UNIVERSITY OF CALIFORNIA, SAN DIEGO

Optimizing and Decoding LDPC codes with Graph-Based Techniques

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering
(Communications Theory and Systems)

by

Amir H. Djahanshahi

Committee in charge:

Professor Laurence B. Milstein, Co-Chair
Professor Paul H. Siegel, Co-Chair
Professor Pamela Cosman
Professor Patrick J. Fitzsimmons
Professor Alexander Vardy

2010
The dissertation of Amir H. Djahanshahi is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_________________________________________

_________________________________________

_________________________________________

_________________________________________

_________________________________________

Co-Chair

_________________________________________

Co-Chair

University of California, San Diego

2010
DEDICATION

To my parents and brother.
“A clever person solves a problem. A wise person avoids it.”

–ALBERT EINSTEIN
TABLE OF CONTENTS

Signature Page . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . iii
Dedication . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . iv
Epigraph . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . v
Table of Contents . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . vi
List of Figures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ix
Acknowledgements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xi
Vita and Publications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiii
Abstract of the Dissertation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiv
Chapter 1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 Graph-Based Linear Codes . . . . . . . . . . . . . . . . . . . . . 2
  1.3 Dissertation Overview . . . . . . . . . . . . . . . . . . . . . . . . 3
Bibliography . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
Chapter 2 LDPC Codes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  2.2 Definitions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
  2.3 BP Decoding . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
    2.3.1 Drawbacks of Cycles . . . . . . . . . . . . . . . . . . . . . . 11
Bibliography . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
Chapter 3 Joint Equalization and Decoding of LDPC Codes in MISO Systems . . 16
  3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
  3.2 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . . 17
    3.2.1 System Model . . . . . . . . . . . . . . . . . . . . . . . . . . 18
    3.2.2 EM Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  3.3 Graph Based Representation of Turbo-Type Receivers . . . . . 21
  3.4 Modified Tanner Graphs . . . . . . . . . . . . . . . . . . . . . . . . 23
    3.4.1 BP over Modified Tanner Graphs . . . . . . . . . . . . . . . 25
    3.4.2 Initialization . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
    3.4.3 Variable-to-Check Messages . . . . . . . . . . . . . . . . . 26
    3.4.4 Check-to-Variable Messages . . . . . . . . . . . . . . . . . 27
  3.5 Edge-Based Message-Passing . . . . . . . . . . . . . . . . . . . . . . 27
    3.5.1 Variable-to-Check Messages . . . . . . . . . . . . . . . . . 29
    3.5.2 Check-to-Variable Messages . . . . . . . . . . . . . . . . . 31
  3.6 Performance Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . 31
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A Tanner graph representing a non-binary LDPC code.</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>A Tanner graph representing a binary LDPC code.</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>(a) Update process at a variable node, (b) Update process at a check node.</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>The turbo-type receiver for MIMO-LDPC systems.</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>A three-layer Tanner graph.</td>
<td>21</td>
</tr>
<tr>
<td>3.3</td>
<td>(a) The Tanner graph associated with a binary parity-check matrix, (b) The corresponding modified Tanner graph when $Mn_1T = 2$.</td>
<td>24</td>
</tr>
<tr>
<td>3.4</td>
<td>(a) A three-layer Tanner graph, (b) A modified Tanner graph.</td>
<td>34</td>
</tr>
<tr>
<td>3.5</td>
<td>BER vs. $E_b/N_0$ for known and unknown channel scenarios.</td>
<td>36</td>
</tr>
<tr>
<td>3.6</td>
<td>BER vs. $E_b/N_0$ for various channel coherence times.</td>
<td>37</td>
</tr>
<tr>
<td>3.7</td>
<td>BER vs. $E_b/N_0$ for different pilot symbols powers.</td>
<td>38</td>
</tr>
<tr>
<td>3.8</td>
<td>BER vs. $E_b/N_0$ for 8-ary QAM and QPSK modulations.</td>
<td>39</td>
</tr>
<tr>
<td>3.9</td>
<td>(a) Computation of a variable-to-check message in a three-layer Tanner graph, (b) Computation of a message from a modified variable node to a modified check node in the modified Tanner graph.</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>Tanner graphs of the basic-binary-image, and an $M_2(GF(2))$ matrixing-image of a $GF(16)$ LDPC code.</td>
<td>66</td>
</tr>
<tr>
<td>4.2</td>
<td>Bit error rate curves of LDPC($n = 80, \lambda(x) = x^2, \rho(x) = x^5$) codes when used over AWGN channels.</td>
<td>71</td>
</tr>
<tr>
<td>4.3</td>
<td>Word error rate curves of LDPC($n = 80, \lambda(x) = x^2, \rho(x) = x^5$) codes when used over AWGN channels.</td>
<td>71</td>
</tr>
<tr>
<td>4.4</td>
<td>Bit error rate curves of LDPC($n = 240, \lambda(x) = x^2, \rho(x) = x^5$) codes when used over AWGN channels.</td>
<td>72</td>
</tr>
<tr>
<td>4.5</td>
<td>Word error rate curves of LDPC($n = 240, \lambda(x) = x^2, \rho(x) = x^5$) codes when used over AWGN channels.</td>
<td>72</td>
</tr>
<tr>
<td>4.6</td>
<td>Bit error rate curves of LDPC($n = 80, \lambda(x) = x^2, \rho(x) = x^3$) codes when used over AWGN channels.</td>
<td>73</td>
</tr>
<tr>
<td>4.7</td>
<td>Word error rate curves of LDPC($n = 80, \lambda(x) = x^2, \rho(x) = x^3$) codes when used over AWGN channels.</td>
<td>73</td>
</tr>
<tr>
<td>4.8</td>
<td>Bit error rate curves of LDPC($n = 240, \lambda(x) = x^2, \rho(x) = x^3$) codes when used over AWGN channels.</td>
<td>74</td>
</tr>
<tr>
<td>4.9</td>
<td>Word error rate curves of LDPC($n = 240, \lambda(x) = x^2, \rho(x) = x^3$) codes when used over AWGN channels.</td>
<td>74</td>
</tr>
<tr>
<td>4.10</td>
<td>Block diagrams of the encoder and the decoder of an extended-binary-image code.</td>
<td>77</td>
</tr>
<tr>
<td>5.1</td>
<td>Bit erasure rate curves of degree-3 variable nodes of LDPC($n, \lambda(x) = \frac{2}{3} x^2 + \frac{12}{25} x^3 + \frac{3}{25} x^8, \rho(x) = \frac{7}{15} x^6 + \frac{8}{15} x^7$) codes when used over binary erasure channel of erasure probability $\epsilon$.</td>
<td>95</td>
</tr>
</tbody>
</table>
Figure 5.2: Bit erasure rate curves of degree-4 variable nodes of LDPC($n$, $\lambda(x) = \frac{2}{5}x^2 + \frac{12}{25}x^3 + \frac{3}{25}x^8$, $\rho(x) = \frac{7}{15}x^6 + \frac{8}{15}x^7$) codes when used over binary erasure channel of erasure probability $\epsilon$.

Figure 5.3: Bit erasure rate curves of degree-9 variable nodes of LDPC($n$, $\lambda(x) = \frac{2}{5}x^2 + \frac{12}{25}x^3 + \frac{3}{25}x^8$, $\rho(x) = \frac{7}{15}x^6 + \frac{8}{15}x^7$) codes when used over binary erasure channel of erasure probability $\epsilon$.

Figure 5.4: Bit erasure rate curves of degree-3 variable nodes of LDPC($n$, $\lambda(x) = \frac{7}{15}x^2 + \frac{8}{15}x^7$, $\rho(x) = x^8$) codes when used over binary erasure channel of erasure probability $\epsilon$.

Figure 5.5: Bit erasure rate curves of degree-8 variable nodes of LDPC($n$, $\lambda(x) = \frac{7}{15}x^2 + \frac{8}{15}x^7$, $\rho(x) = x^8$) codes when used over binary erasure channel of erasure probability $\epsilon$.

Figure 5.6: Bit erasure rate curves of degree-3 and degree-13 variable nodes of LDPC($n = 8000$, $\lambda(x) = \frac{57}{70}x^2 + \frac{13}{70}x^{12}$, $\rho(x) = x^6$) codes when used over binary erasure channel of erasure probability $\epsilon$. 
ACKNOWLEDGEMENTS

As I am looking back, I feel fortunate to benefit from the help of so many people during this six-year journey.

I would like to thank my advisors Professor Laurence B. Milstein and Professor Paul H. Siegel for their support, guidance, and encouragements. I have immensely benefited from their vast knowledge, precious ideas, and advice throughout these years. I owe deepest gratitude to Professor Laurence B. Milstein for his warm personality, meticulous attitude, and his confidence in his students. I also would like to express my deepest appreciation to Professor Paul H. Siegel for his kindness, patience, and wisdom.

I am grateful to Professor Patrick J. Fitzsimmons for his insightful discussions which contributed a lot to this dissertation. I am thankful to Professor Alexander Vardy for generously sharing his great insight and experience in coding theory with me. I also would like to thank Professor Pamela Cosman for her time, help, and effort in serving on my PhD defense committee.

My special thanks go to Professor Jason Schweinsberg, Professor Daniel Rogalski, and Professor Rüdiger Urbanke from whom I have learned a lot.

I wish to thank the ECE and CMRR staff for their excellent administrative support, especially Betty Manoulian, M’Lissa Michelson, Gennie Miranda, Karol Previte, Megan Scott, and Iris Villanueva.

I would like to thank my colleagues at the STAR lab and friends in UCSD for providing a stimulating and enjoyable environment for research and learning. I am grateful to Aravind for our fruitful discussions, especially, for the joint work on the non-binary LDPC codes. I am also grateful to Ali, Brian, Eric, Eirik, Eitan, Federica, Ghazaleh, Han, Hao, Henry, Hossein, Ido, Ismail, Joseph, Junsheng, Koohyar, Mehrdad, Mohammad, Mona, Navid, Ori, Panu, Qi, Raheleh, Rathinakumar, Saeed, Seyhan, Shadi, Shahram, Sharon, Solmaz, Toshio, Xiaojie, Yan, Zeinab, Zheng, and Zsigmond.

Finally, I owe my greatest gratitude to my family, and this dissertation is dedicated to them.

This research was supported in part by the LG Electronics, the Center for Wireless Communications at UCSD, the UC Discovery program, and the NSF under grant No. CCF-0829865.

Chapter 3, in part, is a reprint of the material of the following paper: A. H. Dja-
VITA

2002 B.Sc., Electrical Engineering (Communication Theory and Systems), Sharif University of Technology, Tehran, Iran.

2004 M.Sc., Electrical Engineering (Communication Theory and Systems), Sharif University of Technology, Tehran, Iran.

2010 Ph.D., Electrical Engineering (Communication Theory and Systems), University of California, San Diego.

PUBLICATIONS


ABSTRACT OF THE DISSERTATION

Optimizing and Decoding LDPC codes with Graph-Based Techniques

by

Amir H. Djahanshahi

Doctor of Philosophy in Electrical Engineering

(Communications Theory and Systems)

University of California San Diego, 2010

Professor Laurence B. Milstein, Co-Chair
Professor Paul H. Siegel, Co-Chair

Low-density parity-check (LDPC) codes have been known for their outstanding error-correction capabilities. With low-complexity decoding algorithms and a near-capacity performance, these codes are among the most promising forward error correction schemes.

LDPC decoding algorithms are generally sub-optimal and their performance not only depends on the codes, but also on many other factors, such as the code representation. In particular, a given non-binary code can be associated with a number of different field or ring image codes. Additionally, each LDPC code can be described with many different Tanner graphs. Each of these different images and graphs can possibly lead to a different performance when used with iterative decoding algorithms. Consequently, in this dissertation we try to find better representations, i.e., graphs and images, for LDPC codes.

We take the first step by analyzing LDPC codes over multiple-input single-output (MISO) channels. In an $n_T \times 1$ MISO system with a modulation of alphabet size $2^M$, each group of $n_T$ transmitted symbols are combined and produce one received symbol at the receiver. As a result, we consider the LDPC-coded MISO system as an LDPC code over a $2^{Mn_T}$-ary alphabet. We introduce a modified Tanner graph to represent MISO-LDPC systems and merge the MISO symbol detection and binary LDPC decoding steps into a single message passing decoding algorithm. We present an ef-
icient implementation for belief propagation decoding that significantly reduces the decoding complexity. With numerical simulations, we show that belief propagation decoding over modified graphs outperforms the conventional decoding algorithm for short length LDPC codes over unknown channels.

Subsequently, we continue by studying images of non-binary LDPC codes. The high complexity of belief propagation decoding has been proven to be a detrimental factor for these codes. Thereby, we suggest employing lower complexity decoding algorithms over image codes instead. We introduce three classes of binary image codes for a given non-binary code, namely: basic, mixed, and extended binary image codes. We establish upper and lower bounds on the minimum distance of these binary image codes, and present two techniques to find binary image codes with better performance under the belief propagation decoding algorithm. In particular, we present a greedy algorithm to find optimized binary image codes.

We then proceed by investigation of the ring image codes. Specifically, we introduce matrix-ring-image codes for a given non-binary code. We derive a belief propagation decoding algorithm for these codes, and with numerical simulations, we demonstrate that the low-complexity belief propagation decoding of optimized image codes has a performance very close to the high complexity belief propagation decoding of the original non-binary code.

Finally, in a separate study, we investigate the performance of iterative decoders over binary erasure channels. In particular, we present a novel approach to evaluate the inherent unequal error protection properties of irregular LDPC codes over binary erasure channels. Exploiting the finite-length scaling methodology that has been used to study the average bit error rate of finite-length LDPC codes, we introduce a scaling approach to approximate the bit erasure rates in the waterfall region of variable nodes with different degrees. Comparing the bit erasure rates obtained from Monte Carlo simulation with the proposed scaling approximations, we demonstrate that the scaling approach provides a close approximation for a wide range of code lengths. In view of the complexity associated with the numerical evaluation of the scaling approximation, we also derive simpler upper and lower bounds and demonstrate through numerical simulations that these bounds are very close to the scaling approximation.
Chapter 1

Introduction

1.1 Background

In recent years, communication systems have experienced serious changes due to the new demands and requirements of the wireless era. These changes, yet again, draw attention to one of the oldest and most important goals of communication theory, that is the efficient transmission of information over unreliable channels.

More than sixty years ago, Shannon introduced the concept of reliable data transmission over noisy channels [1]. He proposed a system consisting of an encoder, a channel, and a decoder, and tried to find the minimum transmission redundancy necessary for diminishing the error probability. Shannon showed that, given a communication channel, there exists a limiting data rate, called the capacity of the channel, beyond which reliable transmission is impossible. However, for data rates below the channel capacity information can be transmitted reliably.

The noisy channel coding theorem [1] guaranteed existence of a coding scheme whose decoding error diminishes exponentially in the block-length of the code. However, it did not provide any practical approach for designing the best finite-length codes. Moreover, the random coding scheme proposed in the proof turned out to be a computationally intensive algorithm both in the encoding and decoding parts. Following Shannon’s seminal paper, there has been a large body of work on developing codes that are easy to encode and decode and could approach the channel capacity at the same time.

Elias [2] showed that linear codes, codes whose codewords constitute a linear
vector space over a finite field, can approach the capacity of discrete memoryless channels. Codewords of these codes belong to the null-space of a matrix, i.e., the parity-check matrix of the code. Consequently, these codes can be described based on their parity-check matrices as well. Linear codes can be encoded through the multiplication of the information vector with a generator matrix. As a result, these codes can considerably reduce the encoding complexity too. These facts paved the way for designing linear codes with polynomial decoding time, such as Hamming [3], BCH [4], Reed-Solomon [5], and convolutional codes. Although these codes find practical value, none of them could approach the capacity of additive white Gaussian noise (AWGN) channels. The discovery of capacity-approaching Turbo codes [6] and rediscovery of low-density parity-check (LDPC) codes [7], [8] marked the beginning of a new coding era.

Turbo codes, with a performance just tenths of a decibel shy of the channel capacity [6], established the benchmark of the capacity-approaching coding age. This trend was shortly followed by the rediscovery of LDPC codes. Randomness in design and long block-length, two key properties of Shannon’s random codes, appeared to be the connection between the capacity-approaching codes. These codes achieved outstanding performance close to the channel capacity with an acceptable complexity, and since then, analysis and design of encoding and decoding algorithms for these codes have become one of the most traveled research avenues.

1.2 Graph-Based Linear Codes

Tanner formulated the code design over graphical models in 1981 [9]. One year later, Pearl proposed the belief propagation (BP) algorithm as a message passing algorithm for performing inference over Bayesian networks [10]. Currently, BP is widely used in a variety of engineering disciplines. In fact, most of the signal processing, artificial intelligence, and digital communication algorithms such as Viterbi algorithm, turbo decoding, Kalman filtering [11] and fast Fourier transform are all special cases of BP performed over suitable graphs [12]. BP has also been shown to have practical success in various applications such as decoding of LDPC codes.

Essentially, the error probability behavior of a BP decoder depends on the graph that represents the code. On acyclic graphs, the BP decoder is the optimum decoder [13]. However, the presence of short cycles in the code graph can noticeably
degrade the performance of BP decoders [14].

A number of different graphs can represent the same code. This is easily seen by noting that the parity-check matrix of a code is not a unique matrix. Each row of the parity-check matrix is a constraint that codewords must satisfy. Thereby, the parity-check matrix can alternatively be viewed as a set of linear homogeneous equations called the parity-check equations. Since addition of any two parity-check equations generates another valid parity-check equation, a number of different parity-check matrices can represent the same code.

In the case of non-binary LDPC codes, numerous different field or ring image codes can be associated with a given code. However, each of these images can possibly lead to a different performance when decoded with the BP algorithm.

It is of interest from both theoretical and practical points of view to quantify the tradeoffs between the complexity and performance of decoders. With this as motivation, this dissertation addresses the following question. How can we find better representations (graphs or images, to be precise) for a given code? In particular, we first try to find better representations for binary LDPC codes over multiple-input single-output (MISO) channels, and then set off to find better images of non-binary LDPC codes.

It is known that the performance of BP decoding over erasure channels is mainly governed by the weight distribution of stopping sets [15, 16]. Furthermore, adding redundant parity-checks can improve the performance of BP decoding over these channels. For AWGN channels, [17] shows that in certain scenarios adding redundant parity-check equations can improve the performance of BP decoding. However, finding the best representation for a given code is a hard problem in general, and adding redundant parity-checks does not always improve the decoding performance. In fact, [18] presents scenarios where adding additional check equations degrades the performance of the BP decoder.

1.3 Dissertation Overview

Chapter 2 is dedicated to an introduction on LDPC codes. In Chapter 3, we establish a new approach to represent and decode LDPC codes over MISO channels. Since in an $n_T \times 1$ MISO system with a modulation of alphabet size $2^M$, each group of $n_T$ transmitted symbols are combined and produce one received symbol at the receiver,
we consider the LDPC-coded MISO system as an LDPC code over a $2^{Mn_T}$-ary alphabet. We introduce a modified Tanner graph to represent MISO-LDPC systems and merge the MISO symbol detection and binary LDPC decoding steps into a single message passing decoding algorithm. We also present an efficient implementation for BP decoding that significantly reduces the decoding complexity. With numerical simulations, we show that the proposed decoder outperforms conventional decoders for short length LDPC codes over unknown channels.

In Chapter 4, we take some steps towards finding the best images of non-binary LDPC codes. Non-binary LDPC codes have been shown to provide superb error-correcting performance. However, the high complexity of BP decoding has proven to be a detrimental factor in the implementation of these codes. To get around this problem, we suggest employing lower-complexity decoding algorithms over image codes. Specifically, we introduce three classes of binary image codes for a given non-binary code, and derive upper and lower bounds on the minimum distance of these binary image codes. We also present two techniques to reduce the number of four-cycles in Tanner graphs of these codes. To better understand the complexity-performance tradeoffs, we also consider images over intermediate fields and rings. In particular, we introduce $\mathbb{M}_b(GF(2))$ matrix-ring-image codes for a given non-binary code, where $\mathbb{M}_b(GF(2))$ is the ring of $b \times b$ binary square matrices. We derive a BP decoding algorithm for these codes, and present a technique to obtain better images. With numerical simulations, we demonstrate that the low-complexity BP decoding of “optimized” image codes has a performance very close to the high-complexity BP decoding of the original non-binary code.

Chapter 5, with a view to better understand the performance of finite-length LDPC codes, presents a novel approach to evaluate the inherent unequal error protection (UEP) properties of irregular LDPC codes over binary erasure channels (BECs). Exploiting the finite-length scaling methodology, suggested by Amraoui et. al. [19], we introduce a scaling approach to approximate the bit erasure rates in the waterfall region of variable nodes with different degrees. Comparing the bit erasure rates obtained from Monte Carlo simulation with the proposed scaling approximations, we demonstrate that the scaling approach provides a close approximation for a wide range of code lengths (between 1000 and 8000, to be precise). In view of the complexity as-
associated with the numerical evaluation of the scaling approximation, we also derive simpler upper and lower bounds and demonstrate through numerical simulations that these bounds are very close to the scaling approximation.

Bibliography


Chapter 2

LDPC Codes

2.1 Introduction

LDPC codes were originally invented by Gallager in his PhD thesis [1]. Soon after, due to the computational and storage requirements, they were mainly forgotten for almost thirty years, except for a few significant contributions. Zyablov and Pinsker established a decoding algorithm, similar to the Peeling Algorithm [2], for decoding of codes with sparse parity-check matrices [3]. Then, Tanner generalized Gallager’s construction of LDPC codes and employed a bipartite graph for representation of these codes [4]. Subsequently, Wiberg established the relations between different message passing algorithms [5]. And, MacKay showed that LDPC codes could outperform the best turbo codes [6].

Analysis of LDPC codes over erasure channels [2] motivated the design of LDPC codes with performance very close to the Shannon limit over erasure channels [7]. This was followed by analysis of LDPC codes over AWGN channels [8] and design of capacity-approaching codes over these channels [9]. Later, these concepts were generalized to other channels as well [10] - [12].

In this chapter, we briefly introduce LDPC codes and discuss the detrimental effects upon the decoder performance caused by short cycles in the bipartite graph of LDPC codes.
2.2 Definitions

Linear codes are codes whose codewords constitute a linear vector space over a field $\mathbb{F}$. In particular, a code $C$ over the field $\mathbb{F}$ is called linear if

\[
\forall x_1, x_2 \in C, \forall \alpha_1, \alpha_2 \in \mathbb{F} : \alpha_1 x_1 + \alpha_2 x_2 \in C.
\]

(2.1)

Assume that $C$ has a block-length of $n$. Consequently, codewords of $C$ belong to $\mathbb{F}^n$, and there exists an integer $k$, called the dimension of the code, such that

\[
|C| = |\mathbb{F}|^k.
\]

(2.2)

Codewords of linear codes belong to the null-space of a matrix, named the parity-check matrix of the code. Consequently, these codes can be described based on their parity-check matrices as well.

The linear code $C$ is, alternatively, defined as the set of codewords $x = (x_1, x_2, \ldots, x_n)$ over $\mathbb{F}^n$ that satisfy the parity-check equation

\[
H x^T = 0_{n-k}.
\]

(2.3)

Here, $H$ is an $(n - k) \times n$ parity-check matrix with elements from $\mathbb{F}$, and $0_{n-k}$ is the $(n - k) \times 1$ all-zero vector.

LDPC codes are linear codes that have a sparse parity-check matrix. These codes, like any linear code, can be represented by a bipartite graph called the Tanner graph of the code. The $n$ columns of the parity-check matrix correspond to the variable nodes and the $n - k$ rows of the parity-check matrix to the check nodes of the Tanner graph. There is an edge between variable node $j$ and check node $i$ if $H_{i,j} \neq 0$ and this edge is associated with the constant $H_{i,j}$. For instance, Figure 2.1 represents the Tanner graph of a length-5 LDPC code over $GF(4)$, with parity-check matrix

\[
H = \begin{bmatrix}
\alpha & \alpha^2 & \alpha^3 & 0 & 0 \\
0 & \alpha & \alpha^2 & \alpha^3 & 0 \\
0 & 0 & \alpha & \alpha^2 & \alpha^3
\end{bmatrix},
\]

(2.4)

where $\alpha$ is a primitive element of $GF(4)$.

In binary LDPC codes, that is LDPC codes over $GF(2)$, all the edge constants are the same, namely 1, so the edge constants are not represented. As an example,
Figure 2.1: A Tanner graph representing a non-binary LDPC code.

Figure 2.2 shows the Tanner graph of a length-7, Hamming code associated with the parity-check matrix

\[
H = \begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}.
\] (2.5)

Let us define the degree of a node as the number of edges that are attached to the node. A \((d_v, d_c)\)-regular LDPC code is defined as an LDPC code such that every variable node has degree \(d_v\) and every check node has degree \(d_c\).

In general, the degrees of the nodes can be different. Irregular LDPC codes are codes whose variable nodes, or correspondingly check nodes, have different degrees. An LDPC code is often described by two degree distributions:

\[
\lambda(x) = \sum_{i=1}^{l_{\text{max}}} \lambda_i x^{i-1}, \quad (2.6)
\]

and

\[
\rho(x) = \sum_{i=1}^{r_{\text{max}}} \rho_i x^{i-1}, \quad (2.7)
\]

where \(\lambda_i\) (\(\rho_i\), respectively) gives the fraction of edges connected to variable nodes (check nodes, respectively) of degree \(i\), and \(l_{\text{max}}\) (\(r_{\text{max}}\), respectively) is the maximum variable node (check node, respectively) degree.
Figure 2.2: A Tanner graph representing a binary LDPC code.

Similar to edge perspective degree distributions, node perspective degree distributions can be defined for both variable and check nodes. Define

$$\Lambda(x) \triangleq \sum_{i=1}^{l_{\text{max}}} \Lambda_i x^i = \int_0^x \frac{\lambda(u) du}{\int_0^1 \lambda(u) du},$$

(2.8)

as the variable node perspective degree distribution, where $\Lambda_i$ for $1 \leq i \leq l_{\text{max}}$ represents the fraction of degree-$i$ variable nodes relative to the total number of variable nodes in the Tanner graph. Further, let

$$P(x) \triangleq \sum_{i=1}^{r_{\text{max}}} P_i x^i = \int_0^x \frac{\rho(u) du}{\int_0^1 \rho(u) du},$$

(2.9)

denote the check node perspective degree distribution, where $P_i$ for $1 \leq i \leq r_{\text{max}}$ represents the fraction of degree-$i$ check nodes relative to the total number of check nodes in the Tanner graph.

We generate LDPC codes based on their degree distributions [14]. Specifically, we first define $n$ variable and $n-k$ check nodes. Then, degrees of variable nodes are assigned such that a fraction $\lambda_i$ of the edges is connected to degree-$i$ variable nodes for $i = 1, 2, \ldots, l_{\text{max}}$. Similarly, degrees of check nodes are assigned such that a fraction $\rho_i$ of
the edges is connected to degree-\(i\) check nodes for \(i = 1, 2, \ldots, r_{\text{max}}\). Assume \(E\) denotes the total number of edges in the Tanner graph. We enumerate the edges connected to variable nodes and the edges connected to check nodes from 1 to \(E\). Then, using a pseudo random permutation, we connect the corresponding edges to each other.

2.3 BP Decoding

While maximum-likelihood decoding often has a complexity exponential in the code length, BP decoding provides an optimum performance over acyclic Tanner graphs with complexity linear in the code length.

The BP decoding algorithm operates over Tanner graphs of LDPC codes and proceeds by applying Bayes’ rule locally and iteratively to update the messages over the edges of these graphs [15].

Consider a binary LDPC code transmitted over a binary-input AWGN (BI-AWGN) channel. BP tries to calculate the log-likelihood ratios of bits based on the Tanner graph of the code and runs as described in Algorithm 2.1. Note that BP over BI-AWGN channels is commonly referred to as the sum-product algorithm.

At the start of the sum-product algorithm, the channel messages \(n_c^{(i)}\), for \(i = 1, 2, \ldots, n\), are initialized based on the received information from the channel, i.e., \(y^{(i)}\). In subsequent iterations, variable nodes update the log-likelihood ratio messages based on the received check-to-variable messages and the channel message. Then, the check nodes update the check-to-variable messages based on the received variable-to-check messages.

In order to update a message on an outgoing edge, a node uses all the incoming messages except for the incoming message along that same edge, see Figure 2.3. This is reminiscent of turbo decoding [16], and aims to ensure the circulation of extrinsic information.

2.3.1 Drawbacks of Cycles

The concentration theorem [17] asserts that most randomly generated LDPC codes tend to perform close to the ensemble average for long-enough LDPC codes. Moreover, for long-enough codes, this average behavior is close to the performance of
Algorithm 2.1 Sum-product algorithm [13, 14]

Initialization:
Initialize all messages in the graph to 0.
Initialize the channel message at the $i$th variable node to $m^{(i)}_{c} = \log \frac{\Pr(y^{(i)}|x^{(i)}=0)}{\Pr(y^{(i)}|x^{(i)}=1)}$.

Decoding:
for $l := 1$ to $l$
do 
  for $i := 1$ to $n$
do 
    Consider the variable node $v_i$ of degree $d$.
    Assume $m^{(i)}_{c}$ denotes the channel message.
    Let $m^{(j)}$ ($m^{(j)}_{o}$, respectively) for $j = 1, 2, \ldots, d$ represent the incoming (outgoing, respectively) messages.
    Update the outgoing variable-to-check messages from $v_i$ according to:
    \[ m^{(j)}_{o} = m^{(i)}_{c} + \sum_{t=1, t \neq j}^{d} m^{(t)} \]
  end for
  for $i := 1$ to $n - k$
do 
    Consider the check node $c_i$ of degree $d$.
    Let $m^{(j)}$ ($m^{(j)}_{o}$, respectively) for $j = 1, 2, \ldots, d$ represent the incoming (outgoing, respectively) messages.
    Update the outgoing check-to-variable messages from $c_i$ according to:
    \[ m^{(j)}_{o} = 2\arctanh \prod_{t=1, t \neq j}^{d} \tanh \frac{m^{(t)}}{2} \]
  end for
end for

Termination:
for $i := 1$ to $n$
do 
  Consider the variable node $v_i$ of degree $d$.
  if $m^{(i)}_{c} + \sum_{t=1, t \neq j}^{d} m^{(t)} > 0$ then
    decode $x_i$ to 0.
  else 
    if $m^{(i)}_{c} + \sum_{t=1, t \neq j}^{d} m^{(t)} < 0$ then
      decode $x_i$ to 1.
    else 
      decode $x_i$ with probability 0.5 to 0 and with probability 0.5 to 1.
    end if
  end if
end for
acyclic codes. However, short LDPC codes generally can contain a number of short cycles. These short cycles can significantly degrade the performance of message-passing decoders [18]. Consequently, a number of girth-conditioning\(^1\) techniques have been proposed to design finite-length LDPC codes with improved performance [18] - [21].

**Bibliography**


\(^1\)The girth of a graph is the length of the smallest cycle of the graph.


Chapter 3

Joint Equalization and Decoding of LDPC Codes in MISO Systems

3.1 Introduction

Due to the increasing demand for multimedia data transmission in wireless communications, there is an evident need for increasing the spectral efficiency of wireless channels [1]. MISO systems, with their high spectral efficiency, appear to be an indispensable part of most future cellular systems. In addition, spatial diversity obtained by multiple transmit antennas is crucial for mitigating the effects of fading in wireless channels [2]. Thereby, this chapter is devoted to joint detection and decoding of LDPC-coded MISO systems.

Inspired by the success of turbo codes [3], a turbo-type architecture which we call the turbo-type receiver, can be employed for joint detection and decoding of MIMO-LDPC\(^1\) systems [4] - [9]. The turbo-type receiver is obtained by considering the MIMO symbol detector as the first component of the turbo architecture and the LDPC decoder as the second component.

To exploit the BP algorithm for decoding in turbo-type receivers, we establish a graph-based representation, named the three-layer Tanner graph, and show that decoding using the turbo-type receiver is equivalent to BP decoding over the corresponding three-layer Tanner graph.

\(^1\)Note that MISO-LDPC is a special case of MIMO-LDPC.
In an $n_T \times 1$ MISO system, with a modulation of alphabet size $2^M$, each $n_T$ transmitted symbols are combined and produce one received symbol at the receiver. Considering the MISO-LDPC system as an LDPC code over $2^{Mn_T}$-ary alphabet, we introduce a new approach for joint detection and decoding in these systems. We introduce a modified Tanner graph to illustrate MISO-LDPC systems and establish a BP decoding for modified Tanner graphs. With numerical simulations, we show that the proposed scheme outperforms the traditional decoder for short length codes.

In view of the complexity associated with the straightforward implementation of the BP decoding over modified graphs, we introduce a novel edge-based message-passing (EBMP) algorithm that considerably reduces the decoding complexity.

In some practical scenarios, channel state information is not available at the receiver. Consequently, we explore the unknown channel scenario as well. In particular, we examine the expectation-maximization (EM) algorithm as a technique to iteratively estimate the channel and data. It is known that EM-based receivers, exploiting the decoder information, can improve the channel estimation [9] - [11]. We compare the simulated performance of our proposed algorithm to that of the traditional decoder, and show that the proposed scheme outperforms the traditional one for short length codes in the unknown channel scenario.

The rest of this chapter is organized as follows. Section 3.2 briefly outlines the system model and describes the problem that we are trying to solve. In Section 3.3 a three-layer Tanner graph is introduced to exploit the BP algorithm for decoding in turbo-type receivers. To eliminate the positive feedback due to length-four cycles in the three-layer Tanner graphs, Section 3.4 proposes a new graph-based representation and a BP algorithm for decoding MISO-LDPC systems. Section 3.5 establishes a low-complexity implementation of the BP decoding over the modified graphs. Performance analysis and numerical comparisons are presented in Sections 3.6 and 3.7, and Section 3.8 concludes the chapter.

### 3.2 Problem Formulation

This section serves as an introduction to the proposed joint detection and decoding algorithm and is divided into two parts, namely system model and EM algorithm.
3.2.1 System Model

We consider the standard MISO-LDPC system model:

\[ y_l = \sqrt{\frac{E_s}{n_T}} c_l x_l + n_l, \tag{3.1} \]

where \( E_s \) is the transmitted symbol energy and \( n_T \) represents the number of transmit antennas. Here, \( c_l \) is the \( 1 \times n_T \) channel vector at the \( l \)th time instant, and \( n_l \sim \mathcal{N}(0, \sigma_n) \) represents a sample of AWGN at the \( l \)th time instant. We denote by \( x_l \) (\( y_l \), respectively) the \( n_T \times 1 \) (1 \( \times 1 \), respectively) vector of transmitted symbols (received data, respectively) at the \( l \)th time instant. The transmitted symbols vector \( x_l \) is generated as follows:

\[ x_l = [x_{l,1} \ x_{l,2} \ \cdots \ x_{l,n_T}]^T, \tag{3.2} \]

where \( x_{l,j}, \) for \( 1 \leq j \leq n_T, \) is a member of the alphabet \( \mathcal{X}, \) and corresponds to \( M \) consecutive LDPC bits, i.e.,

\[ x_{l,j} = F \left( x_{(l-1)Mn_T+(j-1)M+1}, \ldots, x_{(l-1)Mn_T+jM} \right), \tag{3.3} \]

where \( x_s, \) for \( 1 \leq s \leq n, \) is the \( s \)th bit of an \( (n,k) \) LDPC code, and \( F \) represents the modulation.

We consider a quasi-static Rayleigh flat fading channel model, where the channel remains constant during each fading block and changes independently from block to block. In other words, we assume that the block-length is equal to the coherence time [12] of the channel. Two different cases are considered for the channel, namely when the channel is known at the receiver and when the channel is unknown at the receiver.

Channel Known at the Receiver

Data detection when the channel is known at the receiver is straightforward. Since the channel is known, we can calculate the apriori probabilities and initialize the BP algorithm.

Channel Unknown at the Receiver

When the channel is not known at the receiver it should first be estimated before data detection can take place. This is the idea behind pilot symbol assisted modulation (PSAM) [13]-[15]. A set of pilot symbols \( p \) is transmitted at the beginning of
each block, the receiver estimates the channel based on the known pilot symbols, and then it decodes the received codeword using the estimated channel.

To unify the equations in the unknown channel scenario, similar to [16], we define:

\[
X_l = \begin{bmatrix} p x_{l,nD+1} & \cdots & x_{l,(l+1)nD} \end{bmatrix} \\
y_l = \begin{bmatrix} y^{(p)}_{l,nD+1} & \cdots & y_{l,(l+1)nD} \end{bmatrix} \\
n_l = \begin{bmatrix} n^{(p)}_{l,nD+1} & \cdots & n_{l,(l+1)nD} \end{bmatrix},
\]

(3.4)

where \( p \) is an \( n_T \times n_P \) matrix containing the pilot symbols and \( X_l \) is the \( n_T \times (n_P + n_D) \) matrix of transmitted pilot and data symbols at the \( l \)th block. \( y_l \) (\( n_l \), respectively) is the \( 1 \times (n_P + n_D) \) vector of received symbols (noise samples, respectively) at the \( l \)th block. Furthermore, \( y^{(p)}_{l} \) (\( n^{(p)}_{l} \), respectively) represents the \( 1 \times n_P \) vector containing the received signals due to the pilot symbols (Gaussian noise samples at the first \( n_P \) time instants of the \( l \)th block, respectively).

Clearly,

\[
y_l = \sqrt{E_s n_T} c_l X_l + n_l,
\]

(3.5)

where \( c_l \) is the \( 1 \times n_T \) channel vector during the \( l \)th block.

While being less complex, PSAM systems suffer from weak performance. Better performance can be attained when the channel estimation and data detection are performed jointly rather than sequentially. One scheme for joint estimation and detection is the EM algorithm. It has been shown that a turbo-EM estimator with the ability to use the soft output of the decoder outperforms the conventional MMSE-based channel estimators [10, 11].

### 3.2.2 EM Algorithm

The EM algorithm is well suited for problems which have both observed data and unobserved hidden data. This algorithm has two steps [17]:

\[
\begin{align*}
\text{E step:} & \quad Q(c_l|c_l^i) = E_{X_l|y_l,c_l^i}[\ln \Pr(y_l, X_l|c_l)] \\
\text{M step:} & \quad c_l^{i+1} = \arg \max_{c_l} Q(c_l|c_l^i)
\end{align*}
\]

(3.6)

\(^2\)Note that each block contains \( n_P + n_D \) MISO symbols.
Figure 3.1: The turbo-type receiver for MIMO-LDPC systems.

These steps should be repeated alternately until convergence in the decoded data. Note that here the transmitted data $X_l$ is the hidden data, the channel vector $c_l$ is the parameter to be estimated, $c_i^l$ is the $i^{th}$ estimate of $c_l$, and $(X_l, y_l)$ is the complete data.

Similar to [9], after some algebraic manipulations, the EM update equations (3.6) can be written as

$$c_i^{l+1} = \sqrt{\frac{H^T}{E_s}} y_l U_l^H R_l^{-1},$$

where

$$R_l = E_{X_l|c_l^I, y_l} [X_l X_l^H],$$

and

$$U_l = E_{X_l|c_l^I, y_l} [X_l].$$

Note that $R_l$ and $U_l$ are computed using the soft output of the LDPC decoder. Moreover, $I_{n_T}$ represents the $n_T \times n_T$ identity matrix.

The EM algorithm always converges, but not necessarily to the global maximum. To avoid the possibility of converging to a local maximum, the initial conditions should be chosen with care. To that end, we use the output of the PSAM estimator to estimate the channel.
3.3 Graph Based Representation of Turbo-Type Receivers

Inspired by [18], which deals with the joint decoding and equalization of LDPC codes over partial-response channels, we construct a three-layer Tanner graph where the second and third layers are, respectively, variable and check nodes of the LDPC code, and the first layer consists of some generalized nodes which we call functional nodes and represent by triangles in the three-layer Tanner graph, see Figure 3.2. Each functional node is connected to $Mn_T$ consecutive LDPC variable nodes from the second layer of the graph, and represents a MISO symbol detector in the turbo-type receiver, shown in Figure 3.1.

The BP algorithm over three-layer Tanner graphs runs similar to the BP algorithm over Tanner graphs of irregular repeat accumulate codes [19]. Nodes act under the assumption that each message communicated to them is a conditional probability, log-likelihood ratio to be exact, on the bit and each received message is conditionally independent of all other received messages.

Variable nodes and check nodes in the three-layer graph follow the same rules
as in the binary LDPC decoder. However to update a message from a given functional node to a given variable node, the functional node uses the messages, i.e., log-likelihood ratios, received from the $Mn_T - 1$ other variable nodes connected to it, together with the information received from the channel, and updates the log-likelihood ratio of the variable node.

For convenience, we henceforth refer to the messages passed from check node (variable node, respectively) decoders to variable node (check node, respectively) decoders in the turbo-type receiver as the check-to-variable (variable-to-check) messages.

We claim that decoding with the turbo-type receiver is formally identical to BP over a three-layer Tanner graph$^3$ and use induction to furnish the proof. Consider a turbo-type receiver and the three-layer Tanner graph associated with the turbo-type receiver. Clearly at the initialization the check-to-variable messages are zero and, as a result, they are the same for both decoders. We claim that if check-to-variable messages in a turbo-type receiver are equal to the associated check-to-variable messages in the three-layer Tanner graph, then after one iteration again the corresponding check-to-variable messages will be the same.

It is not hard to see that if the check-to-variable messages in the three-layer Tanner graph are equal to the corresponding check-to-variable messages in the turbo-type receiver, then the variable-to-functional messages in the three-layer graph will be equal to the corresponding messages passed from variable node decoders to the soft-in soft-out MISO detectors in the turbo-type receiver.

Consider a functional node and its corresponding soft-in soft-out MISO detector. Given that the incoming messages to the functional node are the same as the incoming messages to the soft-in soft-out MISO detector, the corresponding outgoing messages also will be the same. See Appendix 3.A for proof. Subsequently, one can deduce that the variable-to-check messages in the three-layer Tanner graph should be the same as the corresponding variable-to-check messages in the turbo-type receiver. As a result, the corresponding check-to-variable messages also would be the same.

$^3$Note that different scheduling schemes can be considered for BP over three-layer Tanner graphs. In this dissertation, we assume that during each iteration of the BP algorithm, functional nodes first update the functional-to-variable messages, then variable nodes update the variable-to-check messages. Subsequently, check nodes update the check-to-variable messages and, lastly, variable nodes update the variable-to-functional messages.
3.4 Modified Tanner Graphs

For brevity, in the rest of this chapter, we refer to LDPC codes over a $2^Mn_T$-ary alphabet as $2^Mn_T$-ary LDPC codes. Similar to binary LDPC codes, a modified parity-check matrix and a Tanner graph can be defined for $2^Mn_T$-ary LDPC codes. As in the binary LDPC graph, each column of the modified parity-check matrix represents a modified variable node. Since variable nodes in a $2^Mn_T$-ary LDPC code correspond to $Mn_T$ consecutive variable nodes in the original graph, each column of the modified parity-check matrix represents $Mn_T$ consecutive columns of the original parity-check matrix. To derive the modified parity-check matrix, we consider each $Mn_T$ consecutive components in each row of the original parity-check matrix as the corresponding component in the modified matrix.

Each Tanner graph has three parts: variable nodes, check nodes, and the edges connecting the variable nodes to the check nodes. In order to derive the modified Tanner graph, we define how each of the three above mentioned parts are affected. Variable nodes in the modified Tanner graph correspond to $Mn_T$ consecutive variable nodes in the original Tanner graph, hence the number of variable nodes is $\frac{1}{Mn_T}$ times the number of the variable nodes in the original graph.

Given that each check node represents one constraint in the LDPC code, the number of check nodes is not affected by modifying the graph. However, check nodes in the modified graph are connected to $2^Mn_T$-ary variable nodes instead of binary variable nodes.

If any of the $Mn_T$ binary constituent variable nodes of a $2^Mn_T$-ary modified variable node are connected to a check node in the original graph, there would be an edge connecting the $2^Mn_T$-ary variable node to the corresponding modified check node in the modified graph. In the modified Tanner graph, the edges are not all identical. To differentiate the edges in the modified Tanner graph, we label them with the corresponding component in the modified parity-check matrix. Example 3.1 sheds some light on the procedure of deriving the modified parity-check matrix and Tanner graph.

**Example 3.1** Deriving the modified parity-check matrix and Tanner graph from the binary parity-check matrix when $Mn_T = 2$: 

Figure 3.3: (a) The Tanner graph associated with a binary parity-check matrix, (b) The corresponding modified Tanner graph when $Mn_T = 2$.

Consider the following binary parity-check matrix:

\[
H = \begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0
\end{bmatrix}.
\] (3.10)

To derive the modified parity-check matrix, when $Mn_T = 2$, we combine every two consecutive columns in the original parity-check matrix and consider them as a column in the modified parity-check matrix. For instance, the first two columns of the first row of the binary parity-check matrix are 1 and 0, so there should be a value of $[1 \ 0]$ in the first row and first column of the modified parity-check matrix. Following the same procedure, the rest of the modified parity-check matrix can be obtained.

\[
H^{(4)} = \begin{bmatrix}
[1 \ 0] & [0 \ 1] & [1 \ 0] & [1 \ 0] \\
[1 \ 1] & [0 \ 0] & [0 \ 1] & [0 \ 0] \\
[0 \ 0] & [1 \ 1] & [0 \ 1] & [0 \ 1] \\
[0 \ 0] & [1 \ 1] & [0 \ 1] & [0 \ 0]
\end{bmatrix}.
\] (3.11)

Note that the superscript in $H^{(4)}$ represents the alphabet-size over which the parity-check matrix is defined.
Figure 3.3.(b) represents the modified Tanner graph corresponding to $H^{(4)}$. Each quaternary variable node in this graph represents two consecutive variable nodes in the original Tanner graph. For example, $V_1$ corresponds to $v_1$ and $v_2$ in the original binary graph, illustrated in Figure 3.3.(a).

Check nodes in the modified graph represent the same constraints as in the original graph. To determine the constraint imposed by a check node, we write the corresponding values for the variable nodes and edges as length-2 binary vectors, then we use the inner product and summation over $GF(2)$ to determine the constraint. For example, $C_1$ implies

$$\langle [1 \ 0], X_1 \rangle \oplus_2 \langle [0 \ 1], X_2 \rangle \oplus_2 \langle [1 \ 0], X_3 \rangle \oplus_2 \langle [1 \ 0], X_4 \rangle = 0,$$

(3.12) where $\langle \cdot, \cdot \rangle$ ($\oplus_2$, respectively) represents the inner-product (addition, respectively) operator over $GF(2)$. Equation (3.12) can be rewritten as

$$\left( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes_2 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \oplus_2 \left( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes_2 \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \right) = 0,$$

(3.13)

where $\otimes_2$ is the vector multiplication operator over $GF(2)$. Consequently, (3.13) can be rewritten as

$$x_1 + x_4 + x_5 + x_7 = 0 \pmod{2},$$

(3.14) which is the same as the constraint imposed by the first row of $H$, (3.10).

### 3.4.1 BP over Modified Tanner Graphs

BP over modified Tanner graphs is an iterative decoding algorithm which uses length-2$M_{NT}$ vectors of transmitted symbols a-posteriori probabilities as its messages, i.e.,

$$m = \begin{pmatrix} M_{NT} \\ M_{NT} \\ M_{NT} \\ M_{NT} \end{pmatrix} \begin{pmatrix} \Pr([0 \ 0 \ \ldots \ 0]|y), \Pr([0 \ 0 \ \ldots \ 1]|y), \ldots, \Pr([1 \ 1 \ \ldots \ 1]|y) \end{pmatrix}.$$  

(3.15)

$^4$Note that for simplicity of the notation, we suppress the dependence of the message on the iteration number and the edge associated with the message.
For convenience let us define \( m_l \triangleq \Pr \left( \langle l \rangle_{MnT} \mid y \right), \) for \( l = 0, 1, \ldots, 2^{MnT} - 1, \) where \( \langle l \rangle_{MnT} \) denotes the length-\( MnT \) row vector corresponding to the \( MnT \)-bit representation of \( l \).

Similar to LDPC codes over \( GF(q) \) [20] the channel messages are computed based on the received values. Then, the messages would be processed and passed between the modified variable and check nodes.

### 3.4.2 Initialization

Consider a modified variable node \( V_i \). Let \( m^c_i \) be the channel message corresponding to \( V_i \). This message is a length-\( 2^{MnT} \) probability vector whose \( l \)th element \( m^c_i(l) \) is given by

\[
m^c_i(l) = \Pr \left( x_i = \langle l \rangle_{MnT} \mid y_i \right) = \Pr \left( y_i \mid x_i = \langle l \rangle_{MnT} \right) \frac{\Pr(x_i = \langle l \rangle_{MnT})}{\Pr(y_i)}. \tag{3.16}
\]

Assuming equal probability for LDPC codewords, the fraction \( \Pr \left( x_i = \langle l \rangle_{MnT} \right) / \Pr \left( y_i \right) \) becomes independent from \( l \). Let us call this fraction \( \xi \). Hinging on the fact that \( m^c_i \) is a probability vector, one can deduce that

\[
\xi = \frac{1}{\sum_l \Pr \left( y_i \mid x_i = \langle l \rangle_{MnT} \right)}.
\tag{3.17}
\]

### 3.4.3 Variable-to-Check Messages

A degree-\( dv \) variable node with incoming messages \( m^{(j)}_i, j = 1, 2, \ldots, dv - 1, \) and channel message \( m^c \), updates the outgoing message \( m^{(o)}_i \) as follows:

\[
m^{(o)}_i = \kappa m^{(c)}_i \prod_{j=1}^{dv-1} m^{(j)}_i, \tag{3.18}
\]

where \( \kappa \) is chosen to ensure

\[
\sum_{i=0}^{2^{MnT}-1} m^{(o)}_i = 1. \tag{3.19}
\]

Moreover, a degree-\( dv \) variable node with incoming messages \( m^{(j)}, j = 1, 2, \ldots, dv, \) and channel message \( m^c \), can tentatively decode the \( l \)th bit, \( l = 1, 2, \ldots, MnT, \)
as follows:

\[ \hat{x}_l = \arg \max_d \sum_{s \in D_d^{(l)}} m_i^{(c)} \prod_{j=1}^{d_c} m_j^{(j)}, \]  

(3.20)

where \( d \in \{0, 1\} \) and \( D_d^{(l)} \) is defined as

\[ D_d^{(l)} = \{ s : \langle s \rangle_{M_{\text{tr}}T} = d \}. \]  

(3.21)

Here \( \langle s \rangle_{M_{\text{tr}}T} \) is the \( l \)th bit of the vector \( \langle s \rangle_{M_{\text{tr}}T} \).

3.4.4 Check-to-Variable Messages

Consider a-degree \( d_c \) check node with incoming messages \( m^{(j)}, j = 1, 2, \ldots, d_c - 1 \), and let \( e^{(j)} \) be the edge-constant associated with the message \( m^{(j)}, \) for \( j = 1, 2, \ldots, d_c - 1 \). Assume that \( m^{(o)} (e^{(o)}, \) respectively) denotes the outgoing message (edge-
constant associated with the outgoing message, respectively). The incoming messages
are processed at the check node according to

\[ m_i^{(o)} = \theta \sum_{(i_1, \ldots, i_{d_c-1}) \in \mathbb{I}_i(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)})} \prod_{j=1}^{d_c-1} m_{ij}^{(j)}, \]  

(3.22)

where \( \theta \) is a normalization factor to ensure

\[ \sum_{i=0}^{2^{M_{\text{tr}}-1}} m_i^{(o)} = 1. \]  

(3.23)

Furthermore, \( \mathbb{I}_i(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)}) \) corresponds to the local codewords [21] of the
check node and is defined as follows:

\[ \mathbb{I}_i(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)}) \triangleq \langle i_1, \ldots, i_{d_c-1} \rangle : \langle e^{(1)}, i_1 \rangle \oplus \ldots \oplus_2 \langle e^{(d_c-1)}, i_{d_c-1} \rangle = \langle e^{(o)}, i \rangle, \]  

(3.24)

where for convenience we use \( \langle e^{(j)}, i_j \rangle, \) for \( j = 1, 2, \ldots, d_c - 1, \) to represent the inner-
product of \( e^{(j)} \) and \( \langle i_j \rangle_{M_{\text{tr}}} \) over \( GF(2). \)

3.5 Edge-Based Message-Passing

Straightforward implementation of BP over modified graphs is computationally too intensive to be practical. Considering the log-likelihood ratio of the edges as
messages, we propose an alternative algorithm for implementation of BP that considerably reduces the decoding complexity.

Let us define \( Z_e \), for \( e \in GF(2)^{Mn_T} \), as the subset of \( Mn_T \)-ary alphabet, i.e., \( \{0, 1, \ldots, 2^{Mn_T} - 1\} \), whose inner-product with \( i \) is equal to zero,

\[
Z_e \triangleq \{ j : \langle e, j \rangle = 0 \}.
\] (3.25)

Moreover, let \( O_e \), for \( e \in GF(2)^{Mn_T} \), denote the subset of \( Mn_T \)-ary alphabet whose inner-product with \( i \) is equal to one,

\[
O_e \triangleq \{ j : \langle e, j \rangle = 1 \}.
\] (3.26)

It is not hard to see that \( |Z_0| = 2^{Mn_T} \) and \( |O_0| = 0 \), where \( 0_{Mn_T} \) is the length-\( Mn_T \) all-zero vector. The following lemma determines the cardinality of \( Z_e \) and \( O_e \) for any non-zero \( e \).

**Lemma 3.1** For any non-zero \( e \in GF(2)^{Mn_T} \), we have

\[
|Z_e| = |O_e| = 2^{Mn_T - 1}.
\] (3.27)

**Proof:** Let \( w_H(e) \) be the Hamming weight [22] of \( e \). Furthermore, assume \( e_1, \ldots, e_{w_H(e)} \) are the positions of ones in \( e \). Finally, let

\[
\langle j \rangle_{Mn_T} = [j_1 \ j_2 \ \ldots \ j_{Mn_T}],
\] (3.28)

where \( j \in \{0, 1, \ldots, 2^{Mn_T} - 1\} \). Now, one can see that

\[
\langle e, j \rangle = j_{e_1} \oplus 2 j_{e_2} \oplus 2 \ldots \oplus 2 j_{w_H(e)},
\] (3.29)

As a result, \( \langle e, j \rangle = 0 \) for values of \( j \) that have an even number of ones in positions \( e_1, e_2, \ldots, e_{w_H(e)} \). And, similarly, \( \langle e, j \rangle = 1 \) for values of \( j \) that have an odd number of ones in these positions. Considering the fact that for any non-zero \( e \), there exist \( 2^{Mn_T - 1} \) values of \( j \) with an even number of ones in these positions and \( 2^{Mn_T - 1} \) values of \( j \) with an odd number of ones in these positions, one can deduce that

\[
|Z_e| = |O_e| = 2^{Mn_T - 1},
\] (3.30)

as desired. \( \blacksquare \)
Before introducing the EBMP algorithm, let us present a definition.

**Definition 3.1** The log-likelihood ratio of an edge with edge-constant $e$ is defined as follows:

$$m_L(e) \triangleq \ln \frac{\sum_{i \in Z_e} \Pr((i)_{MnT}|y)}{\sum_{i \in O_e} \Pr((i)_{MnT}|y)}.$$  \hfill (3.31)

For brevity, we henceforth drop the subscript $L$ from $m_L(\cdot)$ and also suppress the dependency of the log-likelihood ratio on the edge-constant.

With some algebraic manipulations, one can show that:

**Theorem 3.2** The log-likelihood ratios of the edges are sufficient for calculation of the variable-to-check messages in the BP algorithm.

See Appendix 3.B for the proof.

Moreover, it is not hard to check that:

**Theorem 3.3** The log-likelihood ratios of the edges are sufficient for calculation of the check-to-variable messages in the BP algorithm.

See Appendix 3.C for the proof.

From Theorems 3.2 and 3.3, we deduce that the log-likelihood ratios of the edges are sufficient for implementation of the update equations. Consequently, we introduce the EBMP algorithm where each node just passes the log-likelihood ratio messages. As a result, the nodes just send one scalar message for each edge, as opposed to length-$2^{Mn_T}$ messages passed in the BP algorithm. The EBMP update equations can be formulated as follows.

### 3.5.1 Variable-to-Check Messages

Consider a degree-$d_v$ variable node with incoming BP messages $m^{(j)}$, $j = 1, 2, \ldots, d_v - 1$, and channel message $m^{(c)}$. Assume $m^{(j)}$ ($e^{(j)}$, respectively) denotes the log-likelihood ratio message (the edge-constant, respectively) associated with $m^{(j)}$, for $j = 1, 2, \ldots, d_v - 1$. Finally, let $m^{(o)}$ ($e^{(o)}$, respectively) represent the log-likelihood ratio
message (the edge-constant, respectively) corresponding to the outgoing message. According to (3.18) and (3.31), the outgoing log-likelihood message is updated as follows:

\[
m^{(o)} = \ln \frac{\sum_{i \in Z_e} \kappa m_i^{(c)} \prod_{j=1}^{d_v-1} m_i^{(j)}}{\sum_{i \in O_e} \kappa m_i^{(c)} \prod_{j=1}^{d_v-1} m_i^{(j)}},
\]

or, equivalently,

\[
m^{(o)} = \ln \frac{\sum_{i \in Z_e} m_i^{(c)} \prod_{j=1}^{d_v-1} m_i^{(j)}}{\sum_{i \in O_e} m_i^{(c)} \prod_{j=1}^{d_v-1} m_i^{(j)}}.
\]

Some thought shows that

\[
m^{(o)} = \ln \frac{\sum_{i \in Z_e} m_i^{(c)} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 0\}} m_i^{(j)} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 1\}} m_i^{(j)}}{\sum_{i \in O_e} m_i^{(c)} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 0\}} m_i^{(j)} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 1\}} m_i^{(j)}},
\]

Consider the incoming check-to-variable message \(m^{(j)}\), for \(j = 1, 2, \ldots, d_v - 1\). It is not hard to see that

\[
\forall l, s \in Z_{e^{(l)}} \quad m_s^{(j)} = m_l^{(j)},
\]

and that

\[
\forall l, s \in O_{e^{(l)}} \quad m_s^{(j)} = m_l^{(j)}.
\]

See Appendix 3.B for the proof. Consequently, one can show that

\[
\forall l \in Z_{e^{(l)}} \quad m_l^{(j)} = \frac{e^{m_l^{(j)}}}{1 + e^{m_l^{(j)}}},
\]

\[
\forall l \in O_{e^{(l)}} \quad m_l^{(j)} = \frac{1}{1 + e^{m_l^{(j)}}}.
\]

As a result, (3.34) can be rewritten as

\[
m^{(o)} = \ln \frac{\sum_{i \in Z_e} m_i^{(c)} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 0\}} \frac{e^{m_l^{(j)}}}{1 + e^{m_l^{(j)}}} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 1\}} \frac{1}{1 + e^{m_l^{(j)}}}}{\sum_{i \in O_e} m_i^{(c)} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 0\}} \frac{e^{m_l^{(j)}}}{1 + e^{m_l^{(j)}}} \prod_{j \in \{l : \langle e^{(l)}, i \rangle = 1\}} \frac{1}{1 + e^{m_l^{(j)}}}}.
\]
Multiplying both the numerator and denominator of Equation (3.37) by
\[ \prod_{j=1}^{d_v-1} 1 + e^{m(j)}, \]
we get
\[ m^{(o)} = \ln \sum_{i \in \mathbb{Z}} m_i^{(c)} \prod_{j \in \{i: (e^{(o)}, i) = 0\}} e^{m(j)} \]
\[ \sum_{i \in \mathbb{Z}} m_i^{(c)} \prod_{j \in \{i: (e^{(o)}, i) = 0\}} e^{m(j)}. \]
And the initialization can be done using
\[ m^{(o)} = \ln \sum_{i \in \mathbb{Z}} m_i^{(c)} \sum_{i \in \mathbb{Z}} m_i^{(c)}. \]

### 3.5.2 Check-to-Variable Messages

The update algorithm of the check-to-variable messages in EBMP runs similar to the update algorithm of the check-to-variable messages in BP algorithm over binary Tanner graphs.

Consider a-degree \( d_c \) check node with incoming EBMP messages \( m^{(j)} \), \( j = 1, 2, \ldots, d_c - 1 \), and outgoing message \( m^{(o)} \). The incoming messages are processed at the check node according to
\[ m^{(o)} = \ln \left( 1 + \prod_{j=1}^{d_c-1} \frac{m^{(j)}}{1 + \tanh \frac{m^{(j)}}{2}} \right). \]

### 3.6 Performance Analysis

It is not hard to see that when the modified Tanner graph is acyclic, the EBMP is equivalent to MAP and, as a result, performs the same as or better than any other decoding scheme such as the turbo-type receiver. To compare the performance of receivers over graphs with cycles, e.g. finite-length LDPC codes, we consider two different scenarios:
- Any pair of variable nodes that are connected to a soft-in soft-out MISO symbol detector are not connected to the same check node.
- There exists at least one pair of variable nodes that are connected to a given soft-in soft-out MISO detector and a given check node.

### 3.6.1 First Scenario

In Section 3.3, we showed that decoding using a turbo-type receiver is equivalent to BP decoding over the corresponding three-layer Tanner graph. In this section, we focus on the first scenario, i.e., when there is no pair of variable nodes connected to a given check node and a given soft-in soft-out MISO symbol detector. We show that BP over a three-layer Tanner graph is equivalent to BP over the corresponding modified Tanner graph. Consequently, we use induction to furnish the proof. We start by showing that once the associated variable-to-check messages are the same in both receivers then the corresponding check-to-variable messages would also be the same. Additionally we show that if the corresponding check-to-variable messages are the same in both graphs, modified variable nodes in the modified Tanner graph produce the same variable-to-check messages as the functional and variable nodes in the three layer Tanner graph. Then we discuss the first step of the induction and show that during the initialization, corresponding variable-to-check messages are the same in both graphs.

Each check node in a three-layer Tanner graph corresponds to a check node in the associated modified Tanner graph. Furthermore, in this scenario, the degrees of the corresponding check nodes are the same. Namely, there is a one-to-one correspondence between the set of edges in the three-layer Tanner graph and the set of edges in the modified Tanner graph.

Consider a degree-$d_c$ check node from the three-layer Tanner graph with incoming messages $m^{(i)}$, $j = 1, 2, \ldots, d_c - 1$, and let $m^{(o)}$ denote the outgoing check-to-variable message. Consequently [22]

$$m^{(o)} = \ln \left( \frac{1 + \prod_{j=1}^{d_c-1} \tanh \frac{m^{(i)}}{2}}{1 - \prod_{j=1}^{d_c-1} \tanh \frac{m^{(i)}}{2}} \right).$$

(3.41)
According to (3.40), a degree-$d_c$ modified check node with incoming messages $m^{(j)}$, $j = 1, 2, \ldots, d_c - 1$, also generates the same outgoing message as $m^{(o)}$. Thereby one can deduce that once the corresponding variable-to-check messages are the same, in both decoders, then the corresponding check-to-variable messages would also be the same.

Now, consider a modified variable node in the modified Tanner graph and the corresponding functional and variable nodes in the three-layer Tanner graph. Given that the incoming messages from modified check nodes to the modified variable node are the same as the associated check-to-variable messages in the three-layer Tanner graph, then, the outgoing messages from the modified variable node would also be the same as the associated variable-to-check messages in the three-layer graph. We relegate the proof to Appendix 3.D.

Let us now establish the first step of the induction. In the initialization step of EBMP (BP over modified Tanner graphs, respectively), all the messages passed from modified check nodes (check nodes, respectively) to the modified variable nodes (variable nodes, respectively) in the modified Tanner graph (the three-layer Tanner graph, respectively) are zero. As a result, the associated messages passed from the modified variable nodes to the modified check nodes (in the modified graph) and the variable nodes to the check nodes (in the three-layer graph) are the same, as desired.

### 3.6.2 Second Scenario

We have already seen that, in the first scenario, both the EBMP and turbo-type receiver perform the same. However, it is a hard task to make a rigorous comparison between the performance of algorithms in the second scenario.

In this scenario, there exists at least one pair of variable nodes that are connected to a given functional node and a given check node in the three-layer Tanner graph. Clearly, the pair of variable nodes, the functional node, and the check node form a length-four cycle in the three-layer Tanner graph. However, this cycle does not appear in the modified Tanner graph. In particular, the length-four cycle is transformed to an edge in the modified graph, see Figure 3.4.

Essentially, each cycle in the modified Tanner graph corresponds to a cycle in the turbo-type architecture. Therefore, conversion from a three-layer graph to the modified graph merely eliminates the induced length-four cycles, and does not generate any
Apart from the positive feedback, turbo-type receivers also suffer from an information loss due to the data-processing [23, 24] at the functional nodes. Let us introduce this idea by a simple example.

**Example 3.2** Information loss due to data-processing in three-layer Tanner graphs:

Consider a rate-\( \frac{1}{2} \), length-4, LDPC code \( C \) with parity-check matrix

\[
H = \begin{bmatrix}
1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}.
\]  

(3.42)

Assume that \( M_{n_T} = 2 \). Figure 3.4 shows the three-layer and modified Tanner graphs of \( C \). Clearly, the three-layer Tanner graph contains two length-four cycles, whereas the modified Tanner graph is an acyclic graph.

Let

\[
\Pr(x_1 = s|y_1) = \begin{cases}
0.1 & \text{if } s = [0 \ 0] \text{ or } [1 \ 1] \\
0.4 & \text{if } s = [0 \ 1] \text{ or } [1 \ 0].
\end{cases}
\]  

(3.43)

Consider the message passed from the first check node \( c_1 \) to the third variable node \( v_3 \) during the first iteration of the BP decoding over the three-layer Tanner graph, as shown in Figure 3.4. During the initialization, the functional node \( f_1 \) passes a message of 0 to both \( v_1 \) and \( v_2 \). Subsequently, \( v_1 \) and \( v_2 \) also send a message of 0 to the first check node \( c_1 \), and the first check node \( c_1 \) passes a message of 0 to the third variable node \( v_3 \).
Now, let us consider the associated message, i.e., the message passed from the first modified check node $C_1$ to the second modified variable node $V_2$, in the modified Tanner graph. During the first iteration, the first modified variable node $V_1$ passes a message of $\ln \left( \frac{1}{4} \right)$ to the first modified check node $C_1$, and, as a result, $C_1$ passes a message of $\ln \left( \frac{1}{4} \right)$ to the second modified variable node $V_2$. Comparing the associated check-to-variable messages in both graphs, one can deduce that BP decoding over three-layer Tanner graphs suffers from a data-processing loss.

Due to the positive feedback associated with the length-four cycles and the information loss associated with data-processing, we expect the EBMP to outperform the turbo-type scheme. We also expect the performance difference to depend on the ratio of the number of length-four cycles to the total number of cycles in the three-layer Tanner graph.

### 3.7 Numerical Results

We simulated our proposed algorithm for joint detection and decoding in a $2 \times 1$ MISO-LDPC system for both known and unknown channel cases. We analyzed a random (3,4)-regular LDPC code of length $n = 252$. In all cases, we plot the bit error rate in terms of the $\frac{E_b}{N_0}$. Each plot compares the performance of the proposed EBMP algorithm with the turbo-type receiver. The comparisons all correspond to quasi-static Rayleigh fading channels. Unless explicitly mentioned, the modulation is 8-ary QAM with Gray labeling, i.e., $M = 3$, the pilot symbol power is assumed to be 4 times the data symbols power, and the coherence time $T_c = 8T$, where $T$ represents the symbol time.

Figure 3.5 illustrates the performance of receivers for both known and unknown channels. One can easily see that the proposed scheme outperforms the traditional scheme, i.e., turbo-type receiver, in both scenarios. Moreover, the performance difference in the unknown channel case is larger than the difference in the known channel scenario. The reason for this observation is that in the unknown channel scenario, the soft-in soft-out MISO detectors are “weaker”, i.e., get less information from the received symbols, and consequently generate more positive feedback as opposed to the known channel scenario.

Figure 3.6 represents the effect of the coherence time of the channel on the
performance of both systems over unknown channels. One can see that increasing the coherence time increases the performance difference of the systems. This might be due to the fact that channel estimation errors can be a more significant contributor to erroneous detection in fast fading channels.

Figure 3.7 investigates the effects of the pilot symbol power on the performance of the systems in the unknown channel cases. By increasing the pilot symbol power, the performance difference becomes closer to the known channel case.

Finally, Figure 3.8 demonstrates the effects of the modulation size on the performance of systems over unknown slow fading channels. One can see that increasing the modulation size increases the performance gap between both systems. This is due to the fact that by increasing the modulation size, the number of length-four cycles in the three-layer Tanner graph increases.
3.8 Concluding Remarks

In this chapter, we introduced a new graph-based representation for MISO-LDPC systems which can be used for joint MISO-symbol detection and decoding using message passing algorithms. We presented a novel edge-based message passing algorithm for implementation of BP that considerably decreases the computational complexity of the BP algorithm. To characterize the performance of the turbo-type receiver, we introduced a three-layer Tanner graph. We noted that on acyclic modified graphs, our algorithm is equivalent to MAP decoding. Moreover, in graphs with cycles, we identified scenarios where both algorithms perform the same. Our simulation results show that for short length LDPC codes, the proposed algorithm outperforms the traditional one in the unknown channel scenario.

3.9 Acknowledgement

This chapter, in part, is a reprint of the material of the following paper: A. H. Djahanshahi, P. H. Siegel, and L. B. Milstein, “Decoding on graphs: LDPC-coded


Figure 3.7: BER vs. \( E_b/N_0 \) for different pilot symbols powers.


### Appendix 3.A The Equivalence of Functional Nodes and Soft-In Soft-Out MISO Detectors

Without loss of generality, consider the first functional node and the soft-in soft-out MISO detector associated with it. Assume that \( m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)} \) denote the incoming messages from the first \( Mn_T \) variable nodes (variable node decoders, respectively) in the three-layer Tanner graph (turbo-type receiver, respectively). Let \( m^{(o)} \) denote the message passed from the first soft-in soft-out MISO detector to the first variable node. It is not hard to see that

\[
m^{(o)} = \ln \left[ \frac{\Pr(x_1=0|y_1,m^{(1)},m^{(2)},\ldots,m^{(Mn_T)})}{\Pr(x_1=1|y_1,m^{(1)},m^{(2)},\ldots,m^{(Mn_T)})} \right] - m^{(1)}, \tag{3.44}
\]

\[\text{Note that the soft-in soft-out MISO detector is a special case of the soft-in soft-out MIMO detector [9], [25] - [28].}\]
or, equivalently,

\[
m^{(o)} = \ln \left[ \frac{\sum_{s \in D_0} \Pr(x_1 = s, m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)})}{\sum_{s \in D_1} \Pr(x_1 = s, m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)})} \right] - m^{(1)},
\]

(3.45)

where \( D_0 \) (\( D_1 \), respectively) is the set of all length-\( Mn_T \) binary vectors such that their first bit is zero (one, respectively). Using the Bayes' rule, (3.45) can be written as

\[
m^{(o)} = \ln \left[ \frac{\sum_{s \in D_0} \Pr(y_1 | x_1 = s, m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)}) \Pr(x_1 = s | m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)})}{\sum_{s \in D_1} \Pr(y_1 | x_1 = s, m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)}) \Pr(x_1 = s | m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)})} \right] - m^{(1)}.\]

(3.46)

To compute \( m^{(o)} \), let us make two assumptions:

**Assumption 3.1** Let \( x_1, y_1 \) be independent of the log-likelihood ratios \( m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)} \). In other words,

\[
\Pr(y_1 | x_1 = s, m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)}) = \Pr(y_1 | x_1 = s).
\]

(3.47)

**Assumption 3.2** Let \( m_i, x_i \) be independent of \( \{m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)}\} \setminus \{m_i\} \), i.e.,

\[
\Pr(x_i = s_i | m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)}) = \Pr(x_i = s_i),
\]

\[
e^{(1-s_i)m^{(i)}} \frac{1}{1+e^{m^{(i)}}}.
\]

(3.48)
Consequently,
\[
\Pr \left( x_1 = s \mid m^{(1)}, m^{(2)}, \ldots, m^{(Mn_T)} \right) = \prod_{i=1}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}. \tag{3.49}
\]

Moreover, one can see that
\[
\Pr \left( x_1 = s \mid m^{(2)}, \ldots, m^{(Mn_T)} \right) = \frac{1}{2} \prod_{i=2}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}. \tag{3.50}
\]

Using (3.47) and (3.49), (3.46) can be written as
\[
m^{(o)} = \ln \left[ \frac{\sum_{s \in D_0} \Pr \left( y_1 \mid x_1 = s \right) \prod_{i=2}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}}{\sum_{s \in D_1} \Pr \left( y_1 \mid x_1 = s \right) \prod_{i=2}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}} \right] - m^{(1)}. \tag{3.51}
\]

Some thought shows that
\[
m^{(o)} = \ln \left[ \frac{\frac{e^{m^{(1)}}}{1 + e^{m^{(1)}}} \sum_{s \in D_0} \Pr \left( y_1 \mid x_1 = s \right) \prod_{i=2}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}}{\frac{1}{1 + e^{m^{(1)}}} \sum_{s \in D_1} \Pr \left( y_1 \mid x_1 = s \right) \prod_{i=2}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}} \right] - m^{(1)}, \tag{3.52}
\]
or, equivalently,
\[
m^{(o)} = \ln \left[ \frac{\frac{e^{m^{(1)}}}{1 + e^{m^{(1)}}} \sum_{s \in D_0} \Pr \left( y_1 \mid x_1 = s \right) \prod_{i=2}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}}{\frac{1}{1 + e^{m^{(1)}}} \sum_{s \in D_1} \Pr \left( y_1 \mid x_1 = s \right) \prod_{i=2}^{Mn_T} \frac{e^{(1-s_i)m^{(i)}}}{1 + e^{m^{(i)}}}} \right]. \tag{3.53}
\]

Using (3.50), we have
\[
m^{(o)} = \ln \left[ \frac{\sum_{s \in D_0} \Pr \left( y_1 \mid x_1 = s \right) \Pr \left( x_1 = s \mid m^{(2)}, \ldots, m^{(Mn_T)} \right)}{\sum_{s \in D_1} \Pr \left( y_1 \mid x_1 = s \right) \Pr \left( x_1 = s \mid m^{(2)}, \ldots, m^{(Mn_T)} \right)} \right], \tag{3.54}
\]
and, by assumption 3.1,
\[
m^{(o)} = \ln \left[ \frac{\sum_{s \in D_0} \Pr \left( y_1 = s, m^{(2)}, \ldots, m^{(Mn_T)} \right) \Pr \left( x_1 = s \mid m^{(2)}, \ldots, m^{(Mn_T)} \right)}{\sum_{s \in D_1} \Pr \left( y_1 = s, m^{(2)}, \ldots, m^{(Mn_T)} \right) \Pr \left( x_1 = s \mid m^{(2)}, \ldots, m^{(Mn_T)} \right)} \right], \tag{3.55}
\]
or, equivalently,
\[
m^{(o)} = \ln \left[ \frac{\sum_{s \in D_0} \Pr \left( x_1 = s, m^{(2)}, \ldots, m^{(Mn_T)} \right)}{\sum_{s \in D_1} \Pr \left( x_1 = s, m^{(2)}, \ldots, m^{(Mn_T)} \right)} \right]. \tag{3.56}
\]
Consequently,
\[
m^{(o)} = \ln \left[ \frac{\Pr \left( x_1 = 0 \mid y_1, m^{(2)}, \ldots, m^{(Mn_T)} \right)}{\Pr \left( x_1 = 1 \mid y_1, m^{(2)}, \ldots, m^{(Mn_T)} \right)} \right], \tag{3.57}
\]
which is the same as the message passed from the first functional node to the first variable node, as desired.
Appendix 3.B  Proof of Theorem 3.2

Proof: Consider a degree-$d_c$ check node with incoming messages $m^{(j)}$, $j = 1, 2, \ldots, d_c - 1$, and assume $e^{(j)}$ is the edge-constant associated with the message $m^{(j)}$, for $j = 1, 2, \ldots, d_c - 1$. Let $m^{(o)}$ ($e^{(o)}$, respectively) denote the outgoing message (edge-constant associated with the outgoing message, respectively). According to (3.22), the outgoing message is given by

$$m^{(o)}_i = \theta \sum_{(i_1, \ldots, i_{d_c-1}) \in \mathbb{I}_i(e^{(o)}), e^{(1)}, \ldots, e^{(d_c-1)}} \prod_{j=1}^{d_c-1} m^{(j)}_{ij},$$  \hspace{1cm}  \text{(3.58)}$$

where $\mathbb{I}_i(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)})$ represents the local codewords of the check node, i.e.,

$$\mathbb{I}_i(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)}) = \{ (i_1, \ldots, i_{d_c-1}) : (e^{(1)}, i_1) \oplus \ldots \oplus (e^{(d_c-1)}, i_{d_c-1}) = (e^{(o)}, i) \}.$$  \hspace{1cm}  \text{(3.59)}$$

Clearly,

$$\forall i, l \in \mathbb{Z}_{e^{(o)}} \mathbb{I}_i(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)}) = \mathbb{I}_l(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)}),$$  \hspace{1cm}  \text{(3.60)}$$

and,

$$\forall i, l \in \mathbb{O}_{e^{(o)}} \mathbb{I}_i(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)}) = \mathbb{I}_l(e^{(o)}, e^{(1)}, \ldots, e^{(d_c-1)}).$$  \hspace{1cm}  \text{(3.61)}$$

We know from Lemma 3.1 that $|\mathbb{Z}_{e^{(o)}}| = |\mathbb{O}_{e^{(o)}}| = 2^{M_{nt}} - 1$. Thus, from $2^{M_{nt}}$ elements of the message $m^{(o)}$, half of them are equal to each other and the remaining half are also equal to each other. Since the message vector $m^{(o)}$ is a probability vector, one can deduce that $m^{(o)}$ can be derived from the ratio of these values. Therefore, the logarithm of this ratio, which is the log-likelihood ratio of $e^{(o)}$, is also sufficient for deriving the check-to-variable messages.

\[ \blacksquare \]

Appendix 3.C  Proof of Theorem 3.3

Proof: Consider a degree-$d_c$ check node with incoming messages $m^{(j)}$, $j = 1, 2, \ldots, d_c - 1$. Assume $e^{(j)}$ represents the edge-constant corresponding to message $m^{(j)}$, for $j = 1, 2, \ldots, d_c - 1$. Moreover, let $m^{(o)}$ and $e^{(o)}$, respectively, denote the outgoing message and the edge-constant associated with the outgoing message. According to (3.22), the outgoing message $m^{(o)}_i$, $i = 0, 1, \ldots, 2^{M_{nt}} - 1$, is generated by

$$m^{(o)}_i = \theta \sum_{(i_1, \ldots, i_{d_c-1}) \in \mathbb{I}_i(e^{(o)}), e^{(1)}, \ldots, e^{(d_c-1)}} \prod_{j=1}^{d_c-1} m^{(j)}_{ij},$$  \hspace{1cm}  \text{(3.62)}$$
Since,
\[ \forall i \in \mathbb{Z}_{e(1)} \; \langle i, e^{(1)} \rangle = 0, \quad (3.63) \]
and
\[ \forall i \in \mathbb{O}_{e(1)} \; \langle i, e^{(1)} \rangle = 1, \quad (3.64) \]
one can deduce that
\[ m_i^{(a)} = \emptyset \left[ \left( \sum_{i_1 \in \mathbb{Z}_{e(1)}} m_i^{(1)} \right) \left( \sum_{(i_2, \ldots, i_{d-1}) \in \mathbb{I} \{ \langle e^{(2)}, i_2 \rangle \oplus \cdots \oplus \langle e^{(d-1)}, i_{d-1} \rangle = \langle e^{(o)}, i \rangle \}} d-1 \prod_{j=2}^{d-1} m_j^{(j)} \right) \right] + \left( \sum_{i_1 \in \mathbb{O}_{e(1)}} m_i^{(1)} \right) \left( \sum_{(i_2, \ldots, i_{d-1}) \in \mathbb{J} \{ \langle e^{(o)}, e^{(2)}, \ldots, e^{(d-1)} \rangle = 1 \oplus \langle e^{(o)}, i \rangle \}} d-1 \prod_{j=2}^{d-1} m_j^{(j)} \right), \quad (3.65) \]
where,
\[ \mathbb{I}_i \{ e^{(o)}, e^{(2)}, \ldots, e^{(d-1)} \} \triangleq \{ (i_2, \ldots, i_{d-1}) : \langle e^{(2)}, i_2 \rangle \oplus \cdots \oplus \langle e^{(d-1)}, i_{d-1} \rangle = \langle e^{(o)}, i \rangle \}, \quad (3.66) \]
and
\[ \mathbb{J}_i \{ e^{(o)}, e^{(2)}, \ldots, e^{(d-1)} \} \triangleq \{ (i_2, \ldots, i_{d-1}) : \langle e^{(2)}, i_2 \rangle \oplus \cdots \oplus \langle e^{(d-1)}, i_{d-1} \rangle = 1 \oplus \langle e^{(o)}, i \rangle \}. \quad (3.67) \]
Consequently from \( 2^{MN_T} \) elements of the message \( m^{(1)} \),
\[ \sum_{i_1 \in \mathbb{Z}_{e(1)}} m_i^{(1)}, \]
and
\[ \sum_{i_1 \in \mathbb{O}_{e(1)}} m_i^{(1)}, \]
are sufficient for updating the check-to-variable message. And since each message is a probability vector, these two quantities sum to one. Thus, each of them can be obtained from the logarithm of their ratio. In other words, the log-likelihood ratio of \( e^{(1)} \) is sufficient for updating the check-to-variable message.

So far we have been concerned merely with \( m^{(1)} \). However, this result can easily be extended to other variable-to-check messages as well.
Appendix 3.D  Equivalence of Functional and Variable Nodes to the Modified Variable Nodes

Here, we only consider \((d_v, d_c)\)-regular LDPC codes, however, the results presented can easily be generalized to irregular LDPC codes as well. Without loss of generality, consider the first functional node and \(Mn_T\) variable nodes connected to it in the three-layer Tanner graph. Assume that \(m_{vc}^{(o)}\) represents the outgoing variable-to-check message from the first variable node to the first check node connected to it. Let \(m_{\text{in}}^{(r)}, \) for \(r = 1, 2, \ldots, Mn_Td_v - 1\), denote the \(Mn_Td_v - 1\) incoming check-to-variable messages. Finally, assume that \(m_{fv}^{(o)}\) is the message passed from the functional node to the first variable node and let \(m_{vf}^{(r)}, \) for \(r = 1, 2, \ldots, Mn_T - 1\), represent the \(Mn_T - 1\) variable-to-functional messages from the remaining variable nodes, see Figure 3.9(a).

Figure 3.9: (a) Computation of a variable-to-check message in a three-layer Tanner graph, (b) Computation of a message from a modified variable node to a modified check node in the modified Tanner graph.
It is not hard to see that [22],

\[ m_{vc}^{(o)} = m_{fv}^{(o)} + \sum_{r=1}^{d_v-1} m^{(r)}, \] (3.68)

where, according to (3.53), \( m_{fv}^{(o)} \) is given by

\[
m_{fv}^{(o)} = \ln \left[ \frac{\sum_{s \in \mathcal{D}_0} \Pr(y_1 | x_1 = s) \frac{1}{2} \prod_{r=1}^{Mn_T-1} e^{(1-s_r)m_{vf}^{(r)}}}{\sum_{s \in \mathcal{D}_1} \Pr(y_1 | x_1 = s) \frac{1}{2} \prod_{r=1}^{Mn_T-1} e^{(1-s_r)m_{vf}^{(r)}}} \right],
\] (3.69)

or, equivalently,

\[
m_{fv}^{(o)} = \ln \left[ \frac{\sum_{s \in \mathcal{D}_0} \Pr(y_1 | x_1 = s) \prod_{r=1}^{Mn_T-1} e^{(1-s_r)m_{vf}^{(r)}}}{\sum_{s \in \mathcal{D}_1} \Pr(y_1 | x_1 = s) \prod_{r=1}^{Mn_T-1} e^{(1-s_r)m_{vf}^{(r)}}} \right].
\] (3.70)

Using (3.68) and (3.70), we get

\[
m_{vc}^{(o)} = \ln \left[ \frac{\sum_{s \in \mathcal{D}_0} \Pr(y_1 | x_1 = s) \prod_{r=1}^{Mn_T-1} e^{(1-s_r)m_{vf}^{(r)}}}{\sum_{s \in \mathcal{D}_1} \Pr(y_1 | x_1 = s) \prod_{r=1}^{Mn_T-1} e^{(1-s_r)m_{vf}^{(r)}}} \right] + \sum_{r=1}^{d_v-1} m^{(r)}. \] (3.71)

Now, let us consider the modified Tanner graph. Assume, \( e^{(r)}, r = 1, 2, \ldots, Mn_Td_v - 1 \), represents the edge-constant associated with message \( m^{(r)} \) in the modified Tanner graph and \( e^{(o)} (m_{vc}^{(o)}, \text{respectively}) \) is the edge constant associated with the outgoing message (the outgoing message, respectively), see Figure 3.9.(b). Clearly,

\[
e^{(o)} = \left[ \frac{Mn_T-1}{1 \ 0 \ 0 \ \ldots \ 0} \right].
\] (3.72)

Let us define \( \langle s \rangle_{\infty} \) as the integer value corresponding to the binary vector \( s \). It is not hard to see that

\[
\forall r \in \mathbb{Z}_{e^{(o)}} \quad \langle r \rangle_{Mn_T} \in \mathbb{D}_0,
\] (3.73)

or,

\[
\forall s \in \mathbb{D}_0 \quad \langle s \rangle_{\infty} \in \mathbb{Z}_{e^{(o)}}.
\] (3.74)
Note that any binary length-$Mn_T$ vector $s \in \mathbb{D}_0$ has a 0 in its first position and, as a result, satisfies the condition $\langle s, e^{(0)} \rangle = 0$. Therefore $\langle s \rangle_\infty$ belongs to $\mathbb{Z}_{e^{(0)}}$.

Similarly, one can easily see that

$$\forall r \in \emptyset_{e^{(0)}} \quad \langle r \rangle_{Mn_T} \in \mathbb{D}_1,$$

or,

$$\forall s \in \mathbb{D}_1 \quad \langle s \rangle_{\infty} \in \emptyset_{e^{(0)}}.$$  (3.76)

From (3.16) and (3.17), one can deduce that

$$\Pr \left( y_1 | x_1 = \langle r \rangle_{Mn_T} \right) = \frac{m_r^{(c)}}{\xi}.$$  (3.77)

Combining (3.71), and (3.73) - (3.77), we get

$$m^{(o)}_{vc} = \ln \left[ \sum_{i \in \mathbb{Z}_{e^{(0)}}} \frac{m_i^{(c)}}{\xi} \prod_{r=1}^{Mn_T-1} e^{(1-(i)_{Mn_T,r}) m_r^{(r)}} \right] + \sum_{r=1}^{d_s-1} m^{(r)},$$  (3.78)

or, equivalently,

$$m^{(o)}_{vc} = \ln \left[ \sum_{i \in \emptyset_{e^{(0)}}} \frac{m_i^{(c)}}{\xi} \prod_{r=1}^{Mn_T-1} e^{(1-(i)_{Mn_T,r}) m_r^{(r)}} \right] + \sum_{r=1}^{d_s-1} m^{(r)}.$$  (3.79)

Since [22]

$$m^{(r)}_{ef} = \sum_{l=rd_v}^{(r+1)d_v-1} m^{(l)},$$  (3.80)

one can deduce that

$$(1 - \langle i \rangle_{Mn_T,r}) m^{(r)}_{ef} = (1 - \langle i \rangle_{Mn_T,r}) \sum_{l=rd_v}^{(r+1)d_v-1} m^{(l)}.$$  (3.81)

Hinging on the fact that

$$(1 - \langle i \rangle_{Mn_T,r}) m^{(l)} = \begin{cases} m^{(l)} & \text{if } \langle i \rangle_{Mn_T,r} = 0 \\ 0 & \text{if } \langle i \rangle_{Mn_T,r} = 1 \end{cases}.$$  (3.82)
and,
\[ e^{(l)} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 \end{bmatrix}, \quad (3.83) \]
for \( l = rd_v, \ldots, (r+1)d_v-1 \), one can deduce that
\[
(1 - \langle i \rangle_{MnT,r})m^{(l)} = \begin{cases} m^{(l)} & \text{if } \langle e^{(l)}, i \rangle = 0 \\ 0 & \text{if } \langle e^{(l)}, i \rangle = 1 \end{cases}, \quad (3.84)
\]
or, equivalently,
\[
(1 - \langle i \rangle_{MnT,r})m^{(r)}_{\text{ef}} = \sum_{j \in \{ l: \langle e^{(l)}, i \rangle = 0 \}} m^{(j)}. \quad (3.85)
\]
As a result, (3.79) can be reformulated as
\[
m^{(o)}_{\text{vc}} = \ln \left[ \sum_{i \in Z_{e^{(o)}}} \mathbf{m}_i^{(c)} \prod_{j \in \{ l: \langle e^{(l)}, i \rangle = 0 \}} e^{m^{(j)}} \right] + \sum_{r=1}^{d_v-1} m^{(r)}. \quad (3.86)
\]
Clearly, for \( r = 1, 2, \ldots, d_v-1 \),
\[
e^{(r)} = \begin{bmatrix} \ldots & 0 & 0 & 0 \end{bmatrix}. \quad (3.87)
\]
Consequently, one can deduce that for \( r = 1, 2, \ldots, d_v-1 \),
\[
\langle e^{(l)}, i \rangle = \begin{cases} 0 & i \in Z_{e^{(o)}} \\ 1 & i \in O_{e^{(o)}} \end{cases}. \quad (3.88)
\]
Hence, (3.86) can be reformulated as
\[
m^{(o)}_{\text{vc}} = \ln \left[ \sum_{i \in Z_{e^{(o)}}} \mathbf{m}_i^{(c)} \prod_{j \in \{ l: \langle e^{(l)}, i \rangle = 0 \}} e^{m^{(j)}} \right] + \sum_{r=1}^{d_v-1} m^{(r)}. \quad (3.89)
\]
Comparing (3.89) with (3.38), one can deduce that
\[
m^{(o)}_{\text{vc}} = m^{(o)}_{\text{mve}}, \quad (3.90)
\]
as desired.
Bibliography


Chapter 4

Generalized Binary Images of a Non-Binary LDPC Code

4.1 Introduction

The rediscovery of LDPC codes by MacKay [1, 2] showed that they outperform the best known Turbo codes. With their near capacity performance [1] these codes are among the most promising forward error correction schemes.

The analysis of BP decoding for LDPC codes over AWGN channels [3] paved the way for the design of optimized codes that achieved capacity [4]. The trend was soon followed by the analysis and design of capacity-approaching codes on various other channels [5] - [7]. Non-binary LDPC Codes were shown to further improve the performance of binary LDPC Codes [8].

The performance of message-passing decoders is mainly governed by two competing factors: the minimum distance of the LDPC code and the number of short cycles in the Tanner graph of the code. Having higher weight parity-check matrices improves the minimum distance, but also increases the number of short cycles. In [9] the authors demonstrate that these competing factors can be well balanced by increasing the field size. In fact, [9] claims that LDPC codes defined over large enough fields are generally “good codes.”

The naive implementation of the BP decoder for $GF(q)$ LDPC codes is computationally expensive. With a decoding complexity of $O(q^2)$, BP is far too complex to be
practical for large fields. As a result, there has been quite a bit of work on reducing the complexity of the BP decoder.

In [10, 11], the authors suggested the use of Fourier transforms (Hadamard transforms, to be precise) to reduce the complexity to $O(q \log q)$ without compromising the performance. An FFT-based decoding for a more general class of LDPC codes, namely LDPC codes over Abelian groups was established in [12]. An approximation based on the log-domain implementation of the BP decoder was proposed in [13]. Inspired by the min-sum decoder for binary LDPC codes, [14] introduced a decoding algorithm, called the extended min-sum (EMS), and showed that with an offset and a correction factor EMS can perform quite close to the BP decoder. An alternative implementation of the message-passing decoding using the min-max decoding was established in [15].

Apart from the high computational complexity, non-binary LDPC codes also suffer from a high decoding-memory requirement. This problem is addressed in [16] by allowing the check and variable nodes to operate on different fields. To achieve lower memory requirements than [16], while reducing the computational complexity, we propose decoding a binary image of the non-binary LDPC code, instead.

The decoding algorithms for LDPC codes are suboptimal, hence their performance not only depends on the codes, but also on many other factors such as the code representation. Many different Tanner graphs can be used to describe the same code, and will possibly lead to different performance when used with an iterative decoder [17]. As a result, we set off to find the best binary images of non-binary LDPC codes.

The rest of this chapter is organized as follows. Section 4.2 briefly establishes the required background and notation. Section 4.3 outlines the derivation of different binary images for a given non-binary code. In an attempt to better understand the complexity-performance tradeoffs, Section 4.4 introduces the generalized images of a non-binary LDPC code over intermediate fields and rings. Complexity evaluation and numerical simulations are presented in Sections 4.5, and 4.6, respectively. Finally, Section 4.7 concludes the chapter.
4.2 Channel Model

Even though the codes we consider are non-binary, in this chapter we assume that the transmission takes place over a binary, memoryless channel. In particular, we consider transmission over the binary-input AWGN (BI-AWGN) channel.

A non-binary LDPC code of block-length $n$, dimension $k$, and rate $r = k/n$ is defined as the set of codewords $x = (x_1, x_2, \ldots, x_n)$ over $GF(2^b)$ that satisfy the parity-check equation $Hx^T = 0$. Here, $H$ is an $(n-k) \times n$ sparse parity-check matrix with elements from $GF(2^b)$.

Each symbol $x_i$ in the codeword is transmitted as $b$ bits $x_i^{(2)} = (x_{i1}, x_{i2}, \ldots, x_{ib})$, where the superscript in $x_i^{(2)}$ represents the size of the field that elements of $x_i^{(2)}$, i.e., $x_{i1}, x_{i2}, \ldots, x_{ib}$, belong to. As the transmission takes place over the memoryless BI-AWGN channel, these bits are received as $y_i = (y_{i1}, y_{i2}, \ldots, y_{ib})$, where

$$y_{ij} = (1 - 2x_{ij}) + n_{ij}, \quad \text{for } j = 1, 2, \ldots, b,$$

and $n_{ij} \sim \mathcal{N}(0, \sigma^2)$. For convenience, we will henceforth drop the subscript $i$ when we refer to messages corresponding to a variable node.

4.2.1 BP over $GF(q)$

The BP decoder for a $GF(q)$ LDPC code, where $q = 2^b$, is an iterative message-passing scheme. Messages are vectors of a-posteriori probabilities of the transmitted symbols, i.e.,

$$m = \left(\Pr(0|y), \Pr(1|y), \Pr(\alpha|y), \ldots, \Pr(\alpha^{q-2}|y)\right),$$

where $\alpha$ is a primitive element of $GF(2^b)$. Define $m_i \triangleq \Pr(\alpha^i|y)$, for $i = -\infty, 0, 1, \ldots, q-2$, with $\alpha^{-\infty}$ denoting 0. The messages are processed at variable and the check nodes of the Tanner graph, and are passed between the nodes.

The channel messages $m^{(c)}$ are initialized based on the received values and knowledge of the channel model (4.1). We assume that the channel signal to noise ratio has been estimated correctly. At the first iteration these channel messages are sent from each variable node to neighboring check nodes. At subsequent iterations, a degree-$d_v$ variable node with incoming messages $m^{(j)}$, $j = 1, 2, \ldots, d_v - 1$, and the channel
message $\mathbf{m}^{(c)}$, updates the outgoing message $\mathbf{m}^{(o)}$ as follows:

$$\mathbf{m}_i^{(o)} = \mathbf{m}_i^{(c)} \prod_{j=1}^{d_v-1} \mathbf{m}_j^{(i)}.$$  \hspace{1cm} (4.3)

The complexity of this update process scales as $O(q)$ operations per iteration per variable node. Note that normalization of the output message vectors is necessary to keep output messages as probability mass vectors.

The edge constant $\alpha^l \in GF(q) \setminus \{0\}$, associated with the edge corresponding to the message $\mathbf{m}$ from the variable node $j$ to the check node $i$, results in a permutation of the message vector $\mathbf{m}$ as follows:

$$\bar{\mathbf{m}}_t = \mathbf{m}_{[t-l]} \hspace{0.5cm} \forall \ t = -\infty, 0, 1, \ldots, q-2,$$  \hspace{1cm} (4.4)

where $[t-l] = (t-l) \mod (q-1)$.

These permuted message vectors are processed at check nodes according to

$$\bar{\mathbf{m}}_i^{(o)} = \sum_{(i_1, i_2, \ldots, i_{d_v-1}) \in \mathbb{Z}_q^{d_v-1}} \prod_{j=1}^{d_v-1} \bar{\mathbf{m}}_{i_j}^{(j)},$$  \hspace{1cm} (4.5)

where $\mathbb{Z}_q^{d_v-1} \triangleq \{(i_1, i_2, \ldots, i_{d_v-1}) : \alpha^{i_1} + \alpha^{i_2} + \ldots + \alpha^{i_{d_v-1}} = \alpha^l\}$. This is an $O(q \log q)$-complex update process per iteration per check node when the Fourier transform, i.e., Walsh-Hadamard transform, is used. However, if the update is performed using look-up tables, the complexity is $O(q^2)$ per iteration per check node.

These messages are then unpermuted to get the incoming messages into variable nodes. The message corresponding to an edge with constant $\alpha^l \in GF(q) \setminus \{0\}$ is mapped as follows:

$$\mathbf{m}_t = \bar{\mathbf{m}}_{[t+l]} \hspace{0.5cm} \forall \ t = -\infty, 0, 1, \ldots, q-2.$$  \hspace{1cm} (4.6)

After a predefined number of iterations, the final a-posteriori probabilities are obtained at each variable node as follows:

$$\mathbf{m}_i^{(f)} = \mathbf{m}_i^{(c)} \prod_{j=1}^{d_v} \mathbf{m}_j^{(i)}.$$  \hspace{1cm} (4.7)

The variable node is then decoded to the $GF(q)$-symbol with the highest probability in $\mathbf{m}^{(f)}$.

Note that apart from the operational complexity of the non-binary decoder, there is also a $q$-fold increase in the memory requirements over the binary counterpart.
This complexity of the decoding process has been found to be prohibitive for large fields.

4.2.2 EMS Algorithm

The EMS is an iterative message-passing algorithm wherein the messages are probability distributions as in BP. This algorithm draws inspiration from the min-sum algorithm for decoding binary codes, and differs from BP in the operation performed at the check nodes. To be more specific, (4.5) is approximated as

$$\bar{m}_i^{(o)} = \max_{(i_1, i_2, \ldots, i_{d_c-1}) \in I_{d_c-1}} \left\{ \prod_{j=1}^{d_c-1} \bar{m}_{ij}^{(j)} \right\}. \tag{4.8}$$

EMS has a check node update complexity of $O(q^2)$.

Although EMS itself stands for extended min-sum algorithm, the operations involved in (4.8) are maximization and product rather than minimization and summation. This is due to the fact that the messages we are dealing with are probabilities rather than the log-likelihood ratios.

Since the check node update in EMS algorithm is always smaller than that of BP, an offset or a scaling factor could be introduced and optimized for the best performance [14].

4.2.3 Min-Max Algorithm

The min-max algorithm is another iterative message-passing algorithm aimed at the reduction of the decoding complexity. Similar to the EMS algorithm, min-max differs from BP only in the operations performed at the check nodes.

The min-max algorithm overcomes the underestimation in the check node update of the EMS algorithm (4.8) by overestimating the terms involved in the maximization [15]

$$\bar{m}_i^{(o)} = \max_{(i_1, i_2, \ldots, i_{d_c-1}) \in I_{d_c-1}} \left\{ \min_{j=1,2,\ldots,d_c-1} \bar{m}_{ij}^{(j)} \right\}. \tag{4.9}$$

As in the case of the EMS algorithm, the complexity of the check node update scales as $O(q^2)$. 
As before, while the algorithm is called the min-max algorithm, the operations involved in the check node update (4.9) are max-min, a result of the use of probabilities as messages, rather than log-likelihood ratios.

### 4.3 Binary Images of A Non-Binary LDPC code

Consider a non-binary LDPC code defined over $GF(q)$, for $q = 2^b$. In order to obtain the binary image of a non-binary codeword, we should develop ways to map $GF(q)$ symbols into binary vectors.

Derivation of the parity-check matrix of a binary image code requires mapping of non-binary symbols into binary matrices. In this chapter, we will only consider techniques to map $GF(q)$ symbols into square matrices.

Before addressing the question of how we can obtain binary images of a non-binary LDPC code, let us consider a slightly different question, namely how can we understand all possible isomorphisms $f : GF(q) \rightarrow M_s(GF(2))$, where $M_s(GF(2))$ represents the ring of $s \times s$ binary matrices?

Considering the fact that $M_s(GF(2))$ is an $s^2$-dimensional central simple $GF(2)$ algebra, the maximal field in $M_s(GF(2))$ would have a dimension of $s$ [18]. Consequently, one can deduce that it is impossible to map $GF(q)$ symbols into $M_s(GF(2))$ for $s < b$.

It is known that the companion matrix\(^1\) of a primitive polynomial of $GF(q)$ generates a subset of $M_b(GF(2))$ isomorphic to $GF(q)$. As a result, for the case where $s = b$, we can find a mapping from $GF(q)$ into a subset of matrices in $M_s(GF(2))$.

#### 4.3.1 The Basic Binary Image of A Non-Binary Code

Consider an extension field $GF(q)$, where $q = 2^b$. Let $\alpha$ be a primitive element of the field. Furthermore, let

$$P(x) = x^b + p_{b-1}x^{b-1} + p_{b-2}x^{b-2} + \cdots + p_1x + 1,$$

be the primitive polynomial associated with $\alpha$. To be more specific, $P(x)$ is the minimal polynomial of $\alpha$ over $GF(2)[x]$, where $GF(2)[x]$ is the family of polynomials with coefficients from $GF(2)$.

\(^1\)The companion matrix is defined by the theory of the rational canonical form [19].
Each $GF(q)$ symbol can be associated with a vector of $b$ bits corresponding to the coefficients of the polynomial associated with the symbol in the extension field. Let $v(\cdot): GF(q) \rightarrow GF(2)^b$ be the function from $GF(q)$ symbols to their corresponding binary representation. It is not hard to see that $v(0) = [0 \ 0 \ \ldots \ 0]^T$, and for any non-zero symbol $\beta = \alpha^i \in GF(q)$

$$v(\beta) = [r_0 \ r_1 \ \ldots \ r_{b-1}]^T, \quad (4.11)$$

where $R(x) = r_0 + r_1 x + \cdots + r_{b-1} x^{b-1}$, is the remainder of the division of $x^i$ by $P(x)$, i.e.,

$$x^i = Q(x) P(x) + R(x), \quad (4.12)$$

for some $Q(x) \in GF(2)[x]$.

It can be easily shown that for any two symbols $\beta_1$ and $\beta_2 \in GF(q)$, we have

$$v(\beta_1) \oplus_2 v(\beta_2) = v(\beta_1 + \beta_2), \quad (4.13)$$

where $\oplus_2$ is the modulo-2 summation of the $b \times 1$ binary vectors.

$GF(q)$ symbols can also be represented as $b \times b$ matrices. Let us define

$$M(\beta) \triangleq [v(\beta) \ v(\alpha \beta) \ v(\alpha^2 \beta) \ \ldots \ v(\alpha^{b-1} \beta)], \quad (4.14)$$

where $M(\beta)$ is the $b \times b$ matrix representation of the extension field symbols. It is not hard to check that $M(\alpha)$ is the companion matrix of the primitive polynomial $P(x)$

$$M(\alpha) = \begin{pmatrix} 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & \ldots & 0 & p_1 \\ 0 & 1 & \ldots & 0 & p_2 \\ \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & p_{b-1} \end{pmatrix}. \quad (4.15)$$

The matrix representation of the primitive element of the field $M(\alpha)$ generates a subring of $M_b(GF(2))$ isomorphic to the $GF(2^b)$ field [20]. In particular, the set $S \triangleq \{M(0), M(\alpha)^u : u = 0, 1, \ldots, b-2\}$, constitutes a field of matrices under matrix addition and multiplication operations [21]. For any symbols $\beta_1$ and $\beta_2$ over $GF(q)$,

$$M(\beta_1) \oplus_2 M(\beta_2) = M(\beta_1 + \beta_2)$$

$$M(\beta_1) \otimes_2 M(\beta_2) = M(\beta_1 \beta_2), \quad (4.16)$$
where $\oplus_2$ represents modulo-2 summation of $b \times b$ binary matrices, and $\otimes_2$ is modulo-2 multiplication of $b \times b$ binary matrices. It can be easily shown that

$$M(\beta_1) \otimes_2 v(\beta_2) = v(\beta_1 \beta_2),$$

(4.17)

where for simplicity we let $\otimes_2$ represent modulo-2 multiplication of the $b \times b$ binary matrices with $b \times 1$ binary vectors. From (4.13) and (4.17), one can deduce that a binary image can be derived by essentially writing each non-binary symbol of the non-binary codeword in terms of its $b \times 1$ binary vector representation.

A $GF(q)$ code of blocklength $n$ corresponds to a binary code of length $n^{(2)} = nb$ bits. If $H^{(q)}$ is the parity-check matrix of the non-binary code, the parity-check matrix $H^{(2)}$ of the binary image of this code is obtained by replacing each element of $H^{(q)}$ by corresponding $b \times b$ matrix representation. We call this code the basic-binary-image code. Note that every codeword in the non-binary code corresponds to a unique codeword in the basic-binary-image code, and the basic-binary-image code is of the same size as the non-binary code.

Once we have the parity-check matrix $H^{(2)}$ of the basic-binary-image code, we can work with the Tanner graph of this code to decode the received symbols. We thus reduce the problem of decoding $GF(q)$ LDPC codes to the problem of decoding binary LDPC codes. Consequently, BP can be used to decode the LDPC code at this stage. However, the presence of many short cycles in $H^{(2)}$ proves to be a detrimental factor for BP decoding. To get around this problem, we suggest a greedy algorithm to obtain binary images with fewer short cycles.

### 4.3.2 Mixed Binary Image Codes

In basic-binary-image codes, the binary image of a codeword is obtained by replacing symbols in the codeword with their $b \times 1$ binary representation. The $b$-bit binary representation $v(\cdot)$ is a linear mapping from $GF(q)$ to $GF(2)^b$. Consequently, we can try to employ other linear mappings.

Consider a non-singular matrix $B \in \mathbb{M}_b(GF(2))$. For any symbol $\beta$ over $GF(q)$, define

$$v_B(\beta) \triangleq B \otimes_2 v(\beta).$$

(4.18)
Clearly, \( \mathbf{v}_B(\cdot) \) is also a linear mapping from \( GF(q) \) to \( GF(2)^b \), i.e.,

\[
\mathbf{v}_B(\beta_1) \oplus_2 \mathbf{v}_B(\beta_2) = \mathbf{v}_B(\beta_1 + \beta_2).
\] (4.19)

As a result, \( \mathbf{v}_B(\cdot) \) could be used to obtain other binary images, once a matching matrix representation is derived for it. For any symbol \( \beta \) over \( GF(q) \), define

\[
\mathbf{M}_B(\beta) \triangleq \mathbf{B} \otimes_2 \mathbf{M}(\beta) \otimes_2 \mathbf{B}^{-1},
\] (4.20)

where \( \mathbf{B}^{-1} \in M_b(GF(2)) \) is the inverse of \( \mathbf{B} \), i.e.,

\[
\mathbf{B} \otimes_2 \mathbf{B}^{-1} = \mathbf{B}^{-1} \otimes_2 \mathbf{B} = \mathbf{I}_b,
\] (4.21)

with \( \mathbf{I}_b \in M_b(GF(2)) \) representing the \( b \times b \) identity matrix. For any two \( GF(q) \) symbols \( \beta_1 \) and \( \beta_2 \),

\[
\begin{align*}
\mathbf{M}_B(\beta_1) \otimes_2 \mathbf{M}_B(\beta_2) &= \mathbf{M}_B(\beta_1 + \beta_2) \\
\mathbf{M}_B(\beta_1) \otimes_2 \mathbf{M}_B(\beta_2) &= \mathbf{M}_B(\beta_1 \beta_2),
\end{align*}
\] (4.22)

and,

\[
\mathbf{M}_B(\beta_1) \otimes_2 \mathbf{v}_B(\beta_2) = \mathbf{v}_B(\beta_1 \beta_2).
\] (4.23)

The matrix representation of the primitive element generates a subset of \( M_b(GF(2)) \) isomorphic to \( GF(2^b) \). In other words, the set \( S_B \triangleq \{ \mathbf{M}_B(0), \mathbf{M}_B(\alpha)^u : u = 0, 1, \ldots, b - 2 \} \), is a field under matrix addition and multiplication operations. As a result, a new class of binary image codes can be obtained by writing the non-binary symbols in the codeword in terms of their vector representation \( \mathbf{v}_B(\cdot) \).

Let us make a significant generalization before introducing the mixed-binary-image codes\(^2\). In basic-binary-image codes all non-binary symbols are treated similarly. To be more precise, a given embedding of \( GF(q) \) symbols into \( b \times 1 \) binary vectors is used to transform the non-binary symbols into binary ones. However, similar to the Justesen codes [22], [20], each symbol can be treated differently. In fact, a set of \( n \) non-singular matrices can be chosen to map the non-binary symbols of a given codeword. Define

\[
\mathbf{B} \triangleq (\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_n),
\] (4.24)

\(^2\)According to Skolem-Noether theorem, all the possible \( GF(q) \)-isomorphisms in \( M_b(GF(2)) \) can be generated using \( \mathbf{M}_B(\alpha) \) matrices, for non-singular matrices \( \mathbf{B} \in M_b(GF(2)) \) [18]. Therefore, there is no hope of finding any other embedding of \( GF(q) \) symbols into \( M_b(GF(2)) \).
where \( B_1, B_2, \ldots, B_n \) are nonsingular matrices in \( M_b(GF(2)) \). We suggest to represent the \( i \)th non-binary symbol, for \( i = 1, 2, \ldots, n \), using \( v_{B_i}(\cdot) \). Consequently, a mixture of different mappings can be used to represent various symbols of a given codeword. We call these codes the mixed-binary-image codes.

Now, let us focus on the derivation of the parity-check matrix for mixed-binary-image codes. For any symbol \( \beta \in GF(q) \), define
\[
\tilde{M}_{B_i}(\beta) \triangleq M(\beta) \otimes_2 B_i^{-1}.
\] (4.25)

If \( H(q) \) is the parity-check matrix of the non-binary code, we claim that the parity-check matrix of the mixed-binary-image code \( H^{(2)}_B \) is obtained by replacing the \( GF(q) \) symbols at the \( i \)th column of \( H^{(q)} \), for \( i = 1, 2, \ldots, n \), with their corresponding matrix representation, i.e., \( \tilde{M}_{B_i}(\cdot) \).

To show that \( H^{(2)}_B \) is the parity-check matrix of the mixed-binary-image code, consider the constraint imposed by the \( j \)th non-binary check node, for \( j = 1, 2, \ldots, n-k \),
\[
\sum_{i=1}^{n} H_{j,i}^{(q)} x_i^{(q)} = 0,
\] (4.26)
where \( H_{j,i}^{(q)} \) is the element in the \( j \)th row and \( i \)th column of \( H \), and \( x_i^{(q)} \) is the \( i \)th element of the non-binary code. In a mixed-binary-image code, this constraint would be transformed to
\[
\sum_{i=1}^{n} \tilde{M}_{B_i} \left( H_{j,i}^{(q)} \right) \otimes_2 v_{B_i} \left( x_i^{(q)} \right),
\] (4.27)

Considering the fact that for any two symbols \( \beta_1, \beta_2 \in GF(q) \), we have
\[
\tilde{M}_{B_i}(\beta_1) \otimes_2 v_{B_i}(\beta_2) = v(\beta_1 \beta_2).
\] (4.28)

As a result, (4.27) can be rewritten as follows:
\[
\sum_{i=1}^{n} v \left( H_{j,i}^{(q)} x_i^{(q)} \right).
\] (4.29)

By (4.13), we have
\[
\sum_{i=1}^{n} v \left( H_{j,i}^{(q)} x_i^{(q)} \right) = v \left( \sum_{i=1}^{n} H_{j,i}^{(q)} x_i^{(q)} \right),
\] (4.30)
and, as a result, one can deduce that the equation
\[
\sum_{i=1}^{n} \tilde{M}_{B_i} \left( H_{j,i}^{(q)} \right) \otimes_2 v_{B_i} \left( x_i^{(q)} \right) = v(0),
\] (4.31)
represents the same constraint as (4.26).

Each codeword in a mixed-binary-image code can be obtained by a linear transformation from a codeword in the basic-binary-image code. Assume the non-binary code to be of length \( n \), the mixed-binary-image code would be of length \( n(2) = nb \) bits. Let \( C \) and \( C_B \) represent the basic-binary-image and mixed-binary-image codes, respectively. Clearly,

\[
\forall x^{(2)} \in C \Rightarrow \exists x_B^{(2)} \in C_B : x^{(2)}_B = \text{diag}(B_1, B_2, \ldots, B_n)x^{(2)}, \tag{4.32}
\]

where \( \text{diag}(B_1, B_2, \ldots, B_n) \) is a block-diagonal \( n(2) \times n(2) \) matrix with \( B_1, B_2, \ldots, B_n \) on the primary diagonal.

From the parity-check matrix of the mixed-binary-image code, we can obtain its Tanner graph. There are \( \mu_b \) nonsingular matrices in \( \mathbb{M}_b(GF(2)) \), where

\[
\mu_b \triangleq 2^{\frac{b(b-1)}{2}} [b]_2!
= \prod_{i=0}^{b-1}(2^b - 2^i), \tag{4.33}
\]

with \([b]_2!\) representing the \( q \)-factorial notation [23]. As a result, we can obtain \( \mu_b^n \) different mixed-binary-image codes for a given non-binary code. One can choose the image code with the least number of short cycles to get the best performance when using iterative decoding.

Hinging on the fact that all \( GF(q) \)-symbols belong to \( GF(2^c) \) fields, where \( b \) divides \( c \), another class of binary images can be derived for a given non-binary code. We call these images the extended-binary-image codes. See Appendix 4.A for more details.

### 4.3.3 Minimum Distance and Short Cycles

So far we have presented two different approaches to obtain binary images of a given non-binary code, namely: basic-binary-image and mixed-binary-image codes. The basic-binary-image codes are a special case of mixed-binary-image codes. Consequently, in this section, we only consider the mixed-binary-image codes.

There exist, \( \mu_b^n \) different mixed-binary-image codes for a given \( GF(2^b) \) LDPC code, all of which are linear transformation of one another. In this section, we will try to get a better understanding about the performance of these images.
A number of parameters can be used to evaluate the performance of different codes over BI-AWGN channels. Weight spectrum and minimum distance are only two of these parameters. Evaluation of the weight spectrum of various images seems to be a hard problem to tackle. Consequently, we will try to quantify the minimum distance of image codes, instead.

The following lemma gives upper and lower bounds on the minimum distance of these images.

**Lemma 4.1** If the non-binary linear code has the minimum distance of $W^{(q)}$, then the minimum distance of any binary image of this code $W^{(2)}$ is bounded as follows:

$$W^{(q)} \leq W^{(2)} \leq \frac{bq}{2(q-1)} W^{(q)},$$

(4.34)

**Proof:** To prove this lemma, we first verify the lower bound. If there exists a binary image codeword with Hamming weight less than $W^{(q)}$, then its corresponding non-binary codeword would have a weight less than $W^{(q)}$ as well, which is a contradiction. Consequently, it is impossible to have any binary codeword with weight less than $W^{(q)}$, or, equivalently, $W^{(2)} \geq W^{(q)}$.

Now, let us establish the upper bound. Consider a minimum weight codeword in the non-binary code. Any scalar product of this codeword is also a valid codeword from the non-binary codebook. There are $q$ elements in $GF(q)$, and as a result, there would be $q$ “scalar-product” codewords corresponding to the minimum weight codeword in the non-binary code.

Define the binary span of a given non-binary codeword as the set of all positions in the binary image codeword that correspond to the binary representation of the non-zero elements in the non-binary codeword. The binary span of the minimum-distance codeword has a cardinality of $W^{(q)} b$ bits. Furthermore, all the $q$ scalar-product codewords of the minimum distance codeword have the same binary span. Now, let us just consider binary images of these $q$ scalar-product codewords. One can easily see that these $q$ product codewords constitute a binary code of length $W^{(q)} b$ bits, with $q$ codewords. This code, denoted by $\tilde{C}$, is a linear code and therefore its minimum distance and minimum weight are the same.

One can easily see that

$$\sum_{x \in \tilde{C}} w_H(x) = \frac{q}{2} W^{(q)} b,$$

(4.35)
where \( w_H(x) \) represents the Hamming weight of the codeword \( x \). This is due to the fact that each position in the binary span is zero in half the codewords and one in the remaining half. It’s not hard to see that

\[
\sum_{x \in \tilde{C}} w_H(x) \geq (q - 1)W^{(2)}.
\]  

(4.36)

Putting (4.35) and (4.36) together, we get

\[
W^{(2)} \leq \frac{bq}{2(q - 1)} W^{(q)},
\]  

(4.37)

as desired.

One can generate a mixed-binary-binary image code such that its minimum distance \( W^{(2)} \) is equal to the minimum distance of the non-binary code \( W^{(q)} \). In other words, the lower bound on the minimum distance of the binary codes (4.34) is a tight lower bound. However, other bounds such as the sphere-packing bound or Singleton bound can be applied to derive other upper bounds. Employing the sphere-packing bound [25], we get the following upper bound on the minimum weight

\[
W^{(2)} \leq \max \left\{ d \left| \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{bW^{(q)}}{i} \leq \frac{2^{bW^{(q)}}}{q} \right. \right\},
\]  

(4.38)

The binary image graphs could contain a number of four-cycles. These cycles would degrade the performance of iterative message-passing decoders. Assume that the non-binary graph does not contain any four-cycles. Consequently, four-cycles in the binary graph would correspond to conversion of the non-binary variable and check nodes to binary variable and check nodes. To minimize the problems associated with these cycles, we present two techniques to reduce the number of four-cycles in the binary image graphs.

**Reducing the number of four-cycles due to the conversion of non-binary variable nodes to binary variable nodes**

Consider a length-\( n \) LDPC code over \( GF(q) \), where \( q = 2^b \). Let \( H^{(2)} \) be the parity-check matrix of the basic-binary-image of this code. Consider invertible matrices \( B_1, B_2, \ldots, B_n \in M_b(GF(2)) \), and let \( \mathcal{B} = \{ B_1, B_2, \ldots, B_n \} \). One can easily see that the
parity-check matrix of the mixed-binary-image code corresponding to $B$, i.e., $H^{(2)}_B$, is given by

$$H^{(2)}_B = H^{(2)} \otimes_2 \text{diag} \left( B_1^{-1}, B_2^{-1}, \ldots, B_n^{-1} \right).$$

(4.39)

There are $\mu_b$ nonsingular matrices in $M_b(GF(2))$, and, as a result, $\mu_b^n$ choices for $B$. Consequently, one can explore among all possible choices to find the parity-check matrix with the fewest four-cycles.

**Reducing the number of four-cycles due to the conversion of non-binary check nodes to binary check nodes**

Consider the $j$th non-binary check node for $j = 1, 2, \ldots, n - k$. In a mixed-binary-image code, the constraint imposed by this check node is represented as follows:

$$\sum_{i=1}^{n} H^{(2)}_{j,i} x_i = 0_b,$$

(4.40)

where $H^{(2)}_{j,i}$ is the $b \times b$ matrix corresponding to the element in the $j$th row and $i$th column of the non-binary parity-check matrix, $x_i$ is the $b \times 1$ vector representation of the $i$th element of the non-binary code, and $0_b$ is the $b \times 1$ all-zero vector. For any nonsingular matrix $C_j \in M_b(GF(2))$,

$$\sum_{i=1}^{n} C_j H^{(2)}_{j,i} x_i = 0_b,$$

(4.41)

represents the same constraint as (4.40). As a result, an equivalent parity-check matrix can be generated by replacing $H^{(2)}_{j,i}$s, for $1 \leq i \leq n$, with $C_j H^{(2)}_{j,i}$s. Other equivalent parity-check matrices can be obtained by replacing the rest of the check equations with equivalent constraints. Hence, to further reduce the number of four-cycles, one can search among all invertible matrices $C_j \in M_b(GF(2))$, for $1 \leq j \leq n - k$, to find the equivalent parity-check matrix with the fewest four-cycles.

A variety of loop-reduction algorithms can be devised based on the presented techniques. As an example, in Appendix 4.B we present a greedy loop-reduction algorithm.
4.4 Generalized Images

In Section 4.3, we presented two classes of binary images for a given non-binary code. These image codes can be decoded using message-passing algorithms. As a result, they can significantly reduce the decoding-complexity and the required decoding-memory. However, the binary image codes generally contain a number of four-cycles. These cycles could degrade the performance of message-passing decoders. Consequently, we will also consider the image codes over intermediate fields and rings.

4.4.1 Matrix Ring Image Codes

Consider an LDPC code of length \( n \) over \( GF(q) \), where \( q = 2^b \). Based on the proposed techniques, we can generate \( \mu_b n \) binary images for this code. Consider the Tanner graph of one of these binary image codes. An \( M_s(GF(2)) \) ring-image of this non-binary code is obtained by combining each group of \( s \) consecutive binary variable nodes and considering them as a modified variable node and combining each group of \( s \) consecutive binary check nodes as a modified check node. Let us call these images the matrix-ring-image codes.

In this chapter we only concentrate on matrix-ring-image codes over matrix rings \( M_s(GF(2)) \) where \( s|b \), however the method presented here can be generalized for an arbitrary \( s \) by adding dummy variable and check nodes. The dummy variable nodes are generated by adding all-zero columns to the parity-check matrix of the binary-image-code and the dummy check nodes are obtained by adding all-zero rows to the parity-check matrix. Note that the constraints represented by the dummy check nodes are always satisfied.

A modified parity-check matrix and a modified Tanner graph can be defined for matrix-ring-image codes\(^3\). To derive the modified parity-check matrix, we consider the \( s \times s \) binary matrices consisting of the components in the \( s \) consecutive rows and the \( s \) consecutive columns of the original binary parity-check matrix as the corresponding

---

\(^3\)Note that modified parity-check matrices and modified Tanner graphs, associated with MISO-LDPC systems, are generated by merging \( M_{n_T} \) consecutive variable nodes together. In addition, the matrix-ring-image codes are obtained by combining \( s \) consecutive binary variable nodes and \( s \) consecutive binary check nodes together. For convenience, we refer to the parity-check matrices and the graphs corresponding to matrix-ring-image codes as modified parity-check matrices and modified Tanner graphs, respectively. Note that unlike MISO-LDPC modified Tanner graphs, each modified check node of a modified Tanner graph associated with a matrix-ring-image code represents a number of binary check nodes.
components in the modified matrix.

As in the binary LDPC graph, each column of the modified parity-check matrix represents a modified variable node. Since variable nodes in a matrix-ring-image code correspond to $s$ consecutive variable nodes in the original graph, each column of the modified parity-check matrix represents $s$ consecutive columns of the original parity-check matrix. Similarly, modified check nodes, that are check nodes in the matrix-ring-image code, correspond to $s$ consecutive check nodes in the original graph. Moreover, each row of the modified parity-check matrix represents $s$ consecutive rows of the original parity-check matrix.

If any of the $s$ binary constituent variable nodes of a modified variable node are connected to one of the $s$ component check nodes of a modified check node (in the binary image graph), there would be an edge connecting the modified variable node to the corresponding modified check node in the modified graph. In the modified Tanner graph, the edges are not all identical. To differentiate the edges in the modified Tanner graph, we label them with the corresponding component in the modified parity-check matrix. Example 4.1 sheds light on the procedure of deriving the modified parity-check matrix and Tanner graph for the matrix-ring-image of a non-binary LDPC code.

**Example 4.1** Consider a length $n = 2$ LDPC code over $GF(16)$ with the parity-check matrix

$$H = [\alpha^3 \; \alpha^5].$$

The basic-binary-image code, associated with the primitive polynomial $P(x) = x^4 + x + 1$, is a length $n^{(2)} = 8$ binary code with parity-check matrix

$$H^{(2)} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

And, the $M_2(GF(2))$ matrix-ring-image code, associated with this basic-binary-image,
is a length $n^{(4)} = 4$ code with modified parity-check matrix

$$H^{(4)} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0
\end{bmatrix}.$$  

Figure 4.1 represents the Tanner graph of the original $GF(16)$ LDPC code, the Tanner graph of the basic-binary-image code, and the modified Tanner graph of the $M_2(GF(2))$ matrix-ring-image code.

### 4.4.2 Belief Propagation

The BP decoder for an $M_q(GF(2))$ matrix-ring-image of a $GF(q)$ LDPC code is an iterative decoding algorithm. The messages are length-$2^s$ probability vectors of a-posteriori probabilities of the transmitted symbols, i.e., the messages are

$$m = \left( \Pr([0 0 \ldots 0]|y), \Pr([0 0 \ldots 1]|y), \ldots, \Pr([1 1 \ldots 1]|y) \right). \quad (4.42)$$
Let \( \mathbf{m}_i = \Pr \left( \langle i \rangle | y \right), i = 0, 1, \ldots, 2^s - 1 \) with \( \langle i \rangle \) denoting the length-\( s \) row vector corresponding to the \( s \)-bit representation of \( i \). The channel messages \( \mathbf{m}^{(c)} \) are initialized based on the received values and the knowledge of the channel model. Then the messages would be passed between the modified variable nodes and the modified check nodes, and would be processed at the nodes.

At the initialization, each modified variable node passes the channel messages to the neighboring modified check nodes. At any subsequent iteration, the variable-to-check messages are updated as follows:

\[
\mathbf{m}^{(o)}_i = \mathbf{m}^{(c)}_i \prod_{j=1}^{d_v-1} \mathbf{m}^{(j)}_i, \quad \text{for } i = 0, 1, \ldots, 2^s - 1, \tag{4.43}
\]

where \( d_v \) is the degree of the modified variable node, i.e., the number of neighboring modified check nodes. \( \mathbf{m}^{(o)} \) denotes the outgoing variable-to-check message, and \( \mathbf{m}^{(j)}_i, j = 1, 2, \ldots, d_v - 1 \) represents the incoming check-to-variable messages along the \( d_v - 1 \) remaining edges. Note that normalizing the output message vector \( \mathbf{m}^{(o)} \) is necessary to keep the outgoing messages as a probability vector. The complexity of the modified-variable node update scales as \( O(2^s) \) operations per iteration, per edge, per modified variable node.

Let \( \mathbf{0}_{ss} \) be the \( s \times s \) all-zero matrix. The edge label \( \mathcal{E} \in \mathbb{M}_s(GF(2)) \setminus \mathbf{0}_{ss} \) associated with the edge corresponding to the message \( \mathbf{m} \) from a modified variable node to a modified check node results in the transformation of the vector \( \mathbf{m} \) as follows:

\[
\bar{\mathbf{m}}_k = \sum_{i \in \mathbb{I}_E(k)} \mathbf{m}_i, \quad \text{for } k = 0, 1, \ldots, 2^s - 1, \tag{4.44}
\]

where

\[
\mathbb{I}_E(k) = \left\{ j : 0 \leq j \leq 2^s - 1, \langle j \rangle_s \otimes_2 \mathcal{E}^T = \langle k \rangle_s \right\}. \tag{4.45}
\]

Note that when the edge label \( \mathcal{E} \) is a non-singular matrix, this transformation becomes a permutation.

Consider a degree-\( d_c \) modified check node. The transformed message vectors are processed at this node according to

\[
\bar{\mathbf{m}}^{(o)}_i = \sum_{(i_1,i_2,\ldots,i_{d_c-1}) \in \mathbb{I}_E^{(d_c-1)}} \prod_{j=1}^{d_c-1} \bar{\mathbf{m}}^{(j)}_{i_j}, \quad \text{for } i = 0, 1, \ldots, 2^s - 1, \tag{4.46}
\]
where $\bar{m}^{(o)}$ is the transformed outgoing messages, $\bar{m}_i^{(j)}$, $j = 1, 2, \ldots, d_v - 1$ represents transformed incoming variable-to-check messages along the remaining $d_v - 1$ edges, and

$$\mathbb{J}_i^{(d_v-1)} = \{(i_1, i_2, \ldots, i_{d_v-1}) : \langle i_1 \rangle_s \oplus_2 \langle i_2 \rangle_s \oplus_2 \cdots \oplus_2 \langle i_{d_v-1} \rangle_s = \langle i \rangle_s \}.$$ (4.47)

This is an $O(2^{s} s)$-complex process per iteration, per edge, per modified check node when Fourier (Walsh-Hadamard) transforms are used. If, however, the updates are performed using look-up tables, the complexity is $O(2^{2s})$.

These messages are then untransformed to obtain the check-to-variable messages. The message vector along an edge with label $E \in \mathbf{M}_s(\text{GF}(2)) \setminus \{0\}_s$ is mapped as follows:

$$m_k = \bar{m}_k^{(o)} \langle L_E(k) \rangle_\infty | I_E(\langle L_E(k) \rangle_\infty)|,$$ for $k = 0, 1, \ldots, 2^s - 1$, (4.48)

where $\langle \cdot \rangle_\infty$ denotes the integer representation of a given binary vector, $|\cdot|$ is the cardinality operator, and $L_E(k) \triangleq \langle k \rangle_s \otimes_2 E^T$.

After a predefined number of iterations, the final a-posteriori probabilities are derived at modified variable nodes as follows:

$$m_i^{(f)} = m_i^{(c)} \prod_{j=1}^{d_v} m_i^{(j)}, \text{ for } i = 0, 1, \ldots, 2^s - 1,$$ (4.49)

where $m_i^{(j)}$, $j = 1, 2, \ldots, d_v$ represents the incoming check-to-variable messages, $m_i^{(c)}$ is the channel message, and $m_i^{(f)}$ denotes the a-posteriori message vector. Note that normalization is necessary to get a-posteriori probability vectors from a-posteriori messages.

Consider the $i^{th}$ bit of a modified variable node, for $i = 1, 2, \ldots, s$. Given the final a-posteriori probability vector, the log-likelihood ratio of this bit can be computed as follows:

$$m_i^{(f)} = \ln \left( \frac{\sum_{j \in \mathbb{D}_0^{(i)}} m_j^{(f)}}{\sum_{j \in \mathbb{D}_1^{(i)}} m_j^{(f)}} \right),$$ (4.50)

where

$$\mathbb{D}_0^{(i)} = \{j : \langle j \rangle_{s,i} = 0\}$$

and

$$\mathbb{D}_1^{(i)} = \{j : \langle j \rangle_{s,i} = 1\}.$$ (4.51)
with \( \langle j \rangle_{s,i} \) denoting the \( i \)th element of the \( s \)-bit binary representation of \( j \).

If the log-likelihood ratio of a given bit is greater than or equal to zero, the bit is decoded to 0, otherwise the bit is decoded as a 1.

**Lemma 4.2** If modified Tanner graph of a matrix-ring-image code is acyclic, then modified BP decoding is equivalent to bitwise maximum-likelihood decoding.

The proof of this lemma runs similar to the corresponding proof for binary LDPC codes [26], and consequently we omit the details.

## 4.5 Complexity Evaluation

Table 4.1 presents a comparison between the variable and check node update complexity of the following decoding algorithms for regular \((d_v, d_c)\)-LDPC codes over \(GF(q)\): \(q\)-ary BP, FFT-based \(q\)-ary BP, EMS, min-max, sum-product decoding over binary image codes, and BP decoding over \(M_s(GF(2))\) matrix-ring-image codes, where \(q' = 2^s\).

Note that Table 4.1 gives the worst case complexity orders for sum-product decoding over binary image codes, and BP over matrix-ring-image codes. This table presents the complexity order for the set of variable nodes or check nodes in the image code that correspond to a given variable node or check node in the original \(q\)-ary graph. Moreover, complexity reduction up to \(O(q \log q)\) for EMS is possible by enhancements suggested in [14].

The computational complexity of sum-product decoding over binary image codes grows as \((\log q)^2\) with the Galois field size \(q\), better than all previous algorithms. Moreover BP decoding of \(M_s(GF(2))\) matrix-ring-image codes has a computational complexity growing as \(q' \log q' (\log_q q)^2\), which is less than previous algorithms except for the sum-product decoding over proposed binary image codes.

## 4.6 Numerical Analysis

To empirically investigate the performance of the proposed decoders, we performed Monte Carlo simulations of random LDPC codes of various lengths and degree-distributions over AWGN channels.
In Figures 5.1 and 5.2, we present bit error rate (BER) and word error rate (WER) curves, respectively, for a length-80 (3, 6)-regular LDPC code over GF(16). These figures present the performance of SP decoding of the basic-binary-image and an optimized mixed-binary-image code as well. The mixed-binary-image code is obtained from an optimization with an objective to reduce the number of 4-cycles in the binary image code and contains 185 4-cycles whereas the the basic-binary-image code has 1419 4-cycles. Performance of the BP decoding of the $M_2(GF(2))$ matrix-ring-image code corresponding to the basic-binary-image code and an optimized $M_2(GF(2))$ matrix-ring-image code are also presented in these figures. Again, the optimized matrix-ring-image code is obtained with an optimization aiming at the reduction of the number of 4-cycles in the matrix-ring-image code.

Figures 5.3 and 5.4 compare the performance of decoders for a length-240 (3, 6)-regular LDPC code over GF(16). The Tanner graph of the basic-binary-image code contains 4217 4-cycles whereas the Tanner graph of the optimized mixed-binary-image code contains 581 4-cycles.

Figures 5.5 and 5.6 present the simulation results for a length-80 (3, 4)-regular LDPC code over GF(8). The Tanner graphs of the basic-binary-image code and the optimized mixed-binary-image code, respectively, contain 625 and 37 4-cycles.

Figures 4.8 and 4.9 compare the performance of decoders for a length-240 (3, 4)-regular LDPC code over GF(16). The optimized mixed-binary-image code contains only 278 4-cycles while the basic-binary-image code contains 4680 4-cycles.

Note that in all these simulations, BP decoding of the basic-binary-image code
Figure 4.2: Bit error rate curves of LDPC\((n = 80, \lambda(x) = x^2, \rho(x) = x^5)\) codes when used over AWGN channels.

Figure 4.3: Word error rate curves of LDPC\((n = 80, \lambda(x) = x^2, \rho(x) = x^5)\) codes when used over AWGN channels.
Figure 4.4: Bit error rate curves of LDPC\( (n = 240, \lambda (x) = x^2, \rho (x) = x^5) \) codes when used over AWGN channels.

Figure 4.5: Word error rate curves of LDPC\( (n = 240, \lambda (x) = x^2, \rho (x) = x^5) \) codes when used over AWGN channels.
Figure 4.6: Bit error rate curves of LDPC($n = 80, \lambda(x) = x^2, \rho(x) = x^3$) codes when used over AWGN channels.

Figure 4.7: Word error rate curves of LDPC($n = 80, \lambda(x) = x^2, \rho(x) = x^3$) codes when used over AWGN channels.
Figure 4.8: Bit error rate curves of LDPC\((n = 240, \lambda(x) = x^2, \rho(x) = x^3)\) codes when used over AWGN channels.

Figure 4.9: Word error rate curves of LDPC\((n = 240, \lambda(x) = x^2, \rho(x) = x^3)\) codes when used over AWGN channels.
and $M_2(GF(2))$ matrix-ring-image code associated with the basic-binary-image code suffer from the high error-floor problem. However, the BP decoding of the optimized mixed-binary-image code and the optimized $M_2(GF(2))$ matrix-ring-image code have a performance close to the non-binary BP.

4.7 Concluding Remarks

In this chapter, we investigated different approaches to generate binary images of a given non-binary LDPC code. We introduced two classes of binary images for a given non-binary code, namely: basic-binary-image, and mixed-binary-image codes. We derived upper and lower bounds on the minimum distance of these images, and presented techniques to reduce the number of short cycles in the Tanner graph of the binary image codes. To better understand the complexity-performance tradeoffs, we also considered images over intermediate fields and rings. We derived BP decoding algorithms for these codes, and presented techniques to obtain matrix-ring-image codes with performance close to the BP decoding of the original non-binary LDPC code. With numerical simulations, we showed that optimized image codes perform very close to the high-complexity BP decoding of the original non-binary LDPC code.

4.8 Acknowledgement


Appendix 4.A Extended Binary Image Codes

So far, we have established two techniques to generate binary images of a non-binary code. Given that every $GF(q)$-symbol also belongs to $GF(2^c)$ fields where $b$ divides $c$, we can employ each of the proposed methods to obtain a subset of matrices
over $\mathbb{M}_c(GF(2))$ isomorphic to the $GF(q)$ field and then generate the binary image of the code based on the isomorphic field.

Consider a $GF(2^c)$ field for a $c$ such that $b|c$. Let $\alpha$ be a primitive element of this field, and, furthermore let $\bar{P}(x)$ be the primitive polynomial associated with $\alpha$. Define $\bar{A}$ as the companion matrix corresponding to $\bar{P}(x)$. It’s not hard to see that $\alpha^{(2^c-1)/(2^b-1)}$ generates a subfield of $GF(2^c)$ isomorphic to $GF(2^b)$. Moreover, $\bar{A}^{(2^c-1)/(2^b-1)}$ generates a subset of matrices over $\mathbb{M}_c(GF(2))$ isomorphic to $GF(2^b)$. Consider a non-singular matrix $\bar{B} \in \mathbb{M}_c(GF(2))$. Clearly

$$\bar{B} \otimes_2 \bar{A}^{(2^c-1)/(2^b-1)} \otimes_2 \bar{B}^{-1},$$

also creates a subset of matrices isomorphic to $GF(2^b)$ over $\mathbb{M}_c(GF(2))$. Considering the fact that there are $\mu_c \triangleq 2^{2^{c-1}[c]_2!}$ such non-singular matrices in $\mathbb{M}_c(GF(2))$, one can obtain $\mu_c$ various $GF(q)$-isomorphism over $\mathbb{M}_c(GF(2))$. Consequently, using the suggested techniques, one can generate $\mu_c^n$ different binary images for a given length-$n$ $GF(q)$ code. We name these images the extended-binary-images of the non-binary code.

The extended-binary-image codes can be divided into two separate subclasses: the basic-extended-binary-image codes and the mixed-extended-binary-image codes. Each codeword in these subclasses can be obtained by a linear transformation from a corresponding codeword in another subclass. In addition, the parity-check matrices of these codes are related to each other with a linear transformation. As a result, we will henceforth focus on the basic-extended-binary-image codes.

Consider the matrix field isomorphic to $GF(q)$ based on the companion matrix of a primitive polynomial of $GF(2^c)$. Let $H^{(q)}$ be the parity-check matrix of the non-binary code. To generate the parity-check matrix $H^{(2)}$ of the basic-extended-binary-image code, we should first replace each $GF(q)$-element of $H^{(q)}$ by the corresponding $c \times c$ matrix from the matrix field. This way, a code of blocklength $n$ over $GF(q)$ becomes a length $\tilde{n}_2 = nc$ binary code. Let us call this code $C^{(inner)}$. Every codeword in the non-binary code corresponds to a unique codeword in $C^{(inner)}$, however $C^{(inner)}$ contains a number of codewords that do not belong to the non-binary code. This is due to the fact that $C^{(inner)}$ is the binary image of another equivalent LDPC code over $GF(2^c)$. The equivalent code contains codewords with elements from $GF(2^c)$ that do not necessarily
belong to \( GF(q) \). To get around this problem, we add extra constraints to the parity-check matrix of \( C_{\text{inner}} \).

In \( C_{\text{inner}} \), each \( GF(q) \) symbol is written as a vector of \( c \) bits. This vector corresponds to the coefficients of the polynomial associated with the symbol in \( GF(2^c) \). However, only \( b \) bits are sufficient to represent \( GF(q) \) symbols. Consequently, a code \( C_{\text{outer}} \) can be defined such that \( C_{\text{outer}} \) encodes the \( b \)-bit representation of \( GF(q) \) symbols into their corresponding \( c \)-bit representation. Assume \( \alpha \) is a primitive element of \( GF(q) \). The generator matrix of \( C_{\text{outer}} \) is the \( b \times c \) matrix whose \( i \)th row is the \( c \)-bit representation of \( \alpha^{i-1} \). As a result, one can consider the binary image code as a concatenated code, i.e., a code with \( C_{\text{outer}} \) outer code and \( C_{\text{inner}} \) inner code.

Once we have parity-check matrices of the inner and outer codes, we can use a turbo-like decoder, see Figure 4.10, to decode the transmitted codewords [24]. We can also combine the parity-check matrices of both codes and consider them as one single code. This way, we can use BP or other message-passing algorithms to decode the transmitted codewords.

**Appendix 4.B  A Loop-Reduction Algorithm**

There are \( \mu_b \) nonsingular matrices in \( M_b(GF(2)) \). Let us call these nonsingular matrices \( N_1, N_2, \ldots, \) and \( N_{\mu_b} \).
Consider an LDPC code of block-length $n$ and dimension $k$ over $GF(q)$, where $q = 2^b$. Assume that $H^{(2)}$ is the parity-check matrix of the basic-binary-image of this code. For nonsingular matrices $B_1, B_2, \ldots, B_n \in M_b(GF(2))$, define

$$\mathbb{B} \triangleq \{B_1, B_2, \ldots, B_n\}. \quad (4.52)$$

The parity-check matrix of the mixed-binary-image code corresponding to $\mathbb{B}$, i.e., $H^{(2)}_{\mathbb{B}}$, is given by

$$H^{(2)}_{\mathbb{B}} = H^{(2)} \otimes_2 \text{diag} \left( B_1^{-1}, B_2^{-1}, \ldots, B_n^{-1} \right), \quad (4.53)$$

where $\text{diag} \left( B_1^{-1}, B_2^{-1}, \ldots, B_n^{-1} \right)$ is an $nb \times nb$ block-diagonal matrix with $B_1^{-1}, B_2^{-1}, \ldots, B_n^{-1}$ on the primary diagonal.

Consider $n - k$ nonsingular matrices $C_1, C_2, \ldots, C_{n-k} \in M_b(GF(2))$. Define

$$\mathbb{C} \triangleq \{C_1, C_2, \ldots, C_{n-k}\}, \quad (4.54)$$

and let

$$H^{(2)}_{\mathbb{B}, \mathbb{C}} = \text{diag} \left( C_1, C_2, \ldots, C_{n-k} \right) \otimes_2 H^{(2)}_{\mathbb{B}} \quad (4.55)$$

Finally, let $w \left( H^{(2)}_{\mathbb{B}, \mathbb{C}} \right)$ ($\sigma \left( H^{(2)}_{\mathbb{B}, \mathbb{C}} \right)$, respectively) denote the Hamming weight of $H^{(2)}_{\mathbb{B}, \mathbb{C}}$ (the number of four-cycles in $H^{(2)}_{\mathbb{B}, \mathbb{C}}$, respectively). Clearly, $H^{(2)}_{\mathbb{B}, \mathbb{C}}$ represents the same set of constraints as $H^{(2)}_{\mathbb{B}}$. Since there are $\mu_b$ nonsingular matrices in $M_b(GF(2))$, as a result, there are $\mu_b^n$ choices for $\mathbb{B}$ and $\mu_b^{n-k}$ choices for $\mathbb{C}$. Consequently, one can explore among all the possible $\mu_b^{2n-k}$ possible choices for $H^{(2)}_{\mathbb{B}, \mathbb{C}}$ to find the parity-check matrix with the fewest four-cycles. However, the extensive number of possibilities, for $\mathbb{B}$ and $\mathbb{C}$, makes it impractical to find the parity-check matrix with the least number of four-cycles through a brute-force search. Consequently, we suggest a Turbo-like greedy algorithm, instead.

The optimization problem of finding the parity-check matrix with the fewest four-cycles is computationally intensive. And, instead, we suggest use of Algorithm 5.1 which hopes to find the global optimum by making locally optimal choices at each optimization stage. In other words, since finding $2n - k$ nonsingular matrices $B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_{n-k}$ that result in the parity-check matrix with the fewest number of four-cycles is a hard problem, we suggest to run the optimization for only one matrix at each stage.
Algorithm 4.1 Greedy algorithm for reduction of four-cycles

**Initialization:**

\[ B_1 \leftarrow I_b, B_2 \leftarrow I_b, \ldots, B_n \leftarrow I_b \]
\[ B^\text{opt}_1 \leftarrow I_b, B^\text{opt}_2 \leftarrow I_b, \ldots, B^\text{opt}_n \leftarrow I_b \]
\[ C_1 \leftarrow I_b, C_2 \leftarrow I_b, \ldots, C_{n-k} \leftarrow I_b \]
\[ C^\text{opt}_1 \leftarrow I_b, C^\text{opt}_2 \leftarrow I_b, \ldots, C^\text{opt}_{n-k} \leftarrow I_b \]

**Optimization:**

while exit condition is not satisfied do

for \( i := 1 \) to \( n \) step 1 do

\[ B_i \leftarrow N_1 \]
\[ B^\text{opt} \leftarrow N_1 \]
\[ w_{\text{min}} \leftarrow w (H^{(2)}_{B,C}) \]
\[ \sigma_{\text{min}} \leftarrow \sigma (H^{(2)}_{B,C}) \]

for \( l := 2 \) to \( \mu_b \) step 1 do

\[ B_i \leftarrow N_l \]

if \( \sigma (H^{(2)}_{B,C}) \leq \sigma_{\text{min}} \) and \( w (H^{(2)}_{B,C}) \leq w_{\text{min}} \) then

\[ B^\text{opt} \leftarrow B_i \]
\[ w_{\text{min}} \leftarrow w (H^{(2)}_{B,C}) \]
\[ \sigma_{\text{min}} \leftarrow \sigma (H^{(2)}_{B,C}) \]

end if

end for

\[ B_i \leftarrow B^\text{opt} \]

end for

for \( j := 1 \) to \( n - k \) step 1 do

\[ C_j \leftarrow N_1 \]
\[ C^\text{opt} \leftarrow N_1 \]
\[ w_{\text{min}} \leftarrow w (H^{(2)}_{B,C}) \]
\[ \sigma_{\text{min}} \leftarrow \sigma (H^{(2)}_{B,C}) \]

for \( l := 2 \) to \( \mu_b \) step 1 do

\[ C_j \leftarrow N_l \]

if \( \sigma (H^{(2)}_{B,C}) \leq \sigma_{\text{min}} \) and \( w (H^{(2)}_{B,C}) \leq w_{\text{min}} \) then

\[ C^\text{opt} \leftarrow C_j \]
\[ w_{\text{min}} \leftarrow w (H^{(2)}_{B,C}) \]
\[ \sigma_{\text{min}} \leftarrow \sigma (H^{(2)}_{B,C}) \]

end if

end for

\[ C_j \leftarrow C^\text{opt} \]

end for

end while
Note that \( B_i \), for \( i = 1, 2, \ldots, n \), corresponds to the binary representation of the \( i^{th} \) variable node and \( C_j \), for \( j = 1, 2, \ldots, n - k \), is associated with the binary representation of the \( j^{th} \) check node. Algorithm 5.1 aims to optimize the parity-check matrix by local optimizations from the perspective of these individual variable and check nodes.

It is known that greedy algorithms provide good solutions for problems that have the “greedy choice property” and “optimal substructure” [27]. Consider each of the local optimizations. Clearly, finding the least number of four cycles does not have any of these properties, however finding the lowest weight parity-check matrix has both properties. Heuristically, a low-weight parity-check matrix is a good parity-check matrix and generally contains fewer four-cycles than a higher weight parity-check matrix. So, Algorithm 5.1 aims to choose the parity-check matrix that contains both the least weight and the fewest cycles. Note that different variations of this algorithm can be obtained by changing the optimization criterion. Moreover, an exit condition can be employed to restrict the number of iterations.

**Bibliography**


Chapter 5

Degree-Dependant Finite-Length Scaling for LDPC Codes

5.1 Introduction

LDPC codes, with their superb error correction performance [1], [2], are among the most promising forward error correction schemes for equal error protection (EEP). However, many practical applications, such as robust transmission of video and image require unequal error protection (UEP). The near capacity performance of EEP-LDPC codes over binary erasure channels (BECs) suggests LDPC codes could also offer good performance in UEP scenarios [3]. Consequently, the idea of designing LDPC codes for UEP has been studied in many papers, including [3] - [6]. In [4]-[6], UEP is provided with partially regular LDPC codes optimized using unequal density evolution, while Plotkin-type constructions are employed to design UEP-LDPC codes in [3]. Here, exploiting the finite-length scaling methodology [7], we introduce a new approach to analytically evaluate the inherent UEP properties in finite-length irregular LDPC codes.

It is known that in an irregular LDPC code transmitted over a BEC, variable nodes with larger degrees provide better protection than variable nodes with smaller degrees. In this work, extending the results of [8], [9], we quantify the effect of the variable-node degree on the bit erasure rate of the variable nodes in the waterfall region.

The rest of this chapter is organized as follows. Section 5.2 briefly establishes the required background and notation. Section 5.3 outlines the derivation of the bit
erasure rate (the scaling approximation, upper bounds, and lower bounds) based on the finite-length scaling of LDPC codes. Numerical simulations and performance analysis are presented in Section 5.4, and finally Section 5.5 concludes the chapter.

5.2 Preliminaries

This section is devoted to a concise outline of the peeling decoding algorithm, the traditional algorithm for decoding LDPC codes over BECs.

5.2.1 Peeling Algorithm

The Peeling Algorithm was first introduced in [10] for decoding graph-based codes over BECs and runs as described in Algorithm 5.1. In this algorithm, upon receiving the channel outputs, the known variable nodes send their values to the check nodes connected to them and are removed from the graph. The decoder proceeds by looking for a degree-one check node, i.e., a check node such that all but one of the variable nodes connected to it are known. If it finds one, it computes the value of the unknown variable node, propagates the value of the variable node to all check nodes connected to the variable node, and then removes it from the graph. If the decoder does not find a degree-one check node, then the decoding stops. At this point, the residual graph does not have a degree-one check node. As a result, two cases are possible: either the graph is empty, i.e., the decoding is successful, or the graph is not empty and all the remaining check nodes have degrees greater than one, i.e., the decoding has failed.

Consider decoding an LDPC code with the Peeling Algorithm over a BEC with an erasure probability of $\epsilon$. The threshold erasure parameter, $\epsilon^*$, is the maximum value of $\epsilon$ such that the probability of decoder failure tends to zero for all $\epsilon < \epsilon^*$ as $n$ tends to infinity.

The Peeling Algorithm can be parameterized in terms of a discrete time $t \in \{0, 1, 2, \ldots\}$. Let us denote the ratio of the number of degree-one check nodes, at time $t$, to the number of edges in the original graph by $r_1(t)$. Moreover, Similar to [10], let us define the normalized time $\bar{t}$ as

$$\bar{t} \triangleq \frac{t}{n\Lambda'(1)}. \tag{5.1}$$
Finally, let us define a variable $y$ such that

$$d\tilde{t} = -\epsilon\lambda(y)dy,$$

(5.2)

with the initial condition of $y = 1$ at $\tilde{t} = 0$. The ratio of the number of degree-one check nodes to the edges, i.e., $r_1$, can be parameterized in terms of $y$. For the simplicity of notation, we let $r_1(y)$ represent the ratio of the number of degree-one check nodes to the number of edges parameterized by $y$.

Critical points are defined as points where both $r_1$ and its partial derivative with respect to $y$ vanish. In the rest of this chapter, we concentrate on LDPC codes with only one nontrivial critical point\(^1\).

**Algorithm 5.1 Peeling Algorithm [11]**

**Initialization:**
Forward the value of all known variable nodes along the edges connected to them. Accumulate the received messages at the check nodes and “record” the parity. Erase the known variable nodes along with the edges connected to them.

**Decoding:**

while there is a degree-one check node in the residual graph do

Select a degree-one check node and forward its parity to the variable node connected to it. Erase the degree-one check node and the edge connected to it. Repeat the procedure in the **Initialization** process at the known variable nodes.

end while

**Termination:**

if the residual graph is empty then

the decoding is successful and the codeword is determined.

else

the decoding has failed.

end if

\(^1\)Note that $r_1(y)$ and its derivative always vanish at $y = 0$. 
5.3 Computation of the Bit Erasure Rate: A Scaling Approach

5.3.1 Preliminaries

The evolution of the degree sequence, i.e., the number of nodes with various degrees, in the Peeling Algorithm can be considered as a first-order Markov process [10]. However, deriving the erasure probabilities directly from the Markov model is computationally intractable. See Appendix 5.A.

Amraoui et. al., in their papers [8], [9], and [12], present an abstract setting which allows them to approximate the scaled process of the number of degree-one check nodes in the Peeling Algorithm by a Brownian motion with a parabolic drift. We also exploit the same setting.

Let $X_1^{(0)}$ be the number of degree-one check nodes in the residual graph at time $t$. Define the first passage time of the process $\{X_1^{(0)}\}$ as follows:

$$\tau \triangleq \sup \{ t \mid X_1^{(0)}(\gamma) \geq 0 \ \forall \gamma \leq t \} ,$$

where $X_1^{(0)}(\gamma)$ represents the number of edges attached to degree-one check nodes at time $\gamma$. Consequently, the erasure probability at degree-$d$ variable nodes $P_E^{(d)}$ is given by

$$P_E^{(d)} = \frac{E\{L_d(\tau)\}}{dn\Lambda_d},$$

where $L_d(\tau)$ represents the number of edges connected to degree-$d$ variable nodes at the first passage time. Note that the original graph contains $n\Lambda_d$ degree-$d$ variable nodes and $dn\Lambda_d$ edges connected to degree-$d$ variable nodes. By Bayes’ rule, (5.4) can be rewritten as

$$P_E^{(d)} = \Pr\{\text{decoding failure}\} \cdot \frac{E\{L_d(\tau)\}_{\text{decoding failure}}}{dn\Lambda_d} + \Pr\{\text{successful decoding}\} \cdot \frac{E\{L_d(\tau)\}_{\text{successful decoding}}}{dn\Lambda_d}.$$ (5.5)

When the decoding is successful, the residual graph is empty and consequently $L_d(\tau) = 0$. Thereby,

$$E\{L_d(\tau)\}_{\text{successful decoding}} = 0,$$ (5.6)

and (5.5) can be reformulated as follows:

$$P_E^{(d)} = \Pr\{\text{decoding failure}\} \cdot \frac{E\{L_d(\tau)\}_{\text{decoding failure}}}{dn\Lambda_d}.$$ (5.7)
The problem of the computation of $\Pr\{\text{decoding failure}\}$ has been addressed in [13]. Hence, to derive $P_E^{(d)}$, we focus on computation of $E\{L_d(\tau)|\text{decoding failure}\}$.

Let $t^*$ be the asymptotic critical time of the process $\{X_t^{(0)}\}$, i.e., the time when both $r_1$ and its partial derivative with respect to $y$ vanish. The following lemma gives an upper bound on the tail probability of $|\tau - t^*|$.

**Lemma 5.1** There exist positive constants $\Omega_1, \Omega_2, \delta_0$, and a function $n_0(\delta)$, such that, for any $\delta > \delta_0$, and $n > n_0(\delta)$,

$$\Pr\{|\tau - t^*| \geq \delta n^{6/7}\} \leq \Omega_1 e^{-\Omega_2 \delta^2}. \tag{5.8}$$

The proof of this Lemma is deferred to the Appendix 5.B.

Considering the fact that $|\tau - t^*|$ is small on the scale of $n$, while $|\tau - t^*|$ is large on the scale $O(1)$ of a single step, we compute the probability density function of the first passage time using a ‘continuum’ approach.

### 5.3.2 Scaling Approximations

Define the rescaled trajectory of the number of degree-one check nodes, i.e., $\omega(\cdot) \in \mathbb{R}$, as follows (similar to [13]):

$$\omega(\Gamma_1(t-t^*)) \triangleq \Gamma_2 X_t^{(0)}, \tag{5.9}$$

where,

$$\Gamma_1 = f_s^{(00)} \beta^{-2} n^{-2/3} \Lambda'(1)^{-2} \left( \frac{\partial r_1}{\partial \epsilon} |_{t^*} \right)^{-2},$$

$$\Gamma_2 = \beta^{-1} n^{-1/3} \Lambda'(1)^{-1} \left| \frac{\partial r_1}{\partial \epsilon} |_{t^*} \right|^{-1},$$

and,

$$f_s^{(00)} = \left( \frac{y^s \lambda'(y^s)}{\lambda(y^s)} - \frac{y^{s^2} \lambda'(y^s)^2}{\lambda(y^s)^2} \right) \frac{r_2^2}{\sigma^2} + \frac{y^s \lambda'(y^s)}{\lambda(y^s)} \frac{r_2^2}{\sigma^2},$$

$$\beta = \left[ e^{-4r_2^2(e^s \lambda'(y^s)^2 r_2^2 - x^s(\lambda''(y^s))^2 + \lambda'(y^s) x^s)^2} \right]^{1/3}. \tag{5.10}$$

Note that $e^* = e^s y^s \lambda(y^s)$, $y^s$ denotes the fractional rate of the erasures in check-to-variable messages at the critical point, and $\left( \frac{\partial r_1}{\partial \epsilon} |_{t^*} \right)$ is the partial derivative of $r_1$ with respect to the erasure rate at the critical point. Also,

$$x^* = e^s \lambda(y^s),$$

$$\bar{x}^* = 1 - x^*. \tag{5.11}$$
and,
\[
    r^*_i = \begin{cases} 
    e^* \lambda(y^*) [y^* - 1 + \rho(1 - e^* \lambda(y^*))] & \text{if } i = 1, \\
    \sum_{n-k \geq j \geq i} (-1)^{i+j} \binom{j-1}{i-1} \binom{n-k-1}{j-1} \rho_{n-k}(e^* \lambda(y^*))^j & \text{otherwise.}
    \end{cases}
\]

(5.12)

It can be shown that \( \omega(\cdot) \) can be described as a two-sided Brownian motion with a parabolic drift [13]:
\[
    \omega(\bar{\theta}) = \omega(0) + B(\bar{\theta}) + \frac{\bar{\theta}^2}{2},
\]
where \( B(\bar{\theta}) \) is a two-sided Brownian motion, with \( B(0) = 0 \). Let us denote the first passage time of the rescaled trajectory by \( \bar{\theta}_\tau \)
\[
    \bar{\theta}_\tau \triangleq \sup_{\bar{\theta}} \{ \bar{\theta} | \omega(\gamma) \geq 0 \ \forall \gamma \leq \bar{\theta} \}. \tag{5.14}
\]

Clearly,
\[
    \bar{\theta}_\tau = \Gamma_1 (\tau - t^*). \tag{5.15}
\]

Define the scaled trajectory of the number of edges connected to degree-\( d \) variable nodes, i.e., \( u(\cdot) \in \mathbb{R} \), as follows (similar to [13]):
\[
    u(n^{-2/3}(t - t^*)) = n^{-2/3} (L_d(t) - L_d(t^*)), \tag{5.16}
\]

It can be shown that [13]
\[
    u(\bar{\theta}) = - \frac{dl^*_d}{e^*} \bar{\theta}, \tag{5.17}
\]
where \( l^*_d = e^* \lambda_d y^* \). Combining (5.16) and (5.17), we have
\[
    L_d(t) = L_d(t^*) - \frac{dl^*_d}{e^*} (t - t^*), \tag{5.18}
\]
as a result,
\[
    L_d(\tau) = L_d(t^*) - \frac{dl^*_d}{e^*} (\tau - t^*) = L_d(t^*) - \frac{dl^*_d}{e^* \Gamma_1} \bar{\theta}_\tau. \tag{5.19}
\]

Since the fraction of erasures in check-to-variable messages at the critical time is \( y^* \) and the erasure rate of the channel at the threshold is \( e^* \), we deduce that, with probability \( e^* y^* d \) a degree-\( d \) variable node still remains in the LDPC graph at the critical time. As a
result, the average number of edges connected to degree-$d$ variable nodes at the critical time is given by $dnL_d \infty y^d$. Consequently,

$$E\{L_d(\tau)|\text{decoding failure}\} = dnL_d e^* y^d - \frac{d}{\epsilon L_0} d \epsilon. \ E\{\hat{\theta}_\tau|\text{decoding failure}\}. \ (5.20)$$

We compute the conditional expected value of $\hat{\theta}_\tau$, i.e., $E\{\hat{\theta}_\tau|\text{decoding failure}\}$, using its conditional cumulative probability density function. Let us define

$$F_{\hat{\theta}_\tau}(\gamma|\text{decoding failure}) \triangleq \Pr\{\hat{\theta}_\tau \leq \gamma|\text{decoding failure}\}. \ (5.21)$$

It is not hard to see that

$$E\{\hat{\theta}_\tau|\text{decoding failure}\} = \int_0^\infty 1 - F_{\hat{\theta}_\tau}(\gamma|\text{decoding failure}) d\gamma - \int_{-\infty}^0 F_{\hat{\theta}_\tau}(\gamma|\text{decoding failure}) d\gamma. \ (5.22)$$

From the definition of the conditional cumulative density function, we have

$$F_{\hat{\theta}_\tau}(\gamma|\text{decoding failure}) = \frac{\Pr\{\hat{\theta}_\tau \leq \gamma, \text{decoding failure}\}}{\Pr\{\text{decoding failure}\}}. \ (5.23)$$

Define

$$g(\gamma) \triangleq \Pr\{\omega(\hat{\theta}) > 0 \ \forall \hat{\theta} \leq \gamma\}. \ (5.24)$$

As a result,

$$F_{\hat{\theta}_\tau}(\gamma|\text{decoding failure}) = \frac{1 - g(\gamma)}{\Pr\{\text{decoding failure}\}}. \ (5.25)$$

The problem of the computation of $\Pr\{\text{decoding failure}\}$ has been studied in [13]. Consequently, we focus on the computation of $g(\gamma)$. We proceed by dividing this problem into two cases. The first case corresponds to scenarios where $\gamma \leq 0$, and the second case corresponds to scenarios where $\gamma > 0$.

**Case 1 ($\gamma \leq 0$).** By the definition of $g(\gamma)$,

$$g(\gamma) = \Pr\{\omega(\hat{\theta}) > 0 \ \forall \hat{\theta} \leq \gamma\}
= \int_0^\infty \Pr\{\omega(\hat{\theta}) > 0 \ \forall \hat{\theta} < \gamma|\omega(\gamma) = \xi\} P\{\omega(\gamma) = \xi\} d\xi, \ (5.26)$$

where $\omega(\gamma)$ has a Gaussian distribution [7]. We denote the mean and the variance of $\omega(\gamma)$ as follows:

$$\mu(\gamma) \triangleq E\{\omega(\gamma)\}
\quad \sigma^2(\gamma) \triangleq E\{\omega^2(\gamma)\} - E^2\{\omega(\gamma)\}. \ (5.27)$$
Consequently,
\[ P\{\omega(\gamma) = \xi\} = \frac{1}{\sqrt{2\pi\sigma(\gamma)}} e^{-\frac{(\xi - \mu(\gamma))^2}{2\sigma^2(\gamma)}}. \quad (5.28) \]

One can show that
\begin{align*}
\mu(\gamma) &= \mu(0) + \frac{\gamma^2}{2} \\
\sigma^2(\gamma) &= \sigma^2(0) + |\gamma|.
\end{align*}
\quad (5.29)

Define a process \( \bar{\omega}(\cdot) \) as follows:
\[ \bar{\omega}(\bar{\theta}) \triangleq -\omega(\bar{\theta}) = -\omega(0) - \frac{\bar{\theta}^2}{2} - B(\bar{\theta}). \quad (5.30) \]

It is not hard to see that \(-B(\bar{\theta})\) is also a two-sided Brownian motion process, so we define \( \bar{B}(\bar{\theta}) \triangleq -B(\bar{\theta}) \), where \( \bar{B}(\bar{\theta}) \) is a two-sided Brownian motion process starting at \( \bar{B}(0) = 0 \). Hence,
\[ \Pr\{\omega(\bar{\theta}) > 0 \ \forall \bar{\theta} < \gamma|\omega(\gamma) = \xi\} = \Pr\{\bar{\omega}(\bar{\theta}) < 0 \ \forall \bar{\theta} < -\gamma|\bar{\omega}(-\gamma) = -\xi\}. \quad (5.31) \]

The right-hand side of (5.31) corresponds to the probability that the maximum of a two-sided Brownian motion with a parabolic drift is less than zero. This problem has been studied in [14]. Adapting the results from [14] to our situation, we have
\[ \Pr\{\omega(\bar{\theta}) > 0 \ \forall \bar{\theta} < \gamma|\omega(\gamma) = \xi\} = 2^{-\frac{3}{2}} e^{-\frac{\gamma^2 + 6\gamma \xi}{6}} \int_{-\infty}^{\infty} e^{i\nu \gamma} \frac{\text{Ai}(i(2^{1/3} \nu^2 + 21/3) - \text{Bi}(2^{1/3} \nu) \text{Ai}(i(2^{1/3} \nu + 21/3) \nu d\nu,} \quad (5.32) \]

where \( i = \sqrt{-1} \), and \( \text{Ai}(\cdot) \), and \( \text{Bi}(\cdot) \), are the Airy functions defined on page 446, of [15]. Putting everything together, for \( \gamma \leq 0 \), we have
\[ g(\gamma) = \frac{2^{-\frac{3}{2}}}{\sqrt{2\pi\sigma(\gamma)}} \int_{0}^{\infty} e^{-\frac{(\xi - \mu(\gamma))^2}{2\sigma^2(\gamma)}} e^{-\frac{\gamma^2 + 6\gamma \xi}{6}} \int_{-\infty}^{\infty} e^{i\nu \gamma} \frac{\text{Ai}(i(2^{1/3} \nu^2 + 21/3) - \text{Bi}(2^{1/3} \nu) \text{Ai}(i(2^{1/3} \nu + 21/3) \nu d\nu d\xi.} \quad (5.33) \]

Now let us focus on the second case, i.e., when \( \gamma \geq 0 \).

**Case 2** (\( \gamma > 0 \)). Since \( \bar{B}(0) = 0 \), computation of \( g(\gamma) \) when \( \gamma > 0 \) requires a slightly different approach. It’s not hard to see that
\[ g(\gamma) = \int_{0}^{\infty} P\{\omega(0) = \xi\} \Pr\{\omega(\bar{\theta}) > 0 \ \forall \bar{\theta} < 0|\omega(0) = \xi\} \Pr\{\omega(\bar{\theta}) > 0 \ \forall 0 < \bar{\theta} < \gamma|\omega(0) = \xi\} d\xi. \quad (5.34) \]
Pursuing steps similar to those used in the calculation of \( g(\gamma) \) in the first case, one can show that

\[
\Pr\{\omega(\tilde{\theta}) > 0 \; \forall \tilde{\theta} < 0 | \omega(0) = \xi \} = 2^{-\frac{3}{2}} \int_{-\infty}^{\infty} \frac{\text{Ai}(2 \sqrt[3]{3} \nu \omega B(i 2^{1/3} \nu + 2^{1/3} \xi) - \text{Ai}(2^{1/3} \nu))}{\text{Ai}(2^{1/3} \nu)} d\nu, \tag{5.35}
\]

moreover,

\[
P\{\omega(0) = \xi\} = \frac{1}{\sqrt{2\pi \sigma(0)}} e^{-\frac{(\xi - \mu(0))^2}{2\sigma(0)^2}}. \tag{5.36}
\]

It is not hard to show that

\[
\Pr\{\omega(\tilde{\theta}) > 0 \; \forall 0 < \tilde{\theta} < \gamma | \omega(0) = \xi \} = \Pr\{\omega(\tilde{\theta}) < 0 \; \forall 0 < \tilde{\theta} < \gamma | \omega(0) = -\xi \}. \tag{5.37}
\]

Again using results from [14], we have

\[
\Pr\{\omega(\tilde{\theta}) > 0 \; \forall 0 < \tilde{\theta} < \gamma | \omega(0) = \xi \} = 1 - \int_0^{\gamma} e^{-\frac{3}{2} \nu h_{1, \xi}(\theta)} d\theta, \tag{5.38}
\]

where the function \( h_{1, \xi}(\theta) \) has the Laplace transform

\[
H_{1, \xi}(\omega) = \int_0^\infty e^{-\omega \theta} h_{1, \xi}(\theta) d\theta = \text{Ai}\left(2^{1/3}(\xi + \omega)\right) / \text{Ai}\left(2^{1/3} \omega\right). \tag{5.39}
\]

Putting everything together, we deduce that, when \( \gamma > 0 \),

\[
g(\gamma) = \frac{2^{-\frac{3}{2}}}{\sqrt{2\pi \sigma(0)}} \int_{-\infty}^{\infty} \frac{(-\sqrt{3} \nu \omega(0))^2}{2\sigma(0)^2} \left(1 - \int_0^{\gamma} e^{-\frac{3}{2} \nu h_{1, \xi}(\theta)} d\theta\right) d\nu \int_{-\infty}^{\infty} \frac{\text{Ai}(2 \sqrt[3]{3} \nu \omega B(i 2^{1/3} \nu + 2^{1/3} \xi) - \text{Ai}(2^{1/3} \nu))}{\text{Ai}(2^{1/3} \nu)} d\nu d\xi. \tag{5.40}
\]

### 5.3.3 Bounds on the Bit Erasure Rate

Due to the complexity of the numerical evaluation of \( g(\gamma) \), we now derive upper and lower bounds on \( g(\gamma) \). These bounds can be employed in the computation of upper and lower bounds on the scaling approximation to the bit erasure rate. Let us first present Theorem 5.2 which establishes an upper bound on \( g(\gamma) g(-\gamma) \):

**Theorem 5.2** For any \( \gamma \in \mathbb{R} \),

\[
g(\gamma) g(-\gamma) \leq \Pr\{\text{successful decoding}\} \left(1 - Q\left(\frac{\mu(\gamma)}{\sigma(\gamma)}\right)\right), \tag{5.41}
\]
where \(\mu(\gamma)\) and \(\sigma(\gamma)\) denote the mean and variance of the rescaled trajectory \(\omega(\gamma)\) and are given by (5.27). Furthermore, \(Q(\cdot)\) is defined as follows:

\[
Q(\varphi) = \frac{1}{\sqrt{2\pi}} \int_{\varphi}^{\infty} e^{-\frac{x^2}{2}} dx.
\]

(5.42)

The proof is presented in Appendix 5.C.

**Corollary 5.3** From Theorem 5.2, one can deduce that

\[
g(0) \leq \sqrt{\Pr\{\text{successful decoding}\} \left( 1 - Q\left( \frac{\mu(0)}{\sigma(0)} \right) \right)}.
\]

(5.43)

**Remark** A weaker upper bound on \(g(\gamma)g(-\gamma)\), i.e.,

\[
g(\gamma)g(-\gamma) \leq \Pr\{\text{successful decoding}\},
\]

(5.44)

can be obtained using the fact that positively correlated normal random variables are associated\(^2\) [16].

Now consider the following Lemmas and Theorem, presenting lower and upper bounds on \(g(\gamma)\):

**Lemma 5.4** For any \(\gamma \in \mathbb{R}\),

\[
g(\gamma) \geq \Pr\{\text{successful decoding}\}.
\]

(5.45)

**Proof:** From definition of \(g(\gamma)\), we have

\[
g(\gamma) = \Pr\{\omega(\bar{\theta}) > 0 \ \forall \bar{\theta} \leq \gamma\}
\]

\[
\geq \Pr\{\omega(\bar{\theta}) > 0 \ \forall \bar{\theta}\}
\]

(5.46)

\[
= \Pr\{\text{successful decoding}\}.
\]

\(\blacksquare\)

---

\(^2\)The random variables \(\Psi_1, \ldots, \Psi_u\) are called associated, if the inequality

\[
\text{Cov}[f(\Psi_1, \ldots, \Psi_u), g(\Psi_1, \ldots, \Psi_u)] \geq 0,
\]

holds for any pair of bounded Borel-measurable increasing functions \(f\) and \(g\), where \(\text{Cov}[\cdot, \cdot]\) represents the covariance. Note that a function \(f(\Psi_1, \ldots, \Psi_u)\) is called an increasing function if it is non-decreasing in each of the separate variables \(\Psi_1, \ldots, \Psi_u\) [16].
Theorem 5.5 For any $\gamma \leq 0$,

$$1 - Q\left(\frac{\hat{\mu}_\gamma(0)}{\hat{\sigma}_\gamma(0)}\right) - \frac{\hat{\sigma}_\gamma^2(0)}{2\pi\hat{\sigma}_\gamma(0)} \leq g(\gamma) \leq \min_{M > 0} \sqrt{\left(1 - Q\left(\frac{\hat{\mu}_\gamma(0,M)}{\hat{\sigma}_\gamma(0)}\right) - \frac{(1+M)^{-1/3}}{\sqrt{2\pi\hat{\sigma}_\gamma(0)}} e^{-\frac{\hat{\mu}_\gamma^2(0,M)}{2\hat{\sigma}_\gamma^2(0)}}\right)} \left(1 - Q\left(\frac{\hat{\mu}_\gamma(0,M)}{\hat{\sigma}_\gamma(0)}\right)\right),$$

(5.47)

where

\begin{align*}
\hat{\mu}_\gamma(0) &= \Gamma_2(e^* - e) \frac{\partial R_1}{\partial e} |_* + \frac{1}{2} \gamma^2 \\
\hat{\sigma}_\gamma^2(0) &= \frac{\alpha^2 r_2}{n} \left( \frac{\partial R_1}{\partial e} |_* \right)^2 + |\gamma| \\
\hat{\mu}_\gamma(0,M) &= \Gamma_2(e^* - e) \frac{\partial R_1}{\partial e} |_* + \frac{\gamma^2}{2} (1 + \frac{1}{M}) \\
\hat{\sigma}_\gamma^2(0) &= \frac{\alpha^2 r_2}{n} \left( \frac{\partial R_1}{\partial e} |_* \right)^2 + |\gamma|,
\end{align*}

(5.48)

and $\frac{\partial R_1}{\partial e} |_*$ is the partial derivative of $R_1$, i.e., the number of edges connected to degree-one check nodes, with respect to the erasure rate at the critical point.

We relegate the proof of this theorem to Appendix 5.D.

Lemma 5.6 For any $\gamma \geq 0$,

$$\Pr\{\text{successful decoding}\} \leq g(\gamma) \leq \frac{\Pr\{\text{successful decoding}\} \left(1 - Q\left(\frac{\mu(\gamma)}{\sigma(\gamma)}\right)\right)}{1 - Q\left(\frac{\hat{\mu}_\gamma(0)}{\hat{\sigma}_\gamma(0)}\right) - \frac{\hat{\sigma}_\gamma^2(0)}{2\pi\hat{\sigma}_\gamma(0)}}.$$

(5.49)

Proof: By Lemma 5.4,

$$g(\gamma) \geq \Pr\{\text{successful decoding}\}. \tag{5.50}$$

And by Theorem 5.2,

$$g(\gamma)g(-\gamma) \leq \Pr\{\text{successful decoding}\} \left(1 - Q\left(\frac{\mu(\gamma)}{\sigma(\gamma)}\right)\right). \tag{5.51}$$

As a result,

$$g(\gamma) \leq \frac{\Pr\{\text{successful decoding}\} \left(1 - Q\left(\frac{\mu(\gamma)}{\sigma(\gamma)}\right)\right)}{g(-\gamma)}. \tag{5.52}$$

By Theorem 5.5,

$$g(-\gamma) \geq 1 - Q\left(\frac{\hat{\mu}_\gamma(0)}{\hat{\sigma}_\gamma(0)}\right) - \frac{\hat{\sigma}_\gamma^2(0)}{2\pi\hat{\sigma}_\gamma(0)}, \tag{5.53}$$
Note that $\hat{\mu}_{-\gamma}(0) = \hat{\mu}_\gamma(0)$ and $\hat{\sigma}_{-\gamma}(0) = \hat{\sigma}_\gamma(0)$. Consequently,

\[
\frac{1}{g(-\gamma)} \leq \frac{1}{1 - Q\left(\frac{\hat{\mu}_\gamma(0)}{\hat{\sigma}_\gamma(0)}\right)} - \frac{1}{\sqrt{2\pi}\hat{\sigma}_\gamma(0)} e^{-\frac{\hat{\mu}_\gamma^2(0)}{2\hat{\sigma}_\gamma^2(0)}}.
\]  

(5.54)

Combining (5.52), and (5.54), we get

\[
g(\gamma) \leq \frac{\Pr\{\text{successful decoding}\} \left(1 - Q\left(\frac{\mu(\gamma)}{\sigma(\gamma)}\right)\right) - \frac{1}{\sqrt{2\pi}\sigma(\gamma)} e^{-\frac{\mu^2(\gamma)}{2\sigma^2(\gamma)}}}{1 - Q\left(\frac{\hat{\mu}(\gamma)}{\hat{\sigma}(\gamma)}\right)}. 
\]

(5.55)

\[\blacksquare\]

### 5.4 Numerical Analysis

To empirically investigate the accuracy of the scaling approximations, we performed Monte Carlo simulations of random LDPC code ensembles of various lengths and degree distributions over BECs.

In Figures 5.1, 5.2, and 5.3, we compare the simulated bit erasure rates of degree-3, degree-4, and degree-9 variable nodes with the results obtained from our scaling approximation, and from both the upper and lower bounds for LDPC code ensembles with $\lambda(x) = \frac{2}{5}x^2 + \frac{12}{25}x^3 + \frac{3}{25}x^8$ and $\rho(x) = \frac{7}{15}x^6 + \frac{8}{15}x^7$. In each graph, we present results for codes of length $n = 1000, 2000, 4000$, and $8000$.

In Figures 5.4, and 5.5, we present the simulated bit erasure rates of degree-3 and degree-8 variable nodes as well as the results obtained from our scaling approximation, and both upper and lower bounds for LDPC code ensembles with $\lambda(x) = \frac{7}{15}x^2 + \frac{8}{15}x^7$ and $\rho(x) = x^8$. Again, we consider codes of lengths $n = 1000, 2000, 4000$, and $8000$.

Figure 5.6 presents a comparison between bit erasure rates of degree-3 and degree-13 variable nodes in LDPC code ensembles of length $n = 8000$ with $\lambda(x) = \frac{57}{70}x^2 + \frac{13}{70}x^{12}$ and $\rho(x) = x^6$.

Note that, in all cases, the simulation results are quite close to the scaling approximation. Furthermore, for longer codes, the upper and lower bounds become very close to the simulation results.
5.5 Concluding Remarks

In this chapter, we investigated the UEP properties of finite-length irregular LDPC codes in the waterfall region of the peeling decoder. We introduced a scaling approach to compute the bit erasure rate of variable nodes with a given degree over BECs. Simulation results showed that, for a wide range of code lengths, scaling approximations provide a very close estimate to the bit erasure rate. We further derived upper and lower bounds on the scaling approximation to the bit erasure rate. We showed that these bounds are quite tight, and for larger codes, they become very close to the Monte Carlo simulation results.

5.6 Acknowledgement

This chapter, in part, is a reprint of the material of the following papers: A. H. Djahanshahi, P. H. Siegel, and L. B. Milstein, “On unequal error protection of finite-length LDPC codes over BECs: a scaling approach,” in Proceedings of the IEEE Interna-
Figure 5.2: Bit erasure rate curves of degree-4 variable nodes of LDPC($n, \lambda(x) = \frac{2}{5}x^2 + \frac{12}{25}x^3 + \frac{3}{25}x^8, \rho(x) = \frac{7}{15}x^6 + \frac{8}{15}x^7$) codes when used over binary erasure channel of erasure probability $\epsilon$.


Appendix 5.A  A Markov Model for Peeling Algorithm

Consider decoding an LDPC code from the LDPC($n, \lambda(x), \rho(x)$) ensemble with the Peeling Algorithm. At the time instant $t$, the state of the decoder can be presented using the vector

$$X_t = (R_1, R_2, \ldots, R_{r_{\text{max}}-1}, L_1, L_2, \ldots, L_{l_{\text{max}}})$$

(5.56)

where $L_i$, for $i = 1, 2, \ldots, l_{\text{max}}$, represents the number of edges connected to degree-$i$ variable nodes, and $R_i$, for $i = 1, 2, \ldots, r_{\text{max}}-1$, denotes the number of edges connected to degree-$i$ check nodes in the residual graph [13]. Clearly,

$$\sum_{i=1}^{l_{\text{max}}} L_i = \sum_{i=1}^{r_{\text{max}}} R_i$$

(5.57)
Figure 5.3: Bit erasure rate curves of degree-9 variable nodes of LDPC($n, \lambda(x) = \frac{2}{3}x^2 + \frac{12}{25}x^3 + \frac{1}{5}x^8, \rho(x) = \frac{7}{15}x^6 + \frac{8}{15}x^7$) codes when used over binary erasure channel of erasure probability $\epsilon$.

Figure 5.4: Bit erasure rate curves of degree-3 variable nodes of LDPC($n, \lambda(x) = \frac{7}{15}x^2 + \frac{8}{15}x^7, \rho(x) = x^8$) codes when used over binary erasure channel of erasure probability $\epsilon$. 
Figure 5.5: Bit erasure rate curves of degree-8 variable nodes of LDPC($n, \lambda(x) = \frac{7}{15}x^2 + \frac{8}{15}x^7, \rho(x) = x^6$) codes when used over binary erasure channel of erasure probability $\epsilon$.

Figure 5.6: Bit erasure rate curves of degree-3 and degree-13 variable nodes of LDPC($n = 8000, \lambda(x) = \frac{57}{70}x^2 + \frac{13}{70}x^{12}, \rho(x) = x^6$) codes when used over binary erasure channel of erasure probability $\epsilon$. 
Note that both summations in (5.57) denote the total number of edges in the residual graph. Consequently,

\[ R_{r_{\text{max}}} = \sum_{i=1}^{r_{\text{max}}} L_i - \sum_{i=1}^{r_{\text{max}}-1} R_i, \]  

(5.58)

As a result, \( R_{r_{\text{max}}} \) can be determined in terms of the state vector.

The Peeling Algorithm can be considered as a first-order Markov process with two sets of absorbing states, in particular, the correct decoding state

\[ S_{\text{correct decoding}} = \{ (0,0,\ldots,0) \}, \]

and decoding failure states

\[ S_{\text{decoding failure}} = \{ (0,R_2,R_3,\ldots,L_{1_{\text{max}}}) : (R_2,R_3,\ldots,L_{1_{\text{max}}}) \in \mathbb{N}^{r_{\text{max}}+r_{\text{max}}+2} \setminus \{ (0,0,\ldots,0) \} \}, \]

where \( \mathbb{N} \) denotes the set of non-negative integers

\[ \mathbb{N} = \{ 0,1,2,\ldots \}, \]  

(5.59)

and \( \mathbb{N}^{r_{\text{max}}+r_{\text{max}}+2} \setminus \{ (0,0,\ldots,0) \} \) represents the set of all \( r_{\text{max}}+r_{\text{max}}+2 \)-tuple non-negative integers excluding the origin. Consequently, the problem of computation of the bit erasure rate at variable nodes with a given degree can be considered as the problem of the computation of the absorption probability in a Markov process. To compute this probability, let us first derive the initial and transition probabilities of the Peeling Algorithm.

**Appendix 5.A.1 Initial Probabilities**

Let \( L_i(0) \) (\( V_i(0) \), respectively) for \( i = 1,2,\ldots,1_{\text{max}} \), denote the number of edges connected to degree-\( i \) variable nodes (number of degree-\( i \) variable nodes, respectively) after the initialization of the Peeling Algorithm. Moreover, let \( R_i(0) \) (\( C_i(0) \), respectively) for \( i = 1,2,\ldots,r_{\text{max}} \), denote the number of edges connected to degree-\( i \) check nodes (number of degree-\( i \) check nodes, respectively) after the initialization of the Peeling Algorithm. It is not hard to see that

\[ \Pr\{ L_i(0) = iv_i \} = \Pr\{ V_i(0) = v_i \} = \binom{n_{A_i}}{v_i} e^{v_i} (1-e)^{n_{A_i} - v_i}. \]  

(5.60)
Note that here we use the assumption that each bit experiences an independent BEC. From (5.60), one can deduce that
\[
\Pr \{ L_1(0) = v_1, \ldots, L_{1_{\text{max}}}(0) = 1_{\text{max}}v_{1_{\text{max}}} \} = \prod_{i=1}^{1_{\text{max}}} \left( \frac{nA_i}{v_i} \right) e^{v_i} (1 - \epsilon)^{nA_i - v_i}. \tag{5.61}
\]
After initialization of the Peeling Algorithm, a degree-\(i\) check node can become a degree-\(j\) check node, for \(j = 0, 1, \ldots, i\). Let us define \(u_{i,j}\) as the number of degree-\(i\) check nodes that have become a degree-\(j\) check node after the initialization of the Peeling Algorithm. Consequently, to compute the probability
\[
\Pr (C_1(0) = c_1, C_2(0) = c_2, \ldots, C_{r_{\text{max}}}(0) = c_{r_{\text{max}}}),
\]
or, equivalently, the probability
\[
\Pr (R_1(0) = c_1, R_2(0) = 2c_2, \ldots, R_{r_{\text{max}}}(0) = r_{\text{max}}c_{r_{\text{max}}}),
\]
we should consider all combinations of \(u_{i,j} \in \mathbb{N}\), such that
\[
\sum_{j=0}^{i} u_{i,j} = (n - k)P_i \quad \forall i : 1 \leq i \leq r_{\text{max}}
\]
\[
\sum_{s=1}^{r_{\text{max}}} u_{s,l} = c_l \quad \forall l : 1 \leq l \leq r_{\text{max}}. \tag{5.62}
\]
Remember that \((n - k)P_i\) denotes the number of degree-\(i\) check nodes in the original graph.

Let us define \(\tilde{U}(c_1, \ldots, c_{r_{\text{max}}})\) as the set of parameters \(u_{1,0}, u_{1,1}, \ldots, u_{r_{\text{max}}, r_{\text{max}}}\)
that satisfy the conditions imposed by (5.62). As a result
\[
\Pr \{ R_1(0) = c_1, \ldots, R_{r_{\text{max}} - 1}(0) = (r_{\text{max}} - 1)c_{r_{\text{max}} - 1}, L_1(0) = v_1, \ldots, L_{1_{\text{max}}}(0) = 1_{\text{max}}v_{1_{\text{max}}} \} =
\sum_{(u_{1,0}, \ldots, u_{r_{\text{max}}, r_{\text{max}}}) \in \tilde{U}(c_1, \ldots, c_{r_{\text{max}}})} \prod_{i=1}^{r_{\text{max}}} \left( \frac{(n - k)P_i}{u_{i,0}} \right) \prod_{s=0}^{i} \left( \frac{i}{s} \right) e^{s} (1 - \epsilon)^{i-s} u_{i,s},
\]
where
\[
c_{r_{\text{max}}}) = \frac{1}{r_{\text{max}}} \left( \sum_{j=1}^{r_{\text{max}} - 1} jv_j - \sum_{j=1}^{r_{\text{max}}} jc_j \right), \tag{5.63}
\]
and
\[
\tilde{e} = \frac{\sum_{j=1}^{r_{\text{max}}} jv_j}{\sum_{j=1}^{r_{\text{max}}} jnA_j}. \tag{5.64}
\]
Putting everything together, the initial probabilities can be computed as follows:
\[
\Pr \{ R_1(0) = c_1, \ldots, R_{r_{\text{max}} - 1}(0) = (r_{\text{max}} - 1)c_{r_{\text{max}} - 1}, L_1(0) = v_1, \ldots, L_{1_{\text{max}}}(0) = 1_{\text{max}}v_{1_{\text{max}}} \} =
\prod_{i=1}^{r_{\text{max}}} \left( \frac{nA_i}{v_i} \right) e^{v_i} (1 - \epsilon)^{nA_i - v_i} \left( \sum_{(u_{1,0}, \ldots, u_{r_{\text{max}}, r_{\text{max}}}) \in \tilde{U}(c_1, \ldots, c_{r_{\text{max}}})} \prod_{i=1}^{r_{\text{max}}} \left( \frac{(n - k)P_i}{u_{i,0}} \right) \prod_{s=0}^{i} \left( \frac{i}{s} \right) e^{s} (1 - \epsilon)^{i-s} u_{i,s} \right).
\]
Appendix 5.A.2 Transition Probabilities

Now, let us consider a decoding step of the Peeling Algorithm. First, a degree-one check node is selected. Then, the variable node connected to the check node and all the edges connected to the variable node are peeled from the graph. Assume that the variable node is of degree-$i$. The probability that the other $i-1$ edges of the variable node are connected to $u_1 - 1$ degree-1, $u_2$ degree-2, $u_3$ degree-3, ..., and $u_{\tau_{\text{max}}}$ degree-$\tau_{\text{max}}$ check nodes is given by

$$
\left( \begin{array}{c}
i-1 \\
u_1 - 1, u_2, \ldots, u_{\tau_{\text{max}}} \end{array} \right) q_1^{u_1 - 1} q_2^{u_2} \cdots q_{\tau_{\text{max}}}^{u_{\tau_{\text{max}}}},
$$

(5.65)

if

$$
\sum_{j=1}^{\tau_{\text{max}}} u_j = i.
$$

(5.66)

Otherwise the probability is zero. In (5.65), $q_j$, for $j = 1, 2, \ldots, \tau_{\text{max}}$, is defined as

$$
q_j \equiv \frac{R_j}{\sum_{k=1}^{\tau_{\text{max}}} R_k}.
$$

(5.67)

Assume $S = (R_1, R_2, \ldots, R_{\tau_{\text{max}} - 1}, L_1, L_2, \ldots, L_{\tau_{\text{max}}})$ denotes the current state of the decoder. Additionally, let $S' = (R'_1, R'_2, \ldots, R'_{\tau_{\text{max}} - 1}, L'_1, L'_2, \ldots, L'_{\tau_{\text{max}}})$ represent the state of the decoder after peeling the degree-one check node, the degree-$i$ variable node connected to the check-node, and all the edges attached to the degree-$i$ variable node. One can easily show that [10]

$$
L'_j = \begin{cases} 
L_j & \text{if } j \neq i \\
i - i & \text{otherwise.}
\end{cases}
$$

(5.68)

Furthermore,

$$
R'_j = \begin{cases} 
R_{\tau_{\text{max}}} - \tau_{\text{max}} u_{\tau_{\text{max}}} & \text{if } j = \tau_{\text{max}} \\
R_j - j (u_j - u_{j+1}) & \text{otherwise,}
\end{cases}
$$

(5.69)

or, equivalently,

$$
u_j = \sum_{l=j}^{\tau_{\text{max}}} \frac{R_l - R'_l}{l}.
$$

(5.70)

Putting everything together, one can deduce that the transition probability from state $S = (R_1, R_2, \ldots, R_{\tau_{\text{max}} - 1}, L_1, L_2, \ldots, L_{\tau_{\text{max}}})$ at time $t$ to state $S' = (R'_1, R'_2, \ldots, R'_{\tau_{\text{max}} - 1}, L'_1, L'_2, \ldots, L'_{\tau_{\text{max}}})$ at time $t+1$ is given by [13]

$$
\Pr\{X_{t+1} = S' | X_t = S\} = \left( \begin{array}{c}i-1 \\
u_1 - 1, u_2, \ldots, u_{\tau_{\text{max}}} \end{array} \right) p_1 q_1^{u_1 - 1} q_2^{u_2} \cdots q_{\tau_{\text{max}}}^{u_{\tau_{\text{max}}}},
$$

(5.71)
where

\[ p_i \triangleq \frac{L_i}{\sum_{k=1}^{L_{\text{max}}} L_k}, \]  

(5.72)

and

\[ u_j = \sum_{l=j}^{\ell_{\text{max}}} \frac{R_{l} - R'_{l}}{l}, \]  

(5.73)

with

\[ R_{\ell_{\text{max}}} = \sum_{j=1}^{\ell_{\text{max}}} L_j - \sum_{j=1}^{\ell_{\text{max}}-1} R_j \]

\[ R'_{\ell_{\text{max}}} = \sum_{j=1}^{\ell_{\text{max}}} L'_j - \sum_{j=1}^{\ell_{\text{max}}-1} R'_j. \]  

(5.74)

**Appendix 5.A.3 Computation of the Bit Erasure Rate**

Let us define \( P_{E}^{(d)} \) as the erasure probability of degree-\( d \) variable nodes. One can show that

\[ P_{E}^{(d)} = \frac{1}{nA_d} \sum_{t=0}^{\infty} \sum_{v=1}^{nA_d} \sum_{S \in S_{(d,v)}_{\text{decoding failure}}} \text{Pr}\{X_t = S\}, \]  

(5.75)

where \( \text{Pr}\{X_t = S\} \) represents the probability that the decoder is in state \( S \) at time \( t \), and can be computed recursively using the initial and transition probabilities. We also define

\[ S_{(d,v)}_{\text{decoding failure}} \triangleq \{(0, R_2, \ldots, L_{1_{\text{max}}}) : L_d = dv, (R_2, \ldots, L_{1_{\text{max}}}) \in (\mathbb{N})^{\ell_{\text{max}}+1_{\text{max}}-2} \setminus (0, 0, \ldots, 0)\}, \]

as the subset of states in \( S_{\text{decoding failure}} \) that have \( v \) degree-\( d \) variable nodes.

**Appendix 5.B Proof of Lemma 5.1**

*Proof:* In this appendix, we present the proof of Lemma 5.1. It can be shown that for any positive constant \( \zeta \)

\[ \text{Pr}\left\{ |X_t^{(0)} - \bar{X}_t^{(0)}| \geq \zeta n^{7/12} \right\} \leq 2e^{-\frac{\zeta^2 n^7}{24n^{12}}}, \]  

(5.76)

where \( \Omega_0 \) is a positive constant, independent of \( n \) [13].

Considering the fact that the erasure probability \( \epsilon \) is in a critical window of \( \epsilon - \epsilon^* = O(n^{-1/2}) \), it can be shown that \( \bar{X}_t^{(0)} \) is of order \( O(\sqrt{n}) \) [13]. As a result, for

\[ p_i \triangleq \frac{L_i}{\sum_{k=1}^{\ell_{\text{max}}} L_k}, \]  

(5.72)

Note that here we have assumed that \( L_i - L'_i = i \), for some value of \( i \) such that \( 0 \leq i \leq \ell_{\text{max}} \). Otherwise the transition probability would be zero.
values of \( n \) large enough,
\[
\Pr \left\{ \left| X_t^{(0)} \right| \geq n^{2/3} \zeta \right\} \leq 2e^{-\frac{\zeta^2 t}{200n^3}}. \tag{5.77}
\]

Similar to [13], define
\[
Y_{t-t'} \overset{\Delta}{=} \frac{1}{\kappa_4} \left( X_t^{(0)} - X_{t'}^{(0)} \right),
\tag{5.78}
\]
where \( \kappa_4 \) is a positive constant. Let \( t_i \overset{\Delta}{=} 2^l n^{6/7} \delta^{2/3} \), where \( \delta \) is a positive constant. For the sake of simplicity, here we focus on the case \( t > t' \). The case \( t < t' \) can be treated similarly.

\[
\Pr \{ \min_{t \geq n^{7/3} \delta^{2/3}} Y_t \leq n^{2/3} \delta^{2/3} \} \leq \sum_{l=0}^\infty \Pr \{ \min_{t \leq t+1} Y_t \leq \frac{n^{2/3} \delta^{2/3}}{l} \},
\tag{5.79}
\]
or, equivalently,

\[
\Pr \{ \min_{t \geq n^{7/3} \delta^{2/3}} Y_t \leq n^{2/3} \delta^{2/3} \} \leq \sum_{l=0}^\infty \Pr \{ \min_{t \leq t+1} Y_t - \frac{1}{n} t^2 + \frac{\kappa_5 \delta \sqrt{n}}{\sqrt{n}} \leq \frac{n^{2/3} \delta^{2/3}}{l} - \frac{1}{n} t^2 + \frac{\kappa_5 \delta \sqrt{n}}{\sqrt{n}} t+1 \},
\tag{5.80}
\]
where \( \kappa_5 \) is a positive constant. Adapting a result on the concentration properties of the \( X_t^{(0)} \) from [13], we can rewrite (5.80) as
\[
\Pr \{ \min_{t \geq n^{7/3} \delta^{2/3}} Y_t \leq n^{2/3} \delta^{2/3} \} \leq \Omega_1 \sum_{l=0}^\infty \exp \left\{ -\frac{\Omega_2}{l+1} \left( n^{2/3} \delta^{2/3} - \frac{1}{n} t^2 + \frac{\kappa_5 \delta \sqrt{n}}{\sqrt{n}} t+1 \right)^2 \right\}, \tag{5.81}
\]
where \( \Omega_1 \) and \( \Omega_2 \) are positive constants. After some algebraic manipulations, one can show that
\[
\Pr \{ \min_{t \geq n^{7/3} \delta^{2/3}} Y_t \leq n^{2/3} \delta^{2/3} \} \leq \Omega_1 \sum_{l=0}^\infty \exp \left\{ -\frac{\Omega_2}{2l+1} \left( 2^{2l} - \frac{1}{n} t^2 - \kappa_5 2^{l+1} n^{2/3} \delta^{2/3} \right)^2 \right\}. \tag{5.82}
\]

For \( n > \max \left\{ 1, (2\kappa_5 \delta^{1/3})^{14/5} \right\} \),
\[
\Pr \{ \min_{t \geq n^{7/3} \delta^{2/3}} Y_t \leq n^{2/3} \delta^{2/3} \} \leq \Omega_1 \exp \left\{ -\frac{\Omega_2}{2l+1} \left( 1 - \frac{1}{n} t^2 - 2\kappa_5 n \delta^{2/3} \right)^2 \right\} + \Omega_1 \sum_{l=1}^\infty \exp \left\{ -\frac{\Omega_2}{2l+1} \left( 2^{2l} - 1 - 2^l \right)^2 \right\}. \tag{5.83}
\]

Now it is not hard to show that there exist two positive constants, \( \Omega'_1 \), and \( \Omega'_2 \), such that
\[
\Pr \{ \min_{t \geq n^{7/3} \delta^{2/3}} Y_t \leq n^{2/3} \delta^{2/3} \} \leq \Omega'_1 \exp \left\{ -\Omega'_2 \delta^2 \right\}. \tag{5.84}
\]

Note that \( \Omega'_1 \) and \( \Omega'_2 \) are not necessarily the same as \( \Omega_1 \) and \( \Omega_2 \). Combining the results from (5.77), (5.78), and (5.84) concludes the proof. ■
Appendix 5.C  Proof of Theorem 5.2

Proof: To prove this Theorem we employ the Riemann sum approximation of the integrals. We show that (5.41) holds for any choice of the summation step. Consequently, it will also hold for the exact integrals, i.e., the limiting case when the summation step goes to zero.

Let us define

\[ h(\gamma, \eta) \triangleq \Pr\{\omega(\theta) > 0 \ \forall \theta < \gamma | \omega(\gamma) = \eta \}. \] (5.85)

One can see that

\[ g(\gamma) = \frac{1}{\sqrt{2\pi\sigma(\gamma)}} \int_{0}^{\infty} h(\gamma, \eta)e^{-\frac{(\eta-\mu(\gamma))^2}{\sigma(\gamma)^2}} d\eta. \] (5.86)

Furthermore, it is not hard to show that

\[ \Pr\{\omega(\theta) > 0 \ \forall \theta > \gamma | \omega(\gamma) = \eta \} = h(-\gamma, \eta). \] (5.87)

Additionally,

\[ \Pr\{\text{successful decoding} \} = \frac{1}{\sqrt{2\pi\sigma(\gamma)}} \int_{0}^{\infty} h(\gamma, \eta)h(-\gamma, \eta)e^{-\frac{(\eta-\mu(\gamma))^2}{\sigma(\gamma)^2}} d\eta. \] (5.88)

Now, let us define the following sums:

\[ S_1(\Delta) \triangleq \sum_{l=0}^{\infty} S_1^{(l)}(\Delta) \]
\[ S_2(\Delta) \triangleq \sum_{l=0}^{\infty} S_2^{(l)}(\Delta) \]
\[ S_3(\Delta) \triangleq \sum_{l=0}^{\infty} S_3^{(l)}(\Delta) \]
\[ S_4(\Delta) \triangleq \sum_{l=0}^{\infty} S_4^{(l)}(\Delta), \] (5.89)

where

\[ S_1^{(l)}(\Delta) = \frac{\Delta}{\sqrt{2\pi\sigma(\gamma)}} h(\gamma, l\Delta)e^{-\frac{(l\Delta-\mu(\gamma))^2}{\sigma(\gamma)^2}} \]
\[ S_2^{(l)}(\Delta) = \frac{\Delta}{\sqrt{2\pi\sigma(\gamma)}} h(-\gamma, l\Delta)e^{-\frac{(l\Delta-\mu(\gamma))^2}{\sigma(\gamma)^2}} \]
\[ S_3^{(l)}(\Delta) = \frac{\Delta}{\sqrt{2\pi\sigma(\gamma)}} h(\gamma, l\Delta)h(-\gamma, l\Delta)e^{-\frac{(l\Delta-\mu(\gamma))^2}{\sigma(\gamma)^2}} \]
\[ S_4^{(l)}(\Delta) = \frac{\Delta}{\sqrt{2\pi\sigma(\gamma)}} e^{-\frac{(l\Delta-\mu(\gamma))^2}{\sigma(\gamma)^2}} \] (5.90)

Clearly, \( S_1(\Delta), S_2(\Delta), S_3(\Delta), \) and \( S_4(\Delta) \) represent the Riemann sums corresponding to \( g(\gamma), g(-\gamma), \Pr\{\text{successful decoding} \}, \) and \( 1 - Q(\frac{\mu(\gamma)}{\sigma(\gamma)}) \), respectively. Consequently, to prove the lemma, we show that

\[ S_1(\Delta)S_2(\Delta) \leq S_3(\Delta)S_4(\Delta). \] (5.91)
To prove (5.91), let us first prove the following equality, inspired by a proof of the Cauchy-Schwartz inequality in [17]. Consider two sets of real numbers, i.e., \( \phi_1, \phi_2, \ldots, \) and \( \psi_1, \psi_2, \ldots \). It is not hard to see that

\[
\sum_{l=0}^{\infty} \left( \dfrac{\phi_l}{l!} \sum_{q=0}^{\infty} \psi_q^2 \right) - \left( \sum_{l=0}^{\infty} \phi_l \right)^2 = \sum_{l=0}^{\infty} \phi_l^2 + \sum_{l=0}^{\infty} \sum_{q=0,q \neq l}^{\infty} \dfrac{\psi_q^2}{l!} \phi_l^2
\]

\[
- \sum_{l=0}^{\infty} \phi_l^2 - \sum_{l=0}^{\infty} \sum_{q=0,q \neq l}^{\infty} \phi_l \phi_l
\]

\[
= \sum_{l=0}^{\infty} \sum_{q=0,q \neq l}^{\infty} \dfrac{\psi_q^2}{l!} \phi_l^2 - \sum_{l=0}^{\infty} \sum_{q=0,q \neq l}^{\infty} \phi_l \phi_l
\]

\[
= \sum_{l=1}^{\infty} \sum_{q=0}^{l-1} \left( \phi_l \psi_q - \phi_q \psi_l \right)^2.
\]

Setting \( \phi_l = \sqrt{S_1^{(l)}(\Delta) S_2^{(l)}(\Delta)} \) and \( \psi_l = \sqrt{S_2^{(l)}(\Delta)} \), we have

\[
\sum_{l=0}^{\infty} S_1^{(l)}(\Delta) S_2^{(l)}(\Delta) - \left( \sum_{l=0}^{\infty} \sqrt{S_1^{(l)}(\Delta) S_2^{(l)}(\Delta)} \right)^2 = \sum_{l=1}^{\infty} \sum_{q=0}^{l-1} \left( \sqrt{S_1^{(l)}(\Delta) S_2^{(l)}(\Delta)} - \sqrt{S_1^{(q)}(\Delta) S_2^{(l)}(\Delta)} \right)^2,
\]

or, equivalently,

\[
S_1(\Delta) S_2(\Delta) - \left( \sum_{l=0}^{\infty} \sqrt{S_1^{(l)}(\Delta) S_2^{(l)}(\Delta)} \right)^2 = \sum_{l=1}^{\infty} \sum_{q=0}^{l-1} \left( \sqrt{S_1^{(l)}(\Delta) S_2^{(l)}(\Delta)} - \sqrt{S_1^{(q)}(\Delta) S_2^{(l)}(\Delta)} \right)^2.
\]

Similarly letting \( \phi_l = \sqrt{S_3^{(l)}(\Delta) S_4^{(l)}(\Delta)} \) and \( \psi_l = \sqrt{S_4^{(l)}(\Delta)} \), we get

\[
\sum_{l=0}^{\infty} S_3^{(l)}(\Delta) S_4^{(l)}(\Delta) - \left( \sum_{l=0}^{\infty} \sqrt{S_3^{(l)}(\Delta) S_4^{(l)}(\Delta)} \right)^2 = \sum_{l=1}^{\infty} \sum_{q=0}^{l-1} \left( \sqrt{S_3^{(l)}(\Delta) S_4^{(l)}(\Delta)} - \sqrt{S_3^{(q)}(\Delta) S_4^{(l)}(\Delta)} \right)^2,
\]

or

\[
S_3(\Delta) S_4(\Delta) - \left( \sum_{l=0}^{\infty} \sqrt{S_3^{(l)}(\Delta) S_4^{(l)}(\Delta)} \right)^2 = \sum_{l=1}^{\infty} \sum_{q=0}^{l-1} \left( \sqrt{S_3^{(l)}(\Delta) S_4^{(l)}(\Delta)} - \sqrt{S_3^{(q)}(\Delta) S_4^{(l)}(\Delta)} \right)^2.
\]

Since for any choice of \( l \in \mathbb{N} \) and for any \( \Delta > 0 \),

\[
\sqrt{S_1^{(l)}(\Delta) S_2^{(l)}(\Delta)} = \sqrt{S_3^{(l)}(\Delta) S_4^{(l)}(\Delta)},
\]

(5.97)
we deduce that

\[
S_3(\Delta)S_4(\Delta) - S_1(\Delta)S_2(\Delta) = \sum_{l=1}^{\infty} \sum_{q=0}^{l-1} \left( \sqrt{S_3^{(l)}(\Delta)S_4^{(q)}(\Delta)} - \sqrt{S_3^{(q)}(\Delta)S_4^{(l)}(\Delta)} \right)^2 - \left( \sqrt{S_1^{(l)}(\Delta)S_2^{(q)}(\Delta)} - \sqrt{S_1^{(q)}(\Delta)S_2^{(l)}(\Delta)} \right)^2.
\]

(5.98)

It is not hard to see that

\[
\sqrt{S_3^{(l)}(\Delta)S_4^{(q)}(\Delta)} = \sqrt{S_3^{(q)}(\Delta)S_4^{(l)}(\Delta)}.
\]

(5.99)

Considering the fact that, in (5.98), \( l > q \) one can deduce that

\[
h(-|\gamma|, l\Delta) \geq h(-|\gamma|, q\Delta),
\]

(5.100)

where \(| \cdot |\) denotes the absolute value. As a result

\[
\sqrt{h(-|\gamma|, l\Delta)} \geq \sqrt{h(-|\gamma|, q\Delta)}.
\]

(5.101)

We further claim that

\[
\sqrt{h(|\gamma|, l\Delta)} \geq \sqrt{h(|\gamma|, q\Delta)}.
\]

(5.102)

To establish this claim, let us first consider the definition of \( h(\cdot, \cdot) \),

\[
h(|\gamma|, l\Delta) = \Pr \left\{ \omega(\hat{\theta}) > 0 \quad \forall \theta < |\gamma| \big| \omega(|\gamma|) = l\Delta \right\}
\]

\[
= \int_{-\infty}^{\infty} \Pr \left\{ \omega(\hat{\theta}) > 0 \quad \forall \theta < |\gamma| \big| \omega(0) = \xi_1, \omega(|\gamma|) = l\Delta \right\}
\]

\[
P \left\{ \omega(0) = \xi_1 \big| \omega(|\gamma|) = l\Delta \right\} d\xi_1.
\]

(5.103)

Now define

\[
F_1(x) \triangleq \Pr \left\{ \omega(0) \leq x \big| \omega(|\gamma|) = l\Delta \right\},
\]

(5.104)

and let \( u \) be a uniform random variable between 0 and 1. It’s not hard to see that

\[
\Pr \left\{ F_1^{-1}(u) \leq x \right\} = F_1(x).
\]

(5.105)

Using (5.105), we can rewrite (5.103) as

\[
h(|\gamma|, l\Delta) = \int_{-\infty}^{\infty} \Pr \left\{ \omega(\hat{\theta}) > 0 \quad \forall \theta < |\gamma| \big| \omega(0) = \xi_1, \omega(|\gamma|) = l\Delta \right\}
\]

\[
P \left\{ F_1^{-1}(u) = \xi_1 \right\} d\xi_1.
\]

(5.106)
Similarly, it can be shown that

\[
  h(|\gamma|, q\Delta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Pr \left\{ \omega(\theta) > 0 \; \forall \theta < |\gamma| \; \omega(0) = \xi_0, \omega(|\gamma|) = q\Delta \right\} \Pr \left\{ F_2^{-1}(u) = \xi_0 \right\} d\xi_1 d\xi_2. 
\]

(5.107)

where \( F_2(x) \triangleq \Pr \left\{ \omega(0) \leq x \; \omega(|\gamma|) = q\Delta \right\} \).

Since

\[
  \int_{-\infty}^{\infty} P \left\{ F_1^{-1}(u) = \xi_1, F_2^{-1}(u) = \xi_2 \right\} d\xi_1 = P \left\{ F_2^{-1}(u) = \xi_2 \right\}
\]

\[
  \int_{-\infty}^{\infty} P \left\{ F_1^{-1}(u) = \xi_1, F_2^{-1}(u) = \xi_2 \right\} d\xi_2 = P \left\{ F_1^{-1}(u) = \xi_1 \right\},
\]

(5.108)

we can reformulate (5.106) and (5.107) as follows:

\[
  h(|\gamma|, l\Delta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} P \left\{ F_1^{-1}(u) = \xi_1, F_2^{-1}(u) = \xi_2 \right\} \Pr \left\{ \omega(\theta) > 0 \; \forall \theta < |\gamma| \; \omega(0) = \xi_0, \omega(|\gamma|) = l\Delta \right\} d\xi_1 d\xi_2
\]

\[
  h(|\gamma|, q\Delta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} P \left\{ F_1^{-1}(u) = \xi_1, F_2^{-1}(u) = \xi_2 \right\} \Pr \left\{ \omega(\theta) > 0 \; \forall \theta < |\gamma| \; \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\} d\xi_1 d\xi_2.
\]

(5.109)

As a result

\[
  h(|\gamma|, l\Delta) - h(|\gamma|, q\Delta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} P \left\{ F_1^{-1}(u) = \xi_1, F_2^{-1}(u) = \xi_2 \right\} \left[ \Pr \left\{ \omega(\theta) > 0 \; \forall \theta < |\gamma| \; \omega(0) = \xi_0, \omega(|\gamma|) = l\Delta \right\} - \Pr \left\{ \omega(\theta) > 0 \; \forall \theta < |\gamma| \; \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\} \right] d\xi_1 d\xi_2.
\]

(5.110)

In order to prove (5.102), we need to establish two lemmas first:

**Lemma 5.7** For any \( \xi_2 > \xi_1 \), we have

\[
  P \left\{ F_1^{-1}(u) = \xi_1, F_2^{-1}(u) = \xi_2 \right\} = 0. 
\]

(5.111)

**Proof:** From the definition of \( F_1(x) \), we have

\[
  F_1(x) = \Pr \left\{ \omega(0) \leq x \; \omega(|\gamma|) = l\Delta \right\} = \frac{\Pr \{ \omega(0) \leq x, \omega(|\gamma|) = l\Delta \} \Pr \{ \omega(|\gamma|) = l\Delta \}}{\Pr \{ \omega(|\gamma|) = l\Delta \}} = \int_{-\infty}^{x} \frac{\Pr \{ \omega(|\gamma|) = l\Delta \} \Pr \{ \omega(0) = \xi \} d\xi}{\Pr \{ \omega(|\gamma|) = l\Delta \}}.
\]

(5.112)

Considering the definition of \( \omega(\cdot) \), it is not hard to see that \( \omega(0) \) and \( \omega(|\gamma|) - \omega(0) \) are
two independent Gaussian random variables. Let us define

\[ \mu_1 = E\{\omega(0)\} \]
\[ \sigma_1 = E\{\omega^2(0)\} - E^2\{\omega(0)\} \]
\[ \mu_2 = E\{\omega(|\gamma|) - \omega(0)\} \]
\[ \sigma_2 = E\{(\omega(|\gamma|) - \omega(0))^2\} - E^2\{\omega(|\gamma|) - \omega(0)\}. \] (5.113)

Now, we can reformulate \( F_1(\cdot) \) as

\[
F_1(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(\xi - \mu_1)^2}{2\sigma_1^2}} - \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(l\Delta - \mu_2)^2}{2\sigma_2^2}} d\xi.
\] (5.114)

With some algebraic manipulations, (5.114) can be rewritten as

\[
F_1(x) = \frac{\sqrt{\sigma_1^2 + \sigma_2^2}}{\sqrt{2\pi}\sigma_1\sigma_2} \int_{-\infty}^{x} e^{-\frac{1}{2} \left( \frac{(l\Delta - \mu_2) \sigma_1^2 + \mu_1 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \right)^2} d\xi,
\] (5.115)

or, equivalently,

\[
F_1(x) = 1 - Q \left( \frac{x(\sigma_1^2 + \sigma_2^2) - (l\Delta - \mu_2)\sigma_1^2 - \mu_1 \sigma_2^2}{\sigma_1\sigma_2 \sqrt{\sigma_1^2 + \sigma_2^2}} \right). \] (5.116)

Following the same steps, we get

\[
F_2(x) = 1 - Q \left( \frac{x(\sigma_1^2 + \sigma_2^2) - (q\Delta - \mu_2)\sigma_1^2 - \mu_1 \sigma_2^2}{\sigma_1\sigma_2 \sqrt{\sigma_1^2 + \sigma_2^2}} \right). \] (5.117)

Comparing (5.116) and (5.117), one can deduce that for any choice of \( x \),

\[
F_2(x) \geq F_1(x),
\] (5.118)

and, as a result, for any \( x \) between 0 and 1,

\[
F_2^{-1}(x) \leq F_1^{-1}(x).
\] (5.119)

Consequently, one can deduce that for any \( \xi_2 > \xi_1 \),

\[
P\{F_1^{-1}(u) = \xi_1, F_2^{-1}(u) = \xi_2\} = 0,
\] (5.120)

as desired.
Lemma 5.8 For all $\xi_1 \geq \xi_2$, we have

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} \geq$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\}. \quad (5.121)$$

Proof: For $\xi_2 < 0$,

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\} = 0. \quad (5.122)$$

Consequently, one can deduce that when $\xi_2 < 0$, for any choice of $\xi_1$,

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} \geq$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\}. \quad (5.123)$$

Now, let us concentrate on the scenarios where $\xi_1 \geq \xi_2 > 0$.

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} =$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} =$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} =$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} =$$

As a result,

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} =$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} =$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \right\} =$$

Similarly,

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\} =$$

$$\Pr\left\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\} =$$

To prove (5.121), let us first compare the first right-hand side (RHS) term of (5.126), i.e.,

$$\Pr\{ \omega(\theta) > 0 \quad \forall \theta < \gamma \big| \omega(0) = \xi_1, \omega(|\gamma|) = \Delta \},$$
and the first RHS term of (5.127), i.e.,

\[ \Pr \{ \omega(\tilde{\theta}) > 0 \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| \omega(0) = \xi_1, \omega(|\gamma|) = l\Delta \} = \]

\[ \Pr \{ B(\tilde{\theta}) > -\xi_1 - \frac{\tilde{\theta}^2}{2} \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| \omega(0) = 0, B(\gamma) = l\Delta - \xi_1 - \frac{\alpha^2}{2} \}. \]  

(5.128)

From the definition of \(\omega(\cdot)\), we get

\[ \Pr \{ \omega(\tilde{\theta}) > 0 \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| \omega(0) = \xi_1, \omega(|\gamma|) = l\Delta \} = \]

\[ \Pr \{ B_1(\tilde{\theta}) > -\frac{\xi_1}{2} \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| B_1(0) = \xi_1, B_1(\gamma) = l\Delta - \xi_1 - \frac{\alpha^2}{2} \}. \]

(5.129)

Let us define \(B_1(\tilde{\theta}) \triangleq B(\tilde{\theta}) + \xi_1\). Note that \(B_1(\cdot)\) can also be considered as a two-sided Brownian motion starting at time 0 from \(B_1(0) = \xi_1\). Consequently, (5.128) can be reformulated as follows:

\[ \Pr \{ \omega(\tilde{\theta}) > 0 \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| \omega(0) = \xi_1, \omega(|\gamma|) = l\Delta \} = \]

\[ \Pr \{ B_1(\tilde{\theta}) > -\frac{\alpha^2}{2} \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| B_1(0) = \xi_1, B_1(\gamma) = l\Delta - \xi_1 - \frac{\alpha^2}{2} \}. \]

(5.130)

Define the Brownian bridge \(B_{a,b,l}(\tilde{\theta})\) as the continuous-time stochastic process at time \(\tilde{\theta}\) whose probability distribution is the conditional probability distribution of a Wiener process given the conditions that \(B_{a,b,l}(0) = a\), and \(B_{a,b,l}(l) = b\) [18]. Using the definition of the Brownian bridge, we have

\[ \Pr \{ \omega(\tilde{\theta}) > 0 \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| \omega(0) = \xi_1, \omega(|\gamma|) = l\Delta \} = \]

\[ \Pr \{ B_{a,b,l}(\tilde{\theta}) > -\frac{\alpha^2}{2} \ \forall 0 < \tilde{\theta} < |\gamma| \ \big| B_{a,b,l}(0) = a, B_{a,b,l}(l) = b \}. \]

(5.131)

Combining (5.126) and (5.130), we have

\[ \Pr \left\{ \omega(\tilde{\theta}) > 0 \ \forall \tilde{\theta} < |\gamma| \ \big| \omega(0) = \xi_1, \omega(|\gamma|) = l\Delta \right\} = \]

\[ \Pr \left\{ B_{\xi_1,l\Delta - \xi_1^2,|\gamma|}(\tilde{\theta}) > -\frac{\alpha^2}{2} \ \forall \tilde{\theta} < |\gamma| \right\} \Pr \left\{ \omega(\tilde{\theta}) > 0 \ \forall \tilde{\theta} < 0 \ \omega(0) = \xi_1 \right\}. \]

(5.132)

Similarly, one can see that,

\[ \Pr \left\{ \omega(\tilde{\theta}) > 0 \ \forall \tilde{\theta} < |\gamma| \ \big| \omega(0) = \xi_2, \omega(|\gamma|) = q\Delta \right\} = \]

\[ \Pr \left\{ B_{\xi_2,q\Delta - \xi_2^2,|\gamma|}(\tilde{\theta}) > -\frac{\alpha^2}{2} \ \forall \tilde{\theta} < |\gamma| \right\} \Pr \left\{ \omega(\tilde{\theta}) > 0 \ \forall \tilde{\theta} < 0 \ \omega(0) = \xi_2 \right\}. \]

(5.133)

Note that \(B_{\xi_1,l\Delta - \xi_1^2,|\gamma|}(\cdot)\), can be viewed as \(B_{\xi_2,q\Delta - \xi_2^2,|\gamma|}(\cdot)\) with a linear drift. To be exact, for any \(\tilde{\theta}\), such that \(0 \leq \tilde{\theta} \leq \gamma\), \(B_{\xi_1,l\Delta - \xi_1^2,|\gamma|}(\tilde{\theta})\) is identical in law to [19]

\[ \left\{ B_{\xi_2,q\Delta - \xi_2^2,|\gamma|}(\tilde{\theta}) + [(l - q)\Delta - (\xi_1 - \xi_2)] \frac{\tilde{\theta}}{|\gamma|} + (\xi_1 - \xi_2) \right\}. \]
Putting (5.139) and (5.99) together and considering the fact that
or, equivalently,
\[ \sqrt{\gamma} \] can verify that
\[ h \] as desired.
\[ \blacksquare \]

Putting everything together, we deduce that for any \( \xi \) can easily see that, for any \( \xi \)

Now, let us compare the second term of (5.126) versus the second term of (5.127). One

As a result, one can deduce that

Combining the results from Lemma 5.7, Lemma 5.8, and Equation (5.110), one
can verify that \( h(|\gamma|, l\Delta) - h(|\gamma|, q\Delta) \geq 0 \). As a result

\[ h(|\gamma|, l\Delta) \geq h(|\gamma|, q\Delta), \] (5.137)

or, equivalently,

\[ \sqrt{h(|\gamma|, l\Delta)} \geq \sqrt{h(|\gamma|, q\Delta)}. \] (5.138)

Using (5.101) and (5.138), we have

\[ \max \left\{ \frac{S_1^{(l)}(\Delta) S_3^{(q)}(\Delta)}{S_3^{(q)}(\Delta) S_1^{(l)}(\Delta)}, \frac{S_1^{(q)}(\Delta) S_3^{(l)}(\Delta)}{S_3^{(l)}(\Delta) S_1^{(q)}(\Delta)} \right\} = \max \left\{ \sqrt{\frac{h(|\gamma|, l\Delta) h(|\gamma|, q\Delta)}{h(|\gamma|, q\Delta) h(|\gamma|, l\Delta)}}, \sqrt{\frac{h(|\gamma|, q\Delta) h(|\gamma|, l\Delta)}{h(|\gamma|, l\Delta) h(|\gamma|, q\Delta)}} \right\} \]

\[ = \max \left\{ \frac{S_1^{(q)}(\Delta) S_3^{(l)}(\Delta)}{S_3^{(l)}(\Delta) S_1^{(q)}(\Delta)}, \frac{S_1^{(l)}(\Delta) S_3^{(q)}(\Delta)}{S_3^{(q)}(\Delta) S_1^{(l)}(\Delta)} \right\}. \] (5.139)

Putting (5.139) and (5.99) together and considering the fact that \( \sqrt{S_3^{(l)}(\Delta) S_4^{(q)}(\Delta)} > 0 \),
\[ \sqrt{S_3^{(q)}(\Delta) S_4^{(l)}(\Delta)} > 0, \sqrt{S_1^{(l)}(\Delta) S_2^{(q)}(\Delta)} > 0, \) and \( \sqrt{S_1^{(q)}(\Delta) S_2^{(l)}(\Delta)} > 0 \), one can deduce that

\[ \left( \sqrt{S_3^{(l)}(\Delta) S_4^{(q)}(\Delta)} - \sqrt{S_3^{(q)}(\Delta) S_4^{(l)}(\Delta)} \right)^2 \geq \left( \sqrt{S_1^{(l)}(\Delta) S_2^{(q)}(\Delta)} - \sqrt{S_1^{(q)}(\Delta) S_2^{(l)}(\Delta)} \right)^2. \] (5.140)

Combining (5.140) and (5.98) we get

\[ S_1(\Delta) S_2(\Delta) \leq S_3(\Delta) S_4(\Delta). \] (5.141)

\[ \blacksquare \]
Appendix 5.D Proof of Theorem 5.5

Proof: To prove this theorem, we first verify the lower bound, i.e.,

\[
1 - Q\left(\frac{\hat{\mu}(0)}{\hat{\sigma}(0)}\right) - \frac{1}{\sqrt{2\pi\hat{\sigma}(0)}} e^{-\frac{\hat{\sigma}^2(0)}{2\hat{\sigma}(0)^2}} \leq g(\gamma),
\]

then we establish the upper bound, i.e.,

\[
g(\gamma) \leq \min_{M > 0} \sqrt{\left(1 - Q\left(\frac{\hat{\mu}(0,M)}{\hat{\sigma}(0)}\right) - \frac{1}{\sqrt{2\pi\hat{\sigma}(0)}} e^{-\frac{\hat{\sigma}^2(0,M)}{2\hat{\sigma}(0)^2}} \right) \left(1 - Q\left(\frac{\hat{\mu}(0,M)}{\hat{\sigma}(0)}\right)\right)}}.
\]

From the definition of \( g(\gamma) \), we have

\[
g(\gamma) = \Pr\{\omega(\theta) > 0 \quad \forall \theta \leq \gamma\}
= \Pr\{\omega(0) + B(\theta) + \frac{\theta^2}{2} > 0 \quad \forall \theta \leq \gamma\}
= \Pr\{\omega(0) + \hat{B}_\gamma(\theta) + B(\gamma) + \frac{(\theta + \gamma)^2}{2} > 0 \quad \forall \theta \leq 0\},
\]

where \( \hat{B}_\gamma(\theta) \triangleq B(\theta + \gamma) - B(\gamma) \). Note that \( \hat{B}_\gamma(\theta) \) can also be considered as a Brownian motion starting at \( \hat{\theta} = 0 \), i.e., \( \hat{B}_\gamma(0) = 0 \). It is not hard to see that for any choice of \( \gamma < 0 \),

\[
g(\gamma) \geq \Pr\{\omega(0) + \hat{B}_\gamma(\theta) + B(\gamma) + \frac{\theta^2 + \gamma^2}{2} > 0 \quad \forall \theta \leq 0\}.
\]

Let us define \( \hat{\omega}_\gamma(\hat{\theta}) \) as

\[
\hat{\omega}_\gamma(\hat{\theta}) \triangleq \omega(0) + \hat{B}_\gamma(\hat{\theta}) + \frac{\hat{\theta}^2}{2},
\]

where

\[
\hat{\omega}_\gamma(0) \triangleq \omega(0) + B(\gamma) + \frac{\gamma^2}{2}.
\]

Consequently, (5.145) can be rewritten as

\[
g(\gamma) \geq \Pr\{\hat{\omega}_\gamma(\theta) > 0 \quad \forall \hat{\theta} \leq 0\}
\geq \Pr\{\hat{\omega}_\gamma(\hat{\theta}) > 0 \quad \forall \hat{\theta}\}.
\]

To compute \( \Pr\{\hat{\omega}_\gamma(\theta) > 0 \quad \forall \hat{\theta}\} \), we follow an approach similar to the one used in the
derivation of the word erasure rate in [13].

\[
\Pr\{\hat{w}_r(\hat{\theta}) > 0 \quad \forall \hat{\theta} \} = 1 - \Pr\{\hat{w}_r(0) < 0\} - \Pr\{\min_{\hat{\theta}}[\hat{w}_r(\hat{\theta})] < 0, \hat{w}_r(0) > 0\}
\]

\[
= 1 - \Pr\{\hat{w}_r(0) < 0\} - \int_{\xi=0}^{\infty} \Pr\{\min_{\hat{\theta}}[\hat{w}_r(\hat{\theta})] < 0|\hat{w}_r(0) = \xi\} P\{\hat{w}_r(0) = \xi\} d\xi
\]

\[
\approx 1 - \Pr\{\hat{w}_r(0) < 0\} - P\{\hat{w}_r(0) = 0\} \int_{\xi=0}^{\infty} \Pr\{\min_{\hat{\theta}}[\hat{w}_r(\hat{\theta})] < 0|\hat{w}_r(0) = \xi\} d\xi.
\]

(5.149)

Since \(\Pr\{\min_{\hat{\theta}}[\hat{w}_r(\hat{\theta})] < 0|\hat{w}_r(0) = \xi\}\) diminishes faster than \(P\{\hat{w}_r(0) = \xi\}\), at the last line, we approximated the \(P\{\hat{w}_r(0) = \xi\}\) with \(P\{\hat{w}_r(0) = 0\}\) (similar to [13]). It is not hard to see that

\[
\Pr\{\hat{w}_r(0) < 0\} = Q\left(\frac{\hat{w}_r(0)}{\Delta r(0)}\right),
\]

\[
P\{\hat{w}_r(0) = 0\} = \frac{1}{\sqrt{2\pi} \sigma_r(0)} e^{-\frac{\hat{w}_r(0)^2}{2\sigma_r^2(0)}},
\]

where

\[
\hat{\mu}_r(0) = \gamma^2 (\epsilon^* - \epsilon) \frac{\partial R_2}{\partial \epsilon} |_{\epsilon} + \frac{1}{2} \gamma^2
\]

\[
\delta_r^2(0) = \frac{\sigma_r^2}{\Delta r^2} \left( \frac{\partial R_2}{\partial \epsilon} |_{\epsilon} \right)^2 + |\gamma|.
\]

(5.151)

The probability distribution of the minimum of a Brownian motion with a parabolic drift has been studied in [14]. Adapting the results in computation of

\[
\Pr\{\min_{\hat{\theta}}[\hat{w}_r(\hat{\theta})] < 0|\hat{w}_r(0) = \xi\},
\]

we get

\[
\Pr\{\min_{\hat{\theta}}[\hat{w}_r(\hat{\theta})] < 0|\hat{w}_r(0) = \xi\} = \int_0^{\infty} 1 - K^2(z) dz,
\]

(5.152)

where

\[
K(z) = \frac{1}{2} \int_{-\infty}^{\infty} \frac{\text{Ai}(iv) \text{Bi}(iv+2^{1/3}z) - \text{Bi}(iv) \text{Ai}(iv+2^{1/3}z)}{\text{Ai}(iv)} dv.
\]

(5.153)

With numerical computation, it can be shown that [13]

\[
\int_0^{\infty} 1 - K^2(z) dz \approx 1.
\]

(5.154)

Putting everything together, we get

\[
\Pr\{\hat{w}_r(\hat{\theta}) > 0 \quad \forall \hat{\theta} \} \approx 1 - Q\left(\frac{\hat{w}_r(0)}{\Delta r(0)}\right) - \frac{1}{\sqrt{2\pi} \sigma_r(0)} e^{-\frac{\hat{w}_r(0)^2}{2\sigma_r^2(0)}}.
\]

(5.155)
Consequently, for any $\gamma < 0$,
\[
g(\gamma) \geq \Pr\{\hat{\omega}_\gamma(\hat{\theta}) > 0 \quad \forall \hat{\theta}\}
\approx 1 - Q\left(\frac{\hat{\mu}_\gamma(0)}{\hat{\theta}_\gamma(0)}\right) - \frac{1}{\sqrt{2\pi\hat{\theta}_\gamma(0)}} e^{-\frac{\hat{\sigma}_\gamma^2(0)}{2\hat{\theta}_\gamma(0)}},
\] (5.156)
as desired.

Now, let us focus on verifying the upper bound. Starting with (5.144), we have
\[
g(\gamma) = \Pr\{\omega(0) + \hat{B}_\gamma(\hat{\theta}) + B(\gamma) + \frac{(\hat{\theta} + \gamma)^2}{2} > 0 \quad \forall \hat{\theta} \leq 0\},
\] (5.157)
Clearly, for any choice of $M > 0$,
\[
\frac{(\hat{\theta} + \gamma)^2}{2} \leq \frac{\hat{\theta}^2}{2} + \frac{\gamma^2}{2} + \frac{M\hat{\theta}^2}{2} + \frac{\gamma^2}{2M}
= \frac{\hat{\theta}^2}{2}(1 + M) + \frac{\gamma^2}{2}(1 + \frac{1}{M}).
\] (5.158)
As a result,
\[
g(\gamma) \leq \Pr\{\omega(0) + \hat{B}_\gamma(\hat{\theta}) + B(\gamma) + \frac{\hat{\theta}^2}{2}(1 + M) + \frac{\gamma^2}{2}(1 + \frac{1}{M}) > 0 \quad \forall \hat{\theta} \leq 0\},
\] (5.159)
Define $\tilde{\omega}_\gamma(\cdot)$ as follows:
\[
\tilde{\omega}_\gamma(\hat{\theta}) \triangleq \tilde{\omega}_\gamma(0) + \hat{B}_\gamma(\hat{\theta}) + \frac{\hat{\theta}^2}{2}(1 + M),
\] (5.160)
where
\[
\tilde{\omega}_\gamma(0) \triangleq \omega(0) + B(\gamma) + \frac{\gamma^2}{2}(1 + \frac{1}{M}).
\] (5.161)
Consequently, (5.159) can be rewritten as
\[
g(\gamma) \leq \Pr\{\tilde{\omega}_\gamma(\hat{\theta}) > 0 \quad \forall \hat{\theta} \leq 0\}.
\] (5.162)
Since $\tilde{\omega}_\gamma(\cdot)$ corresponds to a Brownian motion with a parabolic drift, let us define
\[
\tilde{g}_\gamma(\tilde{\varphi}) = \Pr\{\tilde{\omega}(\tilde{\varphi}) > 0 \quad \forall \tilde{\varphi} \leq \tilde{\varphi}\}.
\] (5.163)
From Corollary 5.3, we get
\[
\tilde{g}_\gamma(0) = \Pr\{\tilde{\omega}(\tilde{\varphi}) > 0 \quad \forall \tilde{\varphi} \leq 0\}
\leq \sqrt{\Pr\{\tilde{\omega}_\gamma(\hat{\theta}) > 0 \quad \forall \hat{\theta} \leq 0\} \left(1 - Q\left(\frac{\hat{\mu}_\gamma(0,M)}{\hat{\theta}_\gamma(0)}\right)\right)},
\] (5.164)
where \( \mu_\gamma(0, M) \) and \( \sigma^2_\gamma(0) \), respectively, denote the mean and variance of \( \omega_\gamma(0) \), and are given by

\[
\begin{align*}
\mu_\gamma(0, M) &= \mu(0) + \frac{\gamma^2}{2} (1 + \frac{1}{M}) \\
\sigma^2_\gamma(0) &= \sigma^2(0) + |\gamma|.
\end{align*}
\]

(5.165)

Again with an approach similar to the one used for computation of the word erasure rate in [13], we have

\[
\begin{align*}
\Pr\{\omega(\hat{\theta}) > 0 \ \forall \hat{\theta} \leq \gamma\} &= 1 - \Pr\{\omega_\gamma(0) < 0\} - \Pr\{\min_\delta[\omega_\gamma(\hat{\theta})] < 0, \omega_\gamma(0) > 0\} \\
&= 1 - \Pr\{\omega_\gamma(0) < 0\} - \\
&\int_{\xi=0}^{\infty} \Pr\{\min_\delta[\omega_\gamma(\hat{\theta})] < 0| \omega_\gamma(0) = \xi\} P\{\omega_\gamma(0) = \xi\} d\xi \\
&\approx 1 - \Pr\{\omega_\gamma(0) < 0\} - \\
P\{\omega_\gamma(0) = 0\} \int_{\xi=0}^{\infty} \Pr\{\min_\delta[\omega_\gamma(\hat{\theta})] < 0| \omega_\gamma(0) = \xi\} d\xi.
\end{align*}
\]

(5.166)

Since \( \Pr\{\min_\delta[\omega_\gamma(\hat{\theta})] < 0| \omega_\gamma(0) = \xi\} \) diminishes faster than \( P\{\omega_\gamma(0) = \xi\} \), we approximate \( P\{\omega_\gamma(0) = \xi\} \) with \( P\{\omega_\gamma(0) = 0\} \). Note that this approximation is similar to the approximation used in the derivation of word erasure rate in [13]. It is not hard to see that

\[
\Pr\{\omega_\gamma(0) < 0\} = Q\left( \frac{\mu_\gamma(0, M)}{\sigma_\gamma(0)} \right)
\]

\[
P\{\omega_\gamma(0) = 0\} = \frac{1}{\sqrt{2\pi}\sigma_\gamma(0)} e^{-\frac{\mu^2_\gamma(0, M)}{2\sigma^2_\gamma(0)}}.
\]

(5.167)

Furthermore, using the results from [14] in computation of

\[
\Pr\{\min_\delta[\omega_\gamma(\hat{\theta})] < 0| \omega_\gamma(0) = \xi\},
\]

it can be shown that

\[
\Pr\{\min_\delta[\omega_\gamma(\hat{\theta})] < 0| \omega_\gamma(0) = \xi\} = \int_{0}^{\infty} 1 - \tilde{K}^2(z) dz,
\]

(5.168)

where

\[
\tilde{K}(z) = \frac{1}{2} \int_{-\infty}^{\infty} \frac{Ai(iv)Bi(\sqrt{v+2^{1/3}(1+M)^{1/3}z}) - Bi(iv)Ai(\sqrt{v+2^{1/3}(1+M)^{1/3}z})}{Ai(iv)} dv.
\]

(5.169)

Clearly

\[
\int_{0}^{\infty} 1 - \tilde{K}^2(z) dz = (1 + M)^{-1/3} \int_{0}^{\infty} 1 - \tilde{K}^2(z) dz \\
\approx (1 + M)^{-1/3}.
\]

(5.170)
Putting everything together, we get
\[
\Pr\{\hat{\omega}(\hat{\theta}) > 0 \ \forall \theta \leq \bar{\gamma}\} \approx 1 - Q\left(\frac{\hat{\mu}_{\gamma}(0, M)}{\hat{\sigma}_{\gamma}(0)}\right) - \frac{(1 + M)^{-1/3}}{\sqrt{2\pi \hat{\sigma}_{\gamma}(0)}} e^{-\frac{\hat{\mu}_{\gamma}^2(0, M)}{2 \hat{\sigma}_{\gamma}^2(0)}}.
\]

(5.171)

Consequently, one can deduce that
\[
g(\gamma) \leq \min_{M > 0} \sqrt{\left(1 - Q\left(\frac{\hat{\mu}_{\gamma}(0, M)}{\hat{\sigma}_{\gamma}(0)}\right) - \frac{(1 + M)^{-1/3}}{\sqrt{2\pi \hat{\sigma}_{\gamma}(0)}} e^{-\frac{\hat{\mu}_{\gamma}^2(0, M)}{2 \hat{\sigma}_{\gamma}^2(0)}}\right)(1 - Q\left(\frac{\hat{\mu}_{\gamma}(0, M)}{\hat{\sigma}_{\gamma}(0)}\right))},
\]

(5.172)

as desired.

Bibliography


