under true LLR calculation. We also observe that this method appears to be the optimum in the sense of bit error rate performance too. These results are then extended to the case that the noise power is unknown at the receiver and a performance almost identical to the case that the noise power is perfectly known is obtained[2].

Chapter 5 concludes the thesis. We provide a summary of the contributions of this thesis and suggest problems for further research.

[2]The results of this chapter have been accepted for publication in the Proceedings of IEEE International Symposium on Information Theory (ISIT), Nice, France, 2007 [33].

Chapter 2

Preliminaries and Background

In this chapter, we briefly review some necessary background on channel coding, different communication channels, linear block codes and their graphical representation, LDPC codes and their structure and different design and analysis methods.

In Fig. 2.1, the block diagram of a generic digital communication system is depicted. A communication system generally consists of a transmitter, a channel, and a receiver. The transmitter mainly consists of an analog to digital converter, source encoder, encryptor, channel encoder, and modulator. Channel is the medium through which the information is transmitted. Examples of the channel could be simple copper wire pairs, coaxial cables, optical fibers, free air, a network link, etc. The receiver mainly consists of a demodulator, channel decoder, decryptor, source decoder, and digital to analog converter. In this thesis, we are mainly concerned with the channel encoder and decoder and the channel itself.

2.1 Channel Models and Channel Coding

A communication channel can be viewed as a system whose output depends on its input probabilistically. In fact, the channel is fully characterized by the alphabet of X and Y and the set of conditional probability assignment between them $\mathrm{P}_{Y|X}$, where X denotes the input random variable and Y denotes the output random variable.

In this thesis, we are concerned with channels whose input alphabet is the set of binary symbols and each channel output depends only on the current input, giving rise to a binary-input memoryless channel. Moreover, we only consider channels that are output-symmetric. The most famous channels in this class are the binary

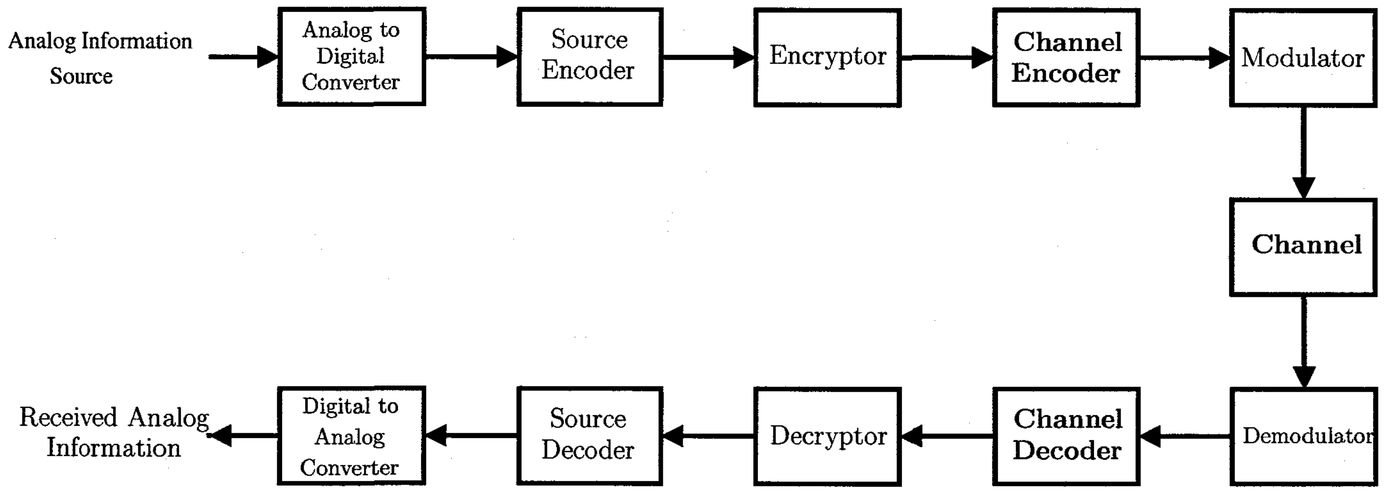

Analog Information Source
Analog to Digital Converter
Source Encoder
Encryptor
Channel Encoder
Modulator
Channel
Demodulator
Channel Decoder
Decryptor
Source Decoder
Digital to Analog Converter
Received Analog Information

Figure 2.1: The block diagram of a generic communication system.

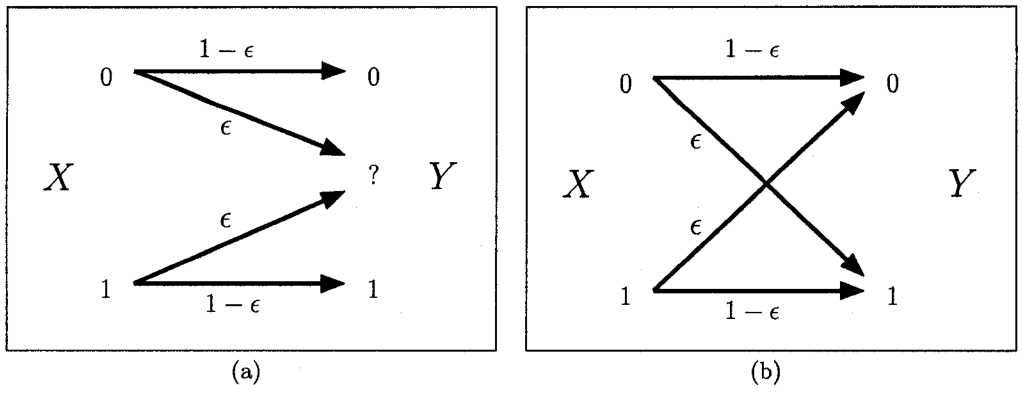


Figure 2.2: Two binary-input memoryless symmetric channels: (a) The BEC(ϵ), and (b) BSC(ϵ).

erasure channel (BEC) and the binary symmetric channel (BSC).

The binary erasure channel model is the simplest channel and is depicted in Fig. 2.2(a). The output is either received correctly with the probability $1-\epsilon$ or as an erasure with probability ϵ. Hereafter, we denote this channel by BEC(ϵ). A real-world example of the BEC is the packet transmission link between two nodes in a data network where a packet is either received correctly or it is lost.

In the BSC, on the other hand, an input symbol is either received correctly or flipped (received as the other symbols). The BSC is illustrated in Fig. 2.2(b). The crossover probability is ϵ in this channel. Hereafter, we call this channel BSC(ϵ). The BSC model is a simplified model of a noisy channel, where the information is affected by noise and is received with error. Many real-world digital communication channels can be modeled with the BSC.

The output-symmetry here (and also for the BEC) means that for these channels

$$\mathrm{P}_{Y|X}(0|0) = \mathrm{P}_{Y|X}(1|1), \tag{2.1}$$

or in other words, 0 and 1 are treated equally with the channel.

Two other famous channel models that will be used in this work are the binary-input additive white Gaussian noise (BIAWGN) channel and binary-input uncorrelated fading channels.

In the BIAWGN channel and binary-input uncorrelated fading channels, the set of output symbols take continuous values in the range $(-\infty, \infty)$. In the BIAWGN(σ_z) channel, the output is given by

$$y = x + z, \tag{2.2}$$

where $x \in \{-1, 1\}$ represents the input signal and z is the Gaussian noise with zero mean and variance σ_z^2. The BIAWGN channel model is of great theoretical and practical importance in communications.

In the binary-input uncorrelated fading channels, The output of the channel is given by

$$y = r \cdot x + z, \tag{2.3}$$

where $x \in \{-1, 1\}$ represents the input signal and z is the Gaussian noise with zero mean and variance σ_z^2. Also $r \geq 0$ is the channel gain which has an arbitrary probability density function (pdf) $f_R(r)$ and changes independently from one channel use to another. This channel model is used in modeling wireless communication channels.

Using the binary phase-shift keying (BPSK) modulation, i.e., the input letters are $+1$ (for binary 0) and -1 (for binary 1), the output-symmetry of the BIAWGN channel and uncorrelated fading channel states that

$$\mathrm{P}_{Y|X}(y|1) = \mathrm{P}_{Y|X}(-y|-1). \tag{2.4}$$

We call the channels presented here binary-input memoryless symmetric (BIMS) channels throughout this work.

One of the most important properties of a channel is its capacity. The capacity of a channel is defined as the maximum mutual information between the channel input and output random variables and is given by

$$C = \max_{p(X)} I(X;Y), \tag{2.5}$$

where $I(X;Y)$ denotes the mutual information of X and Y and $p(X)$ denotes the input probability distribution. For example, $C = 1 - \epsilon$ for the BEC(ϵ), and $C = 1 - h(\epsilon)$ for the BSC(ϵ) where

$$h(\epsilon) = -\epsilon \log_2 \epsilon - (1 - \epsilon) \log_2 (1 - \epsilon) \tag{2.6}$$

is called the binary entropy function [34].

The importance of the channel capacity is mainly due to the noisy-channel coding theorem and its converse. The noisy-channel theorem states that if the channel capacity is C, there exists an encoding and decoding rule under which it is possible to have an arbitrary small probability of error at data transmission rates $R_t < C$ [1].

The converse to the noisy-channel coding theorem states that if $R_t > C$, regardless of the encoding and decoding rule used, the probability of error cannot be less than some positive number [1]. Thus, the noisy-channel coding theorem shows that C is a fundamental limit for reliable data transmission on a channel.

2.2 Linear Block Codes

In this section, we introduce basic concepts of linear block codes, one of the richest classes of codes. The codes we consider in this thesis are binary codes and we assume that the information source is a sequence of binary symbols 0 and 1. Thus, when we refer to linear block codes, we mean binary linear block codes.

In block coding, the information sequence is partitioned in blocks of fixed length k, called message blocks $\mathbf{u}$, representing k information bits. An (n, k) block code is a transformation of message blocks of length k according to a pre-defined rule into blocks of length n $(n > k)$, called codewords $\mathbf{c}$. There are 2^k codewords corresponding to 2^k possible message blocks. A block code is linear if and only if the modulo-2 sum of two codewords is also a codeword [35]. We call n the code's block length and k the dimension of the code.

The *generator matrix* $\mathbf{G}$ of an (n, k) linear block code is defined as the $k \times n$ matrix whose rows are k linearly independent codewords $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ of length n, which generate all the codewords. Thus, if $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ then

$$\begin{aligned} \mathbf{c} &= \mathbf{u} \cdot \mathbf{G} \\ &= (u_0, u_1, \ldots, u_{k-1}) \cdot \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} \\ &= u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + \cdots + u_{k-1}\mathbf{g}_{k-1} \end{aligned} \tag{2.7}$$

Linear block codes are fully specified by the rows of their generator matrix $\mathbf{G}$.

Another matrix associated with linear block codes which is useful in decoding, is their *parity-check matrix.* The parity-check matrix $\mathbf{H}$ of an (n, k) linear block code is an $(n - k) \times n$ matrix such that a vector $\mathbf{c}$ of length n is a codeword if and only if $\mathbf{c} \cdot \mathbf{H}^T = 0$. The rows of $\mathbf{H}$ generate the null space of $\mathbf{G}$, i.e., $\mathbf{G} \cdot \mathbf{H}^T = 0$. In fact, the parity-check matrix gives the parity-check equations on the message bits. Thus, if the rows of $\mathbf{H}$ are denoted by $\mathbf{h}_i = (h_{i0}, h_{i1}, \ldots, h_{i,(n-1)})$ for $i = 0, 1, \ldots, n-k-1$, the parity-check equations are given by

$$c_0 h_{i0} + c_1 h_{i1} + \cdots + c_{n-1} h_{i,(n-1)} = 0. \tag{2.8}$$

These equations show that the codeword bits satisfy $n - k$ even parity constraints. Thus, if any of these equations are not satisfied for the received codeword in the decoder, error has occurred in the channel.

These concepts and definitions will be more clear with an example. Consider a $(7, 4)$ Hamming code which is a simple example of a linear block code [35]. The message sequence has four bits and the codewords have seven bits. The generator matrix $\mathbf{G}$ and the parity-check matrix $\mathbf{H}$ of this code are given by

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

and

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{2.10}$$

Thus, using (2.7), we can obtain the codewords listed in Table 2.1.

The parity-check equations are given by (2.8) as

$$c_0 + c_3 + c_5 + c_6 = 0 \tag{2.11}$$

$$c_1 + c_3 + c_4 + c_5 = 0 \tag{2.12}$$

$$c_2 + c_4 + c_5 + c_6 = 0, \tag{2.13}$$

where the addition is in fact the modulo-2 sum and we will denote it by $\oplus$ hereafter. Therefore, three sets of codeword bits should satisfy even parity constraints. These sets are $\{c_0, c_3, c_5, c_6\}$, $\{c_1, c_3, c_4, c_5\}$, and $\{c_2, c_4, c_5, c_6\}$. This concept has been depicted in Fig. 2.3.

2.3 LDPC Codes: Graphical Representation

An LDPC code is a linear block code which has a sparse parity-check matrix. This means that the parity-check matrix $\mathbf{H}$ of an LDPC code, has a low density of 1's. In fact, the number of 1's in $\mathbf{H}$ grows linearly with the block length n. As for other

Messages	Codewords
(0 0 0 0)	(0 0 0 0 0 0 0)
(0 0 0 1)	(1 1 0 1 0 0 0)
(0 0 1 0)	(0 1 1 0 1 0 0)
(0 0 1 1)	(1 0 1 1 1 0 0)
(0 1 0 0)	(1 1 1 0 0 1 0)
(0 1 0 1)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(0 1 1 1)	(0 1 0 1 1 1 0)
(1 0 0 0)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(1 0 1 0)	(1 1 0 0 1 0 1)
(1 0 1 1)	(0 0 0 1 1 0 1)
(1 1 0 0)	(0 1 0 0 0 1 1)
(1 1 0 1)	(1 0 0 1 0 1 1)
(1 1 1 0)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

Table 2.1: The message sequences and their corresponding codewords for a (7,4) Hamming code.

linear block codes, LDPC codes can be represented by their generator and parity-check matrices. However, for the purpose of analysis, LDPC codes are almost always represented graphically which gives a better insight in analyzing iterative decoding algorithms. We explain how these codes can be graphically represented after giving some definitions.

A *bipartite graph* is a graph where the nodes can be divided into two disjoint sets and the edges of the graph may only connect two nodes from different sets. The graphical representation of linear codes started with the introduction of a bipartite graph called Tanner graph in [3] for linear block codes. Another major step in this field was the introduction of *factor graphs* [36]. A factor graph, is a graph visualizing the factorization of a global multivariate function into simpler local functions. In fact, a factor graph is a bipartite graph whose nodes are classified as variable nodes (corresponding to the function variables) and function (check) nodes (corresponding to the functions).

To see how LDPC codes and in general linear codes can be represented by factor graphs, consider a graph $\mathcal{G}$ with n variable nodes and r check nodes. The variable nodes, which are usually shown by circles, are binary variables (0 or 1) and represent

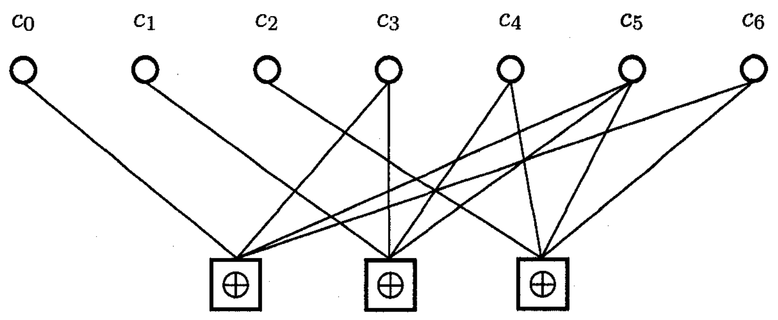


Figure 2.3: The visualization of even parity constraints for a (7,4) hamming code. The code bits are shown by $c_0, c_1, \ldots, c_6$ and circles. Even parity constraints are shown by squares. In other words, the modulo-2 sum of the bits connected to a square is 0.

the codeword bits. The check nodes, shown by squares, represent the even parity constraints on the variable nodes. In particular, a check node c_i shows an even parity constraint given by

$$\bigoplus_{j:\mathsf{v}_j \in n(\mathsf{c}_i)} \mathsf{v}_j = 0 \tag{2.14}$$

where v_j represents the jth variable node, $n(\mathsf{c}_i)$ is the set of all variable nodes connected to c_i and $\oplus$ shows the modulo-2 sum [37]. A sample graph having eight variable nodes and four check nodes is depicted in Fig. 2.4. There are four parity-check equations corresponding to the four check nodes, which can be written as

$$\mathsf{c}_1 : \quad \mathsf{v}_2 \oplus \mathsf{v}_3 \oplus \mathsf{v}_4 \oplus \mathsf{v}_7 = 0 \tag{2.15}$$

$$\mathsf{c}_2 : \quad \mathsf{v}_1 \oplus \mathsf{v}_4 \oplus \mathsf{v}_5 \oplus \mathsf{v}_6 \oplus \mathsf{v}_8 = 0 \tag{2.16}$$

$$\mathsf{c}_3 : \quad \mathsf{v}_1 \oplus \mathsf{v}_2 \oplus \mathsf{v}_3 \oplus \mathsf{v}_5 \oplus \mathsf{v}_6 = 0 \tag{2.17}$$

$$\mathsf{c}_4 : \quad \mathsf{v}_4 \oplus \mathsf{v}_5 \oplus \mathsf{v}_6 \oplus \mathsf{v}_7 \oplus \mathsf{v}_8 = 0. \tag{2.18}$$

If $\mathbf{H}$ is defined as the adjacency matrix of $\mathcal{G}$, then $\mathcal{G}$ gives rise to a linear code of block length n, dimension $k \geq n - r$, and an $r \times n$ parity-check matrix $\mathbf{H}$. In other words, $\mathcal{G}$ represents a binary $r \times n$ parity-check matrix in which the (i, j) entry is 1 if and only if the check node c_i is connected to the variable node v_j. The dimension of the code is equal to $n - r$ if and only if all the parity constraints are linearly independent and this is equivalent to $\mathbf{H}$ being full rank.

When representing an LDPC code by factor graphs, the corresponding graph is

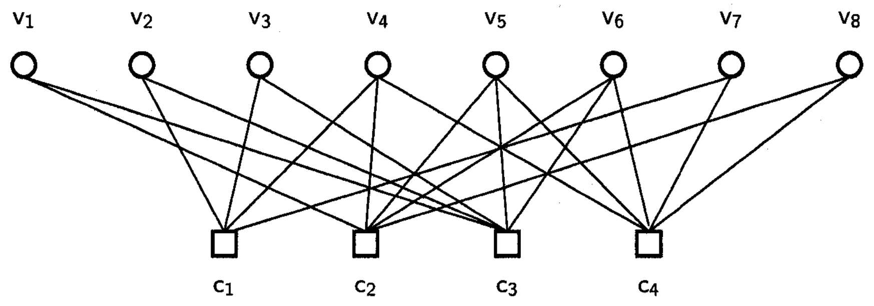


Figure 2.4: A bipartite graph representing an LDPC code. The graph has 8 variable nodes $\mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_8$ and 4 check nodes $\mathsf{c}_1, \ldots, \mathsf{c}_4$.

a sparse factor graph where the number of edges E grows linearly with the number of variable nodes n. Any linear block code has a factor graph representation. If also the factor graph is sparse, we call the code corresponding to the graph an LDPC code. In this work, we are concerned with binary LDPC codes. Therefore, in the remainder of the thesis, when we refer to LDPC codes, we mean binary LDPC codes.

LDPC codes are classified as *regular* or *irregular* based on their structure. A regular LDPC code has variable nodes of a fixed equal degree d_{v} and check nodes of a fixed equal degree d_{c}. For a regular code we have

$$E = d_{\mathsf{v}} \cdot n = d_{\mathsf{c}} \cdot r. \tag{2.19}$$

If viewed from the $\mathbf{H}$ matrix perspective, in a regular LDPC code, $\mathbf{H}$ is a matrix having d_{c} number of 1's in each row and d_{v} number of 1's in each column. An ensemble of $(d_{\mathsf{v}}, d_{\mathsf{c}})$-regular LDPC codes is defined as the ensemble of LDPC codes having variable nodes of degree d_{v} and check nodes of degree d_{c}. Now it is more clear that why the number of 1's in $\mathbf{H}$ or equivalently the number of edges in $\mathcal{G}$ grows linearly with n. The reason is that the number of 1's is fixed in each row or column and the number of rows or columns grow linearly with n.

Irregular LDPC codes were first considered in [11] and it was shown that the performance of LDPC codes can extremely improve by using irregular graphs. In an irregular LDPC code, not all the variable nodes or the check nodes have the same fixed degree. Consequently, the variable and check nodes are defined by two edge degree distributions $\{\lambda_2, \lambda_3, \ldots, \lambda_{d_{\mathsf{v}}}\}$ and $\{\rho_2, \rho_3, \ldots, \rho_{d_{\mathsf{c}}}\}$. In this notation λ_i is the fraction of edges connected to degree-i variable nodes and ρ_i is the fraction of edges connected to degree-i check nodes. Here d_v and d_c refer to the maximum variable and check node degree respectively. Another widely used notation is the representation of the degree sequences by their polynomial generators $\lambda(x) = \sum_{i=2}^{d_\mathsf{v}} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=2}^{d_\mathsf{c}} \rho_i x^{i-1}$ [11]. Using this notation, the relations which contain the degree distributions can be written more easily. In this thesis, we mainly use the latter notation. For irregular codes defined by $\lambda(x)$ and $\rho(x)$, we have

$$n = E\sum_{i=2}^{d_\mathsf{v}} \frac{\lambda_i}{i} = E\int_0^1 \lambda(x)\mathrm{d}x, \tag{2.20}$$

and

$$r = E\sum_{i=2}^{d_\mathsf{c}} \frac{\rho_i}{i} = E\int_0^1 \rho(x)\mathrm{d}x. \tag{2.21}$$

In this thesis, we denote an ensemble of LDPC codes with block length n and degree distributions $\lambda(x)$ and $\rho(x)$ by $\mathcal{C}^n(\lambda(x), \rho(x))$.

As the concentration theorems of [13] show, if the length of the code is large enough, the behavior of all instances of the ensemble concentrates about an expected behavior which corresponds to the infinite-length code with the same degree distribution. Thus, the average performance of an ensemble of LDPC codes is specified by its degree distributions. This explains why LDPC codes are almost always represented only by their degree distributions.

The rate of a $(d_\mathsf{v}, d_\mathsf{c})$-regular LDPC code is given by

$$R = \frac{k}{n} \geq \frac{n-r}{n} = 1 - \frac{d_\mathsf{v}}{d_\mathsf{c}} \tag{2.22}$$

As stated before, if the $\mathbf{H}$ matrix is full rank or all the parity constraints are linearly independent the equality holds in (2.22). However, it is common to ignore the linear dependencies and always consider that $R = 1 - d_\mathsf{v}/d_\mathsf{c}$. The linear dependencies cause the rate to be higher.

Similarly, ignoring the dependencies, the rate of an ensemble of irregular codes defined by $\lambda(x)$ and $\rho(x)$ is given by

$$R = 1 - \frac{\sum_{i=2}^{d_\mathsf{c}} \frac{\rho_i}{i}}{\sum_{i=2}^{d_\mathsf{v}} \frac{\lambda_i}{i}} = 1 - \frac{\int_0^1 \rho(x)\mathrm{d}x}{\int_0^1 \lambda(x)\mathrm{d}x}. \tag{2.23}$$

We conclude this section by an example. Consider an irregular LDPC code $\mathcal{C}^{1000}(\lambda(x), \rho(x))$ with variable node degree distribution $\lambda(x) = 0.4x^2 + 0.4x^5 + 0.2x^8$ and check node degree distribution $\rho(x) = x^8$. Using (2.20) and (2.21), this code has 4500 edges. From the degree distribution, it can be said that 40% of edges are connected to variable nodes of degree three (600 nodes), another 40% to variable nodes of degree 6 (300 nodes) and 20% to variable nodes of degree 9 (100 nodes). Also, it can be understood that there are 500 check nodes all of degree 9. Finally the rate of this code is 0.5, which is given by (2.23).

2.4 LDPC Codes: Decoding

LDPC codes are usually decoded by a class of iterative decoding algorithms called message-passing algorithms. In iterative message-passing decoding algorithms, there are two sources of information about the transmitted codeword available at each iteration. One is the information from the channel called the *intrinsic* information, and the other is the information from the previous iteration called the *extrinsic* information. At each iteration, the decoder combines these two sources of information in an effective way to gain better knowledge about the transmitted codeword.

In these algorithms, the messages passed in each iteration are probabilities or beliefs. These probabilities are passed between the variable and check nodes along the edges connecting them. In fact, in each half iteration, each check node c passes its belief (as a probability) about the variable node v to v based on the beliefs received in the previous iteration from the connecting variable nodes except v. Then, in the next half iteration, each variable node v passes its belief (as a probability) about itself to a connecting check node c based on the beliefs received from the channel and from the connected check nodes except c. Fig. 2.5 shows an example of how the message from a variable node to a check node is calculated in one iteration of the iterative message-passing decoding algorithm. The variable node v_4 passes a message to the check node c_1. This message is calculated in one iteration based on the following procedure. In the first half iteration, c_2 calculates its outgoing message based on the messages it received from v_1, v_5, v_6, and v_8 and sends it to v_4. Also, c_4 calculates its outgoing message based on the messages it received from v_5, v_6, v_7, and v_8 and sends it to v_4. In the next half iteration, v_4 calculates its outgoing message based on the messages received from the channel and from c_2 and c_4 and sends it to c_1.

To further illustrate the idea of iterative message-passing decoding, consider a $(d_{\mathsf{v}}, d_{\mathsf{c}})$-regular LDPC code. In each iteration, the message passed from a check

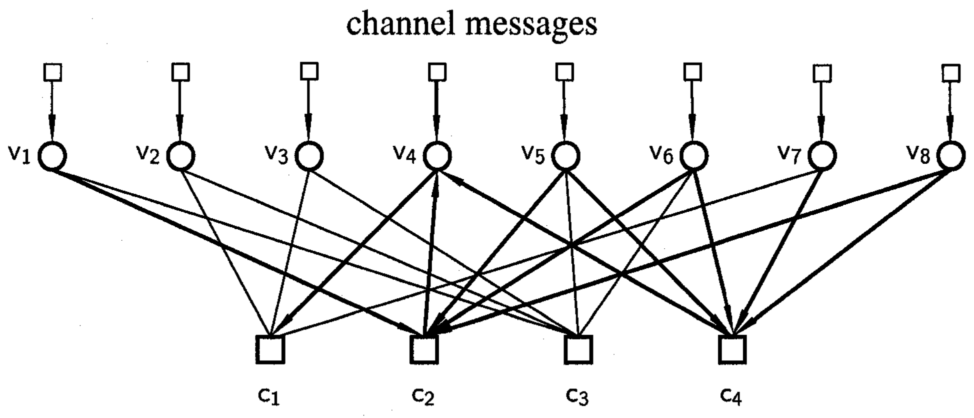


Figure 2.5: An example of how the message from a variable node to a check node is calculated in one iteration of iterative message-passing decoding.

node c to a variable node v is calculated based on the $d_c - 1$ incoming variable node messages. Then, the message passed from a variable node v to check node c is calculated based on the channel observation message and $d_v - 1$ incoming check node messages. This one iteration process can be visualized as a decoding tree of depth one. This decoding tree illustrates the process of message passing between the variable nodes and check nodes in one iteration. The variable nodes output messages are then used in check nodes to start the next iteration. If more iterations are considered, a decoding tree of depth more than one can be obtained. For example, a decoding tree of depth two, showing two message passing iterations is illustrated in Fig. 2.6 for a (3,4)-regular LDPC code.

When the variable nodes only have binary values of $\{0, 1\}$, the probability messages passed along the edges could be the probability of being 0 or the probability of being 1, i.e., P(0) or P(1). It is usually more advantageous to work with log-likelihood ratios (LLRs) instead of probabilities. For binary-valued random variables, LLR is defined as $\log \frac{P(0)}{P(1)}$. Using LLRs, the message update rules become simple and also probabilities very close to 1 and 0 can be represented with higher precision in finite precision computer implementations.

In this section, we first explain the *sum-product* decoding algorithm [36], which is the most powerful iterative message-passing decoding algorithm and we give its message passing and updating rules. After that, we describe the *min-sum* decoding algorithm which is an approximation to the sum-product algorithm. Then, we present how message-passing algorithms can be made simpler on the BEC and BSC

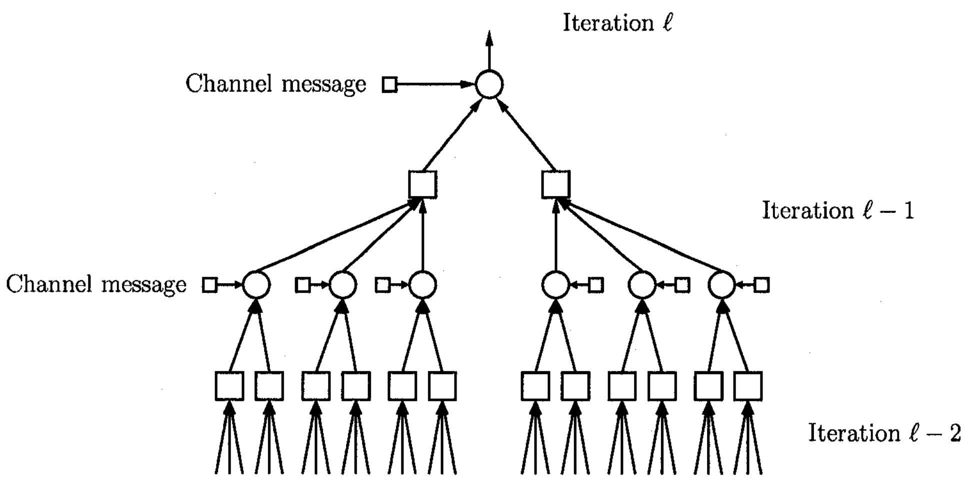


Figure 2.6: The decoding tree of depth 2 of a (3,4)-regular LDPC code. The channel messages are represented by small squares.

using their binary versions.

2.4.1 The sum-product algorithm

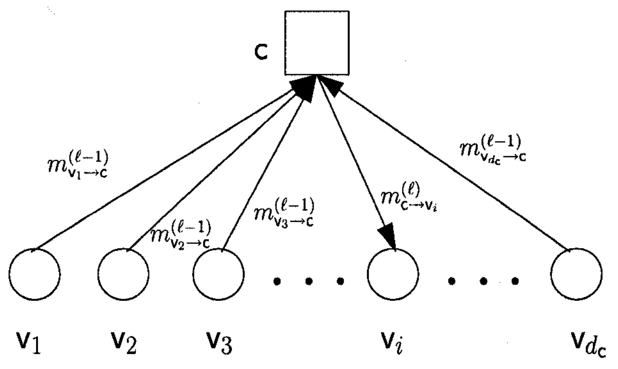
We describe the sum-product algorithm where the messages passed are real valued LLRs. At first, the channel messages m_{0_v} which are called the intrinsic messages are calculated based on channel output values corresponding to each variable node v. In fact, for each channel output y and its corresponding variable node v, the LLR message is

$$m_{0_\mathsf{v}} = \log \frac{\mathrm{P}_{X|Y}(0|y)}{\mathrm{P}_{X|Y}(1|y)}, \tag{2.24}$$

where $x \in \{0,1\}$ is the channel input bit. The variable nodes are initialized by these messages. Then, each variable node v passes its message to all neighboring check nodes c. This message is given by $m^{(0)}_{\mathsf{v}\to\mathsf{c}} = m_{0_\mathsf{v}}$.

At the next step, each check node c calculates its message, denoted by $m^{(1)}_{\mathsf{c}\to\mathsf{v}}$ for each connected variable node v and sends it to v based on the messages received from the connected variable nodes except v. This completes the first half of iteration one. In the next half iteration, based on the incoming check node messages and the intrinsic messages, each variable node calculates its messages and sends them to the connected check nodes. This process continues iteratively.

In more detail, this iterative process can be described by two iterative updating



c
$m^{(\ell-1)}_{\mathsf{v}_1\to\mathsf{c}}$
$m^{(\ell-1)}_{\mathsf{v}_2\to\mathsf{c}}$
$m^{(\ell-1)}_{\mathsf{v}_3\to\mathsf{c}}$
$m^{(\ell)}_{\mathsf{c}\to\mathsf{v}_i}$
$m^{(\ell-1)}_{\mathsf{v}_{d_c}\to\mathsf{c}}$
v_1
v_2
v_3
v_i
v_{d_c}

Figure 2.7: The process of updating messages in a check node of degree $d_{\mathsf{c}} > 4$. The variable node messages from the previous iteration are fed into the check node c and processed to be sent to the variable node v_i.

rules. The check node c sends its message to the variable node v based on the following updating rule [4, 36]:

$$m^{(\ell)}_{\mathsf{c}\to\mathsf{v}} = 2\tanh^{-1}\left(\prod_{\mathsf{v}_i\in n(\mathsf{c})-\{\mathsf{v}\}} \tanh\left(\frac{m^{(\ell-1)}_{\mathsf{v}_i\to\mathsf{c}}}{2}\right)\right), \tag{2.25}$$

where $m^{(\ell)}_{\mathsf{c}\to\mathsf{v}}$ shows the message sent from c to v in the ℓth iteration, $n(\mathsf{c})$ is the set of neighboring variable nodes connected to c, and $m^{(\ell-1)}_{\mathsf{v}_i\to\mathsf{c}}$ denotes the message passed from the variable node v_i to c in the previous iteration. This updating process is shown in Fig. 2.7 for a check node with degree d_{c}. Then, the variable nodes update their messages by [4, 36]

$$m^{(\ell)}_{\mathsf{v}\to\mathsf{c}} = m_{0_{\mathsf{v}}} + \sum_{\mathsf{c}_j\in n(\mathsf{v})-\{\mathsf{c}\}} m^{(\ell)}_{\mathsf{c}_j\to\mathsf{v}}, \tag{2.26}$$

where $n(\mathsf{v})$ is the set of neighboring check nodes connected to v. This updating rule is shown in Fig. 2.8 for a check node with degree d_{c}. Notice that at the start of the decoding, i.e., the initialization, $m^{(0)}_{\mathsf{c}\to\mathsf{v}} = 0$. The messages passed between the nodes in the decoder are called the extrinsic messages.

A decision can be made on the variable node v based on the following rule:

$$V = \begin{cases} 0, & m_{0_{\mathsf{v}}} + \sum_{\mathsf{c}_j\in n(\mathsf{v})} m^{(\ell)}_{\mathsf{c}_j\to\mathsf{v}} > 0 \\ 1, & m_{0_{\mathsf{v}}} + \sum_{\mathsf{c}_j\in n(\mathsf{v})} m^{(\ell)}_{\mathsf{c}_j\to\mathsf{v}} < 0 \end{cases}, \tag{2.27}$$

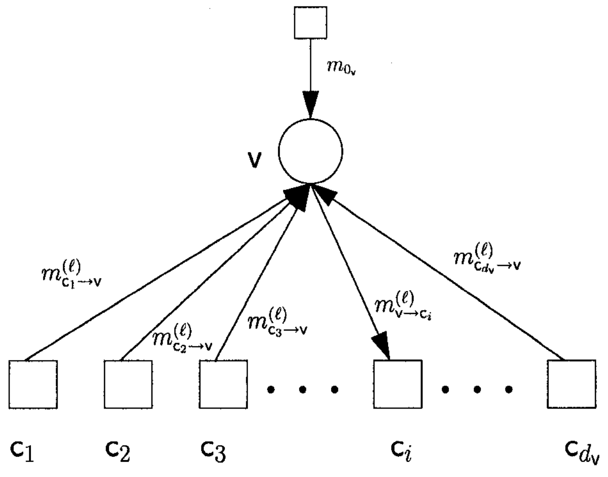


Figure 2.8: The process of updating messages in a variable node of degree $d_v > 4$. The variable node messages from the previous half iteration are fed into the variable node v along with the channel message m_{0_v} and processed to be sent to the check node c_i.

and if $m_{0_v} + \sum_{c_j \in n(v)} m^{(\ell)}_{c_j \to v} = 0$ either $V = 1$ or $V = 0$ can be selected randomly. In fact, for each variable node the extrinsic messages of all neighboring check nodes and the intrinsic message are used to calculate the decision making message $m_{0_v} + \sum_{c_j \in n(v)} m^{(\ell)}_{c_j \to v}$. Since the messages are LLR messages, a positive LLR votes in favor of $V = 0$ and a negative LLR votes in favor of $V = 1$.

As seen in (2.25) and (2.26), the outgoing message is being processed based on the messages received from all the connected nodes except the node which is receiving the outgoing message. This is a key aspect of iterative decoding, since it is needed for the independency of the messages (to be discussed later). As the messages are passed between the nodes in each iteration, the reliability of the variable node messages increase since they receive multiple beliefs about their value at each iteration. In contrast, the reliability of the check node messages decrease as they receive messages from the neighboring variable nodes about their own values. Since these messages are processed so that the neighboring variable nodes satisfy an even parity constraint, the reliability of the outgoing check node message is even less than the reliability of the least reliable incoming message [37]. This process continues until all the errors are corrected by the decoder in a successful decoding.

One of the most important advantages of sum-product and in general belief propagation decoding is that its decoding complexity is low and hence practically implementable for large block lengths. This is due to the fact that in the algorithm, the messages are passed along the edges of a sparse graph (i.e., the number of edges traversed is small). Also, the number of edges are linearly proportional to the block length n and the number of operations are linear in n. Therefore, the decoding complexity of belief propagation decoding is linear in n and thus constant per information bit.

It is worth mentioning that sum-product is in general not as powerful as maximum likelihood (ML) decoding which is the optimum decoding algorithm. In fact, when the code's factor graph has cycles—which will be briefly discussed later—sum-product is sub-optimum. On the other hand if no cycle exists and the code's factor graph is a tree, sum-product is optimum and thus equivalent to ML decoding [4,7,36]. For large enough block lengths, since the effect of cycles is negligible for large number of iterations (see Section 2.5), sum-product performance is very close to the ML decoding. Since sum-product decoding is more practical than ML decoding especially for large block lengths, it is usually used in the decoding of LDPC codes. For detailed analysis on the tradeoff between the decoding performance and complexity per information bit refer to [38–40].

2.4.2 The min-sum algorithm

The min-sum decoding algorithm is a simplified version of the sum-product algorithm [4]. While it is not as effective and powerful as the sum-product algorithm, it is less complex and can be implemented easier.

In this algorithm, the variable node update rule is the same as sum-product and is given by (2.26). The check node update rule is an approximation to (2.25) and is given by [4,36]

$$m_{\mathsf{c}\to\mathsf{v}}^{(\ell)} = \min_{\mathsf{v}_i \in n(\mathsf{c})-\{\mathsf{v}\}} \left| m_{\mathsf{v}_i\to\mathsf{c}}^{(\ell-1)} \right| \prod_{\mathsf{v}_i \in n(\mathsf{c})-\{\mathsf{v}\}} \operatorname{sign}\left(m_{\mathsf{v}_i\to\mathsf{c}}^{(\ell-1)} \right). \tag{2.28}$$

When the magnitude of the messages gets large, (2.28) better approximates (2.25). Thus, in later iterations, where the magnitude of extrinsic messages have become large, the min-sum algorithm performs nearly the same as sum-product.

2.4.3 Gallager's algorithm A and B

The sum-product and min-sum both use *soft decision decoding* which means that real-valued messages are passed. The sum-product is the best algorithm among the message-passing algorithms in terms of the performance. In digital implementations, discretized versions of the sum-product have to be used. The results of [12] show that 11-bit implementation is generally enough and extra bits will not result in noticeable performance improvement. However, this is complicated and in many practical applications it may be difficult to be implemented or may increase the decoding time. In high-throughput applications, very fast decoders are needed. Thus, to reduce the complexity and increase the speed, lower-level discretized versions of sum-product are considered (e.g. 3-bit decoding, 4-bit decoding, etc.). The lowest level of dicretization is 1-bit decoding which is equivalent to passing binary messages and is called *hard decision decoding.*

Gallager's decoding algorithm A and B, both introduced by Gallager in [8], can be used for hard decoding on the BSC. In both of these algorithms the messages passed between the nodes are 0 or 1. In Gallager's algorithm A, the update rule at the check node c is

$$m_{\mathsf{c}\to\mathsf{v}}^{(\ell)} = \bigoplus_{\mathsf{v}_i\in n(\mathsf{c})-\{\mathsf{v}\}} m_{\mathsf{v}_i\to\mathsf{c}}^{(\ell-1)}, \tag{2.29}$$

which is the modulo-two sum of the incoming messages. In the variable nodes, the message updating is based on the following rule:

$$m_{\mathsf{v}\to\mathsf{c}}^{(\ell)} = \begin{cases} m_0, & \exists \mathsf{c}_j \in n(\mathsf{v}) - \{\mathsf{c}\} : m_{\mathsf{c}_j\to\mathsf{v}}^{(\ell)} = m_0 \\ \overline{m_0}, & \text{otherwise} \end{cases}, \tag{2.30}$$

where m_0 is the intrinsic message and $\overline{m_0}$ denotes the binary complement of m_0. In other words, the variable nodes pass the intrinsic messages unless all the incoming extrinsic message disagree with the intrinsic message. In this case, the extrinsic message, passed to the neighboring check node, is equal to the binary complement of the intrinsic message.

Gallager's decoding algorithm B is more powerful than algorithm A but more complex. It is proved in [41] that the algorithm B is in fact the optimum binary message passing algorithm for regular LDPC codes. In this algorithm the check node update rule is the same as (2.29) but the variable update rule is different. Interested reader can refer to [8, 41] for the details.

There are also some modifications of sum-product and min-sum and other message-passing algorithms which can be used in the decoding of LDPC codes [37, 42, 43]. Here, we have presented the most important ones.

2.4.4 Decoding on the BEC

On the BEC, the LLR message is $+\infty$ or $-\infty$ if the corresponding variable node is not erased and is 0 if erased. The check node sends $\pm\infty$ to a variable node if all its incoming messages except the message from the variable node it is sending to are $\pm\infty$ and otherwise it sends 0. The variable node sends 0 if it is erased and sends $\pm\infty$ if and only if there exists at least one $\pm\infty$ incoming check node message other than the one receiving the message. This procedure can be done much easier on the BEC using edge deleting method. Interested reader may refer to [44] for more details.

2.5 LDPC Codes: Analysis methods

2.5.1 Asymptotic analysis methods

As stated in the previous section, in iterative decoding, there are two pieces of information about the transmitted codeword available in the decoder: the intrinsic information which is the observed information from the channel and the extrinsic information which is the information from the previous iterations. The extrinsic information at each round is calculated based on the intrinsic information and the extrinsic information from the previous round. In a successful decoding, the reliability of extrinsic messages improves as the decoding continues iteration by iteration. Therefore, for the analysis of iterative decoding, the statistics of the extrinsic messages are studied in each iteration.

If at every iteration, the incoming messages are independent then the update rules given in the previous section correctly computes the LLRs. This is equivalent to the factor graph being a tree and having no cycles. If the factor graph has cycles with the smallest length—called girth—equal to ℓ, then the neighborhood of a variable node up to depth $\lfloor \frac{\ell}{2} \rfloor$ is a tree and the messages can be assumed independent only for $\lfloor \frac{\ell}{2} \rfloor$ iterations. Therefore, the messages will not be independent for large number of iterations and the existence of cycles makes the messages dependent.

It is worth mentioning that in randomly constructed codes with large enough block lengths, the neighborhood of a fixed depth ℓ of most of the variable nodes is a tree. Therefore, the decoding algorithm calculates the correct LLRs for ℓ iterations in these variable nodes. The fraction of the other nodes having dependent messages is small. Thus, the effect of cycles in the decoding performance is small when the block length is large and their contribution to the probability of error is small and disappears asymptotically. So, for the asymptotic analysis of LDPC codes we can assume that the factor graph is a tree and all the messages are independent.

If linear codes are used and the channel and the update rules have symmetry conditions, then the convergence behavior of iterative decoding is independent of the transmitted codeword. So, for simplicity, it can be assumed that the all-zero codeword is transmitted. The channel symmetry condition is given in (2.4) and the update rules symmetry conditions are given in [13]. The update rules given in Section 2.4 are all symmetric.

Based on the all-zero codeword and message independence assumption, the statistics of the extrinsic messages can be studied. In fact, under the channel symmetry conditions, LLR messages give sufficient statistics for the analysis of the decoding and they have some nice properties. Some of these properties can be found in Appendix A. The most exact analysis method, called *density evolution*, tracks the pdf of the extrinsic messages in each iteration [13]. Throughout this thesis, we assume the all-zero codeword is transmitted when density evolution is used.

Density evolution can be computationally complex in most cases, therefore as an approximation, a representative of the pdf is studied (e.g., extrinsic message error probability [15, 17], mutual information between the transmitted bits and the extrinsic messages [45], and the mean of the extrinsic messages [14], etc.). Moreover, the extrinsic messages' pdf can be fully represented by a single parameter in some cases (e.g., on the BEC and BSC). In these cases, studying the evolution of this single parameter is exact.

Analysis on the BEC and BSC

On the BEC and BSC, since the pdf of the binary messages can be represented by a single parameter, it is sufficient to track the message error probability.

On the BEC(ϵ), this message error rate can be defined as the probability $p_e^{(\ell)}$

that a message passed from the variable nodes to check nodes in iteration ℓ is 0. The message sent by a check node of degree i is in error (or 0) when at least one of its $i-1$ incoming messages is 0. Therefore, the degree-i check node message error probability $q_{e_i}^{(\ell)}$ (or the probability that the message is 0) at iteration ℓ is given by

$$q_{e_i}^{(\ell)} = 1 - (1 - p_e^{(\ell)})^{i-1}. \qquad (2.31)$$

For an irregular LDPC code having check node degree distribution $\{\rho_2, \rho_3, \ldots, \rho_{d_c}\}$, the total check node message error probability is then given by

$$q_e^{(\ell)} = \rho_2 \cdot q_{e_2}^{(\ell)} + \rho_3 \cdot q_{e_3}^{(\ell)} + \cdots + \rho_{d_c} \cdot q_{e_{d_c}}^{(\ell)} \qquad (2.32)$$

or equivalently by

$$q_e^{(\ell)} = 1 - \left(\rho_2 \cdot (1 - p_e^{(\ell)}) + \rho_3 \cdot (1 - p_e^{(\ell)})^2 + \cdots + \rho_{d_c} \cdot (1 - p_e^{(\ell)})^{d_c - 1}\right). \qquad (2.33)$$

Using the polynomial representation of the check node degree distribution $\rho(x)$, the above equation can be shortly written as

$$q_e^{(\ell)} = 1 - \rho(1 - p_e^{(\ell)}). \qquad (2.34)$$

The message sent by a variable node of degree i is in error when the variable node is erased and all the incoming check messages are 0. Therefore, the degree-i variable node message error probability is

$$p_{e_i}^{(\ell+1)} = \epsilon \cdot (q_e^{(\ell)})^{i-1}. \qquad (2.35)$$

For an irregular code having variable node degree distribution $\{\lambda_2, \lambda_3, \ldots, \lambda_{d_v}\}$, the total variable node message error probability is then given by

$$p_e^{(\ell+1)} = \epsilon \cdot \left(\lambda_2 \cdot (q_e^{(\ell)}) + \lambda_3 \cdot (q_e^{(\ell)})^2 + \cdots + \lambda_{d_v} \cdot (q_e^{(\ell)})^{d_v - 1}\right). \qquad (2.36)$$

Again, using the polynomial representation of the degree distribution $\lambda(x)$, the above equation can be shortly written as

$$p_e^{(\ell+1)} = \epsilon \cdot \lambda(q_e^{(\ell)}). \qquad (2.37)$$

The recursive analysis can be written for an irregular LDPC code having degree distributions $\lambda(x)$ and $\rho(x)$ using (2.34) and (2.37) as [44]

$$p_e^{(\ell+1)} = \epsilon \cdot \lambda(1 - \rho(1 - p_e^{(\ell)})). \qquad (2.38)$$

In a successful decoding, the probability of error $p_e^{(\ell)}$ should decrease iteration by iteration and get arbitrarily close to 0. Thus, for a successful decoding we should have

$$\epsilon \cdot \lambda(1-\rho(1-x)) < x \quad \forall x \in (0,\epsilon). \tag{2.39}$$

For the Gallager's decoding algorithm A and the BSC(ϵ), since the messages are 0 and 1, we track the probability $p_e^{(\ell)}$ that the variable node messages sent to the check nodes in iteration ℓ is 1. Using a similar procedure as for the BEC, and from [8,46], we get

$$p_e^{(\ell+1)} = (1-\epsilon)\cdot\lambda\left(\frac{1-\rho(1-2p_e^{(\ell)})}{2}\right) + \epsilon\cdot\left(1-\lambda\left(\frac{1+\rho(1-2p_e^{(\ell)})}{2}\right)\right), \tag{2.40}$$

and for successful decoding we should have

$$(1-\epsilon)\cdot\lambda\left(\frac{1-\rho(1-2x)}{2}\right) + \epsilon\cdot\left(1-\lambda\left(\frac{1+\rho(1-2x)}{2}\right)\right) < x \quad \forall x \in (0,\epsilon). \tag{2.41}$$

Gallager's algorithm B, could also be analyzed in a similar way. For details see [11].

Density evolution for general BIMS channels

The main idea of LDPC code analysis on the BEC and BSC can be extended to other BIMS channels and other decoding algorithms. Density evolution, first proposed by Richardson and Urbanke in 2001 [13], is the general asymptotic analysis method for LDPC codes. It tracks the evolution of the pdf of the extrinsic messages iteration by iteration.

The exact formulation of this method is given in [13,47]. Density evolution is computationally complex and its analytical formulation is not suitable for direct use. Thus numerical analysis is usually done by quantizing the message alphabets and using probability mass functions (pmfs) instead of pdfs. This technique is called *discrete density evolution* [12]. Here, we present some details of this technique for sum-product decoding.

At the first iteration, the decoder is initialized with the pmf of the channel messages. These messages are sent as the variable node messages to the check nodes. Then the incoming pmf is processed at the check nodes based on the check node update rule. Consider the quantizing function $\mathcal{Q}(w)$ as given by

$$\mathcal{Q}(w) = \begin{cases} \lfloor \frac{w}{\Delta} + \frac{1}{2} \rfloor \cdot \Delta, & w \geq \frac{\Delta}{2} \\ \lceil \frac{w}{\Delta} - \frac{1}{2} \rceil \cdot \Delta, & w \leq -\frac{\Delta}{2} \\ 0, & \text{otherwise} \end{cases}, \tag{2.42}$$

where Δ is the quantization interval. For a check nodes with two incoming pmfs p_a and p_b the output pmf is given by

$$p_c[k] = \sum_{(i,j):k\Delta=\mathcal{R}(i\Delta,j\Delta)} p_a[i]p_b[j], \tag{2.43}$$

where

$$\mathcal{R}(a,b) = \mathcal{Q}\left(2\tanh^{-1}\left(\tanh\frac{a}{2}\tanh\frac{b}{2}\right)\right). \tag{2.44}$$

These equations can be implemented using a look-up table. We combine these equations and use the notation $p_c = \mathcal{CHK}(p_a, p_b)$ for a check node operating on two pmf inputs, p_a and p_b. Then, for a check node of degree d_{c} the output rule can be written as

$$\mathrm{CHK}(m_1, m_2, \ldots, m_{d_{\mathrm{c}}-1}) = \mathrm{CHK}(m_1, \mathrm{CHK}(m_2, \ldots, m_{d_{\mathrm{c}}-1})), \tag{2.45}$$

where $\mathrm{CHK}(\cdot)$ is the check node update rule given by (2.25). This implies that the check node operation can be done pairwise. Denoting the variable node message pmf by p_v (which is the same for all messages) and the check node message pmf by p_u, p_u can be computed using

$$p_u = \mathcal{CHK}(p_v, \mathcal{CHK}(p_v, \ldots, \mathcal{CHK}(p_v, p_v), \ldots)). \tag{2.46}$$

At the next half iteration, the output of the check nodes along with the channel pmf is fed into the variable nodes . For a variable node with two incoming pmfs p_a and p_b, the output pmf is given by

$$p_c[k] = p_a[k] * p_b[k], \tag{2.47}$$

where $*$ denotes the discrete convolution. This can be easily implemented using fast-Fourier transform (FFT) techniques. We denote the variable node operation by $p_c = \mathcal{VAR}(p_a, p_b)$. Again for a variable node of degree d_{v}, the update rule can be written as

$$\mathrm{VAR}(m_0, m_1, m_2, \ldots, m_{d_{\mathrm{v}}-1}) = \mathrm{VAR}(m_0, \mathrm{VAR}(m_1, m_2, \ldots, m_{d_{\mathrm{v}}-1})), \tag{2.48}$$

where VAR$(\cdot)$ is the variable node update rule given by (2.26). Using similar reasoning, p_v is given by

$$p_v = \mathcal{VAR}(p_0, \mathcal{VAR}(p_u, \dots, \mathcal{VAR}(p_u, p_u), \dots)), \tag{2.49}$$

where p_0 denotes the intrinsic messages pmf. This completes one iteration. This process can be repeated for many iterations and in each iteration the pmf of the messages can be studied. If the all-zero codeword is assumed to be sent and 0 is mapped to $+1$ and 1 is mapped to -1 (as in BPSK modulation), then the negative tail of the pmf represents the message probability of error in each iteration. If this tail vanishes after some iterations, the decoding is successful.

In density evolution it is assumed that the factor graph is a tree which is not always true. Nevertheless, the concentration theorems of [13] show that when the code length grows, the average behavior of individual instances of the code ensemble and the noise concentrates around its expected behavior. This expected behavior, in turn converges to the behavior of the cycle-free case. Density evolution is a powerful tool for analyzing the asymptotic performance of iterative decoders (when the block length is large). It can also be applied to some other codes defined on graphs which use iterative decoding [48–50].

Due to the high complexity of the density evolution, some approximations to it has also been proposed in the literature. The most important ones are the Gaussian approximation [14] and the semi-Gaussian approximation [15]. In the Gaussian approximation, all the extrinsic messages are considered to have Gaussian distribution. In the semi-Gaussian approximation method, only the variable node messages are considered to be Gaussian. Although these methods are less accurate than the density evolution, their complexity is much less.

Decoding threshold

The decoding threshold of an LDPC code is defined as the worst channel condition for which the message error rate approaches zero as the number of iterative decoding iterations approaches infinity. This fact is proved in [13] that for channel conditions better than the decoding threshold, density evolution converges to arbitrary small message error rate and for channel conditions worse than the decoding threshold, message error rate remains larger than a constant no matter how many iterations performed. The decoding threshold depends on the degree distributions, the decoding algorithm used, and the channel type and is one of the most important properties of an LDPC code.

As an example, in the BEC(ϵ) and BSC(ϵ) with Gallager's decoding algorithm A, the decoding threshold is given by the maximum ϵ for which the inequalities in (2.39) and (2.41) hold, respectively. It is worth mentioning that since density evolution is not accurate for finite-length LDPC codes, there is a gap from performance to the threshold in practice. Our goal in Chapter 3 is to determine this gap and predict the performance of finite-length LDPC codes.

Analysis of the accuracy of discrete density evolution shows that if the messages are represented by 11 bits or more, the error in the decoding threshold is less than 0.001 dB [12].

Extrinsic information chart analysis

Extrinsic information transfer (EXIT) chart analysis is another method of analyzing the asymptotic behavior of iterative decoders. In EXIT charts, the idea is to track the evolution of a representative of the extrinsic messages densities. In other words, the decoder's behavior is analyzed based on evolution of a single parameter. Many parameters could be selected to represent the densities (usually this is a measure of the decoder's success). The most famous parameter for this purpose is the mutual information between the received bits and the extrinsic messages in each iteration [16]. Other parameters which have been considered in the literature are the signal-to-noise ratio (SNR) of the extrinsic messages [51, 52], the extrinsic messages error probability [15, 17], etc. Notice that when the pdf of the extrinsic messages can be fully represented by a single parameter (e.g. in BEC and BSC), EXIT chart analysis is exact and is the same as density evolution. In this thesis, we will use EXIT charts based on the error probability. Thus, when the term EXIT chart is used, we mean EXIT charts that track the message error probability unless otherwise stated.

In an EXIT chart, the error probability of the outgoing extrinsic messages of one iteration p_{out} is expressed as a function of the error probability of the incoming extrinsic messages p_{in} and the intrinsic messages p_0, i.e.,

$$p_{\mathsf{out}} = f(p_{\mathsf{in}}, p_0). \tag{2.50}$$

Then the EXIT chart is the plot of this function using p_{out}-p_{in} coordinates. The

EXIT chart function f also depends on the degree distributions. A sample EXIT chart is plotted in Fig. 2.9. The inverse function f^{-1} is also plotted to better visualize the behavior of the decoder in each iteration. The output error probability p_{out} of an iteration is transferred to the next iteration as a new p_{in} and again a new p_{out} is given and this process continues until convergence. Using EXIT charts it is possible to visualize and determine how many iterations is needed for convergence in the decoder. The more open the EXIT chart (i.e., the tunnel between f and f^{-1} is wide) the less number of iterations is needed. If the EXIT chart gets closed (i.e., the tunnel between f and f^{-1} gets closed), the error probability cannot get smaller than a certain value and the decoder does not converge. This is equivalent to the case that f intersects the 45-degree line. Thus, we define an open EXIT chart the one which is always below the 45-degree line and a closed EXIT chart the one which intersects the 45-degree line.

The decoding threshold can also be approximated by the EXIT chart analysis. It is defined as the worst channel condition p_0^* for which the EXIT chart is open, i.e.,

$$p_0^* = \arg\sup_{p_0}\{f(p_{\text{in}}, p_0) < p_{\text{in}}, \quad \forall p_{\text{in}} : 0 < p_{\text{in}} \leq p_0\}. \tag{2.51}$$

Since an EXIT chart can model the behavior of the decoder in a simple manner, it can be used to design good irregular LDPC codes with desired convergence behavior. In the literature, EXIT charts have been vastly used for LDPC code design (e.g., see [17, 18]). We will briefly discuss how EXIT charts can be used to design good LDPC codes in Section 2.6.

2.5.2 Finite-length analysis methods

Density evolution is an accurate method for the asymptotic analysis of LDPC codes and in general iterative decoding. In the asymptotic analysis methods, the code's block length is considered to be very large or infinite. The accuracy of density evolution and other asymptotic analysis methods degrades when finite-length codes are used. Even for block lengths several tens of thousands large, the performance prediction of density evolution is very poor. For example, on a BIAWGN channel, density evolution calculates a threshold of 1.1015 dB for (3,6)-regular LDPC codes. However, the finite-length performance could be far from the threshold. Fig. 2.10 compares the performance of (3,6)-regular LDPC codes with different block lengths

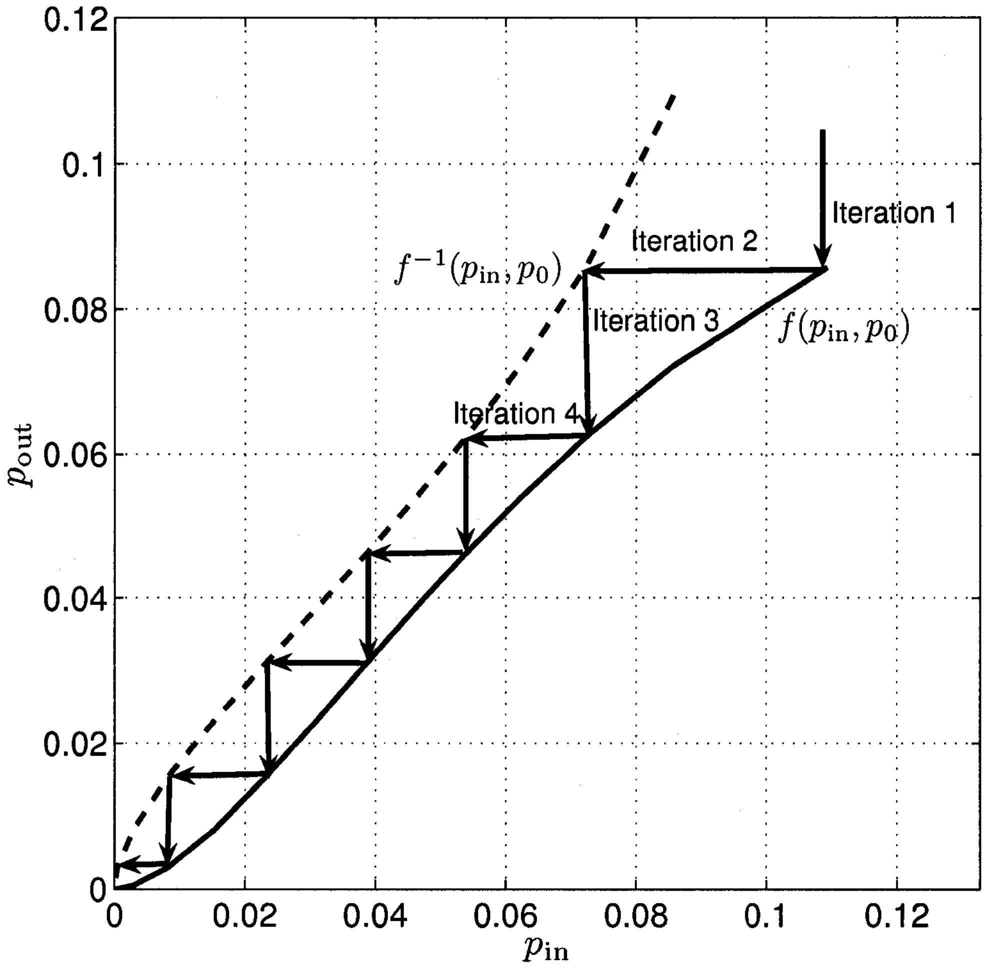


Figure 2.9: An EXIT chart based on message error probability. This EXIT chart corresponds to a (3,6)-regular code on the BIAWGN channel decoded by sum-product. The functions $f(p_{\text{in}}, p_{\text{O}})$ and its inverse are plotted. As seen, it is possible to visualize the decoding behavior and the number of required iterations.

to the threshold. It is seen that the performance gap to the threshold can be significant. In practice, finite-length codes are used. Thus, there is a need for analysis methods capable of predicting accurate performance curves for finite-length LDPC codes.

One of the reasons for the inaccuracy of the asymptotic methods is that for finite-length or small graphs, the tree assumption is valid only for a few iterations. In most cases, these few number of iterations are not enough to reduce the error induced by the channel in the graph. Based on this concept, a combinatorial analysis has been proposed for the finite-length LDPC codes on the BEC in [23]. This analysis is based on considering weaknesses in the code's graph called *stopping sets.*

A stopping set $\mathcal{S}$ is defined as a subset of variable nodes, such that all neighbors of $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice. In other words, a stopping set, is a subset of variable nodes such that no neighboring check node has degree one in the subgraph induced by this subset. The size of $\mathcal{S}$ is the number of its variable nodes. In Fig. 2.11, stopping set sizes of 2, 4 are depicted. Also, it is shown in Fig. 2.12 that how a stopping set may be connected to other parts of the graph.

A stopping set has the property that if all its variable nodes are erased, iterative decoding cannot recover any of these erasures. It can be proved that the set of erasure $\mathcal{E}$ which remains when the iterative decoder stops is equal to the maximal stopping set of $\mathcal{E}$ [23]. A block error occurs when all the variable nodes in a stopping set are erased. Thus, to find the block error probability, it is sufficient to find the probability that a random erasure subset of variable nodes of a randomly chosen element of the code ensemble, contains a nonempty stopping set [23]. Since the variable nodes are erased by the channel randomly, we find the probability that a nonempty stopping set is hit by the randomly erased bits for a random code graph from the code ensemble $\mathcal{C}^n(\lambda(x), \rho(x))$. This can be done by a combinatorial analysis which is extremely complex for block lengths larger than a few hundred bits and even for short block lengths only simple ensembles can be analyzed [23]. This approach has been extended to irregular ensembles in [53].

The performance curves of finite-length LDPC codes can usually be divided into two regions; the *waterfall* region and the *error floor* region. In the waterfall region, the error rate drops significantly with improving channel quality. This region corresponds to low SNR regions. In the error floor region, however, the error rate

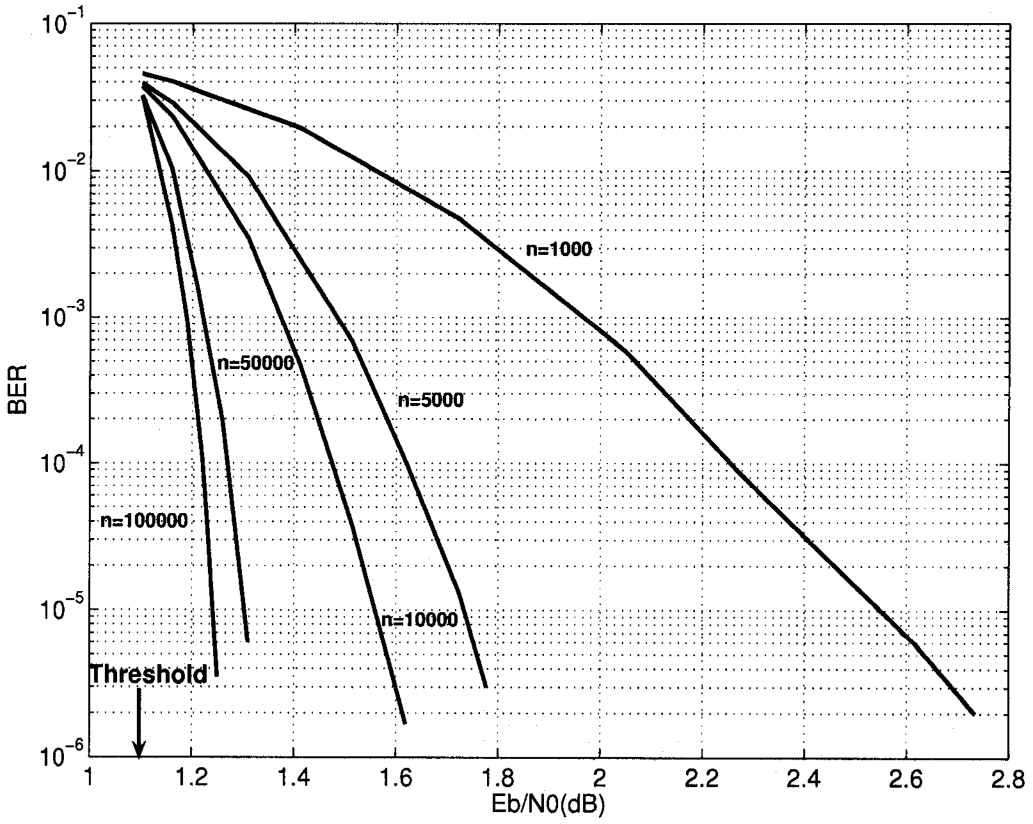


Figure 2.10: Bit error rate of a (3,6)-regular LDPC code on the BIAWGN channel with different block lengths n in the waterfall region. There gap to the threshold gets larger with decreasing n.

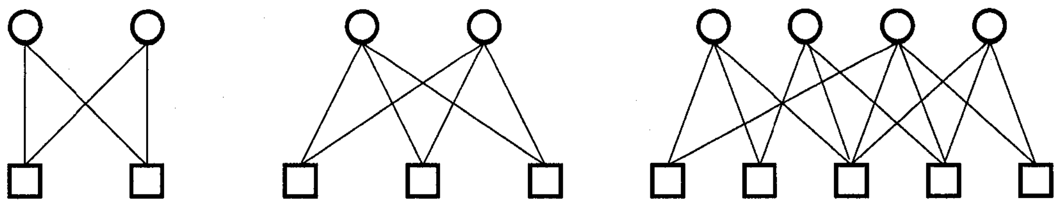


Figure 2.11: Examples of stopping sets of size 2 (left and middle) and 4 (right).

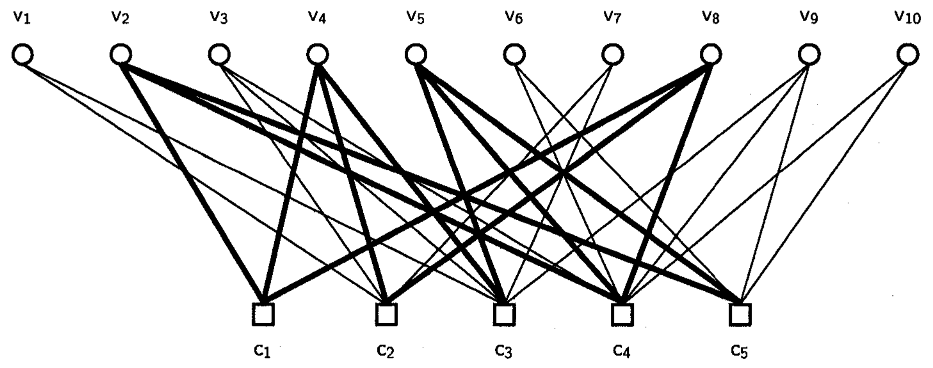


Figure 2.12: The set $\mathcal{S} = \{v_2, v_4, v_5, v_8\}$ is a stopping set of size 4. The bold lines show the subgraph induced by $\mathcal{S}$.

does not drop significantly with improving channel quality and the performance curve flattens out. This region corresponds to high SNR regions. Please see Fig. 2.13 as an example. In the error floor region, the performance degradation is due to the weaknesses in the graph such as stopping sets, trapping sets, etc.

It has been observed in [24] that the average performance of finite-length LDPC codes follow a *scaling law* in the waterfall region. This scaling law has been proved when the transmission takes place on the BEC and is conjectured to be true on the BSC and BIAWGN channels too. In fact, the following refined scaling law has been proved for the BEC(ϵ):

$$\mathrm{E}[\mathrm{P_B}(\mathcal{G},\epsilon)] = Q\left(\frac{\sqrt{n}\left(\epsilon^* - \beta n^{-\frac{2}{3}} - \epsilon\right)}{\alpha}\right) + O(n^{-\frac{1}{3}}), \tag{2.52}$$

where $\mathrm{P_B}$ is the block error probability, n is the block length, $\mathcal{G}$ is a random element of the code ensemble, ϵ^* is the decoding threshold, and

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{t^2}{2}} \mathrm{d}t. \tag{2.53}$$

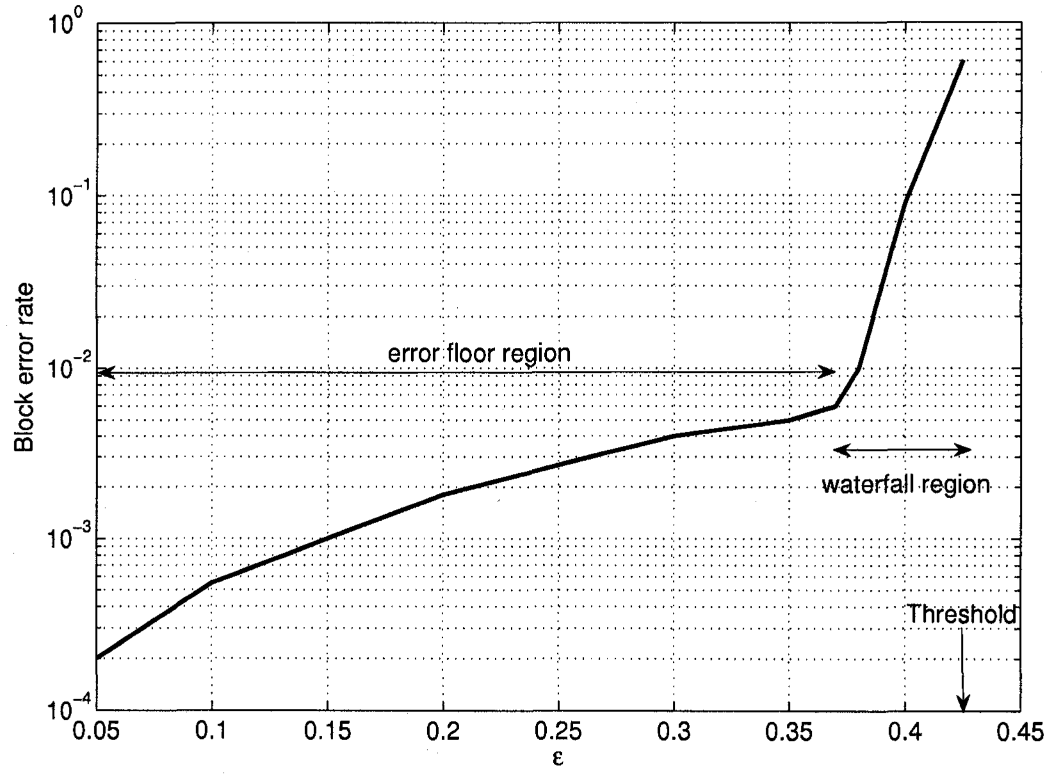


Figure 2.13: Performance curve of a $\mathcal{C}^{1024}(x^2, x^5)$ on the BEC(ϵ). The waterfall and error floor regions are shown in the figure.

The parameter α is the variance coefficient and is computed by a procedure called *covariance evolution*. Also, the term $\beta n^{-\frac{2}{3}}$ represents a shift of the threshold and can be computed by the method described in [24].

The procedure of calculating α and β could be complex especially for irregular ensembles. Also, it is only applicable to the BEC. For the other channels, the parameters are fitted to the data. Thus, there is still a need for a general method capable of analyzing the finite-length behavior of LDPC codes on general BIMS channels. In Chapter 3, we tackle this problem and we propose a simplified version of the scaling law which applies to all BIMS channels.

2.6 LDPC Codes: Design Methods

The goal of designing good LDPC codes is to find good and optimized degree distributions which have a desired performance on a given channel subject to some constraints and conditions. Different measures of performance and optimality and various constraints and conditions could be considered. For example, the degree distributions can be optimized to have the highest decoding threshold while having a minimum code rate [47], or to have the highest rate for a fixed decoding threshold [17], or to have the lowest decoding complexity for a fixed rate and threshold [17,54]. The LDPC code design process is usually done by numerical optimization methods.

When designing LDPC codes, we also need an analysis method capable of measuring the performance. We stated that the EXIT chart method is simple and also gives insight into the convergence behavior. Thus, the EXIT chart method can also be used for the design process. The code design optimization methods become simpler if the EXIT chart is used. In fact, the code design based on maximizing the code rate is transformed to a linear program using EXIT chart [17].

Now, we briefly explain the code design methods that we will use in this thesis. In our methods, we fix the check node degree distribution and optimize the variable degree distribution. This is a reasonable assumption according to the results of [12,47].

One code design approach can be defined as maximizing the rate of the code given a minimum decoding threshold. This optimization can be easily done by linear programming. We will briefly discuss the details here.

Using the sum of probabilities rule, the average output error probability of the variable nodes can be written as

$$p_e^{(\ell)} = \sum_{i=2}^{d_v-1} \lambda_i \cdot p_{e,i}^{(\ell)}, \tag{2.54}$$

where $p_{e,i}^{(\ell)}$ is the message error probability at the output of degree i variable nodes at the ℓth iteration. Based on (2.54), the EXIT chart can be written as

$$f(p_{\text{in}}, p_0) = \sum_{i=2}^{d_v-1} \lambda_i \cdot f_i(p_{\text{in}}, p_0), \tag{2.55}$$

which shows that the EXIT chart of the code is in fact a linear combination of EXIT charts of codes with regular variable node distribution called *elementary* EXIT charts $f_i(p_{\text{in}}, p_0)$. This observation states that the optimization process is equivalent to finding a linear combination of the elementary EXIT charts which maximizes the code rate and is below the 45-degree line given the channel conditions. Using (2.23) and (2.55), the process of maximizing the code rate can be expressed as a linear program by

$$\begin{aligned} \text{maximize} \quad & \sum_i \frac{\lambda_i}{i} \\ \text{subject to} \quad & \lambda_i \geq 0 \\ & \sum_i \lambda_i = 1 \\ & \sum_i \lambda_i f_i(p_{\mathsf{in}}, p_0) < p_{\mathsf{in}}, \quad \forall 0 < p_{\mathsf{in}} \leq p_0 \end{aligned} \tag{2.56}$$

where p_0 is the error probability of the intrinsic (channel) messages which depends on the channel condition. The elementary EXIT charts can be found using the density evolution. Notice that since $\rho(x)$ is fixed, the elementary EXIT charts only depend on λ_i's. Therefore, in each iteration of the optimization routine, λ_i's should undergo a small change and new elementary EXIT charts should be calculated for the next iteration.

Another code design approach is to maximize the decoding threshold of the code given a minimum code rate R_0. This approach is complex in general and is usually done by search-based methods. The idea is to search for degree distributions which satisfy the minimum code rate constraint and their corresponding code can be decoded in the worst channel condition. One of the best search methods to use is *differential evolution* [55] which is in part a hill climbing algorithm and in part a genetic algorithm. Differential evolution has previously been used in the literature for this purpose [19, 47, 56]. In Chapter 4, however, we use the code rate maximization procedure to design threshold maximized codes. The details are presented in Chapter 4.

Chapter 3

An Efficient Analysis of Finite-length LDPC Codes

3.1 Introduction

We stated in Chapter 2 that in most of the LDPC code design and analysis methods, it is assumed that the code's block length is infinite. The performance of LDPC codes and the accuracy of these asymptotic methods depend on the block length and degrade as the block length gets smaller. In practice, where finite-length codes are used, the infinite-length analysis can be quite inaccurate.

It is fair to say that the asymptotic behavior of LDPC codes, when the block length tends to infinity, is well understood by now. Unfortunately, conventional methods such as density evolution or EXIT chart analysis, which are used for asymptotic analysis, may not be accurate enough for finite-length codes. Thus, there has been a growing interest in the analysis of finite-length LDPC codes. Finite-length analysis of other graphical codes has also been a subject of interest. For example, in [57] a lower bound on the bit error rate performance of finite-length turbo codes is obtained.

For LDPC coding on the BEC, a combinatorial analysis has been presented in [23] for regular finite-length ensembles and they have been generalized in [53] for irregular LDPC code ensembles. Also, an upper bound on the error floor of finite-length LDPC codes is given in [58]. These methods give the average performance (block and bit erasure probability) based on the analysis of stopping sets using a combinatorial recursive method. In particular, they calculate the probability that a random subset of erased variable nodes of a randomly chosen element of the code ensemble contains a nonempty stopping set. Unfortunately, these combinatorial recursions are computationally complex and even for practical block lengths only very simple ensembles can be analyzed.

In a recent work [24], through a detailed analysis, it is shown that the performance curves of LDPC codes in the waterfall region, follow a scaling law. This scaling law is motivated by the existing results in statistical physics and is proved for iteratively decoded LDPC ensembles when transmission takes place over the BEC. It is stated that empirically such a scaling seems to be applicable to other channels such as the BSC and BIAWGN channel, and that any function of the channel parameter can be used for stating the scaling law. To hold consistency with the BEC, the capacity of the channel is suggested to be used. The scaling law consists of two extra parameters α and β which depend on the code ensemble and should be calculated. The parameter α represents the variance coefficient and is computable by a procedure called *covariance evolution* [24]. The parameter β represents the coefficient of the shift of the threshold for finite lengths. Although using this method the performance can be predicted accurately, the procedure of calculating α and β, which is only known for the BEC, could be cumbersome especially for irregular codes [24]. Also, this procedure cannot be used for other channels such as the BSC, BIAWGN or fading channels. For these cases, the parameters are simply fitted to the data.

Scaling law is a consequence of the threshold behavior of LDPC decoders for infinite block length. That is, successful decoding is possible if and only if the channel quality is better than a threshold. When the code length is finite, during the transmission of a single codeword, the average channel behavior may not be observed. For example, in a BSC with crossover probability ϵ, each transmitted bit can be flipped by the channel with probability ϵ, but after a finite number of channel uses, say n, the actual number of flipped bits may be more than or less than ϵn. The decoder is influenced by the actual number of flipped bits or the *observed channel quality* during the transmission of a codeword, not the average channel quality.

This view can be used to form a simplified and efficient version of the scaling law which does not have the extra parameters of α and β. To be more specific, since the realization of noise varies from one codeword to another, one can find the probability that the channel—observed during one codeword—is worse than the decoding threshold of the code. This way, assuming that an observed channel worse than the decoding threshold results in a decoding failure, an estimate of the block error performance can be obtained.

This method, however, is not directly applicable to all BIMS channel models. In the BSC (or BEC) the variation of the observed channel is represented by a single parameter. Thus, the observed channel can directly be compared with the decoding threshold of the code and prediction about convergence or failure of the decoder can be made. However, in a Gaussian channel, for example, the actual distribution of noise samples observed over a finite block length is not purely Gaussian. Thus, comparison with the decoding threshold of the code is not directly possible.

In this chapter, we first study the variations of BIMS channels around their average behavior during the transmission of a finite-length codeword. In particular, for BIMS channels whose variations cannot be expressed by a single parameter, we try to model these variations with a single parameter. This way, a direct comparison with decoding threshold and thus an efficient performance analysis of finite-length LDPC codes is made possible.

Using different measures for modeling the observed channel results in different predictions of the code's performance. Thus, we focus on the study, comparison and analysis of different measures such as observed channel capacity or observed bit error rate. We model these channel measures as random variables and find the pdf of these random variables as a function of the block length. Using these pdfs, we effectively estimate the block error rate of the code.

Our approach can provide a very good estimate of the code's block error rate performance (typically within 0.1 dB) in the waterfall region when the code length is more than a few thousand bits (typically more than 2500 bits). A method for finding an accurate estimation of the bit error rate performance from the block error rate estimation is also provided.

Since in our analysis the effects of cycles and stopping sets are ignored, the results become more accurate as the code length gets larger. Moreover, since error floor behavior of LDPC codes is mainly due to the cycles and stopping sets [23, 58], the error floor behavior is not predicted by this analysis. Our method uses the decoding threshold of the code and the code length n to provide an accurate estimate of the performance in the waterfall region and it is applicable to both regular and irregular codes under different decoding algorithms. Also, no fitting of parameters is required.

The rest of this chapter proceeds as follows. In Section 3.2, we study the channel variations around its expected behavior for finite-length codes. The code's performance analysis method is presented in Section 3.3. Simulation results for regular and irregular codes are presented in Section 3.4, where we discuss the results and provide a guideline for designing irregular LDPC codes of finite length. Finally, the chapter is concluded in Section 3.4.

3.2 Channel Variations

3.2.1 One-dimensional channels

In a BEC with erasure rate ϵ or a BSC with crossover probability of ϵ (BEC(ϵ) or BSC(ϵ)), an *observed* bit erasure rate or *observed* bit error probability P_{obs} can be defined after the transmission of any codeword. We have

$$P_{\mathsf{obs}} = \frac{n_\epsilon}{n}, \tag{3.1}$$

where n_ϵ is the number of erasures or errors in the codeword and n is the codeword length. Only when $n \to \infty$, P_{obs} is a constant equal to ϵ. For finite n, however, P_{obs} is no longer a fixed value and varies from one codeword transmission to another. Thus it can be thought as a random variable which has a *scaled* binomial distribution around ϵ. This distribution depends on n and ϵ and its probability mass function (pmf) is given by

$$f_{P_{\mathsf{obs}}}(x) = \binom{n}{nx} \epsilon^{nx}(1-\epsilon)^{n-nx}, \tag{3.2}$$

which is defined wherever nx is an integer and $0 \leq nx \leq n$. If n is large enough, we can approximate this pmf by a continuous Gaussian pdf $\mathcal{N}(\mu_{P_{\mathsf{obs}}}, \sigma^2_{P_{\mathsf{obs}}})$ with mean $\mu_{P_{\mathsf{obs}}} = \epsilon$ and variance $\sigma^2_{P_{\mathsf{obs}}} = \epsilon(1-\epsilon)/n$. Notice that even for short block lengths (a few hundred bits), this approximation is good and that this pdf has negligible values for x outside $[0, 1]$. When n increases, the variations around the mean get smaller and for infinite length the distribution is an impulse at ϵ. As an example, the histogram of P_{obs} is plotted in Fig. 3.1 on the BEC(0.1) when $n = 2000$. It is seen that this histogram can be approximated by a Gaussian pdf.

It is reasonable to assume that the decoder is mainly influenced by the observed erasure rate or error rate of each codeword and not the ϵ itself. Thus, we are interested to model the channel with P_{obs} instead of the fixed value ϵ. This time-varying

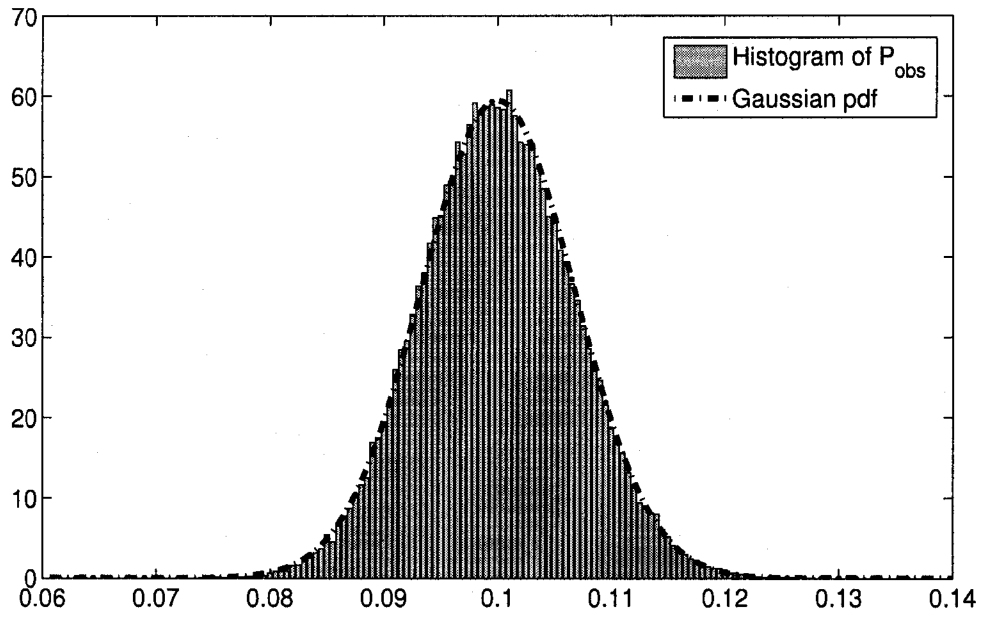


Figure 3.1: Histogram of P_{obs} on the BEC(0.1) when $n = 2000$ based on the simulation. The number of transmitted blocks is 10^5. This histogram is close to a Gaussian pdf with mean 0.1 and variance 4.5×10^{-5}.

interpretation of the channel will be the basis of our block error rate performance analysis.

Notice that one can also define a time-varying *observed* channel capacity. This means that we can assign a probability distribution to the observed channel capacity and treat it as a random variable C_{obs}. For the BEC, we have $C_{\text{obs}} = 1 - P_{\text{obs}}$ and for the BSC we have $C_{\text{obs}} = 1 - h(P_{\text{obs}})$ where $h(\cdot)$ is the binary entropy function given in (2.6).

In the case of the BEC or BSC, a single parameter (P_{obs} or C_{obs}) uniquely defines the observed channel. We call such channels one-dimensional channels.

3.2.2 Multi-dimensional channels

When the observed channel cannot be described with a single parameter, we say the channel is multi-dimensional. For example in a Gaussian channel, the noise has a continuous pdf. However, when the code length is finite, we have a finite number of samples of this continuous pdf at the channel output. The decoder is influenced by these samples, whose pmf varies from one codeword to another and cannot be described by a single parameter.

In this section, we study the variations of multi-dimensional channels and we model these variations by a single parameter. This single parameter can be the observed channel capacity, observed bit error rate, observed noise power, etc. We mainly focus on the observed bit error rate and the observed channel capacity, but a brief discussion on some other channel parameters has been included in Section 3.4. We also limit our discussions to the BIAWGN channel and uncorrelated Rayleigh fading (URF) channel, but a similar approach can be taken for other multi-dimensional channels.

The BIAWGN channel

For the BIAWGN(σ) channel, we have $y = x + z$, where $x \in \{-1, 1\}$ is the input, $z \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian noise, and y is the output of the channel.

Analysis of LDPC codes is usually based on the distribution of the channel log-likelihood ratio (LLR) when the all-zero codeword ($x = +1$) is transmitted. The channel LLR L is defined as

$$L = \log \frac{\mathrm{P}(x = +1|y)}{\mathrm{P}(x = -1|y)}. \tag{3.3}$$

For the BIAWGN(σ), LLRs have a consistent Gaussian distribution [47] with mean $\mu = 2/\sigma^2$ and variance $4/\sigma^2$. For a proof of this property of LLRs on a Gaussian channel and other properties of LLR refer to Appendix A. There is an error in detection when L is negative. For finite n, in the transmission of each codeword, instead of having a pure Gaussian LLR distribution, we have n samples of this distribution to which we can assign P_{obs} and C_{obs}. Clearly, P_{obs} and C_{obs} are two random variables. Our goal is to find the pdf of these random variables.

To calculate the pdf of P_{obs}, consider taking n samples from the consistent Gaussian LLR distribution. Each sample or bit is in error when the LLR is negative. So, the probability of error of each sample is

$$p_0 = Q\left(\frac{1}{\sigma}\right), \tag{3.4}$$

where $Q(\cdot)$ is the Q-function given by (2.53).

Therefore P_{obs}, i.e., the number of all errors divided by n, has the following pmf

$$f_{P_{\mathsf{obs}}}(x) = \binom{n}{nx} p_0^{nx}(1 - p_0)^{n-nx}, \tag{3.5}$$

which is defined on x such that nx is an integer and $0 \leq nx \leq n$. Again, for large n, the pmf can be approximated by a Gaussian pdf $\mathcal{N}(p_0, p_0(1-p_0)/n)$ which has negligible values for x outside $[0,1]$.

Now, we analyze the observed capacity. The capacity of a BIMS channel $\mathfrak{C}$ can be given via the pdf $f_L(x)$ of the LLR by [49,59]

$$C(\mathfrak{C}) = 1 - \int_{-\infty}^{\infty} \log_2(1+e^{-x}) f_L(x) \mathrm{d}x = 1 - \mathrm{E}_L[\log_2(1+e^{-L})]. \tag{3.6}$$

For a proof of this equation refer to Appendix A. For a BIAWGN(σ) channel (3.6) can be written as

$$C(\mathrm{BIAWGN}(\sigma)) = 1 - \frac{\sigma}{\sqrt{8\pi}} \int_{-\infty}^{\infty} \log_2(1+e^{-x}) e^{\frac{-\left(x-\frac{2}{\sigma^2}\right)^2}{8/\sigma^2}} dx. \tag{3.7}$$

As discussed before, for finite n we have n samples $X_1, X_2, \ldots, X_n$ of the channel LLR distribution. Thus, an observed channel capacity C_{obs} can be defined according to (3.6) and using the sample average instead of the expected value, i.e.,

$$C_{\mathsf{obs}} = 1 - \frac{1}{n} \sum_{i=1}^{n} \log_2\left(1+e^{-X_i}\right). \tag{3.8}$$

For the BIAWGN(σ) channel we have $X_i \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ for $i = 1, 2, \ldots, n$. Now define $Y_i = \log_2\left(1+e^{-X_i}\right)$. It is easy to verify that the distribution of Y_i is

$$f_{Y_i}(y) = \frac{\sigma \ln 2}{2\sqrt{2\pi}} \frac{2^y}{2^y - 1} e^{-\frac{(-\ln(2^y-1)-\mu)^2}{8/\sigma^2}} \quad \text{for} \quad y > 0. \tag{3.9}$$

All Y_i are independent and identically distributed (i.i.d.) since all X_i are i.i.d. Therefore, from (3.8) and the Central Limit theorem we find that if n is large enough (even for a short LDPC code n is large enough), C_{obs} has a Gaussian distribution $\mathcal{N}(\mu_{C_{\mathsf{obs}}}, \sigma^2_{C_{\mathsf{obs}}})$. To define this distribution, we need its mean and variance. The mean and variance of C_{obs} can be computed using the mean and variance of Y_i by

$$\mu_{C_{\mathsf{obs}}} = \mathrm{E}[C_{\mathsf{obs}}] = 1 - \mathrm{E}[Y_i], \tag{3.10}$$

and

$$\sigma^2_{C_{\mathsf{obs}}} = \mathrm{Var}[C_{\mathsf{obs}}] = \mathrm{Var}[Y_i]/n. \tag{3.11}$$

So, the main task is to find the mean and variance of Y_i which, using the distribution of X_i, can be written as

$$\mathrm{E}[Y_i] = \frac{\sigma}{\sqrt{8\pi}} \int_{-\infty}^{\infty} \log_2(1+e^{-x}) e^{\frac{-\left(x-\frac{2}{\sigma^2}\right)^2}{8/\sigma^2}} dx \tag{3.12}$$

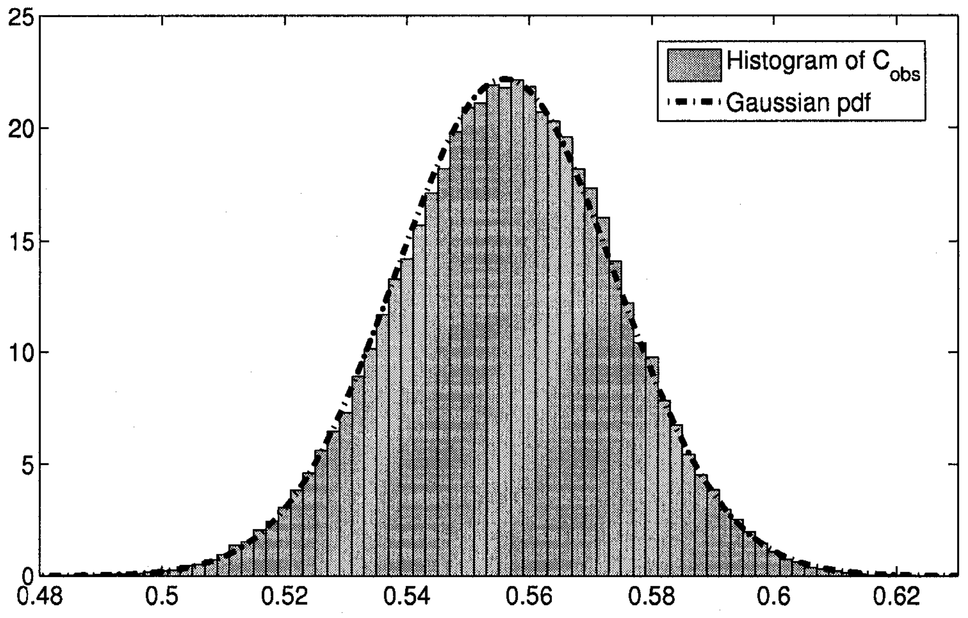


Figure 3.2: Histogram of C_{obs} on the BIAWGN(0.9) channel with $n = 2000$ based on the simulation. The number of transmitted blocks is 10^5. This histogram is close to a Gaussian pdf with mean 0.556 and standard deviation 0.018.

$$\mathrm{Var}[Y_i] = \frac{\sigma}{\sqrt{8\pi}} \int_{-\infty}^{\infty} (\log_2(1+e^{-x}))^2 e^{\frac{-\left(x-\frac{2}{\sigma^2}\right)^2}{8/\sigma^2}} \mathrm{d}x - (\mathrm{E}[Y_i])^2. \tag{3.13}$$

The above integrals cannot be solved analytically and we need to compute them numerically. For example, for $\sigma = 0.9$ and $n = 2000$ we have $\mu_{C_{\text{obs}}} \simeq 0.556$ and $\sigma_{C_{\text{obs}}} \simeq 0.018$. The histogram of C_{obs} for this case is plotted in Fig. 3.2 based on simulation. It can be seen that this histogram can be approximated by a Gaussian pdf.

The mean of C_{obs} is equal to the capacity of the channel when the block length is infinite. Notice that $\mu_{C_{\text{obs}}}$ is only a function of the noise power and $\sigma^2_{C_{\text{obs}}}$ is a function of both the noise power and the block length. As the block length increases, the variance decreases.

Uncorrelated Rayleigh fading channel

For the case of URF(σ) channel, the same approach can be applied with some modifications. In this case, we have $y = r \cdot x + z$ where $x \in \{-1, 1\}$, $r \geq 0$ is the channel fading gain with a normalized Rayleigh distribution $f_R(r) = 2re^{-r^2}$,

$z \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian noise, and y is the output of the channel. We assume that the receiver knows r as a perfect side information (SI). Here, the channel LLR is calculated as

$$L = \log \frac{\mathrm{P}(x = +1|y, r)}{\mathrm{P}(x = -1|y, r)}. \tag{3.14}$$

Then, the pdf of L, assuming that the all-zero codeword ($x = +1$) transmitted, is given by [19] as

$$f_L(x) = \frac{\sigma}{\sqrt{2\pi}} \exp\left(\frac{-x(\sqrt{2\sigma^2+1}-1)}{2}\right) \int_0^\infty \exp\left(-\frac{\left(\frac{\sigma^2}{2r}x - r\sqrt{2\sigma^2+1}\right)^2}{2r}\right) \mathrm{d}r. \tag{3.15}$$

To calculate the pdfs of P_{obs} and C_{obs}, we consider taking n samples from this LLR distribution. Following a similar approach to the BIAWGN channel, the pdf of P_{obs} is given by (3.5) with

$$p_0 = \int_{-\infty}^{0} f_L(x) \mathrm{d}x. \tag{3.16}$$

Also, C_{obs} is given by (3.8) and it has a Gaussian distribution. However, calculating $\mu_{C_{\text{obs}}}$ and $\sigma^2_{C_{\text{obs}}}$ is slightly different since the pdf of X_i is not Gaussian and is given by (3.15). Thus we write

$$\mu_{C_{\text{obs}}} = 1 - \int_{-\infty}^{\infty} \log_2(1 + e^{-x}) f_L(x) \mathrm{d}x, \tag{3.17}$$

and

$$\sigma^2_{C_{\text{obs}}} = \frac{1}{n}\left(\int_{-\infty}^{\infty} (\log_2(1 + e^{-x}))^2 f_L(x) \mathrm{d}x - (1 - \mu_{C_{\text{obs}}})^2\right). \tag{3.18}$$

Now that we have interpreted the channel behavior and characterized two important parameters of the channel, we will use them to analyze the performance of finite-length LDPC codes.

3.3 Measuring Block and Bit Error Rate

3.3.1 Block error probability

In this section we present a method to predict the block error probability. We discuss the method for multi-dimensional channels. The same method applies to one-dimensional channels.

Following the discussion about the channel variations we model a BIMS channel $\mathfrak{C}(\sigma)$ parameterized with σ by either the random variable P_{obs} or C_{obs} with pdf $f_{P_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)$ and $f_{C_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)$ respectively. Also, the cumulative density functions (cdf) are given by $\mathrm{P}(P_{\mathsf{obs}} \leq x) = F_{P_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)$ and $\mathrm{P}(C_{\mathsf{obs}} \leq x) = F_{C_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)$. Notice that the pdf and the cdf are parameterized by the channel, its parameters and the block length of the code.

To calculate the average block error probability of an ensemble of LDPC codes we calculate the probability that the observed channel behaves worse than the code's decoding threshold. We can use either P_{obs} or C_{obs} for this purpose. For one-dimensional channels, it does not matter what parameter is used to model the channel variations. For multi-dimensional channels, since the variations cannot be fully modeled with a single parameter, the choice of parameter can affect the results.

Notice that this method ignores the effect of cycles, stopping sets and other weaknesses in the factor graph of the code. In other words, it assumes that the factor graph of the code is a tree and the code is long enough for density evolution purposes. However, the effect of atypical channel behavior is not ignored.

The procedure is as follows: First, calculate the probability of error (capacity) of a channel whose parameter is at the decoding threshold of the code σ^* assuming $n \to \infty$, and call it p_{th} (c_{th}). Then, for each σ and finite n compute $f_{P_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)$ ($f_{C_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)$). Using this distribution calculate the probability that $P_{\mathsf{obs}} > p_{\mathsf{th}}$ ($C_{\mathsf{obs}} < c_{\mathsf{th}}$). Therefore, we can write

$$\mathrm{P_B}(\mathfrak{C}(\sigma), n, \lambda, \rho) = 1 - F_{P_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, p_{\mathsf{th}}) = \int_{p_{\mathsf{th}}}^{1} f_{P_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)\mathrm{d}x, \tag{3.19}$$

and

$$\mathrm{P_B}(\mathfrak{C}(\sigma), n, \lambda, \rho) = F_{C_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, c_{\mathsf{th}}) = \int_{0}^{c_{\mathsf{th}}} f_{C_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x)\mathrm{d}x, \tag{3.20}$$

where $\mathrm{P_B}$ is the average block error probability of an ensemble of LDPC codes with length n and degree distribution pair λ and ρ (simplified notations for $\lambda(x)$ and $\rho(x)$ defined in Section 2.3). These give us two measures for the performance. The capacity method is more fundamental and seems to be a more meaningful choice, but as we will see in the next section, in most cases P_{obs} gives more accurate results. For the BIAWGN and URF channels, from (3.5), we have

$$f_{P_{\mathsf{obs}}}(\mathfrak{C}(\sigma), n, x) \simeq \frac{1}{\sqrt{2\pi}\sigma_{P_{\mathsf{obs}}}} e^{-\frac{(x-\mu_{P_{\mathsf{obs}}})^2}{2\sigma^2_{P_{\mathsf{obs}}}}}, \tag{3.21}$$

where $\mu_{P_{\text{obs}}}$ is given by (3.4) for the BIAWGN(σ) channel, and by (3.16) for the URF(σ) channel. For both channels we have

$$\sigma^2_{P_{\text{obs}}} = \frac{p_0(1-p_0)}{n}. \tag{3.22}$$

Also, for C_{obs} we have

$$f_{C_{\text{obs}}}(\mathfrak{C}(\sigma), n, x) \simeq \frac{1}{\sqrt{2\pi}\sigma_{C_{\text{obs}}}} e^{-\frac{(x-\mu_{C_{\text{obs}}})^2}{2\sigma^2_{C_{\text{obs}}}}}, \tag{3.23}$$

where for the BIAWGN(σ) channel, $\mu_{C_{\text{obs}}}$ and $\sigma^2_{C_{\text{obs}}}$ are derived according to (3.10) and (3.11) and by (3.17) and (3.18) for the URF(σ) channel. In both (3.21) and (3.23) we have $0 \leq x \leq 1$. Therefore, we can write

$$\mathrm{P_B}(\mathfrak{C}(\sigma), n, \lambda, \rho) = Q\left(\frac{\mu_{C_{\text{obs}}} - c_{\text{th}}}{\sigma_{C_{\text{obs}}}}\right), \tag{3.24}$$

and

$$\mathrm{P_B}(\mathfrak{C}(\sigma), n, \lambda, \rho) = Q\left(\frac{p_{\text{th}} - \mu_{P_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}\right). \tag{3.25}$$

In Fig. 3.3, the proposed block error probability prediction method is visualized on the BIAWGN channel using the pdfs of P_{obs} and C_{obs}.

Extending the results to one-dimensional channels is straightforward. In the BEC(ϵ) and BSC(ϵ) we only need to consider P_{obs} since C_{obs} gives the same results. Therefore, we change σ to ϵ and p_{th} to ϵ^* in (3.19). Here, ϵ^* is the decoding threshold of the code. We write

$$f_{P_{\text{obs}}}(\mathrm{BEC}(\epsilon), n, x) \quad = \quad f_{P_{\text{obs}}}(\mathrm{BSC}(\epsilon), n, x) \simeq \frac{1}{\sqrt{2\pi}\sigma_{P_{\text{obs}}}} e^{-\frac{(x-\mu_{P_{\text{obs}}})^2}{2\sigma^2_{P_{\text{obs}}}}}, \tag{3.26}$$

where $\mu_{P_{\text{obs}}} = \epsilon$ and $\sigma^2_{P_{\text{obs}}} = \epsilon(1-\epsilon)/n$ and $0 \leq x \leq 1$.

This method of predicting the block error probability is applicable to different decoding algorithms (e.g. sum-product, min-sum, Gallager A, etc.). In fact, the effect of the decoding algorithm is only seen on the decoding threshold of the code. Each decoding algorithms has a corresponding decoding threshold. Therefore, it is possible to calculate p_{th} and c_{th} for each algorithm. To predict the block error probability, the corresponding p_{th} and c_{th} are used in our method.

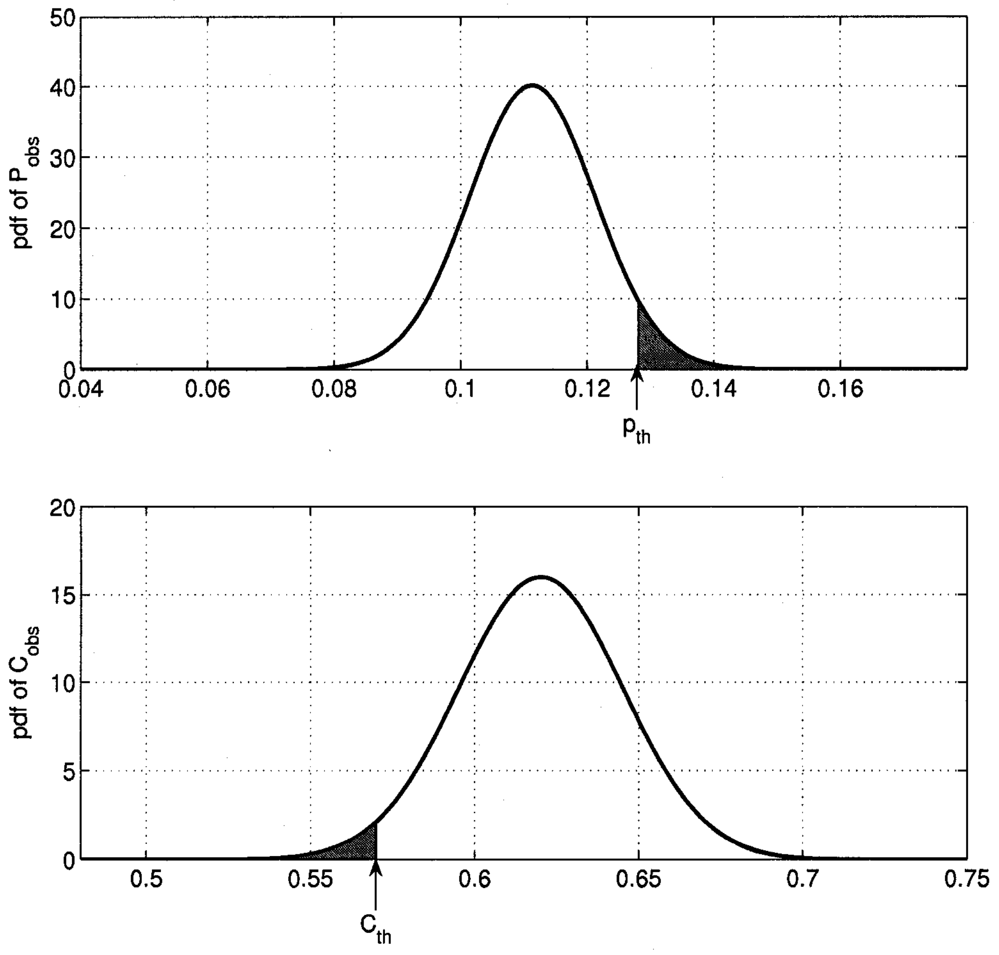


Figure 3.3: The proposed block error probability prediction method using the pdfs of P_{obs} and C_{obs} and (3.24) and (3.25). A (3,6)-regular LDPC code of length 10^3 has been used on the BIAWGN(0.83) channel with $\sigma^* = 0.8809$, $p_{\text{th}} = 0.1281$, and $c_{\text{th}} = 0.5708$. The shaded area represents the estimated block error probability.

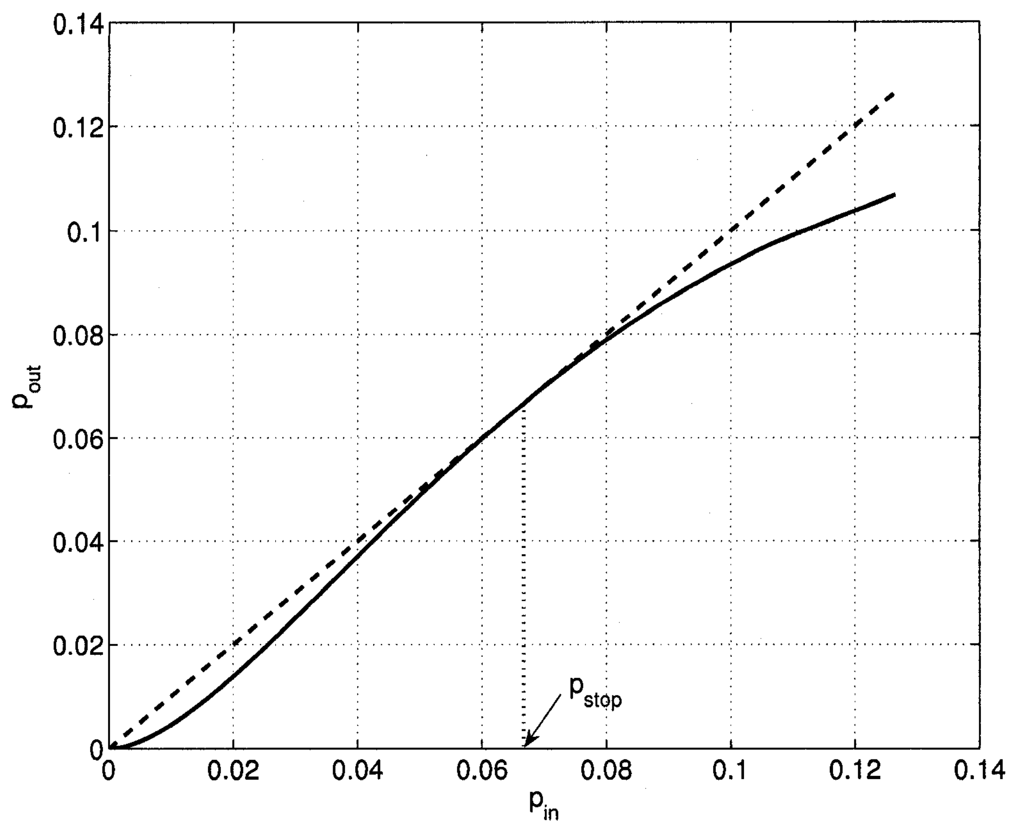


Figure 3.4: Plot of an EXIT chart for a (3,6)-regular code showing the point that the EXIT chart gets closed (p_{stop}).

3.3.2 Bit error probability

We can use the block error probability to derive bit error probability using an EXIT chart analysis. For the cases that the channel behaves worse than the threshold, the decoding process is done until it gets stuck and the error probability of the messages cannot get smaller than a certain amount p_{stop}. This can be visualized as the point where the EXIT chart[1] gets closed and intersects the line $p_{\mathsf{out}} = p_{\mathsf{in}}$. This has been visualized in Fig. 3.4 for a (3,6)-regular code at its decoding threshold. If the EXIT chart closes at only one point, we can assume that p_{stop} does not change significantly for different channel parameters worse than the threshold. Although p_{stop} can change for channel parameters much worse than the threshold, these cases occur very rarely.

Using p_{stop} as the variable-to-check message error probability, regardless of the

[1]EXIT charts can be defined based on different measures. Here we assume EXIT charts that track the evolution of message error probability.

pdf of these messages, one can calculate the check-to-variable message error probability using

$$q_{\mathsf{stop}} = \frac{1-\rho(1-2p_{\mathsf{stop}})}{2}. \tag{3.27}$$

Assuming that these check-to-variable messages are used to make a decision on the variable nodes, one can calculate the bit error probability α from q_{stop}. This implies that when the decoding is not successful, an α fraction of bits are in error. Each block is in error with probability $\mathrm{P_B}(\mathfrak{C}(\sigma), n, \lambda, \rho)$. Thus, the average bit error probability is $\alpha\mathrm{P_B}(\mathfrak{C}(\sigma), n, \lambda, \rho)$, i.e.,

$$\mathrm{P_b}(\mathfrak{C}(\sigma), n, \lambda, \rho) = \alpha\mathrm{P_B}(\mathfrak{C}(\sigma), n, \lambda, \rho). \tag{3.28}$$

Now the question is how to calculate p_{stop} and α. For the BEC and BSC under hard decoding, analytic relations of density evolution are simple and can be used for this purpose. For the sum-product algorithm and the BIAWGN, we use the Gaussian approximation method of [14]. For the URF channel and also min-sum decoding on BIAWGN channel, we use density evolution [13,19].

For the BEC(ϵ), same as (2.38), we have

$$p_e^{(\ell+1)} = \epsilon \cdot \lambda(1-\rho(1-p_e^{(\ell)})), \tag{3.29}$$

and for the BSC(ϵ) and Gallager's decoding algorithm A, same as (2.40), we write

$$p_e^{(\ell+1)} = (1-\epsilon)\cdot\lambda\left(\frac{1-\rho(1-2p_e^{(\ell)})}{2}\right) + \epsilon\cdot\left(1-\lambda\left(\frac{1+\rho(1-2p_e^{(\ell)})}{2}\right)\right), \tag{3.30}$$

where $p_e^{(\ell)}$ denotes the message error probability in the ℓth iteration. To find the point that the EXIT chart gets closed and find p_{stop}, we put $p_{\mathsf{stop}} = p_e^{(\ell+1)} = p_e^{(\ell)}$.

To calculate α we first define the *node-perspective* variable node degree distribution $\Lambda(x) = \sum_{i=2}^{d_v} \Lambda_i x^i$, which is given by

$$\Lambda(x) = \frac{\int_0^x \lambda(t)\mathrm{d}t}{\int_0^1 \lambda(t)\mathrm{d}t}. \tag{3.31}$$

In other words, Λ_i is defined as the fraction of variable nodes having degree i.

For the BEC(ϵ), using (3.27), we calculate α as

$$\alpha = \epsilon\sum_{i=2}^{d_v} \Lambda_i \cdot q_{\mathsf{stop}}^i, \tag{3.32}$$

and for the BSC(ϵ) as

$$\alpha = \sum_{i=2}^{d_v} \Lambda_i \left((1-\epsilon)q_{\text{stop}}^i + \epsilon \cdot (1-(1-q_{\text{stop}}^i))\right). \tag{3.33}$$

For the BIAWGN channel and the sum-product algorithm, similar to [14], we define

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{-\infty}^{\infty} \tanh \frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & x > 0 \\ 1, & x = 0. \end{cases} \tag{3.34}$$

Using the above function and the degree distribution pair $(\lambda(x), \rho(x))$ and assuming consistent Gaussian distribution for the messages in each iteration, the mean of the check-to-variable messages at each iteration $m_u^{(\ell)}$ is given by

$$m_u^{(\ell)} = \sum_{j=2}^{d_c} \rho_j \phi^{-1}\left(1 - \left[1 - \sum_{i=2}^{d_v} \lambda_i \phi\left(m_{u_0} + (i-1)m_u^{(\ell-1)}\right)\right]^{j-1}\right), \tag{3.35}$$

where $m_{u_0} = 2/\sigma^2$ is the mean of channel LLR messages and $m_u^{(0)} = 0$. Using (3.35), we find $m_u^{(\ell)}$ such that $m_u^{(\ell)} = m_u^{(\ell-1)}$. This is the point that the EXIT chart closes.

For a degree i node, we can define a decision-making variable, which is the sum of the channel LLR message and i independent check-to-variable LLR messages (see Equation (2.27)). Therefore, the error rate of a degree i node is $Q\left(\sqrt{\frac{m_{u_0}+im_u^{(\ell)}}{2}}\right)$ and the bit error probability is therefore

$$\alpha = \sum_{i=2}^{d_v} \Lambda_i Q\left(\sqrt{\frac{m_{u_0} + im_u^{(\ell)}}{2}}\right). \tag{3.36}$$

3.4 Examples and Discussion

3.4.1 Regular codes

Example 1 Consider $C^n(x^2, x^5)$ regular LDPC codes on the BSC(ϵ) with Gallager's decoding algorithm A. The code is randomly constructed and all cycles of length 4 are removed from the code's factor graph. The decoding threshold of the code is $\epsilon^* = 0.03946$. In Fig. 3.5, the simulation results reported in [24] are compared to our prediction method for $n = 1024$ and $n = 4096$. It is seen that the prediction method closely approximates the simulation results.

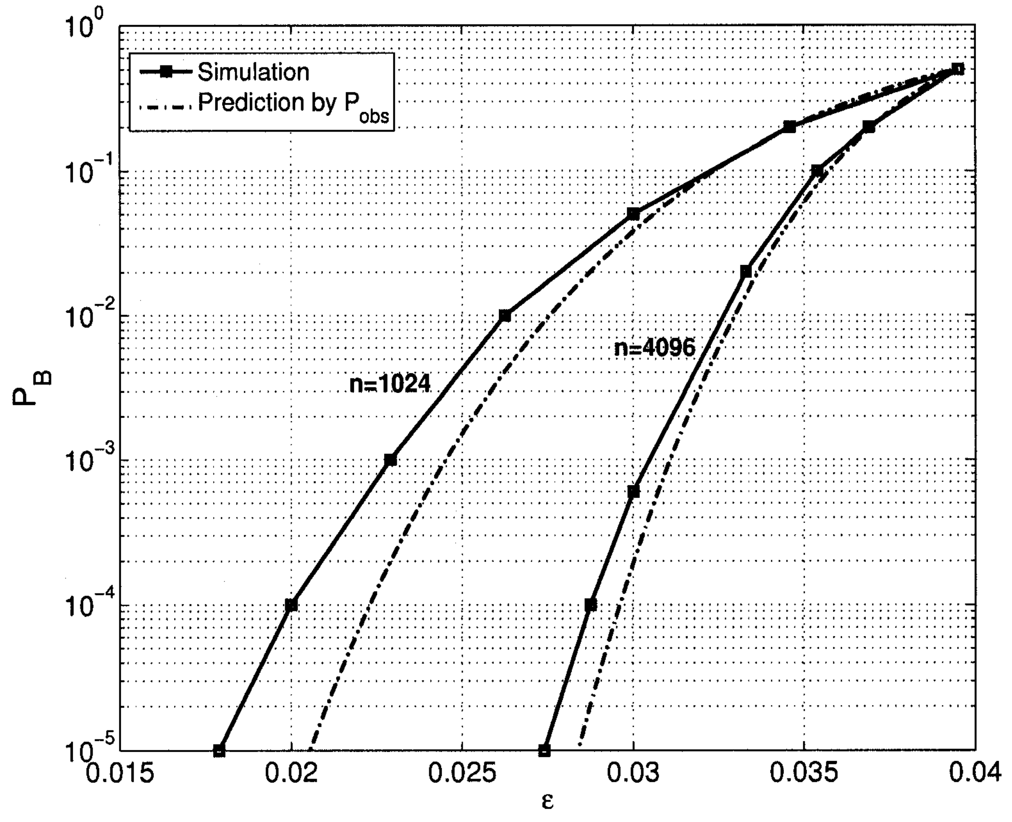


Figure 3.5: Comparison between the block error probability of a $\mathcal{C}^n(x^2, x^5)$ regular code on the BSC(ϵ) channel under Gallager's decoding algorithm A using simulation (solid curve) and our best prediction method (dashed curve).

Example 2 Consider the same code as in Example 1 on a BIAWGN(σ) channel. The decoding threshold of this code is $\sigma^* = 0.8809$ using the sum-product decoding algorithm. We use (3.24) and (3.25) to predict the block error probability.

The simulation results for $n = 10^4$ is depicted in Fig. 3.6. For this code we have $p_{\text{th}} = 0.1281$ and $c_{\text{th}} = 0.5708$. Using (3.35) and (3.36) we get $\alpha = 0.063$ and then we can compute the bit error probability using (3.28). At a bit error rate of 10^{-5} the gap between the actual result and the best prediction is about 0.032 dB while at this bit error rate, the gap between the actual result and the infinite-length performance prediction is 0.46 dB. As seen in the figure, modeling the channel variations with P_{obs} results in a better performance prediction. In Fig. 3.7, the actual performance for different n is compared with the best prediction method which is based on using P_{obs}. As seen in the figure, the prediction error decreases as n increases and vanishes for $n \geq 50,000$ where there is still a 0.2 dB gap from the infinite length performance prediction. We also found that for $n > 2500$, the estimation error is less than 0.1 dB at a bit error rate of 10^{-5}.

Next, we consider the min-sum decoding algorithm. The same $\mathcal{C}^{10^4}(x^2, x^5)$ regular code is decoded by the min-sum algorithm on the BIAWGN(σ) channel. In this case, we have $\sigma^* = 0.8223$, $p_{\text{th}} = 0.1120$, and $c_{\text{th}} = 0.6182$ and using density evolution we get $\alpha = 0.049$. The simulation result is depicted in Fig. 3.8. At a bit error rate of 10^{-5} the gap between the actual result and the best prediction is about 0.026 dB.

Example 3 Now, we use the same $\mathcal{C}^{10^4}(x^2, x^5)$ code on the URF(σ) channel decoded under the sum-product algorithm. The threshold of the code on this channel assuming perfect SI at the receiver is $\sigma^* = 0.7031$. The simulation result is depicted in Fig. 3.9. For this code we have $p_{\text{th}} = 0.1454$ and $c_{\text{th}} = 0.5684$. Using density evolution we get $\alpha = 0.059$ and then we can compute the bit error probability using (3.28). It is seen that the prediction method closely approximates the simulation results (less than 0.036 dB at bit error rate of 10^{-5}).

3.4.2 Irregular codes

Example 4 Consider the BIAWGN(σ) channel and a rate-1/2 irregular LDPC code $\mathcal{C}^{10^4}(\lambda(x), \rho(x))$ specified by the degree distribution pair $\lambda(x) = 0.4x^2 + 0.4x^5 + 0.2x^8$ and $\rho(x) = x^8$. The decoding threshold of this codes is $\sigma^* = 0.8461$ using the sum-

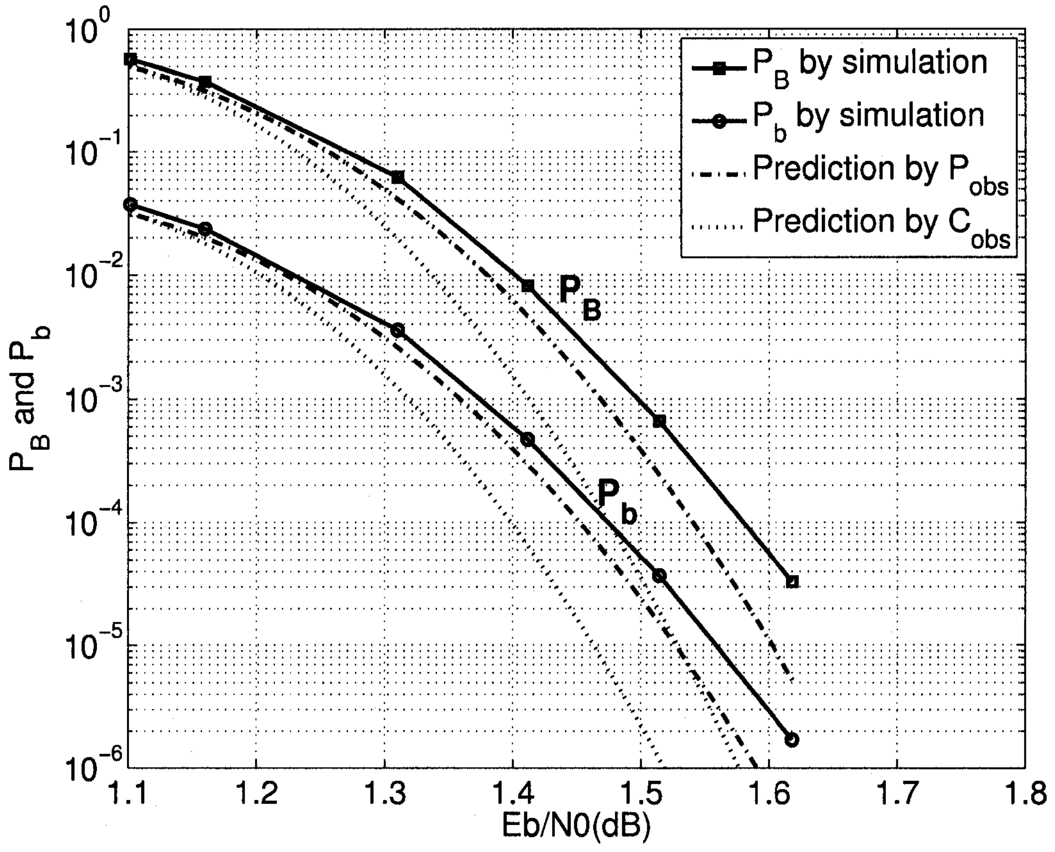


Figure 3.6: Comparison between the block error probability and bit error probability of a $\mathcal{C}^{10^4}(x^2, x^5)$ regular LDPC code on the BIAWGN channel using simulation and our prediction methods. The decoding algorithm is sum-product.

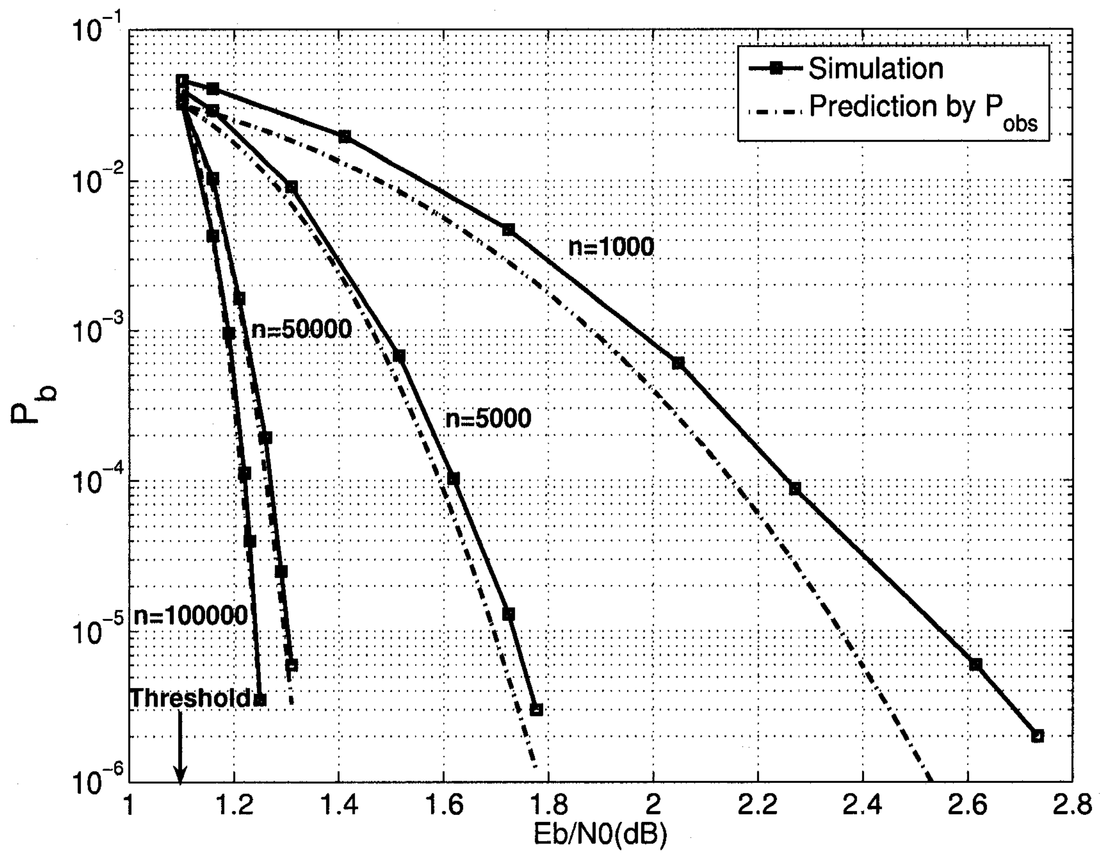


Figure 3.7: Comparison between the bit error probability of a $\mathcal{C}^n(x^2, x^5)$ regular LDPC code on the BIAWGN channel using simulation (solid curve) and our best prediction method (dashed curve) for different block lengths n.

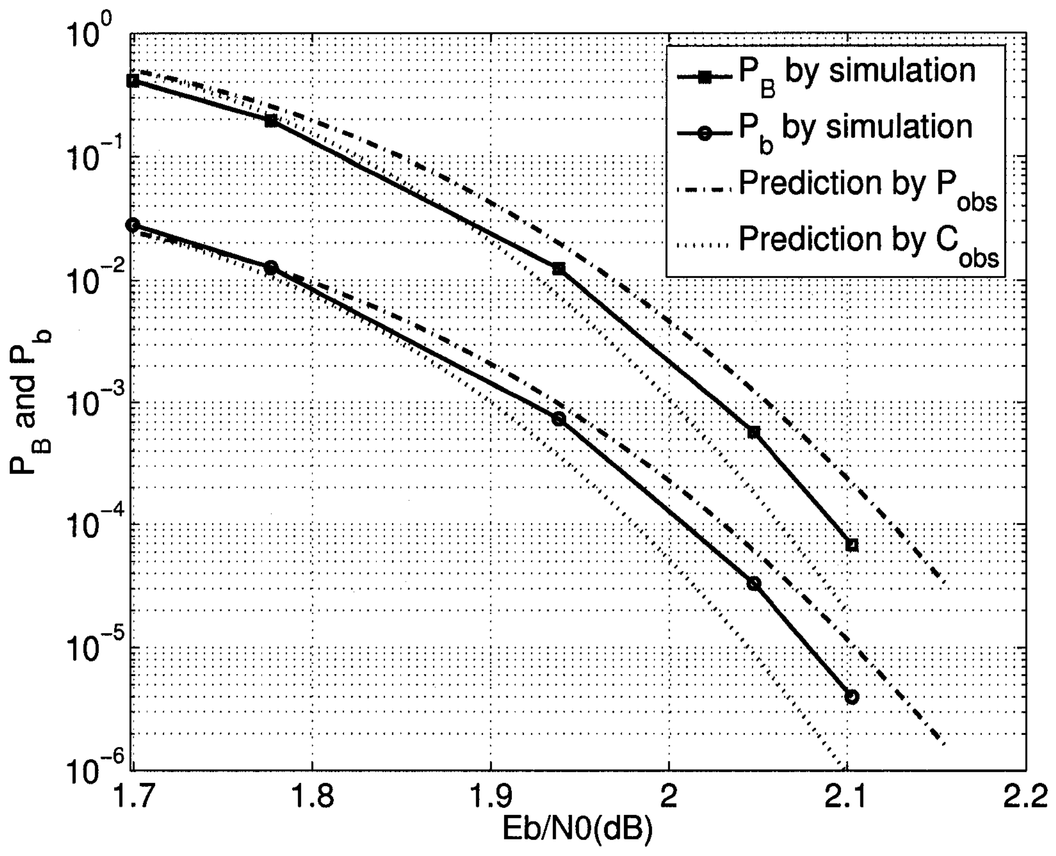


Figure 3.8: Comparison between the block error probability and bit error probability of a $\mathcal{C}^{10^4}(x^2, x^5)$ regular LDPC code on the BIAWGN channel using simulation and our prediction methods. The decoding algorithm is min-sum.

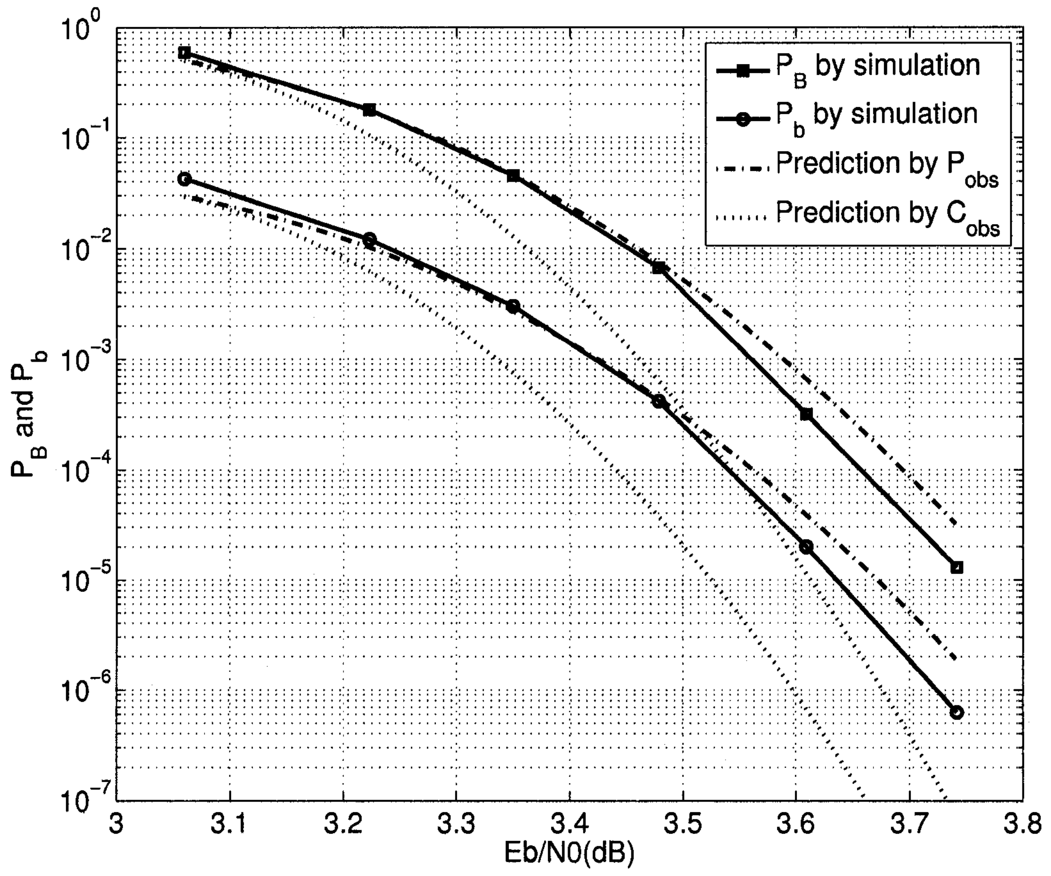


Figure 3.9: Comparison between the block error probability and bit error probability of a $\mathcal{C}^{10^4}(x^2, x^5)$ regular LDPC code on the URF channel using simulation and our prediction methods.

product algorithm and we have $p_{\text{th}} = 0.1186$ and $c_{\text{th}} = 0.5985$ and $\alpha = 0.083$. The code is randomly constructed and any cycle of length 4 is removed.

Fig. 3.10 shows the performance curves and compares our results with the simulation results. As in the regular case, we see that our prediction method closely approximates the actual performance. Again, P_{obs} gives more accurate results. At a bit error rate of 10^{-5}, the gap between the actual result and the best prediction is about 0.035dB.

It is worth mentioning that for codes whose EXIT charts near the threshold are very tight (i.e., their EXIT charts are very close to the 45-degree line when the channel parameter is near the threshold) or close at multiple points (i.e, their EXIT charts hit the 45-degree line at multiple points when the channel parameter is near the threshold), α cannot be calculated accurately. Hence, while the block error probability can still be predicted, the bit error probability cannot be found.

3.4.3 Discussion

As the results show, by considering the channel variations around its expected behavior, we are able to describe the code's performance in the waterfall region for different decoding algorithms, different code lengths and over different channel models. In other words, even by ignoring the presence of cycles in the code's factor graph, it is possible to predict the performance of LDPC codes of a few thousand bits or longer in the waterfall region quite accurately. Most practical scenarios are therefore covered by this simple analysis.

The concentration theorems of [13] show that for large n, the average behavior of individual instances of the code converges to the cycle-free case. As stated in [13], the concentration theorems predict pessimistic values for n and effects of the cycles are milder than predicted. As our results suggest and since our method ignores the presence of cycles, for even medium length LDPC codes the effect of cycles is negligible in the waterfall region.

As seen in the examples, for multi-dimensional channels, P_{obs} and (3.25) predict higher $\mathrm{P_B}$ values than C_{obs} and (3.24). This can be explained as follows. To analyze and compare the results of (3.24) and (3.25), we assume $\frac{1-c_{\text{th}}}{p_{\text{th}}} \approx \frac{1-\mu_{C_{\text{obs}}}}{\mu_{P_{\text{obs}}}}$ which is an appropriate assumption since we are not too far from the code's threshold in the waterfall region. The accuracy of this assumption is visualized in Fig. 3.11,

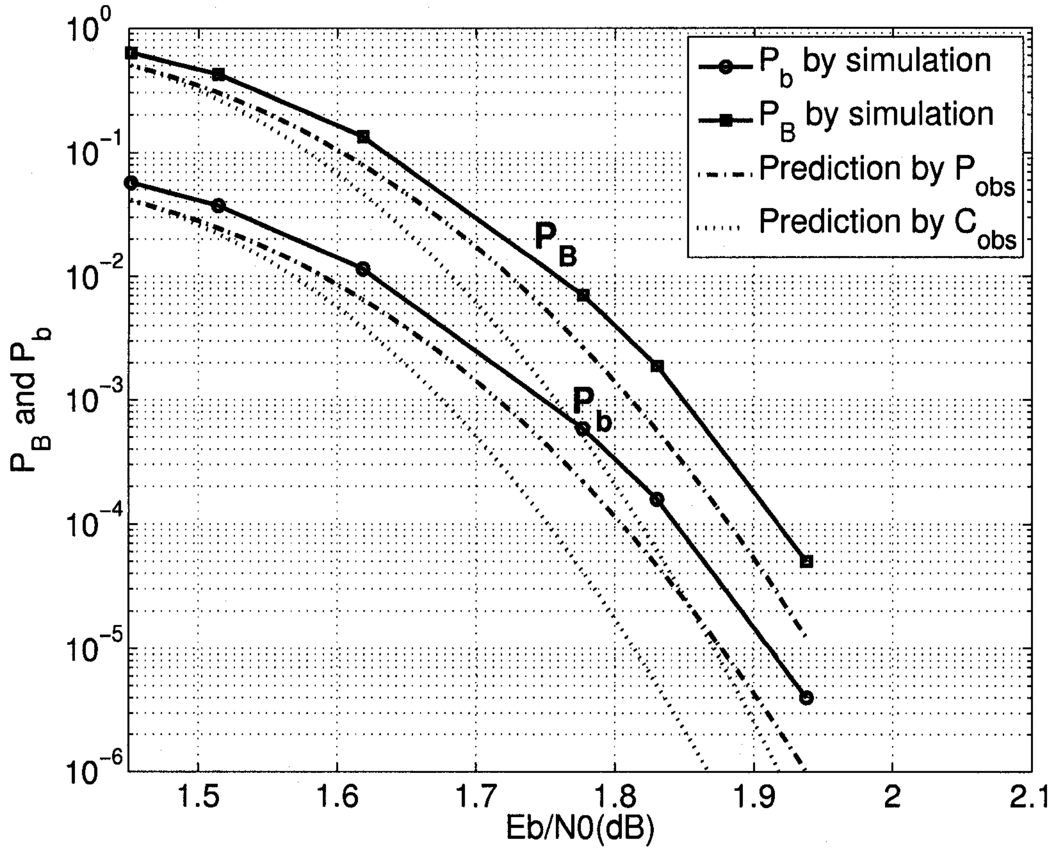


Figure 3.10: Comparison between the block and bit error probability of an irregular code with $n = 10^4$ using simulation and our prediction methods.

where $1-\mu_{C_{\text{obs}}}$ is plotted versus $\mu_{P_{\text{obs}}}$ for a BIAWGN channel. Also, the upper and lower bounds of $1-\mu_{C_{\text{obs}}}$ are depicted which correspond to the BSC and BEC respectively [60, 61].

In Fig. 3.12, $\gamma = \frac{1-\mu_{C_{\text{obs}}}}{\mu_{P_{\text{obs}}}}$ $(\approx \frac{1-c_{\text{th}}}{p_{\text{th}}})$ and $\frac{\sigma_{C_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}$ are depicted versus the SNR for a BIAWGN channel. It is seen that $\gamma > \frac{\sigma_{C_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}$ for different values of E_s/N_0. Therefore, $\frac{\gamma}{\sigma_{C_{\text{obs}}}} > \frac{1}{\sigma_{P_{\text{obs}}}}$ and then $\frac{\gamma(p_{\text{th}}-\mu_{P_{\text{obs}}})}{\sigma_{C_{\text{obs}}}} > \frac{p_{\text{th}}-\mu_{P_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}$. Thus, we get

$$\frac{(1-c_{\text{th}})-(1-\mu_{C_{\text{obs}}})}{\sigma_{C_{\text{obs}}}} = \frac{\mu_{C_{\text{obs}}}-c_{\text{th}}}{\sigma_{C_{\text{obs}}}} > \frac{p_{\text{th}}-\mu_{P_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}. \tag{3.37}$$

Since $Q(\cdot)$ is a strictly decreasing function, we conclude that

$$Q\left(\frac{\mu_{C_{\text{obs}}}-c_{\text{th}}}{\sigma_{C_{\text{obs}}}}\right) < Q\left(\frac{p_{\text{th}}-\mu_{P_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}\right), \tag{3.38}$$

which means that the block error probability predicted by C_{obs} and (3.24) is always less than what is predicted by P_{obs} and (3.25) for a BIAWGN channel. The same discussion also holds for the URF channel.

As shown through the examples, P_{obs} seems to give more accurate results. One conclusion here is that the decoder is more sensitive to the number of errors rather than the capacity. This can be argued in two ways. (1) In a check node, the output probability of error in each iteration q_e is given as a function of the input probability of error p_e by $q_e = \frac{1-\rho(1-2p_e)}{2}$. Thus, q_e only depends on p_e and the degree distribution $\rho(x)$ and not the input LLR pdf. In other words, different input LLR pdfs with the same p_e give the same q_e. As a result, the number of errors play an important role in the decoder. (2) In all practical decoders, the maximum LLR value is finite. Therefore, at high SNRs, the LLRs that have large absolute values cannot be represented accurately and their value is clipped to the maximum LLR value. This clipping influences the observed channel capacity, but has no effect on the observed error rate. Notice that in all examples, at high SNRs, P_{obs} gives a much better estimate of the performance.

Other parameters of the channel could also be used in modeling the channel variations and predicting the finite-length LDPC codes' performance. One such parameter is the observed LLR mean, defined as $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, where X_i denote the received LLR samples. In [14], LLR mean together with a Gaussian approximation on the pdf of messages is used to approximate density evolution. This measure, however, is not an appropriate parameter for modeling the quality of a channel. Notice that on the BEC(ϵ), $\overline{X} = \infty$. In other words, channel variations in terms of ϵ

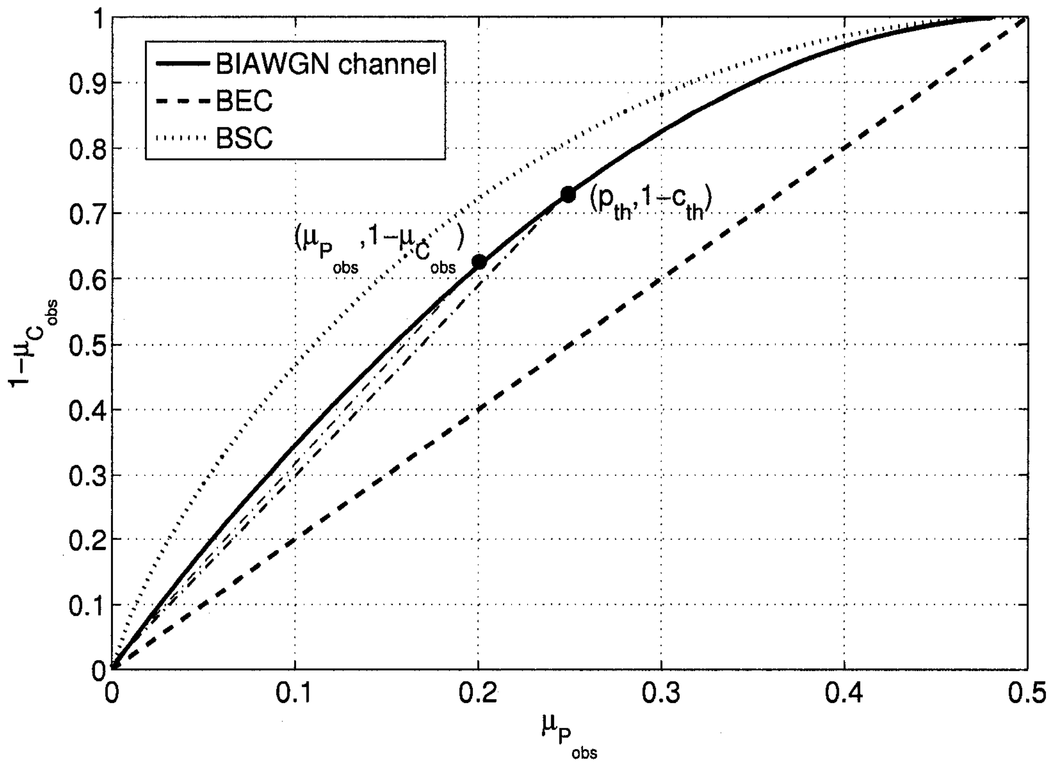


Figure 3.11: Plot of $1 - \mu_{C_{obs}}$ versus $\mu_{P_{obs}}$ for the BSC, BIAWGN, and BEC channels. The slopes of the lines connecting $(\mu_{P_{obs}}, 1 - \mu_{C_{obs}})$ and $(p_{th}, 1 - c_{th})$ to the origin, are approximately equal.

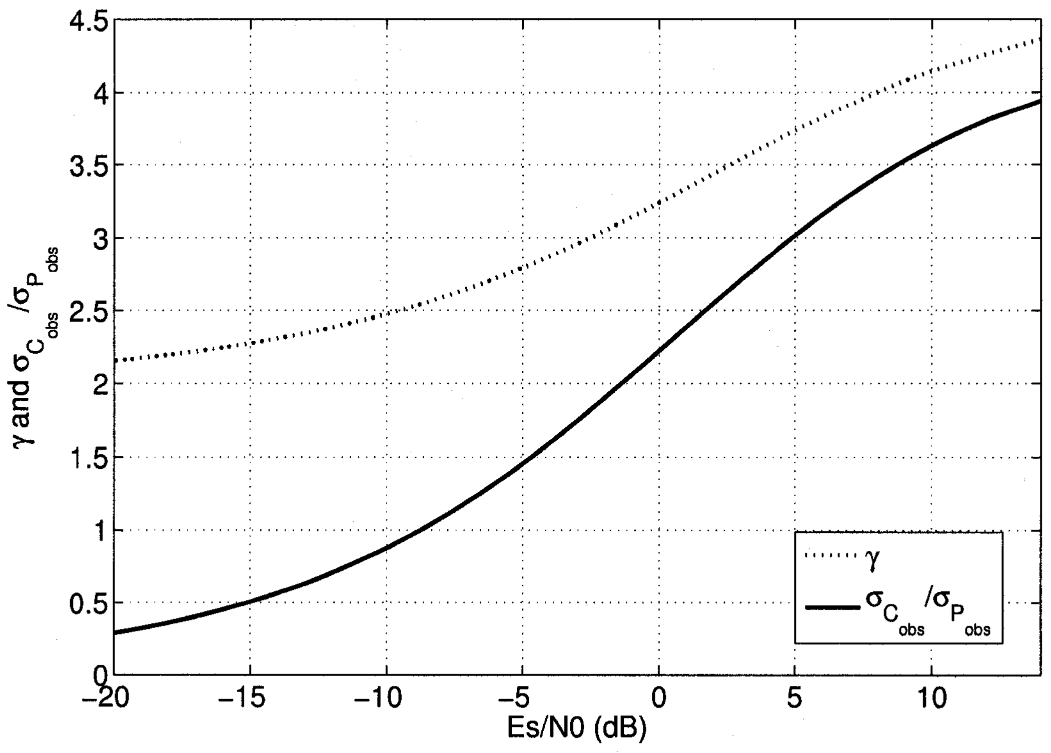


Figure 3.12: Comparison of γ and $\frac{\sigma_{C_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}$ as a function of SNR (E_s/N_0) for the BIAWGN channel. It is seen that $\gamma > \frac{\sigma_{C_{\text{obs}}}}{\sigma_{P_{\text{obs}}}}$ for different values of $\frac{E_s}{N_0}$.

are projected on an infinite change in variations in terms of $\overline{X}$. On other channels a similar problem exist. On a BIAWGN channel, for example, when capacity changes from 0 to 1, $\overline{X}$ varies from 0 to ∞, thus a small change in capacity may translate to a huge change in $\overline{X}$. The analysis based on $\overline{X}$, however, allows for small variations in $\overline{X}$ (due to the averaging). Thus an accurate representation of true channel variations is not captured by $\overline{X}$.

Another channel parameter which can be used is the observed noise power $\frac{1}{n}\sum_{i=1}^{n} z_i^2$, where z_i denote the received noise samples. This measure usually provides a good approximation of the performance. At high SNRs, however, it can be argued that this measure provides a block error rate estimation much below the estimated block error rate using the observed probability of error or observed capacity.

Compared to the scaling law [24], our method gives slightly less accurate results. From any practical point of view, however, our method gives accurate enough estimation of the performance for codes of a few thousand bits. In addition, our method is simpler, does not have any additional parameters, and does not require any curve fitting.

We can also use this method in the design of finite-length LDPC codes. One code design approach can be defined as maximizing the code rate given a target decoding threshold value. In our method, using the block length and the channel parameter, we select an appropriate decoding threshold which results in a desired block error rate. That is, for channel $\mathfrak{C}(\sigma_0)$ and block length n, if a code with a block error rate better than p_{error} is needed, choose the decoding threshold according to

$$p_{\mathsf{th}} \geq F_{P_{\mathsf{obs}}}^{-1}(\mathfrak{C}(\sigma_0), n, 1 - p_{\mathsf{error}}) \tag{3.39}$$

where $F_{P_{\mathsf{obs}}}^{-1}$ is the inverse cdf of P_{obs}. In other words, we have chosen p_{th} such that the probability that $P_{\mathsf{obs}} > p_{\mathsf{th}}$ is less than p_{error}. Then, we compute the decoding threshold according to $\sigma^* = 1/Q^{-1}(p_{\mathsf{th}})$. Now, we assume an infinite length code and maximize the rate for this threshold. This design approach holds for the BIAWGN channel. However, extending the results to the BEC, BSC and URF channel is straightforward.

3.5 Conclusion

A simple method for finite-length LDPC codes analysis was introduced. First, we studied the channel behavior for the finite-length case and we modeled the channel as a varying channel from the codewords perspective. Using this interpretation, we studied two parameters of the observed channel (probability of error and capacity) and modeled them as random variables. After obtaining the distribution of these random variables analytically, we were able to predict the performance (block and bit error probability) of finite-length LDPC codes. Our results suggest that when the block length is a few thousand bits, even by ignoring the effects of cycles, we are able to predict the performance in the waterfall region closely.

We observed that in multi-dimensional channels, modeling the channel variations with P_{obs} gives more accurate results rather than modeling with C_{obs}. Therefore, P_{obs} is suggested to be used in modeling the channel variations and analysis of finite-length LDPC codes. Also, we showed that the block error probability given by P_{obs} is greater than what is given by C_{obs}.

This simple method is applicable to both regular and irregular codes. We also suggested a method to choose the decoding threshold of the code, based on its block length.

Chapter 4

Optimum Linear LLR Calculation for Iterative Decoding on Fading Channels

4.1 Introduction

There have been many advances in iterative decoding techniques and it has been shown that using graphical codes such as LDPC codes [8] and turbo codes [2] associated with iterative decoding, the Shannon limit on many channels (e.g., additive white Gaussian noise channel) can be approached [12]. Therefore, these codes have also been proposed for wireless fading channels [19].

Application of LDPC codes on Rayleigh fading channel is pioneered in [19], where a detailed study of performance and code design is conducted. This work is later extended to time-selective complex fading channels [62], to Rician fading channels [63], and also to Rayleigh block fading channels [64]. The application of turbo codes on Rayleigh fading channels is also studied in [65].

For soft iterative decoding, LLRs at the output of the channel are calculated. The process of computing LLRs depends on whether or not a perfect knowledge of the channel parameters exists at the receiver. The capacity of the fading channel is also affected with the availability of channel parameters at the receiver [65].

In a recent work [66], based on a detailed study on the effects of the channel mismatch, an efficient solution is proposed for uncorrelated Rayleigh block fading channels which does not need a knowledge of the channel noise power. The solution is to consider the decoding threshold of the code as an estimate of the channel noise power. This optimum choice is a property of the code and not the channel and it gives a very close performance to that of a system which has a perfect knowledge of the channel noise power.

An uncorrelated fading channel can be modeled with a fading gain r and an additive Gaussian noise $z \sim \mathcal{N}(0, \sigma_z^2)$. When r is known at the receiver as a perfect side information (SI), LLRs are linear functions of the channel output [19], and exact LLR computation depends on a perfect knowledge of σ_z at the receiver.

In order to have SI at the receiver, channel estimation techniques must be used. These techniques increase the complexity of the system, can cause a significant overhead, and are themselves subject to imperfections. For high throughput wireless applications, the receiver may not be able to handle the extra complexity or overhead. This chapter provides an alternative solution which does not require channel estimation, yet provides better performance compared to the existing solutions that use fixed fading gain estimates in the decoder.

With no SI available at the receiver, LLRs are complicated functions of the channel output [67] and depend on the pdf of r. An approximate LLR, however, can be computed as a linear function of the channel output [67]. The coefficient of this linear function depends on a fixed estimate $\hat{r}$ of the channel fading gain and a knowledge of σ_z. Previous work assume that σ_z is known and use the expected value of r for $\hat{r}$ [19,65]. While the expected value of the fading gain is the minimum mean square error estimation of r, it is not guaranteed that this choice provides the optimum performance in the decoder.

In a general setup (which includes famous fading channel models such as uncorrelated Rayleigh and Rician fading channels), we propose the following question: Assume that the pdf of r is known at the receiver, but the channel instantaneous fading gain is not. Also assume that LLRs are to be computed as linear functions of the channel output. What linear approximation provides the optimum decoding performance?

This question is studied in this chapter and the following contributions are made: (1) When σ_z is known, we find a linear LLR approximation which allows for the maximum achievable code rate on the channel. We prove that the optimum linear approximation is unique and we observe that, on a Rayleigh fading channel, it closely approaches the capacity under true LLR calculation. This solution can significantly outperform LLR calculation based on the expected value of r. We also design irregular LDPC codes which approach this maximum achievable rate. (2) When neither r nor σ_z is known at the receiver, we propose a linear LLR calculation technique which guarantees the convergence of the decoder over the widest possible range of σ_z. The performance of this solution is almost identical to the case that σ_z is perfectly known. We design appropriate irregular LDPC codes for this case too.

This chapter is organized as follows. Section 4.2 reviews some preliminaries and studies the proposed approaches. Section 4.3 studies the problem when σ_z is known to the receiver. Section 4.4 extends our results to the case that σ_z is unknown. Section 4.5 concludes the chapter.

4.2 Preliminaries and Approaches

4.2.1 System model

Consider the following channel model. The output of the channel is given by

$$y = r \cdot x + z, \tag{4.1}$$

where $x \in \{-1, 1\}$ represents the input signal and z is the Gaussian noise with zero mean and variance σ_z^2. Also $r \geq 0$ is the channel gain which has an arbitrary pdf $f_R(r)$ and changes independently from one channel use to another. Uncorrelated fading channels fall into this system model where r represents the channel fading gain.

4.2.2 LLR definition and distributions

For soft decoding, LLRs are usually computed and used. As we said in Section 2.5, analysis of some iterative decoders is based on the pdf of LLRs under the assumption that the all-zero codeword ($x = +1$) is transmitted [47].

For the model in (4.1), the conditional pdf of y is given by

$$f_{Y|X,R}(y|x,r) = \frac{1}{\sqrt{2\pi}\sigma_z} \exp\left(-\frac{(y - x \cdot r)^2}{2\sigma_z^2}\right), \tag{4.2}$$

which represents a Gaussian distribution with mean $x \cdot r$ and variance σ_z^2. We are interested to compute the channel LLRs and their pdf.

Ideal SI

When we have ideal SI, the channel fading gain r is known for each received bit. Also, the receiver knows the noise power. Therefore, the LLR is given by [19]

$$l = \log \frac{\mathrm{P}(x=+1|y,r)}{\mathrm{P}(x=-1|y,r)} = \frac{2}{\sigma_z^2} y \cdot r, \tag{4.3}$$

which is a linear function of y. The pdf of l is given by (3.15).

No SI

When no side information is available at the receiver, the channel LLR is

$$l = \log \frac{\mathrm{P}(x=+1|y)}{\mathrm{P}(x=-1|y)}, \tag{4.4}$$

which can be a complicated function of y in general. For instance, on a normalized Rayleigh channel (i.e., $f_R(r) = 2re^{-r^2}$) we have

$$l = \log \frac{\Phi(y/\sqrt{2\sigma_z^2(1+2\sigma_z^2)})}{\Phi(-y/\sqrt{2\sigma_z^2(1+2\sigma_z^2)})}, \tag{4.5}$$

where $\Phi(z) = 1 + \sqrt{\pi} z e^{z^2} \mathrm{erfc}(-z)$ and erfc$(\cdot)$ represents the complementary error function [67]. This LLR is a complicated function of y and hard to be calculated in the decoder. Also, calculating the LLR pdf is difficult. To simplify the LLR calculation, motivated by (4.3), we write $\hat{l}$ as

$$\hat{l} = \frac{2}{\hat{\sigma}_z^2} y \cdot \hat{r} = \alpha y, \tag{4.6}$$

where $\hat{\sigma}_z^2$ represents the receiver's estimate of the Gaussian noise variance σ_z^2 and $\hat{r}$ represents a fixed receiver's estimate of the fading gain r. Here, $\alpha = 2\frac{\hat{r}}{\hat{\sigma}_z^2}$. This linear representation of LLR is consistent with the results in [67] which states that the LLR can be approximated by a linear function of y (also see Fig. 4.3). This approach is also consistent with existing work which assumes that σ_z is known and uses expected value of r (E$[r]$) as $\hat{r}$ [19].

The conditional pdf of $\hat{l}$ is

$$f_{\hat{L}|R}(\hat{l}|r) = \frac{\hat{\sigma}_z^2}{2\hat{r}\sigma_z\sqrt{2\pi}} \exp\left(-\frac{(\hat{l} - 2r\hat{r}/\hat{\sigma}_z^2)^2}{8\hat{r}^2\sigma_z^2/\hat{\sigma}_z^4}\right). \tag{4.7}$$

To get the unconditional pdf of $\hat{l}$, (4.7) should be averaged over the density of r. For example, for the normalized Rayleigh fading channel we have

$$\begin{aligned} f_{\hat{L}}(\hat{l})_{\{\sigma_z,\alpha\}} &= \frac{2\Delta^2}{\alpha\sigma_z\sqrt{2\pi}} \exp\left(-\frac{\Delta^2}{\alpha^2\sigma_z^2}\hat{l}^2\right) \times \\ & \left[\exp\left(-\frac{\Delta^2}{2\alpha^2\sigma_z^4}\hat{l}^2\right) + \frac{\Delta\sqrt{2\pi}}{2\alpha\sigma_z^2}\hat{l}\ \mathrm{erfc}\left(-\frac{\Delta}{\alpha\sigma_z^2\sqrt{2}}\hat{l}\right)\right] \end{aligned} \quad (4.8)$$

where $\Delta = \sqrt{\frac{\sigma_z^2}{2\sigma_z^2+1}}$. This pdf is parameterized by σ_z and α. If $\hat{\sigma}_z = \sigma_z$ and $\hat{r} = \mathrm{E}[r]$, i.e., $\alpha = 2\frac{\mathrm{E}[r]}{\sigma_z^2}$, (4.8) reduces to the distribution in [19, Eq. 16].

4.2.3 Capacity

The capacity of a BIMS channel can be given via the pdf $f_L(l)$ of the LLR by [49,59]

$$C = 1 - \mathrm{E}_l[\log_2(1+e^{-l})] = 1 - \int_{-\infty}^{\infty} \log_2(1+e^{-l}) f_L(l)\mathrm{d}l. \quad (4.9)$$

$$C = 1 - \mathrm{E}_l[\log_2(1+e^{-l})] = 1 - \int_{-\infty}^{\infty} \log_2(1+e^{-l}) f_L(l)\mathrm{d}l. \quad (4.9)$$

The above relatic

(i.e., $f_L(-l) = e^{-l}f_L(l)$). For the proof of (4.9) and the properties of LLR refer to Appendix A.

The channel capacity C can be computed in two cases: with ideal SI or no SI. In each case, their corresponding LLR distribution should be used in (4.9). In the absence of SI at the receiver, the quantity calculated by putting $f_{\hat{L}}(\hat{l})$ in (4.9) called $\hat{C}$, is not the channel capacity since $\hat{l}$ is a linear approximation and not the true LLR. Also, since $f_{\hat{L}}(\hat{l})$ is not consistent, $\hat{C}$ does not represent the maximum achievable transmission rate under linear LLR calculation of (4.6). It is proved in [68] that $\hat{C}$ achieves its maximum when true LLRs are used. This hints that maximizing $\hat{C}$ under some approximate LLR calculation method would provide a good approximation of the true LLRs. Moreover, we observe that by maximizing $\hat{C}$ with respect to α, $f_{\hat{L}}(\hat{l})$ becomes a nearly consistent distribution. Thus, by maximizing $\hat{C}$, a very good estimate of the maximum achievable transmission rate under a linear LLR calculation is obtained. Moreover, through examples we show that the maximized $\hat{C}$ is extremely close to C (in the absence of SI) which further justifies our approach (see Fig. 4.2).

4.2.4 LDPC codes decoding and analysis

Some of the results of this chapter are shown through analysis and design of LDPC codes. As stated in Chapter 2, many different message-passing algorithms can be used for the decoding of LDPC codes. In this chapter, our focus will be on the sum-product algorithm [36].

For the channel model of (4.1), the decoding threshold σ_z^* of an ensemble of LDPC codes is defined as the maximum noise standard deviation σ_z for which the bit error probability of the message-passing decoder gets arbitrarily small when the code length is growing [13, 47] if and only if $\sigma_z \leq \sigma_z^*$. This σ_z^* depends on whether SI is available at the receiver or not.

4.2.5 Code design

In this chapter, two code design processes associated with two measures of performance are used. One can be defined as maximizing the threshold of the code over its degree distributions given a target code rate and another one, can be defined as maximizing the code rate over its degree distributions given the channel LLR pdf. This pdf generally depends on the channel type and its parameters. In our case, it depends on the noise power, the pdf of r, and the availability of SI. When no SI is available and (4.6) is used, the channel LLR pdf also depends on α in (4.6). We will use both of these definitions in this chapter and will clarify which one we use. We will refer to the first definition as the threshold maximization design and to the second one as the rate maximization design.

4.3 Optimum Linear LLR Calculation

When no SI is available at the receiver, one can calculate the LLRs linearly via (4.6) as an approximation to (4.4). The objective is to find the optimal linear approximation. Different measures of optimality can be considered. Existing work assumes that σ_z is known and chooses $\hat{r} = \mathrm{E}[r]$. This choice of $\hat{r}$ is optimum in the sense of minimum mean square error $\mathrm{E}[|r - \hat{r}|^2]$. Here, we find the linear approximation which maximizes $\hat{C}$. We call this linear approximation *maximum-capacity* linear-approximation (MCLA).

Maximizing $\hat{C}$ requires a knowledge of σ_z and pdf of r. These are needed for finding the pdf of $\hat{l}$ and thus optimizing its corresponding capacity. So, we first assume that these pieces of information are available. Later we generalize our results to the case that σ_z is unknown. When σ_z is known, without loss of generality we set $\hat{\sigma}_z = \sigma_z$ in (4.6) and we find the optimum choice of $\hat{r}$. Notice that with $\hat{r}$ one can adjust α and thus the pdf of $\hat{l}$ as needed.

MCLA maximizes $\hat{C}$. Interestingly, we observe that the maximized $\hat{C}$ is extremely close to C based on true LLR calculation. To show that this optimization is meaningful in practice, we design irregular LDPC codes that approach the maximum achievable transmission rate C. Thus, by maximizing $\hat{C}$ via linear LLR approximation we are providing a simple solution for close to capacity performance.

MCLA is also expected to result in improved performance in iterative decoders. That is, for a fixed code, computing LLRs according to MCLA should improve bit error rate (BER) performance. Our simulation results will support this claim, but the following two arguments can also be provided to justify this choice.

1. MCLA provides the maximum $\hat{C}$ and thus the maximum gap between the code rate R and the capacity $\hat{C} \simeq C$. Thus one expects improved BER performance.

2. Since $\hat{l}$ is not the true LLR, under any linear LLR calculation, $\hat{C} \leq C$. Under a good linear approximation, pdf of $\hat{l}$ is close to that of true LLRs and thus $\hat{C}$ is close to C. Hence, a minimized $C - \hat{C}$ (through maximizing $\hat{C}$) indicates a good LLR approximation and hence an improved performance.

As mentioned, our simulation results show that MCLA indeed improves the performance compared to existing work based on choosing $\hat{r} = \mathrm{E}[r]$. Moreover, though not rigorously proved, MCLA appears to be the optimum choice in terms of BER performance too. Thus, our proposed method is based on maximizing $\hat{C}$ over $\hat{r}$ for fixed $\hat{\sigma}_z$ and σ_z, and we define

$$\hat{r}_{\mathsf{opt}} = \arg\max_{\hat{r}} \hat{C}. \tag{4.10}$$

The following theorem suggests that finding $\hat{r}_{\mathsf{opt}}$ can be done very efficiently.

Theorem 1 *For a fixed $\hat{\sigma}_z$ and σ_z, there exists a unique $\hat{r}$ which maximizes $\hat{C} = 1 - \mathrm{E}_{\hat{l}}[\log_2(1 + e^{-\hat{l}})]$.*

Proof:

$$\hat{C} = 1 - \mathrm{E}_{\hat{l}}[\log_2(1 + e^{-\hat{l}})] = 1 - \mathrm{E}_y[\log_2(1 + e^{-\frac{2\hat{r}}{\hat{\sigma}_z^2}y})]$$

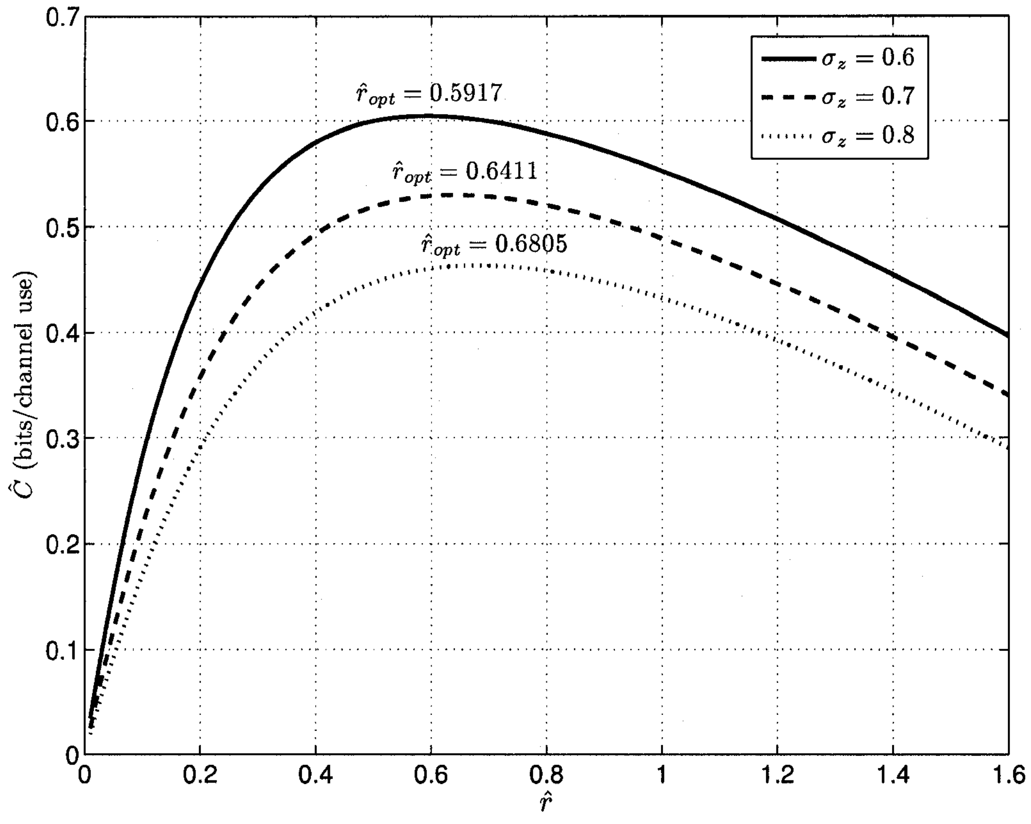


Figure 4.1: $\hat{C}$ for different σ_z and $\hat{r}$ on a normalized Rayleigh fading channel. The curves are concave and the maximizing point is unique. Moreover, the maximizing point is not much sensitive to σ_z.

$$\begin{aligned}\frac{d^2\hat{C}}{d\hat{r}^2} &= -\mathrm{E}_y\left[\frac{d^2}{d\hat{r}^2}\left(\log_2(1+e^{-\frac{2\hat{r}}{\hat{\sigma}_z^2}y})\right)\right] \\ &= \mathrm{E}_y\left[\frac{-\left(\frac{2y}{\hat{\sigma}_z^2}\right)^2 e^{-\frac{2\hat{r}}{\hat{\sigma}_z^2}y}}{\left(1+e^{-\frac{2\hat{r}}{\hat{\sigma}_z^2}y}\right)^2 \ln 2}\right] < 0\end{aligned}$$

The above expression is negative since the term inside the expected value is always negative. Therefore, $\hat{C}$ is a concave function of $\hat{r}$ and there exists a unique maximum in $\hat{r} = \hat{r}_{\mathsf{opt}}$. This theorem is valid for any distribution of $r \geq 0$. ■

Maximizing $\hat{C}$, therefore, is a straightforward task because it is a one-variable convex-optimization problem and can be solved very efficiently by simple numerical techniques. Different $\hat{C}$ curves are depicted in Fig. 4.1 for some $\hat{r}$ and the case $\hat{\sigma}_z = \sigma_z$. Notice that $\hat{r}_{\mathsf{opt}}$ is not very sensitive to σ_z.

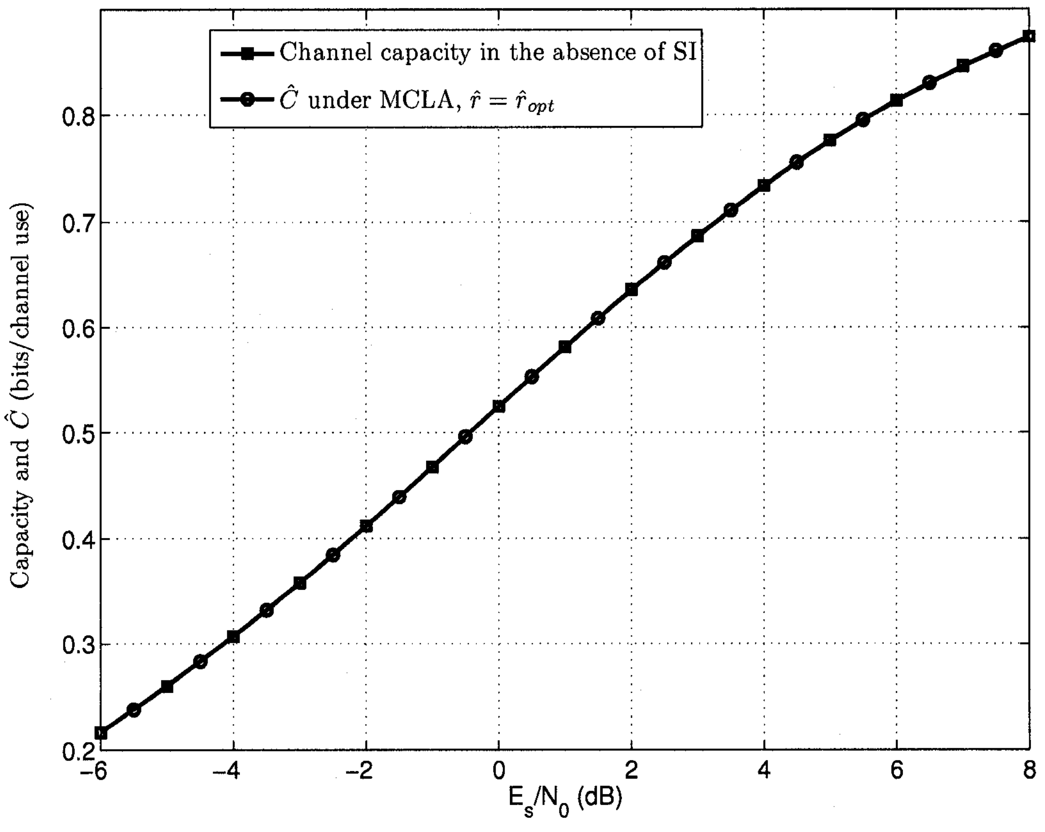


Figure 4.2: Comparison between the highest achievable transmission rate in the case of true LLR calculation and $\hat{C}$ on a normalized Rayleigh fading channel. It is assumed that $\hat{\sigma}_z = \sigma_z$.

Fig. 4.2 shows that we can get very close to the channel capacity under MCLA. Simulations show that $\hat{r} = \mathrm{E}[r]$ can result in significant performance loss especially in high SNR regions where the capacity is high and high-rate codes are used.

The following two examples support our above mentioned results. In Example 1, we design an irregular LDPC code which approaches the capacity that is maximized by MCLA. Example 2 shows improved BER performance under MCLA.

Example 1: Consider an uncorrelated normalized Rayleigh fading channel with $\sigma_z = 0.7436$. The channel capacity is 0.5 bits/channel use in the absence of SI and $\hat{C} = 0.4999$ can be achieved using $\hat{r}_{\text{opt}} = 0.6594$ (compare with $\mathrm{E}[r] = 0.8862$). In Fig. 4.3, the exact LLR values obtained form (4.5) are compared to the optimum linear approximation under MCLA and linear approximation with $\mathrm{E}[r]$.

We design a code based on rate maximization under MCLA. We assume a fixed $\rho(x) = x^8$ and a maximum variable node degree d_v of 30. Under MCLA and 11-bit

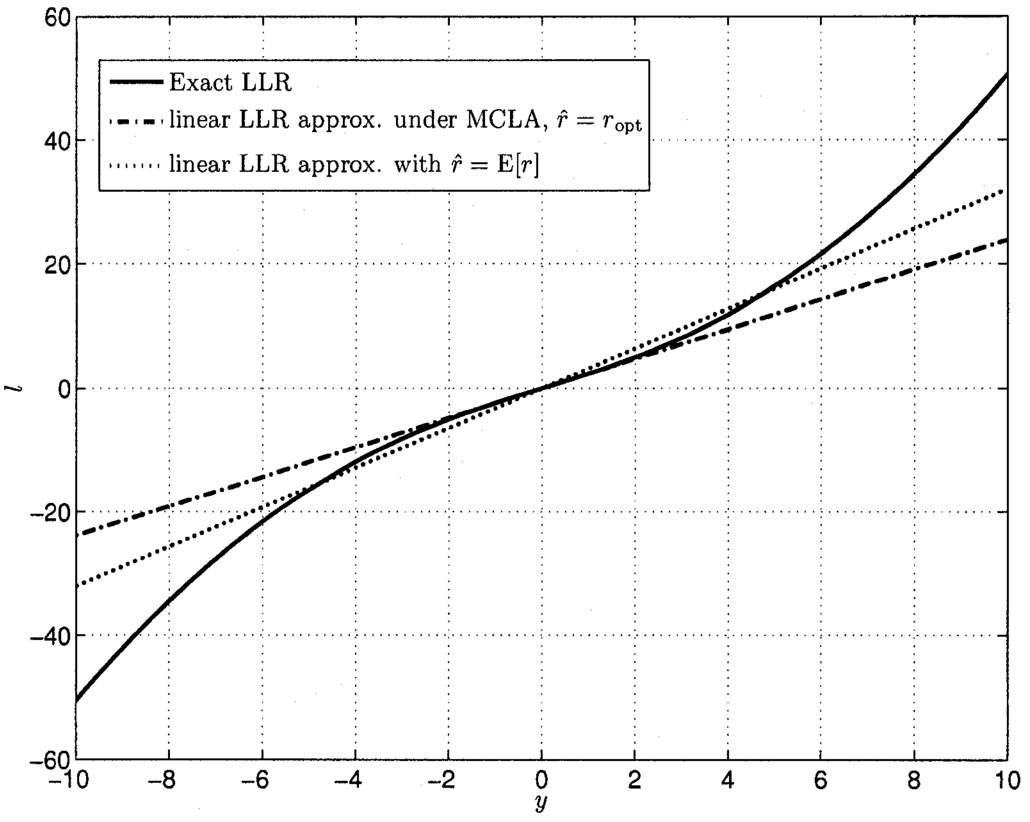


Figure 4.3: Comparison of exact LLRs and linear approximations for a normalized Rayleigh fading channel with $\sigma_z = 0.7436$ and no SI available at the receiver. Under MCLA we have $\hat{r}_{\text{opt}} = 0.6594$. It is seen that MCLA gives a nearly perfect linear approximation when $|y|$ is small.

decoding, and allowing a maximum of 300 iterations, the optimized code is given in Table 4.1 (Code1). The optimization can be done easily using linear programming and the method described in Section 2.6. The rate of the designed code is 0.4889. The designed code has almost approached $\hat{C}$ and also the capacity C of the channel with no SI.

Example 2: To show that MCLA also improves the BER of the code, a $\mathcal{C}^{10^4}(x^2, x^5)$ LDPC code is simulated on an uncorrelated normalized Rayleigh fading channel. Fig. 4.4 shows the BER of the code with and without SI at the receiver. When SI is not available and σ_z is known, two cases have been plotted. One is when $\hat{\sigma}_z = \sigma_z$ and $\hat{r} = \mathrm{E}[r]$ and the other is under MCLA (i.e., $\hat{\sigma}_z = \sigma_z$ and $\hat{r} = \hat{r}_{\text{opt}}$). The decoding threshold of the code is 4.06 dB with $\hat{r} = \mathrm{E}[r]$ and no SI, 3.82 dB under MCLA, and 3.06 dB with perfect SI. The figure shows considerable BER improvement under MCLA. This improvement is about 0.3 dB at BER of 10^{-5}. The gap between the thresholds with SI and with no SI under MCLA is 0.76 dB. At this code rate, existence of SI results in about 0.74 dB improvement [19, 65]. Thus, MCLA shows a minor extra gap (0.02 dB) compared to true LLR calculation.

We have chosen a (3,6)-regular LDPC code since most of its results exist in the literature. More significant performance improvement can be seen for other codes. For example, the decoding threshold of a (4,16)-regular code, which is a code of rate 3/4 is 7.69 dB with $\hat{r} = \mathrm{E}[r]$ and no SI, and 6.94 dB under MCLA on a normalized Rayleigh fading channel. The BER performance of this code is plotted in Fig. 4.5 when no SI is available under $\hat{r} = \mathrm{E}[r]$ and under MCLA. At the BER of 10^{-5} the performance improvement is about 0.75 dB.

From Fig. 4.1, it is seen that $\hat{r}_{\mathsf{opt}}$ decreases when σ_z decreases. This implies that the distance between $\hat{r}_{\mathsf{opt}}$ and $\mathrm{E}[r]$ increases. Therefore, the performance improvement is usually more significant for low values of σ_z where high rate codes are used.

4.4 Noise Power Unknown at the Receiver

Under MCLA, the pdf of $\hat{l}$ is a linear transformation of the pdf of y. In fact, $\alpha_{\mathsf{opt}} = 2\frac{\hat{r}_{\mathsf{opt}}}{\sigma_z^2}$ would give the linear transformation whose associated capacity (given by (4.9)) is maximum. When σ_z is unknown, the distribution of y is not known at the receiver. Therefore, finding the optimum linear transformation is somewhat meaningless. However, for any σ_z one can find α_{opt}. Thus, α_{opt} is a function of σ_z. It is also obvious from Theorem 1, that α_{opt} is unique for each σ_z. Thus, α_{opt} will be denoted as $\alpha_{\mathsf{opt}}(\sigma_z)$ afterwards.

Since σ_z is unknown, $\alpha_{\mathsf{opt}}(\sigma_z)$ is also unknown. However, one can find α such that a given code has the widest range of convergence over changes of σ_z. This way, we ensure that the code is robust to the changes in the noise power (e.g., when the code is used on different channels with different noise powers). This is equivalent to finding α such that the decoding threshold is maximum. Now, we explain how such α could be found.

The basic idea for finding such α is to maximize the achievable transmission rate at the highest noise standard deviation that the code can tolerate under MCLA. To do this, for a given code, we must find the largest σ_z, referred to as σ_z^*, such that

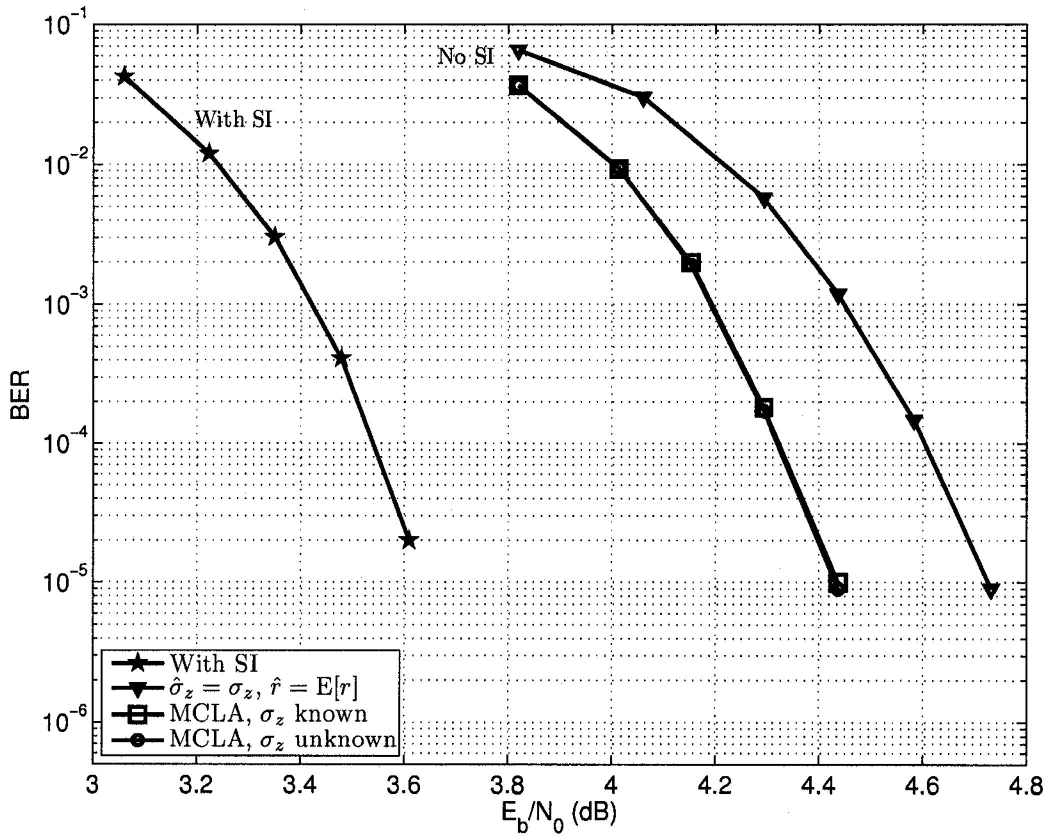


Figure 4.4: Comparison of BER for a $\mathcal{C}^{10^4}(x^2, x^5)$ LDPC code in different cases on a normalized Rayleigh fading channel. The performance of MCLA remains almost the same regardless of whether σ_z is known or not.

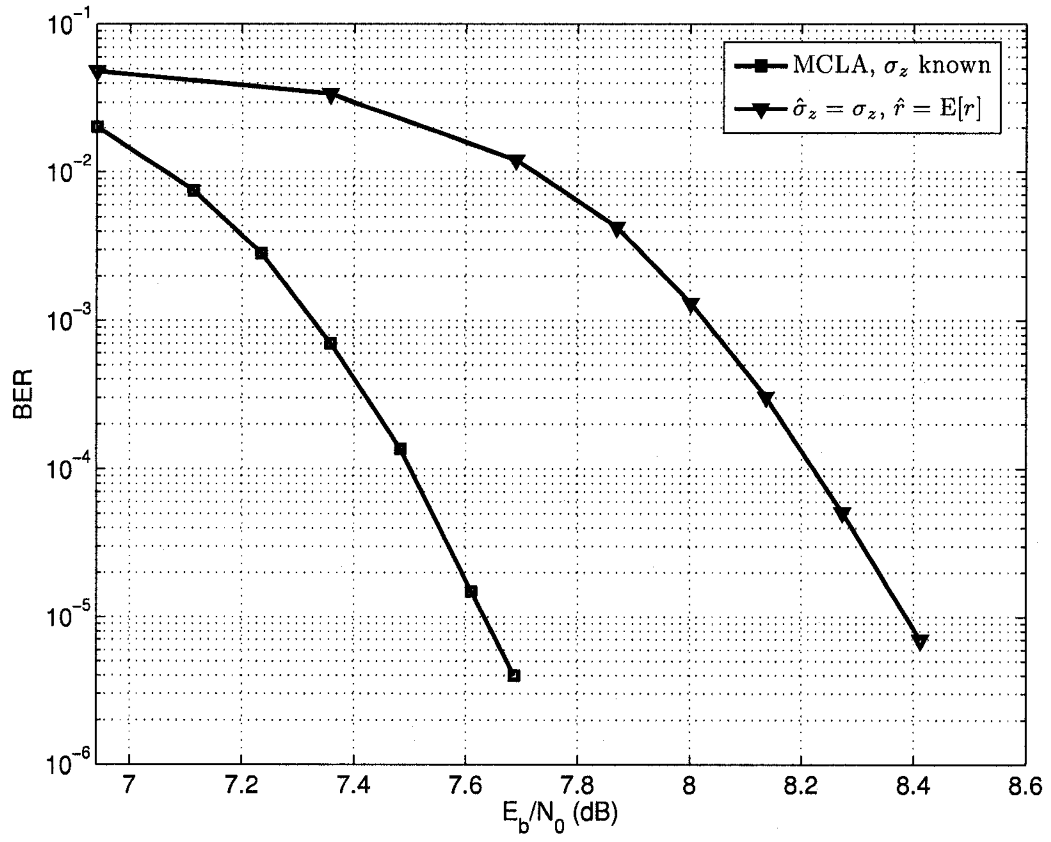


Figure 4.5: BER performance of a $\mathcal{C}^{10^4}(x^3, x^{15})$ LDPC code on a normalized Rayleigh fading channel when no SI is available at the receiver. There is about 0.75 dB performance improvement at BER of 10^{-5} under MCLA.

the code still converges to zero error rate when LLRs are obtained using (4.6) with $\alpha = \alpha_{\mathsf{opt}}(\sigma_z^*)$. Finding σ_z^* can be done efficiently through a binary search as follows:

1. Start by a small σ_{z1} and a large σ_{z2} and calculate $\alpha_{\mathsf{opt}}(\sigma_{z1})$ and $\alpha_{\mathsf{opt}}(\sigma_{z2})$ so that the density evolution converges using (4.8) with $\alpha = \alpha_{\mathsf{opt}}(\sigma_{z1})$ and does not converge using (4.8) with $\alpha = \alpha_{\mathsf{opt}}(\sigma_{z2})$.

2. If $\sigma_{z2} - \sigma_{z1} < \delta$, put $\sigma_z^* = \sigma_{z1}$ and $\alpha = \alpha_{\mathsf{opt}}(\sigma_{z1})$ and stop. Here, δ represents the binary search accuracy.

3. Calculate $\sigma_{z3} = \frac{\sigma_{z1}+\sigma_{z2}}{2}$ and $\alpha_{\mathsf{opt}}(\sigma_{z3})$. Run the density evolution using (4.8) with $\alpha = \alpha_{\mathsf{opt}}(\sigma_{z3})$. If it converges, let $\sigma_{z1} = \sigma_{z3}$. Otherwise, let $\sigma_{z2} = \sigma_{z3}$.

4. Go to step 2.

This choice of α gives the widest convergence range over σ_z, because it is the optimum α in the worst channel condition. When the channel condition improves, this choice of α is no longer optimum, but we expect that even with a sub-optimal α, convergence to be achieved due to improvement in the channel condition [66]. The fact that channel improvement outweighs sub-optimality of α can also be justified recalling that $\hat{r}_{\mathsf{opt}}$ (and hence α_{opt}) is not very sensitive to σ_z. Our simulation results on LDPC codes will confirm that this choice of α provides the widest convergence range.

In order to measure the performance we do as follows. For various σ_z and different values of α (including $\alpha_{\mathsf{opt}}(\sigma_z)$), we find the required number of density evolution iterations to achieve a target message error-rate (MER) p_t for a given LDPC code. We use the required number of iterations $\ell^*(p_t)$ as a comparison measure and to identify the range of convergence. Using density evolution, letting $\mathcal{D}^{(\ell)}_{\mathsf{v}\to\mathsf{c}}(\cdot)$ represent the pdf of the variable-to-check messages in iteration ℓ, and denoting the required number of iterations by $\ell^*(p_t)$, we have

$$\ell^*(p_t) \equiv \min\left\{\ell \in \mathbb{Z}^* : \int_{-\infty}^{0} \mathcal{D}^{(\ell)}_{\mathsf{v}\to\mathsf{c}}(\zeta)\cdot \mathrm{d}\zeta \leq p_t\right\}, \tag{4.11}$$

where $\mathbb{Z}^*$ represents the set of non-negative integers.

In Fig. 4.6, different values of α are used and $\ell^*(p_t)$ is plotted for different values of σ_z for $\mathcal{C}^\infty(x^2, x^5)$. It is seen that by using $\alpha = \alpha_{\mathsf{opt}}(\sigma_z^* = 0.6442) = 2.9634$, the code has the widest convergence range. Interestingly, while this choice of α is not optimum for all values of σ_z, the resulted $\ell^*(p_t)$ is always very close to the curve based on known σ_z under MCLA. This observation can also be made from Fig. 4.4 where unknown σ_z under MCLA results almost the same BER as MCLA with known σ_z.

4.4.1 Code design under MCLA

We proposed a method which gives the widest convergence region under MCLA for a given code even when the noise power is not known at the receiver. We also observed that the performance is extremely close to the case when the receiver knows the noise power. Now, the question is how one can design and optimize an LDPC code of a fixed rate which provides the widest convergence region under MCLA when σ_z is unknown at the receiver. This design procedure is a threshold maximizing design.

In conventional threshold maximizing design procedures [12,19,47,56], the optimization starts with a certain noise standard deviation σ_z and a set of initial degree distributions. In each round of the optimization, σ_z is increased slightly and the degree distributions are updated according to the rate constraint and the best degree distribution (giving the smallest error rate in density evolution) is found. This procedure stops when no degree distribution is found whose error rate converges to zero after so many density evolution iterations.

Our design procedure can be done in a similar way. The difference is that in each round of the optimization, for each σ_z, $\alpha_{\text{opt}}(\sigma_z)$ should be calculated and the LLR distribution is calculated using $\alpha = \alpha_{\text{opt}}(\sigma_z)$ in (4.8). Since when ρ is fixed, maximizing the code rate can be easily solved by linear programming, we apply the code rate maximization method to this problem.

Suppose that we want to design an LDPC code of rate R_0 which has the widest convergence region under MCLA. The procedure is outlined as follows:

1. Start with an initial degree distribution pair $(\lambda_0(x), \rho_0(x))$ and a large σ_z and let $\alpha = \alpha_{\text{opt}}(\sigma_z)$ such that the code's rate R given by (2.23) is lower than the target rate R_0 and the code is decoded successfully using density evolution and (4.8) as the initial channel LLR distribution.

2. Optimize the code's degree distributions by maximizing the code rate for the noise standard deviation σ_z and the LLR distribution in (4.8). The final rate is denoted by R.

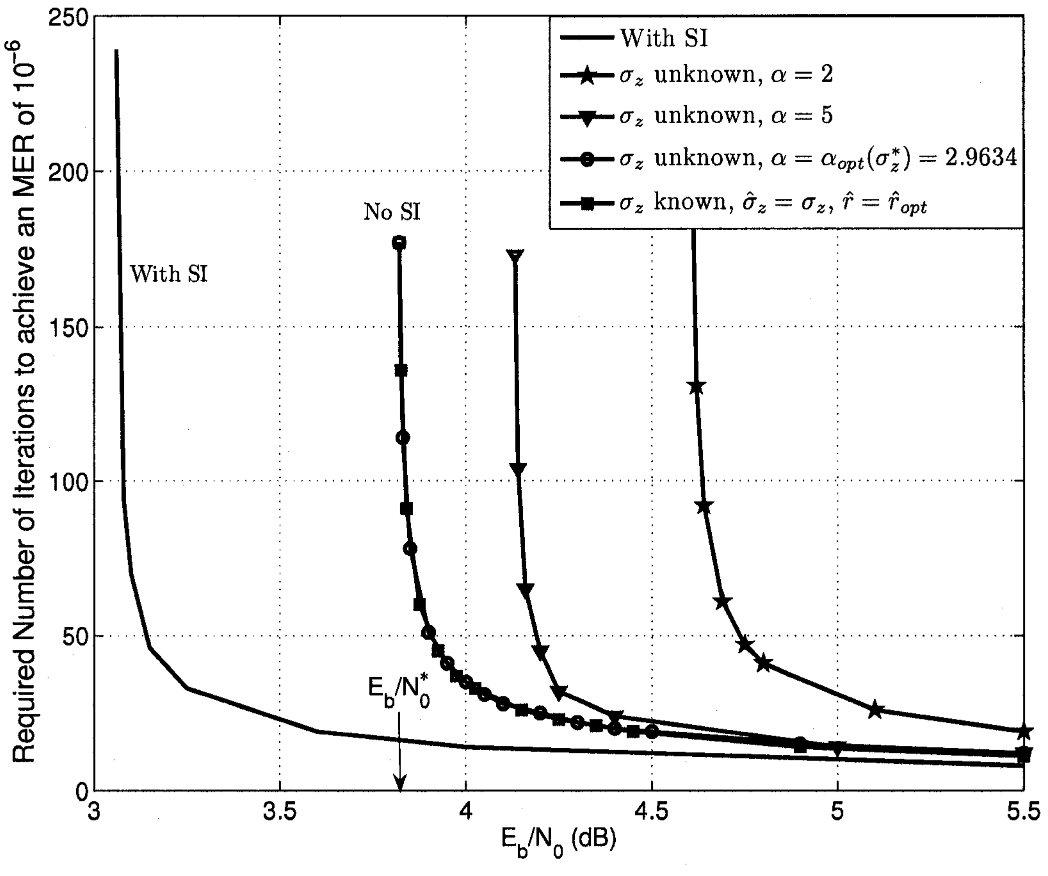


Figure 4.6: Comparison between the performances of $\mathcal{C}^{\infty}(x^2, x^5)$ on a normalized Rayleigh fading channel with or without SI using different α.

	Code1	Code2
λ_2	0.1916	0.1943
λ_3	0.2244	0.2341
λ_4	0.0057	0.0064
λ_5	0.0109	0.0113
λ_6	0.0427	0.0340
λ_7	0.1187	0.0994
λ_8	0.0297	0.0474
λ_9	0.0121	0.0205
λ_{10}	0.0147	0.0119
λ_{11}		0.0171
λ_{15}	0.0157	0.0228
λ_{20}	0.0314	0.0627
λ_{29}	0.0382	0.0680
λ_{30}	0.2649	0.1715
ρ_9	1.0000	1.0000
Rate	0.4889	0.5000
σ_z^*	0.7436	0.7274
$E_b/N_0{}^*$ (dB)	2.57	2.76

Table 4.1: Good LDPC codes designed under MCLA. Code1 is a rate maximized and Code2 is a threshold maximized code.

3. If $R > R_0$ then $\sigma_z := \sigma_z + \delta$ (δ is a small positive constant) and $\alpha = \alpha_{\mathsf{opt}}(\sigma_z)$ and go to step 2. If $R < R_0$ then $\sigma_z := \sigma_z - \delta$ and $\alpha = \alpha_{\mathsf{opt}}(\sigma_z)$ and go to step 2.

4. If $R = R_0$ then $\sigma_z^* = \sigma_z$ and stop.

This procedure gives the degree distributions of an ensemble of LDPC codes with rate R_0 which has the largest decoding threshold σ_z^* or the widest convergence region under MCLA. To decode this code, the decoder should use (4.6) with $\alpha = \alpha_{\mathsf{opt}}(\sigma_z^*)$.

Using the procedure outlined above, one designed code is reported in Table 4.1 (Code2). Again, 11-bit decoding under MCLA is used, the maximum number of iterations allowed is 300 and $d_v = 30$. The threshold of the designed code is 2.76 dB. This code has the largest decoding threshold among all the codes with the rate 0.5 when the fading gain and the noise power are not known at the receiver.

4.5 Conclusion

We proposed a new method for linear LLR calculation on fading channels when channel fading gain is not known at the receiver. Our method is optimum in the sense of maximum achievable rate on the channel. We showed that on a Rayleigh channel, the maximum achievable rate using this method is extremely close to the channel capacity. Compared to existing work, which uses the expected value of the fading gain for LLR calculation, we reported considerable performance improvement at no extra decoding cost.

We then extended our approach to the cases that the additive noise power of the fading channel is also unknown at the receiver. With a careful choice of linear LLR calculation, we were able to obtain a performance almost identical to the previous case, where the additive noise power was known.

For applications that channel estimation results in significant overheads or suffers from severe imperfections, our proposed solution can be of interest.

While we verified some of our results through study and design of LDPC codes on Rayleigh channel, our approach for maximizing the achievable transmission rate and convergence range of the decoder is general. The only reason for using LDPC codes is that, they can approach theoretical limits and thus verify some of our asymptotic results.

Chapter 5

Conclusion

In this chapter we summarize the results and the contributions of this thesis. We also present some new questions and discuss possible future research.

5.1 Contributions

This thesis has two main contributions in the field of LDPC codes and in general iterative decoding.

In the first contribution, we proposed a simple method for finite-length LDPC codes' analysis on symmetric channels. Using a time-varying interpretation of the channel for finite-length codes, we analyzed the channel behavior by modeling two important parameters of the observed channel as random variables. These two parameters were the error rate and capacity of the observed channel. Next, we derived the distribution of these random variables analytically. Using these distributions and assuming that the decoding failure is the result of an observed channel worse than the decoding threshold of the code, we were able to obtain the block error probability of finite-length LDPC codes.

Two different channel classes were studied. In the first class, i.e., one-dimensional channels, the channel variations can be fully modeled by a single parameter. In the second class, i.e., multi-dimensional channels, the channel variations cannot be modeled by a single parameter. We studied the effects of modeling multi-dimensional channels with one parameter and we analyzed the results. We concluded that the iterative decoder is more sensitive to the number of errors than the capacity and the error rate method predicts more accurate results.

Our results suggest that when the block length is a few thousand bits, even by ignoring the effects of cycles, we are able to predict the performance in the waterfall region closely.

We also proposed guidelines for choosing the decoding threshold of the code based on its block length which can be used in the design process of finite-length LDPC codes.

In the second contribution, we considered iterative decoding on fading channels when the channel fading gain is not known at the receiver. We proposed an optimum and efficient linear approximation method to approximate the channel LLRs. This method is optimum in the sense of maximum achievable rate on the channel, which was shown to be extremely close to the channel capacity on a Rayleigh fading channel. We designed LDPC codes for this purpose and showed that they can approach this rate.

We extended this method to the case when the channel fading gain and also the additive noise power are not known at the receiver. We showed that a performance almost identical to previous case, where the additive noise power was known, can be achieved. This is an interesting result for applications that channel estimation results in significant overheads or suffers from severe imperfections.

This method is applicable to iterative decoding in general and not only to LDPC codes. We used LDPC codes due to the fact that they can approach theoretical limits and thus verify some of our asymptotic results. We verified our results for Rayleigh fading channel. But our method is applicable to other uncorrelated fading channels such as uncorrelated Rician fading channel, uncorrelated Nakagami-m fading channel, etc.

5.2 Possible Future Research

In this section we present some new problems and briefly suggest possible future research direction.

Our analysis of the channel variations for finite-length codes in Chapter 3 can be used to analyze the behavior of other codes which show a threshold behavior. For example, the method can be extended to predict the performance of finite-length turbo codes and fountain codes [26]. Also, the channel analysis method can be used with some modifications in analyzing the non-asymptotic capacity of finite-length codes.

Our method of optimum linear LLR calculation is probably the first step in a promising research direction. We proposed the method of maximum-capacity linear LLR approximation for binary phase shift keying modulation on uncorrelated fading channels. Possible future work can be done on extending this method to higher order modulations.

The channels considered in our method were uncorrelated or flat fading channels. Another new problem is to propose this idea for frequency-selective or correlated fading channels. Other research directions can include extending this work to multi-level coding, bit-interleaved coded modulation (BICM) [69], orthogonal frequency division multiplexing (OFDM) systems, and multiple-input multiple-output (MIMO) channels.

One useful application of the general idea presented in Chapter 4 could be to develop new methods of detection and decoding when no channel parameter estimation (or limited estimation) is performed at the receiver. The approach could be based on a new definition of capacity, i.e., detection capacity. A capacity (a limit on data transmission rate) can be assigned to different detection strategies that are based on various assumptions on the channel parameters. Thus, we have a comparison measure which can be used for adjusting the detection parameters and finding the best strategy. Moreover, the concept of detection capacity can be extended to decoding too. This way, decoding can be optimized according to existence or absence of channel knowledge at the receiver. These solutions can be extremely important in time-varying channels where channel estimations techniques are subject to imperfections, cause a significant overhead and increase the receiver's complexity.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and Oct. 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes (1)," in *Conf. Rec. IEEE ICC '93*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.

[3] R. M. Tanner, "A recursive approach to low complexity codes." *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[4] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, Sweden, 1996.

[5] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 219–230, 1998.

[6] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm." *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 140–152, 1998.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[8] R. G. Gallager, "Low-Density Parity-Check codes," Ph.D. dissertation, M.I.T press, Cambridge, MA, 1963.

[9] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of Low Density Parity Check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, August 1996.

[10] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.

[11] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[12] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density perity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[13] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[14] 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 Trans. Inform. Theory*, vol. 47, no. 2, pp. 657–670, 2001.

[15] M. Ardakani and F. R. Kschischang, "A more accurate one-dimensional analysis and design of irregular LDPC codes," *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2106–2114, 2004.

[16] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, May 2001.

[17] M. Ardakani, T. H. Chan, and F. R. Kschischang, "EXIT-chart properties of the highest-rate LDPC code with desired convergence behavior," *IEEE Commun. Lett.*, vol. 9, no. 1, pp. 52–54, Jan. 2005.

[18] M. Ardakani, B. Smith, W. Yu, and F. R. Kschischang, "Complexity-optimized LDPC codes," in *Proceedings of the Allerton Conference on Communication, Control, and Computing*, Allerton, Sept. 2005.

[19] J. Hou, P. H. Siegel, and L. B. Milstein, "Performance analysis and code optimization of low-density parity-check codes on Rayleigh fading channels," *IEEE J. Select. Areas Commun.*, vol. 19, no. 5, pp. 924–934, May 2001.

[20] H. Futaki and T. Ohtsuki, "Low-density parity-check (LDPC) coded OFDM systems," in *Vehicular Technology Conference, 2001. IEEE VTC*, vol. 1, 2001, pp. 82–86.

[21] J. Hou, P. H. Siegel, L. B. Milstein, and H. D. Pfister, "Capacity-approaching bandwidth-efficient coded modulation schemes based on low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 9, pp. 2141–2155, Sept. 2003.

[22] A. Sanderovich, M. Peleg, and S. Shamai, "LDPC coded MIMO multiple access with iterative joint decoding," *IEEE Trans. Inform. Theory*, vol. 51, no. 4, pp. 1437–1450, April 2005.

[23] C. Di, D. Proietti, T. Richardson, E. Telatar, and R. Urbanke, "Finite length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570–1579, 2002.

[24] A. Amraoui, "Asymptotic and finite-length optimization of LDPC codes," Ph.D. dissertation, EPFL, Lausanne, 2006.

[25] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," in *Proceedings of the 36th Allerton Conference on Communication, Control, and Computing, Sept. 1998*, 1998, pp. 201–210.

[26] M. Luby, "LT codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, Allerton, 2002, pp. 271–282.

[27] A. Shokrollahi, "Raptor codes," Digital Fountain, Inc., Tech. Rep. DF2003-06-001, June 2003.

[28] "Digital Video Broadcasting (DVB)," European Standard (Telecommunication series), Std. ETSI EN 302 307 V1.1.1, 2004.

[29] "Optional B-LDPC coding for OFDMA PHY," IEEE Std. IEEE 802.16e-04/78, 2004.

[30] T. J. Richardson, "Error-floors of LDPC codes," in *Proc. of the 41st Annual Allerton Conference on Communications, Control and Computing*, Sept. 2003, pp. 1426–1435.

[31] R. Yazdani and M. Ardakani, "An efficient analysis of finite-length LDPC codes," accepted for publication in the Proc. of IEEE International Conference on Communications (ICC'07), Glasgow, Scotland, 2007.

[32] ——, "Analysis of finite-length LDPC codes on symmteric channels," 2007, submitted for publication in IEEE Transactions on Communications.

[33] ——, "Optimum linear LLR calculation for iterative decoding on fading channels," accepted for publication in the Proc. of IEEE International Symp. on Inform. Theory (ISIT'07), Nice, France, 2007.

[34] R. G. Gallager, *Information Theory and Reliable Communication.* New York, NY, USA: John Wiley & Sons, Inc., 1968.

[35] S. Lin and D. J. Costello, *Error Control Coding, Second Edition.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.

[36] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[37] M. Ardakani, "Efficient analysis, design and decoding of low-density parity-check codes," Ph.D. dissertation, University of Toronto, 2004.

[38] D. Burshtein, M. Krivelevich, S. Litsyn, and G. Miller, "Upper bounds on the rate of LDPC codes," *IEEE Trans. Inform. Theory*, vol. 48, pp. 2437–2449, Sep. 2002.

[39] I. Sason and G. Wiechman, "Performance versus complexity per iteration for low-density parity-check codes: An information-theoretic approach," in *Proc. of the Fourth International Symp. on Turbo Codes and Related Topics*, Munich, Germany, April 2006.

[40] H. D. Pfister, I. Sason, and R. Urbanke, "Capacity-achieving ensembles for the binary erasure channel with bounded complexity," *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2352–2379, July 2005.

[41] M. Ardakani and F. R. Kschischang, "Properties of optimum binary message-passing decoders," *IEEE Trans. Inform. Theory*, vol. 51, no. 10, pp. 3658–3665, Oct. 2005.

[42] J. Chen, R. M. Tanner, C. Jones, and Y. Li, "Improved min-sum decoding algorithms for irregular LDPC codes," in *Proc. of International Symposium on Information Theory*, Sept. 2005, pp. 449–453.

[43] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002.

[44] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, 2001.

[45] S. ten Brink, "Iterative decoding trajectories of parallel concatenated codes," in *Proc. 3rd IEE/ITG Conf. Source, Channel Coding*, Munich, Germany, Jan. 2000, pp. 75–80.

[46] L. Bazzi, T. Richardson, and R. Urbanke, "Exact thresholds and optimal codes for the binary symmetric channel and Gallager's decoding algorithm A," *IEEE Trans. Inform. Theory*, vol. 50, no. 9, pp. 2010–2021, 2004.

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

[48] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on density evolution," *IEEE J. Select. Areas Commun.*, vol. 19, no. 5, pp. 891–907, May 2001.

[49] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Trans. Inform. Theory*, vol. 52, no. 5, pp. 2033–2051, 2006.

[50] A. Roumy, S. Guemghar, G. Caire, and S. Verdu, "Design methods for irregular repeat-accumulate codes," *IEEE Trans. Inform. Theory*, vol. 50, no. 8, pp. 1711–1727, August 2004.

[51] D. Divsalar, S. Dolinar, and F. Pollara, "Low complexity turbo-like codes," in *Proc. 2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, 2000, pp. 73–80.

[52] H. E. Gamal and A. R. J. Hammons, "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 671–686, Feb. 2001.

[53] J. Zhang and A. Orlitsky, "Finite-length analysis of LDPC codes with large left degrees," in *Proc. IEEE International Symp. on Inform. Theory*, Lausanne, Switzerland, 2002.

[54] W. Yu, M. Ardakani, B. Smith, and F. Kschischang, "Complexity-optimized low-density parity-check codes for gallager decoding algorithm B," in *Proc. IEEE International Symp. on Inform. Theory*, Sept. 2005, pp. 1488–1492.

[55] R. Storn and K. Price, "Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces," Berkeley, CA, Tech. Rep. TR-95-012, 1995.

[56] A. Shokrollahi and R. Storn, "Design of efficient erasure codes with differential evolution," in *Proc. of GECCO'99.*

[57] J. W. Lee and R. E. Blahut, "Lower bound on BER of finite-length turbo codes based on EXIT characteristics," *IEEE Commun. Lett.*, vol. 8, no. 4, April 2004.

[58] T. J. Richardson, A. Shokrollahi, and R. Urbanke, "Finite-length analysis of various low-density parity-check ensembles for the binary erasure channel," in *Proc. IEEE International Symp. on Inform. Theory*, 2002.

[59] T. J. Richardson and R. Urbanke, *Modern Coding Theory*, 2007, preprint, available online at http://lthcwww.epfl.ch/mct/index.php.

[60] S. Huettinger and J. Huber, "Information processing characteristics and BER-bounds for concatenated coding schemes," *Euro. Trans. Telecomms.*, vol. 16, pp. 137–149, 2005.

[61] M. E. Hellman and J. Raviv, "Probability of error, equivocation, and the chernoff bound," *IEEE Trans. Inform. Theory*, vol. 16, pp. 368–372, July 1970.

[62] K. Fu and A. Anastasopoulos, "Performance analysis of LDPC codes for time-selective complex fading channels," in *GLOBECOM'02*, vol. 2, Nov. 2002, pp. 1279–1283.

[63] J. Lin and W. Wu, "Performance analysis of LDPC codes on Rician fading channels," *Higher Education Press and Springer-Verlag*, 2006.

[64] J. Xiaowei, A. W. Eckford, and T. E. Fuja, "Design of good low-density parity-check codes for block fading channels," in *MILCOM 2004*, vol. 2, 2004, pp. 1054–1059.

[65] E. K. Hall, "Performance and design of Turbo codes on Rayleigh fading channels," Master's thesis, University of Virginia, May 1996.

[66] P. Zarrinkhat, M. Ardakani, and R. Yazdani, "Channel estimation considerations for iterative decoding in wireless communications," accepted for publication in the *Proceeding of IEEE International Conference on Communications (ICC'07)*, Glasgow, Scotland, 2007.

[67] J. Hagenauer, "Viterbi decoding of convolutional codes for fading- and burst-channels," in *Zurich seminar on digital communications*, 1980.

[68] M. Ardakani, "On the properties of LLR on binary-input output-symmetric channels," 2007, submitted for publication in IEEE Commun. Lett.

[69] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation." *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 927–946, 1998.

Appendix A

Properties of LLR in Binary-input Memoryless Symmetric Channels

In this appendix, we briefly review the properties of LLR in binary-input memoryless symmetric (BIMS) channels. We state the relationship between the channel capacity and the probability density function (pdf) of the channel LLRs and present some theorems with their proofs. Most of the results of this appendix can be found in [47, 59].

The input binary symbols are $\{0, 1\}$, however, it is usually convenient to map the symbols to $\{-1, +1\}$ as in the binary phase-shift keying (BPSK) modulation. Thus, $0 \longleftrightarrow +1$ and $1 \longleftrightarrow -1$.

Definition 1: On a BIMS channel with transition probabilities $\mathrm{P}_{X|Y}(x|y)$, the associated channel LLR function $l(y)$ is given by

$$l(y) = \log \frac{\mathrm{P}_{X|Y}(+1|y)}{\mathrm{P}_{X|Y}(-1|y)}. \tag{A.1}$$

This is the function which maps the output y to the channel LLR. Since the channel output is a random variable Y, its associated LLR $l(Y)$ is also a random variable which has a pdf. It is worth mentioning that $l(y)$ is a sufficient statistics for decoding on BIMS channels. In the belief propagation decoding, the channel LLR messages which correspond to the intrinsic messages are calculated using (A.1).

Assuming that the all-zero codeword is transmitted over the channel, the pdf of $l(y)$ denoted by $f_L(l)$ can be computed for different BIMS channels. It is not hard to see that for the BEC(ϵ), $f_L(l) = \epsilon\delta(l) + (1-\epsilon)\delta(l-\infty)$ and for the BSC(ϵ),$f_L(l) =$

$\epsilon\delta(l - \frac{\epsilon}{1-\epsilon}) + (1-\epsilon)\delta(l - \frac{1-\epsilon}{\epsilon})$, where $\delta(\cdot)$ is the Dirac delta function. For the BIAWGN(σ) channel $l(y) = \frac{2}{\sigma^2} \cdot y$, which has a Gaussian distribution with mean $\frac{2}{\sigma^2}$ and variance $\frac{4}{\sigma^2}$. For the URF channel, depending on the availability of SI at the decoder, $l(y)$ is defined in (4.3) and (4.6) and its pdf is given by (3.15) and (4.8), respectively. These distributions can be used in density evolution as the intrinsic messages pdf to analyze the behavior of LDPC codes.

Definition 2 A pdf f is called consistent or symmetric if $f(-x) = e^{-x} f(x)$ for all $x \in \mathbb{R}$ [47].

Theorem 1 In BIMS channels, under the all-zero codeword assumption, the LLR pdf $f_L(l)$ is consistent [47].

Proof: From the channel symmetry condition we have

$$\begin{aligned} l(y) &= \log \frac{\mathrm{P}_{X|Y}(1|y)}{\mathrm{P}_{X|Y}(-1|y)} \\ &= \log \frac{\mathrm{P}_{Y|X}(y|1)}{\mathrm{P}_{Y|X}(y|-1)} \\ &= \log \frac{\mathrm{P}_{Y|X}(-y|-1)}{\mathrm{P}_{Y|X}(-y|1)} \\ &= \log \frac{\mathrm{P}_{X|Y}(-1|-y)}{\mathrm{P}_{X|Y}(1|-y)} = -l(-y). \end{aligned}$$

Thus,

$$\begin{aligned} f_L(-u) &= \mathrm{P}_{Y|X}(y \in l^{-1}(-u)|1) \\ &= \mathrm{P}_{Y|X}(-y \in l^{-1}(u)|1) \\ &= \mathrm{P}_{Y|X}(y \in l^{-1}(u)|-1) \\ &= e^{-u}\mathrm{P}_{Y|X}(y \in l^{-1}(u)|1) \\ &= e^{-u} f_L(u). \end{aligned}$$

It should be noted that the LLR pdf of the BEC, BSC, BIAWGN, and uncorrelated fading channels are all consistent. Another interesting fact is that the consistency of the densities is conserved under density evolution. In other words, the pdf of the extrinsic messages is consistent in each iteration of density evolution.

For a Gaussian pdf $\mathcal{N}(\mu, \sigma^2)$, the consistency condition can be simplified and is equivalent to $\sigma^2 = 2\mu$. As we stated, in the BIAWGN(σ) channel $l(y) = \frac{2}{\sigma^2} \cdot y$, which has a Gaussian distribution with mean $\frac{2}{\sigma^2}$ and variance $\frac{4}{\sigma^2}$. Therefore, the consistency condition holds.

Two BIMS channels are *equivalent* if their LLR pdfs are equal to each other [47]. Based on this definition we have the following lemma.

Lemma Channel Equivalence Lemma: If $f_L(y)$ is a consistent pdf and if for a BIMS channel $\mathrm{P}_{Y|X}(y|1) = f_L(y)$, then the channel LLR pdf is equal to $f_L(y)$.

Proof:

$$\begin{aligned} \log \frac{\mathrm{P}_{Y|X}(y|1)}{\mathrm{P}_{Y|X}(y|-1)} &= \log \frac{\mathrm{P}_{Y|X}(y|1)}{\mathrm{P}_{Y|X}(-y|1)} \\ &= \log \frac{f_L(y)}{f_L(-y)} \\ &= \log \frac{f_L(y)}{e^{-y} f_L(y)} = y. \end{aligned}$$

Definition 3 The error probability associated with the LLR pdf $f_L(l)$ is given by

$$p_{\text{error}} = \int_{-\infty}^{0} f_L(l)\mathrm{d}l. \tag{A.2}$$

This is a useful property since it enables us to calculate the error probability in each iteration of density evolution using the pdf of the extrinsic messages.

Now, we prove that the capacity of a BIMS channel, which has a consistent LLR pdf, can be calculated from the LLR pdf under the all-zero codeword assumption.

Theorem 2 The capacity of a BIMS channel is given via its LLR pdf $f_L(l)$ by [49,59]

$$C = 1 - \int_{\infty}^{\infty} \log_2 (1 + e^{-l}) f_L(l)\mathrm{d}l = 1 - \mathrm{E}_l[\log_2 (1 + e^{-l})]. \tag{A.3}$$

Proof: Due to the symmetry of the channel, an equiprobable distribution on X maximizes the mutual information. Thus,

$$\begin{aligned} C &= I(X;Y) = H(Y) - H(Y|X) \\ &= \int_{-\infty}^{\infty} \left(-\mathrm{P}_Y(y) \log_2 \mathrm{P}_Y(y) + \frac{1}{2} \sum_{x=\pm 1} \mathrm{P}_{Y|X}(y|x) \log_2 \mathrm{P}_{Y|X}(y|x) \right) \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \frac{1}{2} \sum_{x=\pm 1} \mathrm{P}_{Y|X}(y|x) \log_2 \frac{\mathrm{P}_{Y|X}(y|x)}{\frac{1}{2}\left(\mathrm{P}_{Y|X}(y|1) + \mathrm{P}_{Y|X}(y|-1)\right)} \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \mathrm{P}_{Y|X}(y|1) \log_2 \frac{\mathrm{P}_{Y|X}(y|1)}{\frac{1}{2}\left(\mathrm{P}_{Y|X}(y|1) + \mathrm{P}_{Y|X}(y|-1)\right)} \mathrm{d}y, \end{aligned}$$

where we have used the fact that $\mathrm{P}_{Y|X}(-y|-x) = \mathrm{P}_{Y|X}(y|x)$. Now, by the channel equivalence lemma we have $\mathrm{P}_{Y|X}(y|-1) = f_L(-y) = e^{-y} f_L(y)$ and thus,

$$C = \int_{-\infty}^{\infty} f_L(y) \log_2 \frac{f_L(y)}{\frac{1}{2} f_L(y)\left(1 + e^{-y}\right)} \mathrm{d}y = \int_{-\infty}^{\infty} f_L(y) \left(1 - \log_2(1 + e^{-y})\right) \mathrm{d}y.$$

Notice that Theorem 2 enables us to calculate the capacity of an arbitrary BIMS channel based on its LLR pdf. For an arbitrary consistent LLR pdf, say $g_L(l)$, (A.3) calculates the true capacity of a BIMS channel implied by $g_L(l)$.