LDPC CODING FOR MAGNETIC STORAGE: LOW-FLOOR DECODING ALGORITHMS, SYSTEM DESIGN, AND PERFORMANCE ANALYSIS

by

Yang Han

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2008
As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Yang Han entitled “LDPC Coding For Magnetic Storage: Low Floor Decoding Algorithms, System Design, and Performance Analysis” and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

William E. Ryan, Ph.D.  Date: August 7, 2008

Michael W. Marcellin, Ph.D.  Date: August 7, 2008

Bane Vasic, Ph.D.  Date: August 7, 2008

Final approval and acceptance of this dissertation is contingent upon the candidate’s submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director: William E. Ryan, Ph.D.  Date: August 7, 2008
STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: ___________________________ Yang Han
ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my supervisor Professor William E. Ryan not just for all the precious advices, but also for the encouragements and for placing trust in me. He guided me through the years, helped me to grow passion in academic research and changed me from a naive fresh graduate from college to what I am now. Because of his support and understanding, I had a pleasant and valuable experience during my Ph.D study in the ECE department, University of Arizona. This dissertation would have been impossible without his guidance.

I would like to thank Professor Bane V. Vasic and Professor Michael M. Marcellin for agreeing to be members of my dissertation committee and all the valuable suggestions that make this dissertation better.

I wish to give special thanks to Dr. Yuan Xing Lee of LSI Corporation. He made it possible for me to work as a summer intern at Hitachi GST, which gave me precious industrial experience. I appreciate his valuable advice and generous help on both academic and non-academic issues. I wish to thank Dr. Zongwang Li at LSI Corporation for the suggestions and discussions on the very interesting error floor research, which enriched my knowledge a lot.

I am grateful to my labmates in the Signal Processing and Coding Lab for their help and friendship, including Yifei Zhang, Yan Li, Bo Xia, Michael Yang, Zuqiang Tang, Fei Peng, Shadi Abu-Surra, Lingling Pu and Ling Su. Their suggestions and discussions enlightened me a lot and refined my understanding of the field.

The research work would not have been possible without the financial support of the International Storage Industry Consortium (INSIC). I appreciate their support throughout both my master’s and doctoral studies that help me a great deal in bringing this work to a successful end. I also appreciate the financial support of the NASA Goddard Space Flight Center during my doctoral program. I also appreciate the advice and help from Wai Fong of NASA.

I would also thank Tami Whelan for handling all the paperwork necessary for the completion of my degree.

The support and encouragement I have received from my husband, Fan, my parents and my parents in law throughout my life cannot be measured. Their enlightenment and positive attitude gave me the strength and confidence to follow my dreams.
DEDICATION

To my parents and my husband Fan.
TABLE OF CONTENTS

LIST OF FIGURES .................................................... 8
LIST OF TABLES ...................................................... 11
ABSTRACT ............................................................. 12

CHAPTER 1 INTRODUCTION ......................................... 14
  1.1 Channel Coding Overview .................................. 14
  1.2 Research Environment Surrounding This Dissertation .... 15
  1.3 Dissertation Outline ........................................ 16

CHAPTER 2 ERROR FLOORS: THE CAUSE AND LEVEL PREDICTION . 18
  2.1 The Error Floor Phenomenon and Trapping Sets ............. 18
  2.2 The Weakness of Message-Passing Decoders .................. 22
  2.3 Definitions Related to Trapping Sets ...................... 24
  2.4 Trapping Set Enumeration .................................. 27
  2.5 Trapping Set Evaluation .................................... 33
     2.5.1 IS for Memoryless Channels ......................... 35
     2.5.2 IS for Channels with Memory ....................... 37
  2.6 Conclusions .................................................. 41

CHAPTER 3 LOW-FLOOR DECODERS FOR LDPC CODES .......... 43
  3.1 Codes Under Study .......................................... 45
     3.1.1 The Margulis Code .................................. 45
     3.1.2 The Low-Rate QC Code ............................... 47
     3.1.3 The High-Rate QC Code ............................. 48
  3.2 Bi-Mode Syndrome-Erasure Decoder ........................ 49
     3.2.1 Margulis Code Solution .............................. 51
     3.2.2 Low-Rate QC Code Solution ......................... 54
     3.2.3 Modifications for General Codes ................... 56
  3.3 Concatenation and Bit/State-Pinning ....................... 59
     3.3.1 Pinning Techniques for AWGN Channels ............... 59
     3.3.2 Pinning Techniques for Partial Response Channels .. 63
  3.4 Generalized-LDPC Decoder ................................ 74
     3.4.1 G-LDPC Decoder I .................................... 75
TABLE OF CONTENTS – Continued

3.4.2 G-LDPC Decoder II ........................................... 80
3.4.3 G-LDPC Decoder III ........................................... 83
3.4.4 G-LDPC Decoders for Partial Response Channels .......... 84
3.5 Complexity Discussion ........................................... 85
3.6 Conclusions ....................................................... 89

CHAPTER 4 DUAL-MODE DECODING OF PRODUCT CODES
WITH APPLICATION TO TAPE STORAGE ......................... 91
4.1 Packet-level Codes .............................................. 92
  4.1.1 Packet-LDPC Codes ....................................... 93
  4.1.2 Packet-RS Codes .......................................... 94
4.2 Maximum-Likelihood Decoding on The Erasure Channel .......... 95
4.3 Dual-Mode Decoding for Tape Storage ........................... 97
  4.3.1 Dual-Mode Decoding of RS Codes ......................... 97
  4.3.2 Dual-Mode Decoding of LDPC Codes ...................... 106
4.4 Conclusions ....................................................... 108

CHAPTER 5 CONCATENATING A STRUCTURED LDPC CODE
AND A CONSTRAINT CODE ...................................... 111
5.1 RLL Code Solution .............................................. 112
5.2 DC-free Code Solution .......................................... 115

CHAPTER 6 PERFORMANCE ANALYSIS OF LDPC-CODED
MAGNETIC RECORDING SYSTEMS ................................. 120
6.1 Perpendicular Magnetic Recording Channel Model ................ 121
6.2 Channel Detector .............................................. 124
6.3 The S-IRA LDPC Code .......................................... 127
6.4 Performance Results ............................................ 129
6.5 Conclusions ....................................................... 135

CHAPTER 7 CONCLUSIONS AND RECOMMENDATIONS FOR
FUTURE RESEARCH ............................................... 138

APPENDIX A TRAPPING SET ENUMERATION FOR G-LDPC
DECODERS ......................................................... 141

REFERENCES ......................................................... 143
LIST OF FIGURES

2.1 Performance of a quasi-cyclic IRA (4544,4096) code on the binary-input AWGN channel .................................. 19
2.2 An example of trapping set ................................................. 20
2.3 (a) A simple Tanner graph. (b) A possible 2-cover of (a). (c) Another possible 2-cover. □ = check node. ○ = variable node. .................. 23
2.4 Computation trees of $c_1$ in the original Tanner graph and $c_{11}$ in the cover graph ............................................. 23
2.5 The Margulis code trapping set behavior as a function of number of iterations ...................................................... 26
2.6 An example of oscillating periodic trapping sets of the Margulis code ................................................................. 27
2.7 Trapping set search based on cycle combinations ................. 29
2.8 A trapping set search tree .................................................. 31
2.9 Trapping set search based on tree search: An example based on the (5,4) trapping set ................................................. 32
2.10 Trapping set evaluations for the Margulis code: (12,4) and (14,4) trapping sets ......................................................... 38
2.11 Error-floor predictions for the Margulis code ..................... 39
2.12 System model of ISI channel with AWGN ............................ 40
3.1 Trapping Set Subgraphs of the Margulis and QC Codes .......... 46
3.2 Trees for syndrome-erasure flagging of the bi-mode decoder .... 51
3.3 Distinguishing the two dominant trapping sets of Margulis code in its bi-mode decoder ............................................. 53
3.4 Performance of the bi-mode decoder on the Margulis code ...... 55
3.5 Performance of the bi-mode decoder on the low-rate QC code .... 57
3.6 Performance of the bi-mode partial-soft-erasure decoder on the IEEE 802.3an standard LDPC code .......................... 58
3.7 Pinning one bit removes the entire trapping set iteratively ........ 60
3.8 Performance of BCH-Bit-Pinning decoder for the Margulis code .................. 62
3.9 Performance of BCH-Bit-Pinning decoder for the low-rate QC code .... 64
3.10 System diagram of pre-pinning ............................................. 67
3.11 Trellis pruning for a single pinned bit: (a) PR1, (b) EPR4 ......... 68
3.12 Performance of pre-pinning on the PR1 channel .................... 69
3.13 Performance of pre-pinning on the EPR4 channel .................. 70
3.14 System diagram of post-pinning ........................................... 71
3.15 Performance of post-pinning on PR1 channel ....................... 72
LIST OF FIGURES – Continued

3.16 Performance of post-pinning on EPR4 channel .......................... 73
3.17 An example of generalized graph by combining checks ........ 76
3.18 The Margulis code G-LDPC I grouping ................................. 77
3.19 Performance of G-LDPC decoders on the Margulis code .... 78
3.20 New trapping sets of the Margulis code with G-LDPC I decoder ... 79
3.21 The QC code G-LDPC I grouping ........................................ 80
3.22 New trapping sets of the QC code with G-LDPC I decoder .......... 80
3.23 Performance of G-LDPC decoders on the low-rate QC code .... 81
3.24 The Margulis code G-LDPC III grouping ............................... 84
3.25 The QC Code G-LDPC III grouping .................................... 85
3.26 Performance of G-LDPC decoder III on the high-rate QC code on PR channel ...................................................... 86
3.27 Performance comparison of all decoders on AWGN channel .... 90

4.1 Performance of binary-RS codes with ML decoding on the BEC. .... 98
4.2 ECMA-319 standard data format. ......................................... 99
4.3 Structure of a Dual-Mode ECMA-319 sub data set. .................. 100
4.4 Performance of ECMA-319 with dual-mode decoding. .............. 101
4.5 Structure of a long RS code data set. ..................................... 103
4.6 Performance of long RS data set on AWGN with dual-mode decoding. 104
4.7 Performance of long RS data set on AWGN plus one erased track with dual-mode decoding. ............................. 105
4.8 Structure of data set with a 12-bit RS column code. .................. 106
4.9 Performance of RS/LDPC data set on AWGN with dual-mode decoding. 107
4.10 Performance of RS/LDPC data set on AWGN plus one erased track with dual-mode decoding. ............................. 109

5.1 Reverse concatenation of LDPC and RLL code. ........................... 113
5.2 Performance of the RLL solution on the AWGN channel. .......... 114
5.3 Structure of 50/51 RLL Code. ........................................... 116
5.4 Reverse concatenation of LDPC and DC-free RLL. .................... 117
5.5 Performance of the DC-free RLL solution. ................................ 119

6.1 Continuous-time channel model and receiver front end. ............ 121
6.2 Second-order continuous-time channel model. .......................... 122
6.3 Second-order discrete-time channel model. ............................. 123
6.4 System model with LDPC code. .......................................... 124
6.5 Filter training block diagram. .......................................... 126
6.6 Covariance functions of the LDPC decoder inputs .................... 128
6.7 Performances on PMR channel with 5% jitter noise .................. 130
LIST OF FIGURES – Continued

6.8 Performances on PMR channel with 50% jitter noise ............... 132
6.9 Performances on PMR channel with 95% jitter noise ............... 134
6.10 Correlation functions of detector inputs. ......................... 136
6.11 The impact of various interleavings on $FER_{all}$ .................. 136
6.12 The user density gain of LDPC code over RS code with 95% jitter noise 137


**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Erasure Tree Node Count</td>
<td>51</td>
</tr>
<tr>
<td>3.2</td>
<td>Bit Assignments for BCH Pinning on Margulis Code</td>
<td>61</td>
</tr>
<tr>
<td>3.3</td>
<td>List of dominant trapping set classes of the high-rate QC code</td>
<td>66</td>
</tr>
<tr>
<td>3.4</td>
<td>Trapping Set List of Margulis Code with G-LDPC I</td>
<td>77</td>
</tr>
<tr>
<td>3.5</td>
<td>Trapping Set List of Margulis Code with G-LDPC II</td>
<td>82</td>
</tr>
</tbody>
</table>
ABSTRACT

With recent developments in forward error correction (FEC) schemes, low-density parity check (LDPC) codes have experienced tremendous popularity due to their performance near Shannon’s limits. In this dissertation, several different aspects of LDPC coding and its applications to magnetic storage are investigated. One of the most significant issues that is still not well understood and impedes the use of LDPC codes in many communication and storage systems is the error-rate floor phenomenon associated with their iterative decoders. By delineating the fundamental principles behind the problem, we extend to partial response channels algorithms for predicting the error rate performance in the floor region for the binary-input AWGN channel. We develop three classes of decoding algorithms for mitigating the error floor by directly tackling the cause of the problem: trapping sets. In our experiments, these algorithms provide multiple orders of improvement over conventional decoders at the cost of various implementation complexity increases, from low to moderate.

Product codes are widely used in magnetic and optical recording systems where errors are both isolated and bursty. A dual-mode decoding technique for Reed-Solomon-code-based product codes is proposed, where the second decoding mode involves maximum-likelihood erasure decoding of the binary images of the Reed-Solomon codewords. By exploring a tape storage application, we demonstrate that this dual-mode decoding method dramatically improves the performance of product codes. Moreover, the complexity added by the second decoding mode is manageable. We also show the performance of this technique on a product code which has an LDPC code in the columns.

Run-length-limited (RLL) codes are ubiquitous in today’s disk drives. Using RLL codes has enabled drive designers to pack data very efficiently onto the platter
surface by ensuring stable symbol-timing recovery. We consider a concatenation system design with an LDPC code and an RLL code as components to simultaneously achieve desirable features such as: soft information availability to the LDPC decoder, the preservation of the LDPC code’s structure, and the capability of correcting long erasure bursts.

Finally, we analyze the performance of LDPC-coded magnetic recording channel in the presence of media noise. We employ advanced signal processing for the pattern-dependent-noise-predictive channel detectors, and demonstrate that a gain of over 1 dB or a linear density gain of about 8% relative to a comparable Reed-Solomon is attainable by using an LDPC code.
CHAPTER 1

INTRODUCTION

This dissertation deals with system/algorithms design and performance analysis of low-density parity-check (LDPC) codes applied to the magnetic recording channel.

1.1 Channel Coding Overview

Since the introduction of information theory by C. E. Shannon in 1948 [1], reliable communication over noisy channels had become a major research topic. Shannon’s coding theorem asserts that there exist codes with code rates arbitrarily close to a channel parameter called channel capacity which provide probabilities of error arbitrarily close to zero. However, his theory does not describe how to construct the codes, the encoders, or the decoders. Since then, people have been searching for powerful coding schemes that can achieve what is promised by Shannon’s theorem. Fifty years later, codes for the Gaussian channel have been discovered which come close to the fundamental limits. These breakthroughs have greatly influenced the advancement in many application areas, such as wireless communication, magnetic data storage, deep space communications, etc.

Of all practical error correction methods known to date, turbo codes and low-density parity-check codes, both of which can be viewed as iteratively decodable codes, come closest to approaching the Shannon limit. LDPC code was first invented by Gallager in 1960 [2]. Although revolutionary, these codes were forgotten for a long time, mainly because practical implementations of iterative decoding algorithms associated with these codes were not possible at that time. Throughout the 1970’s and 1980’s, the most widespread technique combined Reed-Solomon codes with Viterbi-decoded short constraint convolutional codes. Then, in 1993, the discovery of turbo codes by Berrou, Glavieux, and Thitimajshima [3] introduced the
principle of turbo decoding, which iteratively exchanges information between elementary decoders. The researchers of [3] and others have shown that long turbo codes easily perform within 1 dB of the Shannon’s capacity limits on a number of channels. The turbo concept is now applied to block codes as well as other parts of a digital transmission system, such as detection, demodulation and equalization.

Shortly after the invention of turbo codes, the rediscovery of LDPC codes by MacKay and Neal [4], and the work of Wiberg, Loeliger, and Koetter [5] on codes on graphs and message-passing iterative decoding initiated a flurry of research on LDPC codes, which continues to these days. LDPC codes can easily exceed the performance of turbo codes when long codewords are allowed. Furthermore, LDPC codes naturally admit parallel decoding architectures, offering much higher speed that turbo codes.

1.2 Research Environment Surrounding This Dissertation

Many properties of LDPC codes have been well studied by coding theorists during the past 12 years. For the case of infinite-length codes, the so-called density evolution technique was first introduced by Luby et al. [7] for the binary erasure channel and then by Richardson, Shokrollahi, and Urbanke [8] [9] for more general channels. These results give valuable insights into how to design good long LDPC codes. However, there is no clear connection to the design of moderate-length codes which is actually the interest of most practical systems. Moreover, a very significant problem of finite-length LDPC codes is the error-rate floor phenomenon, which is not considered by the density evolution technique at all.

The error-floor problem is characterized as a sudden change of the slope in a code’s performance curve in the very high signal-to-noise ratio (SNR) region. In [14], Koetter and Vontobel derived a framework for the finite-length analysis of iterative decoders of LDPC codes by introducing the concept of graph-cover decoding and fundamental polytope. They provide a fundamental understanding and intuition behind the nature of iterative decoders and the cause of error floors. However, there
is still no systematic method to characterize the performance, especially at high SNRs at which many systems operate and where coding performance is difficult to evaluate with simulations.

Magnetic data storage, including disk and tape drives, represents one application for which the error-floor problem would preclude the use of LDPC codes. Magnetic recording systems can be viewed as a special kind of communication channel, where the communication is conducted over time instead of over space, hence the name magnetic recording channel (MRC). For many years, RS codes have been the error-correction code of choice, for its relatively low complexity, its maximum minimum-distance property, and its ability to correct bursts. Recently, LDPC codes together with their iterative decoders have been considered for state-of-the-art disk drives, which provide substantial performance gain over RS codes. Although MRC allows longer (typically better) LDPC codes than some other applications, this channel introduces special challenges to system designers, such as severe intersymbol interference, colored pattern-dependent noise, and a requirement for a very low sector error rate. While LDPC codes have been widely acclaimed for their near-capacity performance, these issues have to be properly addressed in order to deliver a stable channel solution. In the light of this great variety of requirements, we make great strides toward analyzing and solving the error floor problem. We also consider customized system designs for tape storage, concatenation problem of an LDPC code with a runlength limited (RLL) code, and the estimation of the density increase that LDPC codes provide relative to RS codes.

1.3 Dissertation Outline

This dissertation is organized as follows. Chapter 2 introduces the fundamentals of error floor phenomenon, particularly those that will be used in later chapters of this dissertation. An effective two-step procedure for predicting the performance in the very high SNR region is also presented for both memoryless channels and channels with memory. Equipped with the results in Chapter 2, Chapter 3 proposes three
general classes of iterative decoding algorithms to lower the error floor by directly
tackling the trapping sets that lead to the floor. The effectiveness of these meth-
ods are demonstrated by simulations and the semi-analytic technique introduced in
Chapter 2. In Chapter 4, we propose a dual-mode decoding technique for product
codes and explore its application to tape storage. We study the very practical issue
of concatenating an LDPC code with an RLL code in Chapter 5 with the goal of
achieving several desirable features simultaneously. In Chapter 6, we examine the
applicability of LDPC codes to magnetic storage via computer simulations using
a perpendicular recording model with pattern-dependent media noise. The perfor-
manence improvement in terms of both SNR gain and user density gain is investigated.
Finally, Chapter 7 provides some conclusions of this dissertation and suggestions for
future work.
CHAPTER 2

ERROR FLOORS: THE CAUSE AND LEVEL PREDICTION

Since the discovery and rediscovery of low-density parity-check (LDPC) codes [2][4], they have been intensely researched by coding theorists and practitioners over the past decade due to their near-capacity performance when decoded with moderate-complexity iterative decoders. Many of their properties have been well studied, yet the error-rate floor phenomenon has remained an open problem, and is one of the most significant impediments to the use of LDPC codes in many communication and storage systems. This phenomenon can be characterized as an abrupt decrease in the slope of a code’s performance curve from the moderate-SNR waterfall region to the high-SNR floor region [10]. Since many systems such as data storage devices and optical communication systems require extremely low error rates, such error floors become the major factor in limiting the deployment of LDPC codes in products, and solving the error floor problem has been a critical issue during the past decade. Exploring the error floors for practical LDPC codes is very difficult because software simulations at very low error rates take months to collect a sufficient number of frame errors for a confident estimation. In this chapter, we discuss the cause of the error floor problem by exploring properties of iterative decoders. We then present a practical semi-analytical method for estimating the floor level which exploits the importance sampling (IS) theorem [10]. The result due to Richardson [10] is explained in greater detail in this chapter than in [10], and is then generalized to partial response channels for later application in Chapter 3.

2.1 The Error Floor Phenomenon and Trapping Sets

The error floor phenomenon was first described by MacKay and Postol in [11] as a sudden slope change of the error rate performance curve at high SNRs. For example,
Figure 2.1: Performance of a quasi-cyclic IRA (4544,4096) code on the binary-input AWGN channel.
shown in Fig. 2.1 is the error rate performance of a rate-0.9 (4544, 4096) quasi-cyclic (QC) irregular-repeat-accumulate (IRA) LDPC code proposed in [31]. This change of slope was attributed to the existence of low-weight near-codewords rather than low-weight codewords. In 2003, Richardson [10] introduced the notion of trapping sets to describe configurations in the Tanner graphs of codes that cause failures of specific decoding schemes. They discovered that performance of LDPC codes with message-passing decoding in the low error-rate region is determined by the count and structure of the low-weight near-codewords, or trapping sets, rather than the minimum distance of the code.

**Definition 2.1.1.** A \((w, v)\) near-codeword is a binary word of the same length of the code, with weight \(w\) and resulting in \(v\) unsatisfied checks.

**Definition 2.1.2.** A \((w, v)\) trapping set is a set of \(w\) variable nodes (VNs) which induces a subgraph with \(v\) odd-degree check nodes (i.e., \(v\) unsatisfied checks) and an arbitrary number of even-degree check nodes (CNs).

Note the support of a \((w, v)\) near-codeword is exactly a \((w, v)\) trapping set. Fig. 2.2 shows an example of a \((5,4)\) trapping set which has 8 even-degree check nodes which we call mis-satisfied check nodes. The definition of trapping set is very generic. From the view of the code itself, any subgraph of the Tanner graph may be a trapping set depending on the channel and decoding algorithms, and hence
there are $2^n - 1$ different trapping sets in a code of length $n$. Note that we can view the subgraph induced by a codeword as a special trapping set with $v = 0$. Thus any analysis or evaluation method for trapping sets can be applied to codewords for studying the mis-correction rate of the code. On the other hand, not all of them are in the decoding output space; among those which can be a decoder output, not all are of equal importance in terms of the contributions to performance. For example, the trapping sets of the maximum-likelihood (ML) decoder are the non-zero LDPC codewords; for iterative decoding on the binary erasure channel (BEC) a trapping set is called a stopping set, which is defined as a subset of variable nodes whose neighboring checks are connected to the set at least twice. However, there is no simple characterization of trapping sets on the additive white Gaussian noise (AWGN) channel or other more complicated channels. To understand the error floor region, the question of what can and what cannot be a trapping set is not the critical question. What is important is to identify the most relevant trapping sets that dominate the performance in the floor region. This will be one of the main focuses of this chapter. The most relevant trapping sets of a given LDPC code are dependent on the nature of decoding algorithm, decoder scheduling, fixed-point quantization parameters, channel model, number of iterations, etc. Loosely speaking, when $w$ and $v$ are relatively small, errors in each of the $w$ VNs create $v$ parity check failures and tend to lead to situations from which the iterative decoder cannot escape. However the size of the trapping set is not the only relevant parameter. Even for fixed $v$, it sometimes happens that $(w, v)$ trapping sets fail at a higher rate than $(w', v)$ trapping sets with $w \geq w'$. The reason that low-weight trapping sets dominate the error floor region can be explained by understanding the decoders as discussed later in Section 2.2. Throughout the dissertation, to simplify the language, we shall often use “trapping set” when we are referring to the induced subgraph.
2.2 The Weakness of Message-Passing Decoders

Aside from its asymptotically capacity-achieving performance, one of the advantages of LDPC codes is the class of iterative decoding algorithms that are computationally far less demanding than the ML decoder and can be implemented with low complexity high-throughput hardware architectures. However, the behavior of such message-passing decoders for the case of finite-length codes is still not well understood, and simulations are still the only tool available to evaluate the performance. Iterative decoders are susceptible to trapping set problems because they work locally in a distributed-processing fashion to (hopefully) arrive at a globally optimum decision. Of course, iterative decoders are vulnerable to cycles in the graph on which the decoder is based, for cycles often form message-passing loops and lead the iterative decoder away from the maximum-likelihood codeword. Furthermore, trapping sets are unions of several cycles.

In [14], Koetter and Vontobel derived a framework for the finite-length analysis of iterative decoders of LDPC codes by introducing the concept of graph-cover decoding and fundamental polytope. Consider the simple Tanner graph shown in Fig. 2.3(a), which corresponds to a repetition code of length $n = 2$ with two codewords $C = \{(00), (11)\}$. Any $x$-covers of the code can be obtained by a copy-and-permute procedure: the base Tanner graph is copied $x$ times, and then the edges of the individual replicas are permuted among the replicas. Fig. 2.3(b) and (c) depict two examples out of 16 possible 2-covers of the Tanner graph, and the union of the codewords that satisfy the 2-covers is $C_{(2)} = \{(0000), (1111), (0101), (1010), (1001), (0110)\}$. Since message-passing algorithms are graph-based algorithms, we can visualize the decoding process by looking at the computation tree generated from one particular check node (say $c_1$ and $c_{11}$). It can be easily seen (Fig. 2.4) that any locally operating decoder cannot distinguish whether it is operating on the computation tree of $c_1$ or $c_{11}$ because the two trees share the same local graphical structure (similarly for $c_2$ and $c_{22}$). Consequently, the iterative decoder cannot determine whether it is acting on the Tanner graph of
the code itself or on some finite cover of the graph.

In this example, the correspondence of the codewords (0000) and (1111) in $C_{\{2\}}$ to the two codewords in $C$ can be established by considering liftings (repetition of the codewords of $C$). However, the other 4 codewords of $C_{\{2\}}$ together with many other codewords of finite-covers represent pseudo codewords of the original Tanner graph. The decoder decodes to the “pseudo signal” that has the highest correlation with the channel output, and the set of pseudo-codewords (related to trapping sets) from all finite graph covers compete with the transmitted codeword for being the “best” solution. In other words, the major difference between iterative decoders and the ML decoder is that the message-passing decoders base their decisions by gathering
information locally, whereas the ML decoder searches for the optimal solution by using global information and check constraints. This causes the performance gap between the two type of decoders. Obviously, the ML decision space (the set of non-zero codewords) is a subspace of the iterative decoder decision space (the set of codewords and near-codewords). This is the intuition behind one of the low-floor decoder techniques proposed in Chapter 3.

A dominant trapping set sub-graph is an “isolated” aggregation of cycles, lacking sufficient extrinsic information for the decoder to break from it. The failure mechanism can be explained as follows. As shown in Fig. 2.2, a trapping set typically has a larger number of mis-satisfied check nodes than unsatisfied check nodes. In the initial stage of message-passing decoding, due to the presence of certain noise samples, variable nodes internal to one particular trapping set experience a large increase in their reliability estimates while possessing incorrect bit values (signs). Further, this information gets propagated to other variable nodes in the trapping set, some of which already have unreliable bit estimates themselves. After this initial biasing, external variables outside of the trapping set usually start to correct their initially incorrect estimates. However, by that time the variable nodes in the trapping set have already significantly biased their decisions toward the wrong values. Since there are very few unsatisfied check nodes capable of detecting errors within trapping sets, and the extrinsic information to the internal variable nodes are not strong enough to revoke the decisions of biased messages from the mis-satisfied checks, this erroneous information persists in the decoder graph until decoding ceases.

### 2.3 Definitions Related to Trapping Sets

Now we introduce some important definitions related to trapping sets.

**Definition 2.3.1.** An elementary \((w, v)\) trapping set is a trapping set for which all check nodes in the induced sub-graph have either degree one or degree two.

Based on extensive computer simulations reported in literature, it was observed that the majority of trapping sets that exhibit a strong influence on the error-floor
level are of the elementary form. Furthermore, although check nodes of odd degree larger than one are possible, they seem to be very unlikely within small trapping sets. This fact will be used for designing low error floor decoders in the next chapter.

**Definition 2.3.2.** Trapping set characterization in terms of error pattern behaviors w.r.t to iteration numbers.

1. A *Stable Trapping Set* (also called *fixed-point trapping set*) is a trapping set responsible for all erroneous decisions at the end of the decoding process, and from which the decoder cannot escape with increased number of iterations.

2. A *Periodic Oscillating Trapping Set* exhibits a periodic pattern that repeats as decoding iterations continue.

3. An *Aperiodic Oscillating Trapping Set* contains error patterns from which erroneous messages propagate to a number of external variables and exhibit a random pattern.

The behavior of each trapping set class is shown in Fig. 2.5. Stable trapping sets can be described as absorbing states, which are the most harmful ones to the error floor performance because they are more likely to cause decoding failure in the high SNR region. Oscillating trapping set behavior is attributed to the dynamics of the message exchange in which a small number of VNs propagate incorrect messages through their neighboring unsatisfied checks. As the result, these make some of their other neighboring VNs admit incorrect values, which are propagated further to more VNs. As the number of incorrect VNs increases, so does the number of neighboring checks, and eventually there is a sufficient number of unsatisfied checks to enforce the correct values, thus the number of erroneous VNs starts to decrease until a certain point when the incorrect messages of a small number of VNs start to propagate again.

We have observed from extensive simulations that some stable trapping sets may exhibit oscillating behavior. For example, as depicted in Fig. 2.6, the fixed-point (12,4) trapping sets of the Margulis code can show up as two oscillating (6,18)
Figure 2.5: The Margulis code trapping set behavior as a function of number of iterations.
trapping sets with period 2, whose variable nodes have no intersection and the union of the two (6, 18) trapping sets form the (12, 4) trapping set. Such periodic trapping sets are also important since they can be viewed as alternative appearances of stable trapping sets. In the rest of this section, we focus on searching for dominant trapping sets on channels such as the AWGN channel and certain partial response (PR) channels, where no simple characterization exists. We show how to use these trapping sets to predict the error floor level of any given LDPC code.

2.4 Trapping Set Enumeration

Finding the minimum distance of LDPC codes, or in other words the smallest trapping set of ML decoders, which characterizes the ML decoding performance, is in general an NP-hard problem. For iterative decoders, there is no analytical method for characterizing the performance, especially at high SNR where many systems operate and where coding performance is difficult to estimate because of the diminishingly small number of errors. Unlike the case of ML decoding or iterative decoding on the BEC, where the distribution of trapping sets within ensembles of graphs can be analyzed using combinatorial methods, trapping sets of message-passing decoders on channels other than the BEC depend not only on the graphical structure of the code, but also on many other system implementation factors. All of these make the
search for dominant trapping sets even harder. Furthermore, though combinatorial characterization on the code’s shortest cycles to find some small trapping sets is possible, it becomes inefficient for trapping sets with \( w \geq 8 \) on a code with girth 6 or higher, which could still be dominant and the inaccurate enumeration of which may lead to less accurate predictions. Therefore, a combinatorial solution becomes a hard problem, and empirical or semi-empirical methods are the only choices for trapping set enumeration.

The existing methods in the literature for finding trapping sets can be divided into two categories: one is by graph search and heuristics, such as in [10][15][32] where trapping sets can be identified using software or FPGA hardware simulations; the other category is an importance sampling procedure which uses multi-bit and multi-level deterministic error impulse injection [19][20]. Although the simulation method is the most straight-forward solution which captures the system condition completely and the error patterns collected will most likely be the dominant trapping sets, to collect a nearly complete list of all the dominant trapping sets through simulations is practically unrealistic for some well-designed LDPC codes. On the other hand, trapping set-induced subgraphs typically possess certain combinatorial properties of the shortest cycles of a code, hence some graph search algorithms are usually used together with simulations for such enumeration tasks, especially for the class of quasi-cyclic LDPC (QC-LDPC) codes which possess a certain level of isomorphism that can simplify the graph search algorithms. For example in [32], several dominant trapping sets were first observed through simulations, and then the code’s Tanner graph was searched for these known configurations and thus the multiplicities of these trapping sets were obtained.

In 2002, an error-impulse method was proposed in [16] for computing minimum distances based on the ability of the soft-input decoder to overcome error impulse input patterns. In [17], Hu et al. proposed a similar randomized algorithm to tackle the minimum distance problem, which searches codewords locally around the all-zero codeword perturbed by noise, anticipating that the resultant nearest nonzero codewords will most likely contain the minimum-Hamming-weight codeword whose
Hamming weight is equal to the minimum distance of the linear code. These methods have been generalized to the case of trapping set search.

For example, the trapping set search algorithm in [19] combines a graph search technique with importance sampling simulations (Fig. 2.7): for any given bit of the LDPC code, first, the Tanner graph is searched for all of the shortest cycles that involve this bit (called the root bit); secondly, the union of the bits of these cycles is formed to obtain a set of bits which must contain all of the bits that can form any trapping set related to the root bit (this is because trapping sets are formed by overlapping short cycles); thirdly, an error-impulse of magnitude two is injected to each bit in the set, which essentially flips the signs of these bipolar (±1) bits; lastly, run a fixed number of decoding iterations with such biased cycles and observe the resulting trapping sets in the decoding algorithm. This method was justified in [19] on a short MacKay code and a rate-0.5 length-2000 code. However, the effectiveness of this technique for high-rate practical-length codes which have
much larger number of shortest cycles associated with each bit is diminished because the set of error impulses is so large that the iterative decoder is overwhelmed and converges to random, long error events.

Another method using a similar principle was proposed by Cole et al. in [20]. The idea is to search the graph locally by constructing a tree with a root variable node from which all edges descend. And then the variable nodes on different tiers of the tree are given different levels of error impulses as illustrated in Fig. 2.8. For example, consider a regular-\((d_v, d_c)\) LDPC code with \(d_v \geq 3\) and girth \(g \geq 6\). There are \(d_v\) checks adjacent to the root variable node, each of which has \(d_c - 1\) variable nodes other than the root node. To construct the tree, \(N_{TS}\) out of \(d_v\) checks are picked for the first layer of the check nodes, and then for each check only one of the \(d_c - 1\) variable nodes will be picked at a time to form the second tier of variable nodes on the tree. Hence there are \(N_{TS}\) variable nodes on the second layer. Since the girth of the code exceeds 4, the second tier of variable nodes will be distinct, and there are totally \(\binom{d_v}{N_{TS}}(d_c - 1)^{N_{TS}}\) possible trees and, hence, possible multi-bit impulses for each root. \(N_{TS}\) is typically set to two or three based on the observation on trapping set subgraphs that there exists at least one bit that is adjacent to two or three mis-satisfied checks which are included in the tree. The rest of the tree is constructed by including all descendants from tier-2. Bits on the first and second tiers are biased with additive impulses \(\varepsilon_1\); bits on the third layer are biased with \(\varepsilon_2\) and all other bits are biased with a multiplicative scalar \(\gamma \in [0, 1]\). The biased sequence is then passed through the decoder to observe the trapping set patterns. Fig. 2.9 illustrates how this method works for the \((5, 4)\) trapping set example.

This multi-bit, multi-level impulse method is an exhaustive search procedure which searches through a large number of impulses for each VN (column in the \(H\) matrix), and then repeats the procedure for every variable node. For example, suppose we have a length-4544 QC-LDPC code with circulant size \(Q = 64\), \(d_v = 5\) and \(d_c = 50\). If we set \(N_{TS} = 3\), then there are 1,176,490 different error impulses. With the isomorphism property of QC-LDPC codes taken into account, the total number of LDPC decodings is \(N/Q \times \binom{d_v}{N_{TS}}(d_c - 1)^{N_{TS}} = 83,530,790\). The levels
Figure 2.8: A trapping set search tree.
Figure 2.9: Trapping set search based on tree search: An example based on the (5,4) trapping set. $\bullet$ = Root VN (tier-1). $\bigcirc$ = VNs. $\square$ = mis-satisfied CNs. $\blacksquare$ = unsatisfied CNs. $N_{TS} = 4$. Numbers on the VNs are tier indices. Within the trapping sets, four out of five VNs are biased with error impulse $\varepsilon_1$ and one is biased with impulse $\varepsilon_2$. 
of error impulses ($\varepsilon_1$, $\varepsilon_2$ and $\gamma$) have to be chosen heuristically for a given LDPC code such that the decoder is sensitive to decoding failures. Consequently the list of decoding error patterns collected could be very large, which serves as a list of candidate trapping sets whose importance will be evaluated using importance sampling as discussed further in Section 2.5 or some simplified ranking procedure [20]. The advantage of this method is the flexibility with respect to different code lengths and code rates. Furthermore, the number of decodings required is still manageable because the error impulses are deterministic, therefore no Monte-Carlo simulations are required.

These error impulse methods can be customized for the search of trapping sets on many unconventional decoding algorithms such as the ones in Chapter 3 that improve the error floor by explicitly targeting the trapping sets of a conventional iterative decoder. For example, some post-processing techniques for lowering the floor are only executed when the normal decoding gets trapped in a trapping set error event. Thus their efficiency can be characterized as the impact of these techniques on the trapping sets. By applying error impulses on the trapping sets of the normal decoder and observing how the post-processing decoders eliminate/improve trapping set error events, one can quantify how many orders of improvement that the post-processing solution can provide. A two-step importance sampling procedure for the error floor prediction of the so-called generalized LDPC decoders is described in A.

### 2.5 Trapping Set Evaluation

In [10], Richardson proposed a semi-analytical method to predict the performance of LDPC codes in the floor region. In his method, it is assumed that a nearly complete list $T$ of the dominant trapping set candidates was found first, which can be achieved by using the methods described in Section 2.4. Then, the decoder failure is decomposed into the summation of the effects from the individual trapping sets
to obtain a lower bound of the frame error rate (FER):

$$\text{FER} = \Pr \left\{ \bigcup_{T} \xi_T \right\} = \sum_{T} \Pr\{\xi_T\} \geq \sum_{T \in T} \Pr\{\xi_T\}, \quad (2.1)$$

where $\xi_T$ denotes the set of decoder inputs that cause a failure on a particular trapping set $T$, i.e., all VNs in $T$ are in error and no VNs outside $T$ are in error. Note that $\bigcup_T \xi_T$ represents all decoder inputs that cause any decoding failure, and the second equality of Equation (2.1) is due to the fact that the partitions of the decoder input space according to decoding outcomes are disjoint. In the very high SNR region, most of the error events are due to trapping set error events. For example, for the Margulis code, two trapping set classes contribute to 98% of the error floor performance. Therefore, the lower bound is tight using these two trapping sets and can be used to predict the error floor level. Here, a *trapping set class* is defined as the set of isomorphic trapping sets which contribute equally to the error floor performance.

Next, the contribution of each trapping set class to the FER is evaluated by using an importance sampling method [18][19][20] which biases the received channel realizations (decoder inputs) in a manner that produces errors events more frequently, thus facilitating the error rate measurement. The biased measurement $\Pr\{\xi_T|B\}$ is converted to the unbiased measurement $\Pr\{\xi_T\}$ according to:

$$\Pr\{\xi_T\} = E_B[\Pr\{\xi_T|B\}] = \int_{-\infty}^{\infty} \Pr\{\xi_T|B=b\} f(B=b) \, db \quad (2.2)$$

where $B$ is the random variable with pdf $f(B=b)$ that controls noise biasing, $E_B$ represents the expectation over $B$ and the conditional error rate $\Pr\{\xi_T|B=b\}$ is obtained by simulations. Since $\Pr\{\xi_T|B=b\} \gg \Pr\{\xi_T\}$, the conditional error rate is more easily measured, and thus large simulation speed gain is achieved.

Note that the importance sampling here is a statistical biasing in contrast to the deterministic biasing in Section 2.4. Obviously, the most important question
regarding the application of the importance sampling procedure to estimate the low error rate performance is: how to provoke more frequent decoder failures to achieve higher simulation gains through statistically known noise biasing. Sections 2.5.1 and 2.5.2 answer this question for channels without and with memory, respectively.

2.5.1 IS for Memoryless Channels

Before generalizing the IS method to channels with memory, we first review the case for memoryless channels by focusing on the binary-input AWGN channel. Consider an LDPC code of length $N$. Suppose the channel noise sequence is denoted by $n_1, n_2, \cdots, n_N$, which are i.i.d. $\mathcal{N}(0, \sigma^2)$ Gaussian random variables, and the channel inputs are equi-probable binary ($\pm 1$) sequences. Without loss of generality, we can assume the all-zeros codeword is transmitted. Then the scaled channel outputs are given by $y_i = \frac{2}{\sigma^2}(1 + n_i)$, $i = 1, \cdots, N$. Consider a trapping set $T$ with $w$ variable nodes having bit indices $I_T = (i_1, i_2, \cdots, i_w)$ and denote all indices outside of $T$ by $I_T^\complement$. Then partition the $N$-tuple noise vector into two subvectors: trapping set-related noise, $n_T = (n_i, i \in I_T)$, and trapping set-unrelated noise, $n_T^\complement = (n_i, i \notin I_T)$. Therefore, the probability of a failure due to this trapping set can be written as:

$$
\Pr\{\xi_T\} = \int_{-\infty}^{\infty} \int \Pr\{\xi_T|n_1 \cdots n_N\} \Pr\{n_1 \cdots n_N\} dn_1 \cdots dn_N
= \int_{n_T} \int \Pr\{\xi_T|n_T, n_T^\complement\} \Pr\{n_T\} \Pr\{n_T^\complement\} dn_T dn_T^\complement
= \int_{n_T} \Pr\{n_T^\complement\} \int \Pr\{\xi_T|n_T^\complement, S\} \Pr\{n_T^\complement|S\} \Pr\{S\} \Pr\{S\} dn_T^\complement dS
= \int_{n_T} \Pr\{n_T^\complement\} \int \Pr\{\xi_T|n_T^\complement, S\} \Pr\{S\} dS dn_T^\complement,
$$

(2.3)

where $S$ is the random variable that controls the biasing particularly on $n_T$. This is due to the observation through extensive simulations that the failure rate on one particular trapping set is strongly dependent on $n_T$, and therefore only the noise
samples related to the trapping sets are biased to introduce more frequent trapping set failures. Hence, \( \Pr\{\mathbf{n}_T|\mathbf{S}\} = \Pr\{\mathbf{n}_T\} \), \( \Pr\{\xi_T|\mathbf{n}_T, \mathbf{S}\} = \Pr\{\xi_T|\mathbf{n}_T, \mathbf{S}\} \), from which the fourth line of Equation (2.3) arises. Now, we show how to select a single, easy-to-control random variable \( \mathbf{S} \) to bias the vector \( \mathbf{n}_T \) effectively.

Intuitively, for the trapping set-related samples, it is desirable to have a biased noise sample \( n' \) which translates the channel output \( y = \frac{2}{\sigma^2}(1 + n) \) into \( y' = \frac{2}{\sigma^2}(1 + n') \sim \frac{2}{\sigma^2}(-1 + \delta) \), leading to a likely bit error. This suggests the biased noise samples should be in the direction of \((-2, -2, \ldots, -2)\) for vector \( \mathbf{n}_T \).

Now, consider the operation of changing from the standard basis \( (\alpha_1, \ldots, \alpha_w) \) to a new orthonormal basis \( (\gamma_1, \ldots, \gamma_w) \) for \( \mathbf{n}_T \in \mathbb{R}^w \), where we fix the first basis vector to \( \gamma_1 = (-2, \ldots, -2) / \sqrt{4w} = (-1, \ldots, -1) / \sqrt{w} \), i.e., the normalized trapping set direction vector. The purpose here is to impose various levels of biasing in the direction that is sensitive to trapping set-induced error events, through adjusting the coordinate of \( \gamma_1 \), and then observe the decoding behavior. The rest of the \( w - 1 \) new basis vectors can be calculated using the Gram-Schmidt process to orthogonalize the set of vectors \( \{\gamma_1, \alpha_2, \alpha_3, \ldots, \alpha_w\} \). Thus, we express the subvector \( \mathbf{n}_T = (n_{i_1}, \ldots, n_{i_w}) = n_{i_1}\alpha_1 + n_{i_2}\alpha_2 + \cdots + n_{i_w}\alpha_w \) as \( b_1\gamma_1 + b_2\gamma_2 + \cdots + b_w\gamma_w \), where \( b_1, \ldots, b_w \) are obtained by projecting \( \mathbf{n}_T \) onto the new basis. Further, the first coordinate \( b_1 \) is substituted by a parameter \( b \) for controlling different levels of noise biasing, whose values are drawn according to the random variable \( \mathbf{S} \). Then we have \( \mathbf{n}_T' = b\gamma_1 + b_2\gamma_2 + \cdots + b_w\gamma_w = \frac{b}{\sqrt{w}}(-1, \ldots, -1) + b_2\gamma_2 + \cdots + b_w\gamma_w \). Since \( \mathbf{n}_T \) is a Gaussian vector with zero mean and covariance matrix \( \sigma^2 I_{w \times w} \), \( \mathbf{S} \) has Gaussian distribution:

\[
\Pr\{\mathbf{S} = b\} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{b^2}{2\sigma^2}} \quad (2.4)
\]

Then we have

\[
\Pr\{\xi_T\} = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} \sum_{\mathbf{n}_T} \Pr\{\mathbf{n}_T\} \Pr\{\xi_T|\mathbf{n}_T, \mathbf{S} = b\} e^{-\frac{b^2}{2\sigma^2}} db
\]

\[
= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} \Pr\{\xi_T|\mathbf{S} = b\} e^{-\frac{b^2}{2\sigma^2}} db. \quad (2.5)
\]
Thus, during the Monte-Carlo simulations used to estimate $\Pr\{\xi_T|S = b\}$, $n_T$ is substituted by $n_T'$ according to the value of $b$, and $n_T$ is the normal Gaussian vector $\mathcal{N}(0, \sigma^2)$.

Fig. 2.10 summarizes the error floor evaluation at SNR=2.8 dB of a (12, 4) trapping set (solid lines) and a (14, 4) trapping set (dashed lines) in the graph of the Margulis code. The conditional error rate ($\Pr\{\xi_T|S = b\}$) curves were simulated with different levels of noise biasing $b$. The distribution of $S = b$ is the Gaussian $\mathcal{N}(0, \sigma^2)$, and the product of the conditional error rate and the Gaussian pdf is integrated to obtain the contributed error rate of the trapping set being examined. Finally, the overall FER at 2.8 dB is the summation over all trapping set classes of the contributed rates multiplied by the multiplicity of each trapping set class, which is 1320 for both the (12,4) and the (14,4) classes for this code.

The FER prediction curve is shown in Fig. 2.11, where the two predicted data points are indicated by black dots. In fact, the curve can be extrapolated from one data point [10] by making $\sigma$ vary locally such that the simulated conditional error rate changes relatively little in comparison with the changes of the biasing variable pdf. We remark that the prediction curve closely matches the direct simulation.

2.5.2 IS for Channels with Memory

In this section, a discrete-time intersymbol interference (ISI) channel with i.i.d. uniform binary inputs and AWGN is considered. Partial response channels are widely used in data storage systems to increase the recording density by controlling the ISI to a desired target response known as the partial response target. Shown in Fig. 2.12 is a simple LDPC-coded PR channel, where the input sequence $u_k$ is first encoded and mapped to a bi-polar sequence $a_k \in \{\pm 1\}$. This sequence is then convolved with the channel PR target $h(D) = 1 + h_1D + h_2D^2 + \cdots + h_LD^L$. Note that we assume the channel is perfectly equalized to the PR target, and hence the channel output $r_k$ is equivalent to the ideal PR response sequence $y_k$ plus the additive white Gaussian noise $n_k$. Here $L$ is the memory length the channel. A channel detector (typically BCJR or soft-output Viterbi detector) is necessary to
Figure 2.10: Trapping set evaluations for the Margulis code: (12,4) and (14,4) trapping sets.
Figure 2.11: Error-floor predictions for the Margulis code on the AWGN channel and the EPR4 channel.
Figure 2.12: System model of ISI channel with AWGN.

calculate soft information of each bit for the iterative LDPC decoder.

Similar to the case of memoryless channels, in order to predict the error floor level, a list of dominant trapping set is obtained first, and then the decoding failure rate in the error floor region is estimated by Equation (2.1). However on PR channels, simulation using the all-zeros codeword is not sufficient to fully represent the system performance, so random binary input sequences are used. In the effort to derive the IS solution for such channels with memory, the key question is again how to identify and bias the most relevant noise samples to cause trapping set failures efficiently. Suppose the trapping set being evaluated has \( w \) variable nodes: \( v_{i_1}, v_{i_2}, \cdots, v_{i_w} \). Due to the memory of the channel, for one variable node \( v_{i_j} \), the noise samples that determine the output of the detector at the time instance \( i_j \) are \( n_{i_j}, n_{i_j+1}, \cdots, n_{i_j+L} \). Thus, the set of noise samples related to the LDPC decoder inputs corresponding to all \( w \) trapping set variable nodes is \( n_T = \{n_{i_1}, n_{i_1+1}, \cdots, n_{i_1+L}, n_{i_2}, n_{i_2+1}, \cdots, n_{i_2+L}, \cdots, n_{i_w}, n_{i_w+1}, \cdots, n_{i_w+L}\} \), which has up to \((L+1)w\) samples. Further, since the \( n_k \)’s are i.i.d. Gaussian random variables, \( n_T \) can be isolated from all other irrelevant samples, which we again denote by \( n_T \). Thus, Equation (2.3) is still valid for such channels.

Now we want to extract from the vector \( n_T \) a simple biasing parameter \( S \) that affects the failure rate strongly. Let’s reconsider the system diagram in Fig. 2.12. From the LDPC code’s point of view, the dashed block in the figure can be treated as a certain noise generating mechanism. The goal is to bias \( n_T \) such that the equivalent noise to the LDPC code forces the decoder to converge to a trapping set-induced error event \( \hat{a} \) (\( N \)-bit word) instead of the codeword transmitted, \( a \). Let \( y = \)
(y_1, \cdots, y_N) where y_k = \sum_{i=0}^{L} h_i a_{k-i}, and \hat{y} = (\hat{y}_1, \cdots, \hat{y}_N) where \hat{y}_k = \sum_{i=0}^{L} h_i \hat{a}_{k-i}.

Thus, the Euclidean distance between the two vectors in the channel trellis on which the detector operates can be expressed as \( D = \| \hat{y} - y \| = \| \Delta \| \). Evidently, the vector \( \Delta \) indicates the direction toward which the noise vector \((n_1, n_2, \cdots, n_N)\) should be biased. Note that \( \Delta \) has \( w_T \leq (L + 1)w \) non-zero entries with indices in \( I_T = \{i_1, i_1 + 1, \cdots, i_1 + L, \cdots, i_w, i_w + 1, \cdots, i_w + L\} \). Hence, the subvector \( \Delta_T = (\Delta_i | i \in I_T) \) is the direction to bias \( n_T \). Given these, apply the same change of basis procedure for \( n_T \in \mathbb{R}^{w_T} \) as in Section 2.5.1 with the first new basis vector \( \gamma_1 = \Delta_T / \| \Delta_T \| \). Therefore \( n_T \) is biased according to a random variable \( S \), where \( \Pr\{S = b\} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{b^2}{2\sigma^2}} \), such that:

\[
\mathbf{n}'_T = b \gamma_1 + b_2 \gamma_2 + \cdots + b_{w_T} \gamma_{w_T} = \frac{b}{\| \Delta_T \|} \Delta_T + b_2 \gamma_2 + \cdots + b_{w_T} \gamma_{w_T}. \tag{2.6}
\]

To demonstrate the IS floor estimation method, we give an example of the Margulis code on the EPR4 channel for which \( h(D) = 1 + D - D^2 - D^3 \). Let’s consider a (12, 4) trapping set whose variable nodes are separated by at least the length of the PR target \((L + 1 = 4)\). Assume a codeword \( a = (a_k | k = 1, 2, \cdots, N, a_k \in \{\pm1\}) \) is sent over the EPR4 channel, and then the 12-tuple subvector containing all bits in the trapping set is given by \((a_{i_1}, a_{i_2}, \cdots, a_{i_{12}})\). In this case, we have \( w_{T_{i_2}} = 48, \Delta_{T_{i_2}} = (-2a_{i_1}[1, 1, -1, -1], -2a_{i_2}[1, 1, -1, -1], \cdots, -2a_{i_{12}}[1, 1, -1, -1]) \) and \( \| \Delta_{T_{i_2}} \| = 8\sqrt{3} \). The conditional error rate is simulated against the biasing parameter \( b \) in the interval \((0, 8\sqrt{3}]\). Fig. 2.11 presents the error rate curves from direct simulation and IS prediction, which match with each other closely.

We remark that, unlike \([21]\), our derivation of the IS solution does not rely on the assumption of separated trapping set bit locations. Note also that this assumption does not hold for all (12, 4) trapping sets of the Margulis code.

### 2.6 Conclusions

In this chapter, we first introduced the error floor phenomenon and its fundamental cause, and then the trapping set behaviors were comprehensively interpreted. A
two-step performance prediction procedure through trapping set enumeration and evaluation was described systematically for channels with or without memory. We also presented results for AWGN channel and the partial response channel, which prove that the IS method is very efficient in evaluating the low error rate performance that is out of the reach of conventional Monte-Carlo simulations. We remark that this method can be further generalized to channels with pattern-dependent media noise, which is a more realistic model for the magnetic recording systems.
CHAPTER 3

LOW-FLOOR DECODERS FOR LDPC CODES

In an effort to solve or mitigate the floor problem, many attempts have been made by designing LDPC codes with low floors ([23]-[34]). However, code design techniques such as cycle optimization and trapping set optimization have not been very successful in terms of eliminating girth $g \geq 6$ cycles and weight $w \geq 10$ trapping sets for practical length/rate code. Thus the performance gain in the error floor region has been limited. On another hand, only a few papers focus on decoder-based strategies for lowering floors. For example, [35] and [36] attempt to lower the floors of LDPC codes by modifying the scheduling of the decoders. Reference [38] proposes an averaged decoding algorithm which decreases the frequency of trapping set error events by averaging the messages over several iterations to prevent sudden change of the magnitude of a VN’s reliability. In [39], the floors are reduced by utilizing linear programming-based decoding. In [37] a post-processor is proposed which exploits the knowledge of the deleterious subgraphs (trapping sets) for a given LDPC code, and lowers the floor by inverting the bits of a known trapping set after the decoder becomes trapped in that trapping set. We discuss more about this decoder in Section 3.2. Also, [40] investigates error floors by hardware simulation. These approaches in the literature involving decoder improvements for lower LDPC code error floors can be divided into two categories: decoder modification (DM) and post-processing (PP). In this chapter, we propose three decoder-based strategies: a bi-mode decoder based on erasure decoding and its generalizations, bit-pinning/state-pinning decoders with/without outer algebraic codes, and three generalized-LDPC (G-LDPC) decoders. The bi-mode decoder is essentially a PP technique, the bit-pinning approach is a mixture of the DM and PP categories, and the G-LDPC decoders belong to the DM category. We will present these decoding techniques in the order just given, which is in the order of increasing complexity.
Equipped with the techniques discussed in Chapter 2, we shall explore specific codes to determine the trapping sets which dominate the floor of their iterative decoding performance curves. The most dominant trapping sets can be determined from computer simulations in the floor region. Moreover, for most practical codes, because the trapping set-induced subgraphs associated with the error floor are relatively small, and because their cardinalities are not too large, it is feasible to discover, enumerate, and evaluate all of the dominant trapping sets. Hence, once we obtain the trapping set information for an LDPC code, we explicitly target the known trapping sets with novel, custom-tailored iterative decoder designs. These low-floor decoders lower the floor by orders of magnitude. Although the specific trapping sets depend on decoder specifics such as algorithm type and message word size, the applicability and effectiveness of our low-floor techniques do not.

We first consider the binary-input additive white Gaussian noise (AWGN) channel and the sum-product algorithm (SPA) decoder with floating point precision as our baseline system. Later we generalize the solutions to PR channels with a detector that employs the BCJR algorithm and an LDPC decoder the employs the SPA. We chose two LDPC codes to demonstrate the effectiveness of these decoders on AWGN channel: 1) the rate-0.5(2640, 1320) Margulis code which is notorious for its trapping set-induced floors [10], [11], and 2) a short quasi-cyclic (QC) rate-0.3 (640, 192) code that we have derived from a (4544, 4096) QC-LDPC code to reduce simulation time. For the results on PR channel, we simulated a rate-0.78 (2048, 1600) QC-LDPC code also derived from the (4544, 4096) QC-LDPC code.

The rest of this chapter is organized as follows. Section 3.1 briefly reviews the LDPC codes under study and their dominant trapping sets. Section 3.2 describes the bi-mode decoder that recovers trapping sets by a post-processing erasure decoding algorithm. In Section 3.3, we propose the bit-pinning decoder/detector which utilizes one or more outer algebraic codes. In Section 3.4, we explore the concept of generalized LDPC Tanner graphs and their G-LDPC decoders. The idea is to decode conventional LDPC codes as if they were G-LDPC codes, with the goal of eliminating trapping sets error events. The algorithm for finding the trapping sets
of G-LDPC decoders is also presented in Appendix A, from which the frame error rate (FER) performance of G-LDPC decoders can be accurately predicted using the method in Chapter 2. Section 3.5 discusses the complexity cost of these decoders in comparison with the conventional sum-product algorithm decoder. Finally in Section 6.5, we draw some conclusions.

3.1 Codes Under Study

The LDPC codes we consider have the following structure: the parity-check matrix $H$ can be conveniently arranged into an $M \times N$ array of $Q \times Q$ permutation matrices and $Q \times Q$ zero matrices. This structure simplifies the analysis of the code because the Tanner graph possesses an automorphism of order $Q$, and thus simplifies the search for all of the trapping sets of a code. Obviously, if the $Q \times Q$ permutation matrices are circulant, the code is quasi-cyclic (QC), a characteristic that facilitates encoder and decoder implementations. We will refer to the $Q$ variable nodes associated with a column of permutation matrices as a VN group and the $Q$ constraint nodes associated with a row of permutation matrices as a CN group. A VN of such a code can be represented by an integer pair $(v_p, v_o)$, where $v_p \in \{0, 1, \cdots, N - 1\}$ is the index of the VN group and $v_o \in \{0, 1, \cdots, Q - 1\}$ is the offset within the VN group. Similarly, a CN can be represented using the pair $(c_p, c_o)$, where $c_p \in \{0, 1, \cdots, M - 1\}$ and $c_o \in \{0, 1, \cdots, Q - 1\}$.

3.1.1 The Margulis Code

The Margulis construction of a regular $(3, 6)$ Gallager code has been well studied [11][12][13]. The rate-1/2 (2640,1320) Margulis code used in this paper is generated from the special linear group $SL_2(11)$ for which $Q = 11$. Its parity check matrix can be expressed as a $120 \times 240$ array of $11 \times 11$ permutation matrices. The “weakness” of this algebraically constructed code is a relatively high error floor due to trapping sets, as discovered by MacKay and Postol in [11] using computer simulations. The source of the error floor is 1320 isomorphic $(12,4)$ trapping sets and 1320 isomorphic
Figure 3.1: Trapping sets of the Margulis and QC codes decoded by the SPA decoder. T.S. = trapping set. $\bigcirc$ = VNs. $\square$ = mis-satisfied CNs. $\blacksquare$ = unsatisfied CNs.
(14,4) trapping sets, both types depicted in Fig. 3.1. As can be observed in the
figure, a (14,4) trapping set has a structure similar to a (12,4) trapping set, and in
fact each (14,4) trapping set contains a unique (12,4) trapping set as a subgraph.
Hence, there is a 1-1 correspondence between the (12,4) and the (14,4) trapping
sets. For both trapping set classes, half of the bits errors are systematic errors and
half are parity errors.

This code is extremely difficult to deal with in terms of decoder design (as we
will show later) because its trapping sets are highly entangled in a way that is more
involved than what we just described. For instance, every VN of the code belongs
to six different (12,4) trapping sets. It is for this reason that we have studied the
Margulis code: to devise decoders that lower the floor of codes with (and without)
entangled trapping sets.

In Chapter 2, we demonstrated the IS method for AWGN and PR channels ap-
plied to this Margulis code. It was shown in [10] that the 1320 (12,4) trapping
sets account for about 75% of the error floor performance, and the (14,4) trapping
sets account for about 23%. In this chapter, we use a floating point software SPA
decoder simulator which we have verified produces the same performance and pre-
diction curves as those in [10]. It is known that quantization potentially affects
the error floor, and various implementations of the sum-product algorithm together
with various levels of quantization could lead to different trapping sets. We con-
tend that our algorithms will work on any decoder implementation because these
algorithms lower floors by eliminating the trapping set errors of any given decoder
implementation.

3.1.2 The Low-Rate QC Code

Targeting the 4096-bit sector size of disk drives, a rate-0.9(4544, 4096) QC IRA code
was proposed in [31] using PEG-like and ACE algorithms. The column weights of
this code are 5 and 2, and its parity check matrix is a $7 \times 71$ array of $64 \times 64$ circulant
matrices. The performance of this code on the binary-input AWGN channel with
a SPA decoder and a MS decoder implemented on a field-programmable gate array
(FPGA) is shown in Fig. 2.1. Observe that the code’s frame error rate (FER) curve displays a floor near $10^{-8}$. This floor is attributable to one class of $(5,5)$ trapping sets and 12 classes of $(5,7)$ trapping sets (Fig. 3.1). Since this code is quasi-cyclic with $Q = 64$, and each variable node of a trapping set belongs to a different variable node group, every trapping set class has 64 isomorphic trapping sets.

Because simulations of this $(4544, 4096)$ code are extremely time-consuming, we instead studied a derivative QC code whose $H$ matrix is a submatrix of the original $H$ matrix. The derived rate-0.3 $(640, 192)$ QC code has a circulant size $Q = 64$ and column weight $w_c = 5$. The $H$ matrix is a $7 \times 10$ array of $64 \times 64$ circulant permutation matrices. We observed in our simulations two trapping set classes with 64 isomorphic trapping sets in each class, with a representative from each class shown in Fig. 3.1. The $(5,5)$ trapping set is the dominant one, as it contributes to over 90% of the error floor level.

We remark that it is very common for structured LDPC codes to have one or two most dominant trapping set classes because their structured graphs lead to isomorphic classes of trapping sets. Unstructured LDPC codes are more likely to have a more diverse trapping set collection, but these codes are rarely of interest in practice. Still, the techniques in the paper can be straightforwardly extended to unstructured codes as well.

3.1.3 The High-Rate QC Code

Another derived code is a rate-0.78 $(2048, 1600)$ QC code that possesses the 9 most-dominant trapping set classes of isomorphism 64 from the original $(4544, 4096)$ code. With this shorter code, the error floor is more easily reachable using software simulations, and any floor improvement method demonstrated to be effective on this shorter code will have a similar effect on the longer rate-0.9 code.

Before we introduce our floor-lowering techniques, we first emphasize that the trapping sets indicated above for the $(4544, 4096)$ code and its derivative $(640, 192)$, $(2048, 1600)$ codes were found via simulations on the binary-input AWGN channel. In general, when one applies a code to a different channel, or even a different decoder,
the trapping sets can change. We have observed that the dominant trapping sets observed on the binary-input AWGN channel are also the dominant trapping sets for the PR1 and EPR4 channels. Thus, our techniques for these two PR channels will target the aforementioned classes of \((5, 5)\) and \((5, 7)\) trapping sets.

### 3.2 Bi-Mode Syndrome-Erasure Decoder

In \cite{37}, the authors propose a post-processing technique to lower the error floor by using a look-up table of known trapping sets. After conventional SPA decoding, this table is used to process the residual error blocks much like a syndrome decoder. A drawback to this approach is, if the cardinality of trapping sets is large, the implementation complexity of the look-up table may be very costly. Inspired by this technique, we propose another post-processing decoder, which avoids look-up-table decoding. In our technique, post-processing involves simple graph-based erasure decoding, so that our overall decoder has two decoding modes.

As discussed in Section 2.3, there are three types of error events in the error-floor region: (1) unstable error events, which dynamically change from iteration to iteration and for which \(w\) and \(v\) are typically large; (2) stable trapping sets, for which \(w\) and \(v\) are typically small and thus are the main cause of the error floor; and (3) oscillating trapping sets, which periodically vary with the number of decoder iterations and which are sometimes subsets of stable trapping sets. The targets of the bi-mode decoder are the dominant stable trapping sets and some of the dominant periodic oscillating trapping sets. In the first mode, SPA decoding is performed with a sufficient number of iterations for the decoder to reach one of the three error event situations just listed. The second mode is the post-processing mode, which is activated only when the syndrome weight of an error event falls into the set of syndrome weights of the target trapping sets. The key role of the second mode is, using syndrome information, to produce an erasure set that contains all of the VNs of the trapping set reached by the decoder, thus resulting in a pure binary erasure channel (BEC) with only correct bits and erasures. Iterative erasure decoding based
on the LDPC code's graph can then resolve all of the erasures, including the bits that were originally part of the trapping set error event.

The look-up table post-processing decoder in [37] is based on the fact that there is usually a 1-1 correspondence between a set of unsatisfied CNs and a trapping set. It was also observed that in the error-floor region, after running extensive Monte Carlo simulations, most of the error patterns correspond to the elementary trapping sets [41] whose induced subgraphs have only degree-one and degree-two CNs. That is, for an elementary trapping set, the unsatisfied CNs are usually connected to the trapping set exactly once (there is one bit error per unsatisfied CN). This is the case for the Margulis and QC codes we study. However, the applicability of the bi-mode decoding algorithm is not limited to codes with elementary trapping sets.

Assume now that the decoder converges to a trapping set and suppose an unsatisfied CN has degree $d_c$. The goal of that CN is to find which one of the $d_c$ neighbors is in error. Our experiments have shown that the LLR magnitude of the bit in error is not necessarily the smallest among the $d_c$ VNs connected to the unsatisfied CN. Thus, we propose flagging as erasures all of the $d_c$ neighbors of the unsatisfied CN and then performing iterative erasure decoding in the neighborhood of that unsatisfied CN.

The set of erasures is generated by erasing the VNs of $v$ trees whose roots are the $v$ unsatisfied CNs, as in Fig. 3.2. The depth of the trees depends on the trapping set structures of the code being decoded. Consider a regular LDPC code with row weight $d_c$ and column weight $d_v$. For a $(w, v)$ trapping set, the number of nodes in each level of the trees is listed in Table 3.1. For a trapping set with $w \leq v$, i.e., every VN is connected to at least one unsatisfied CN, trees with two levels can cover the whole trapping set. Otherwise, when $w > v$, bigger trees are necessary. As the trees grow, the number of erasures grows exponentially and the whole trapping set will eventually be covered. However, the growth must be limited, because too many erasures may form so-called stopping sets (trapping sets on the BEC), and overwhelm the decoder. As will be explained below for the Margulis code, for some codes, additional steps are necessary to produce a smaller erasure set covering the
Figure 3.2: Syndrome-erasure flagging trees. ■ = unsatisfied CN. ○ = VN in trapping set. ⊙ = VN outside trapping set. □ = mis-satisfied CN. ☐ = CN outside trapping set.

whole trapping set. We now present the bi-mode decoding solutions for the Margulis and QC codes.

<table>
<thead>
<tr>
<th>Level</th>
<th>Node Type</th>
<th>Node Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CN</td>
<td>$v$</td>
</tr>
<tr>
<td>2</td>
<td>VN</td>
<td>$d_v$</td>
</tr>
<tr>
<td>3</td>
<td>CN</td>
<td>$\leq d_v(d_v - 1)$</td>
</tr>
<tr>
<td>4</td>
<td>VN</td>
<td>$\leq d_v(d_v - 1)(d_c - 1)$</td>
</tr>
</tbody>
</table>

3.2.1 Margulis Code Solution

We observed from simulations that the Margulis code with an SPA decoder is sometimes trapped in two oscillating (6,18) trapping sets or two oscillating (7,21) trapping sets or two (6,18) and (7,21) trapping sets, all with period two. The union of the (6,18) pair is a (12,4) trapping set, and that of the (7,21) pair is a (14,4) trapping set. Section 2.3 discusses the dynamic behavior of such periodic trapping sets.
In the decoder solution for this code, the target trapping sets include the stable trapping sets as well as the periodic oscillating configurations. It is obvious that the oscillating trapping sets and some moderate-length unstable error events with \( w \leq v \) can be flagged with two-level trees and recovered successfully. As depicted in Fig. 3.3, any (12,4) trapping set can be flagged and recovered using trees with four levels; however, four-level trees cover only 10 VNs of any (14,4) trapping set. If 6-level trees are used, the erasure decoder is overwhelmed by too many erasures.

To solve this problem, an auxiliary step is necessary. As discussed in Section 3.1.1, any (14,4) trapping set contains a (12,4) subset plus two additional VNs. From the syndrome weight itself, the decoder cannot distinguish which of these two trapping set classes it is dealing with. The algorithm described below which resolves this issue is based on the following observation from the structure of the trees generated from the (12,4) and (14,4) trapping sets: As seen in Fig. 3.3 for either trapping set, there exists exactly two VNs in the second level which share a common level-3 CN. These two VNs are what distinguish the (12,4) and (14,4) trapping sets (examine Fig. 3.3). Toggling the values of these VNs will switch a (12,4) trapping set to a (14,4) trapping set, and vice versa.

Note that \( c_a, v_{a_1}, v_{a_2} \) in Algorithm 1 are unique for any (12,4) or (14,4) trapping set. The difference is: if a (14,4) trapping set is reached in the first decoding mode, \( v_{a_1} \) and \( v_{a_2} \) are incorrect; otherwise when a (12,4) trapping set is reached, the two VNs have correct bit values. Note also that when repeating Steps 1 to 3, the decoder adds to the erasures already flagged in the first round of Steps 1 to 3. Flipping the two bits of a (14,4) trapping set essentially corrects two bit errors and turns it into a (12,4) trapping set. However, since both trapping set classes have the two unique VNs identified in Step 4, the decoder still cannot distinguish which trapping set it is operating on. Thus, if a (12,4) trapping set is reached, \( v_{a_1} \) and \( v_{a_2} \) are flipped anyway, resulting in a (14,4) trapping set. For this case, repeating Steps 1 to 3 simply flags as erasures more VNs that are not in error, but does not overwhelm the erasure decoder. By using this algorithm, 352 erasures will be produced for any (12, 4) or (14,4) trapping set, which can be recovered by the
Figure 3.3: Margulis code erasure flagging with 4-level trees (the numbers on the VNs are tree level indices). Flipping the two additional VNs causes the switching between the two trapping sets.

Algorithm 1 Auxiliary Algorithm for (12,4) and (14,4) Trapping Sets

1. Take the four unsatisfied CNs and create four trees each of four levels with these CNs as roots.

2. Erase all the VNs in the second level, and mark them with flag “A”.

3. Erase all the VNs in the fourth level, and mark them with flag “B”.

4. For each CN in level-3, count how many of its neighboring VNs have flag “A”. Once a CN $c_a$, with two neighboring VNs ($v_{a_1}$ and $v_{a_2}$) that are flagged with “A” is found, stop the search and go to Step 5.

5. Flip the bit values of $v_{a_1}$ and $v_{a_2}$, and re-calculate the syndrome, producing a new set of four unsatisfied CNs.

6. Repeat Steps 1 to 3 with the new syndrome information and then go to Step 7.

7. Including all the VNs with both A and B type erasure flags, perform iterative erasure decoding in that neighborhood until all of the erasures are resolved.
LDPC code successfully within 3 or 4 iterations.

The performance results for this bi-mode decoder with the Margulis code are presented in Fig. 3.4. For the SPA curves in the water fall regime, 100 LDPC codeword errors were collected; in the floor region, 50 codeword errors were collected, except for the last point at 2.7dB, which corresponds to 20 codeword errors. For the bi-mode curves, the numbers are similar, except the last three data points contain at least 10 block errors. Simulations shows that no more (12,4) or (14,4) trapping sets remains. No floor is observed down to FER $\sim 10^{-8}$ for the bi-mode decoder.

We remark that the bi-mode decoder has the following advantages relative to other floor-lowering techniques:

1. Beside stable trapping sets and oscillating trapping sets, this technique can also handle some unstable error events with $w \leq v$.

2. No outer codes are employed, so the gain is achieved without any code rate loss.

3. The erasure decoding post-processing has very low computational complexity, which is equivalent to solving linear equations with binary unknowns, and thus involves only a collection of binary XORs.

4. The concept of employing the erasure decoding second mode can be generalized for combating trapping sets of different graphical structures.

### 3.2.2 Low-Rate QC Code Solution

For both trapping set classes of the short QC code, every VN is associated with at least one unsatisfied CN ($w \leq v$). Hence, one level of VNs per tree is enough to include a whole trapping set in the erasure set. Whenever the SPA decoder gets trapped in an error event with syndrome weight 5 or 7, the decoding enters the second mode. 34 erasures are flagged for any (5,5) trapping set and 54 are flagged for any (5,7) trapping set, all of which can be recovered with one erasure decoding iteration. The performance of this code with bi-mode decoder is presented
Figure 3.4: Performance of the bi-mode decoder on the Margulis code compared with that of the SPA decoder.
in Fig. 3.5, with 100 block errors collected, except for the last two data points for the bi-mode curves which correspond to 20 block errors. No floor is seen down below FER $\sim 10^{-6}$, so that the floor is lowered by at least two orders of magnitude relative to the SPA decoder.

3.2.3 Modifications for General Codes

As discussed above, the limitations of this bi-mode technique lie in the structure of the code’s trapping sets with respect to the erasure correcting capability of the code with iterative decoding. With certain generalization/modifications custom-tailored to tackle different trapping sets, this simple yet effective bi-mode technique can apparently be applied to many LDPC codes. One scenario that requires modification is when the erased bits contain a stopping set, which usually happens if the trapping set has a large number of check violations and/or the code’s graph has large check node degrees. One solution is to use a soft-erasing method which retains the soft information (LLRs) of the bits outside of the erasure set, while erasing the erasure bits by setting their LLRs to zeros, and then performing the conventional iterative decoding. Another variation of the bi-mode technique is a partial-soft-erasing technique, which erases (by setting LLRs to zeros) the neighboring bits of one of the unsatisfied checks and then performs normal decoding; if the decoder fails to converge, the decoder is reset with erasures based on another unsatisfied check and this procedure is repeated until the decoder converges or all unsatisfied checks have been used.

For example in [22], the authors show that the (2048, 1664) Reed-Solomon-based LDPC code adopted in the IEEE 802.3an 10GBASE-T standard possesses an isomorphic class of (8,8) trapping sets. These (8,8) trapping sets are easily resolved using our partial-soft-erasing bi-mode decoder which requires on average around 20 additional decoding iterations to correct the entire trapping set. Additionally, there are some less important (8,12) and (7,12) trapping sets which can also be eliminated by this technique. The performance of the partial-soft-erasure bi-mode decoder is demonstrated on the RS-based LDPC code in Fig. 3.6.
Figure 3.5: Performance of the bi-mode decoder on the low-rate QC code compared with that of the SPA decoder.
Figure 3.6: Performance of the bi-mode partial-soft-erasure decoder on the IEEE 802.3an standard LDPC code.
3.3 Concatenation and Bit/State-Pinning

It is well known that error floors of LDPC codes are usually a consequence of frequently occurring block errors with a small number of bit errors with certain isolated graph structure in each block error. A natural solution to lowering the floor in this case is to concatenate the LDPC code with a high-rate outer algebraic code to clean up the residual systematic errors, at the expense of a code-rate loss. Currently the serial concatenation of an outer code with the LDPC code has in fact been incorporated into the digital video broadcasting (DVB-S2) standard [53]. We propose more effective concatenation and pinning techniques which reduce the code rate loss by exploiting the trapping set knowledge of the LDPC code of interest.

3.3.1 Pinning Techniques for AWGN Channels

The idea behind “pinning” is to, via external measures, “fix” one or more of the bits in each problematic trapping set so that the stabilized subgraph structure is disturbed, and then let the iterative decoder proceed to correct the rest of the bits. As shown in Fig. 3.7(a), our experiment on AWGN channel proves that when one of the variable nodes in a (5, 5) trapping set of the QC LDPC code is pinned by setting its bit value to 0 (or 1) prior to encoding and correspondingly setting to the maximum (or minimum) value the log-likelihood ratio (LLR) input to the decoder, the entire trapping set is corrected with two additional LDPC decoder iterations. Similarly, a (5, 7) trapping set in Fig. 3.7(b) needs at most two additional iterations; a (12, 4) trapping set of the Margulis code in Fig. 3.7(c) can be corrected with six iterations.

Simulations have shown that this procedure offers a substantial improvement in the floor region [32]. However, this solution is not as effective for some codes, such as the Margulis code, which have a large number of overlapped trapping sets due to the substantial code rate loss introduced. The code rate loss due to direct concatenation or bit-pinning alone can be reduced by combining the two methods. Therefore, we consider such combinations by exploring the trapping set information
of a code under study.

In one solution to this problem, one or more high-rate outer codes can be concatenated with the LDPC code, where the bit assignments in outer codes are arranged such that, whenever a trapping set error event occurs, at least one (systematic) bit is corrected by an outer algebraic decoder. The goal is to ensure that the short, low-complexity, high-rate outer codes target the trapping sets of interest. The multiple decoders are coordinated as follows. First, a sufficient number of LDPC decoder iterations is performed to guarantee that either all errors are corrected or a stable trapping set is reached. Then the outer hard-decision decoder(s) correct some of the residual errors and feed(s) back the signs of the corrected bits to the LDPC decoder, which pins the absolute values of LLRs corresponding to these bit to the maximum possible value. Then the LDPC decoder continues to iterate. With probability near
one, the LDPC decoder will correct both systematic and parity errors caused by a trapping set within a few additional iterations.

**Margulis Code Solution**

Since each of the dominant trapping set classes of the Margulis code correspond to either 6 or 7 systematic bits, a single \( t \)-error-correcting BCH code with \( t = 7 \) is an obvious solution. In this case, no pinning is necessary. We choose the \( t = 7 \) \((1320,1243)\) BCH code with roots in \( \text{GF}(2^{11}) \), which is shortened from a primitive \((2047,1970)\) BCH code. The code rate of the overall system is reduced from 0.5 to 0.47, corresponding to a 0.26 dB rate loss.

We can reduce the code rate loss by considering multiple BCH codes of higher rates which exploits the fact that the trapping sets of this code are highly overlapped. We found that four BCH codes with \( t = 1 \) were sufficient when allowing feedback from the BCH decoders to the LDPC decoder. The BCH code bit assignments to the Margulis code bits are shown below. For convenience, the bit assignments are listed by VN group indices \( v_p \) since all 11 bits in the VN group will be part of the BCH codeword.

<table>
<thead>
<tr>
<th>Table 3.2: Bit Assignments for BCH Pinning on Margulis Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>((n,k))</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>(132,124)</td>
</tr>
<tr>
<td>(77,70)</td>
</tr>
<tr>
<td>(55,49)</td>
</tr>
<tr>
<td>(33,28)</td>
</tr>
</tbody>
</table>

The overall code rate is 0.49 (a 0.086 dB loss). The FER and BER curves of concatenation with the single-BCH-code (no pinning) and four-BCH-code (with pinning) solutions are shown in Fig. 3.8, respectively. All curves have at least 20 frame error occurrences in the range of \( E_b/N_0 \geq 2.5 \) dB. We observe that both solutions lower the floor beyond the reach of our simulations, and that the code-rate
Figure 3.8: Performance of the SPA, serial concatenation and BCH-Bit-Pinning decoders for the Margulis code.
loss for the four-BCH-code solution is about 0.2 dB less than that of the single-BCH-code solution, as indicated above.

Short QC Code Solution
As discussed in Section 3.1.2, the (640, 192) QC code has two trapping set configurations. We can concatenate that code with two $t = 1$ binary BCH (64, 58) codes. The bits of one BCH code belong to 64 different (5,5) trapping sets, and those of the other belong to 64 different (5,7) trapping sets. Thus, whenever a trapping set is in error, one of its five bits is correctable by one BCH code. Once corrected, this information is fed back to the LDPC decoder which pins its LLR magnitude to the maximum possible (with the appropriate sign). The overall code rate is reduced from 0.3 to 0.2813 (0.28 dB rate loss). The simulation results presented in Fig. 3.9 show no floor down to FER $\sim 10^{-7}$. 20 LDPC codeword errors were collected in high SNR region.

3.3.2 Pinning Techniques for Partial Response Channels
The current section extends the pinning approach to LDPC-coded partial response (PR) channels, which are applicable to magnetic and optical storage. We consider two partial response targets: PR1 and EPR4, and compare two pinning alternatives: (1) a pre-pinning technique which fixes selected trapping set bits prior to transmission and (2) a post-pinning approach which utilizes information from outer BCH decoders to pin bits in trapping sets. We also compare the performance attained when pinning is performed at the channel detector and at the LDPC decoder. Our simulations on PR1 and EPR4 channels demonstrate the solutions for the code chosen for this study, the 0.78(2048,1600) QC LDPC code.

Pre-Pinning
As explained in the Section 3.3.1 for the binary-input AWGN channel, the pinned bits need not be transmitted and, hence, pre-pinning is in some sense equivalent to code shortening for this channel. Where shortening and pinning differ is that
Figure 3.9: Performance of the SPA, serial concatenation and BCH-Bit-Pinning decoders for the low-rate QC code.
shortening usually involves fixing consecutive data bits to 0’s (or 1’s) without regard to the location of trapping set bits. But there is an additional difference in the case of partial response channels: as will be made clearer in the next paragraph, because PR channels are not memoryless, the pinned bits must be transmitted.

The pinned bit locations can be made available at the LDPC decoder (bit pinning) or the PR channel detector (trellis pruning). The pinned bits at the detector prunes branches from the detector trellis, but also passes the pinned bit information to the decoder. Thus, the trellis pruning option also provides bit pinning at the decoder. We examine the performance of the bit pinning case to investigate the gain that the pruned detector trellis provides relative to bit pinning only at the decoder. Trellis pruning was explored in detail in the context of convolutional codes by Collins and Hizlan in [42]. For their scenario, the pruning of trellis branches in the convolutional decoder was done by a high-reliability outer Reed-Solomon code. A similar approach, which we call post-pinning, is explored in the next section. Also, Bertand and Labeau [43] apply these ideas to trellis-based decoding of linear block codes based on removing some edges and states from the trellis representation of the code.

The (2048,1600) code has the 10 dominant trapping set classes listed in Table 3.3, each of which is represented by the indices of variable node group \( v_p \) and offsets within the group \( v_o \) in the graph of the code. From the first column of Table 3.3, one can observe that the four variable node groups with \( v_p = 0, 1, 2 \) and 3, a total of 256 variable nodes, cover all of the 640 trapping sets in the sense that each trapping set contains exactly one of these 256 bits. Hence, pinning these 256 bits allows the decoder to correct all of the other bits in any given trapping set.

Fig. 3.10 shows the system diagram of pre-pinning method when applied to a partial response channel. The 256 bits corresponding to variable node groups \( v_p = 0, 1, 2, 3 \) are pinned to ‘1’ which, together with 1344 user bits, are encoded to produce a 2048-bit LDPC codeword. As indicated in the figure, there are two possible ways for the receiver to use this bit pinning information. For the “bit-pinning” option (cf. Fig. 3.10), the decoder is designed so that the variable node
Table 3.3: List of dominant trapping set classes of the high-rate QC code

<table>
<thead>
<tr>
<th>$v_p$</th>
<th>$v_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,5) trapping sets</td>
<td>(5,5) trapping sets</td>
</tr>
<tr>
<td>14 0 29 4 9</td>
<td>0 22 18 61 0</td>
</tr>
<tr>
<td>(5,7) trapping sets</td>
<td>(5,7) trapping sets</td>
</tr>
<tr>
<td>10 12 16 22 2</td>
<td>0 9 23 9 10</td>
</tr>
<tr>
<td>11 21 27 2 6</td>
<td>0 49 32 14 10</td>
</tr>
<tr>
<td>0 21 26 2 6</td>
<td>0 17 47 29 25</td>
</tr>
<tr>
<td>21 17 27 5 3</td>
<td>0 26 59 15 3</td>
</tr>
<tr>
<td>13 20 23 28 3</td>
<td>0 49 10 33 17</td>
</tr>
<tr>
<td>0 22 23 30 0</td>
<td>0 29 30 19 57</td>
</tr>
<tr>
<td>15 16 25 1 6</td>
<td>0 61 53 32 14</td>
</tr>
<tr>
<td>16 21 24 29 1</td>
<td>0 0 16 62 38</td>
</tr>
<tr>
<td>17 18 31 19 2</td>
<td>0 30 40 30 14</td>
</tr>
</tbody>
</table>

Inputs corresponding to the pinned bits are set to the maximum possible value with the correct signs. For the “trellis-pruning” option shown in Fig. 3.10, the pinned bit locations are designed into the PR trellis detector. The pruning effect of a single pinned bit is demonstrated in Fig. 3.11 for the PR1 and EPR4 trellises, which have channel transfer functions $1 + D$ and $1 + D - D^2 - D^3$, respectively. Observe that since the PR1 channel has a memory of one, a single pinning bit effectively pins the trellis to a single state. For EPR4, the pinning bit halves the number of trellis branches in the corresponding trellis stage.

Clearly, trellis pruning improves performance as it can reduce the Euclidean distance multiplicities or increase the minimum Euclidean distance. However it is necessary to account for the power used in transmitting these known bits. The code rate of the pre-pinning scheme drops from $\frac{1600}{2048} = 0.7813$ to $\frac{1344}{2048} = 0.6563$, a 0.76 dB rate loss. We will see that the effects of the code rate loss and the improved trellis distance spectrum essentially cancel each other out.

We note in passing that, by choosing these pinned bits to have the value ‘1’, it is possible to permute the encoder output so that these 1’s are uniformly spread
Figure 3.10: System diagram of pre-pinning.

to reside in every eighth bit of the channel sequence (256 pinning bits out of 2048 code bits). This provides a runlength constraint which guarantees that there are at most seven 0’s between consecutive 1’s. Such runlength constraints aid bit-timing recovery and are used in all magnetic and optical storage devices.

We present the simulation results, in terms of frame error rate (FER) and bit error rate (BER) versus SNR, of pre-pinning on PR1 and EPR4 channels in Figs. 3.12 and 3.13, respectively. In these simulations, 100 local LDPC decoder iterations were performed, but we allowed no “global” iterations between the LDPC decoder and the BCJR channel detector. As seen in the figures, both pre-pinning approaches (bit-pinning and trellis-pruning) reduce the error floor by at least two orders of magnitude. Moreover, for trellis-pruning, there is a negligible performance loss in waterfall region. Trellis pruning provides a gain of about 0.4 dB over bit pinning in
FER $\sim 10^{-6}$ region.

**Post-Pinning**

Although there is effectively no code rate loss due to the pinning bits for the trellis-pruning option of the pre-pinning technique, it is possible to provide trellis-pruning in a more power-efficient manner. In this section, we study trellis-pruning with the assistance of an outer algebraic code via a method we call *post-pinning*. As discussed in Section 3.3.2, 256 specific bits cover the 640 trapping sets of the $(2048,1600)$ code and, if their values are known to the decoder, then the decoder is immune to the trapping sets. However, as opposed to pre-pinning where those bits are known at the outset due to bit pinning, with post-pinning the values of the 256 bits can be learned through the addition of a short BCH code which protects those bits with great reliability. Specifically, we used a $(256,240,t = 2)$ BCH code to protect the 256 bits. Because at most one trapping set error occurs at the LDPC decoder output,
Figure 3.12: Performance of pre-pinning on the PR1 channel.
Figure 3.13: Performance of pre-pinning on the EPR4 channel.
at most one of the 256 bits is expected to be in error after a sufficient number of LDPC decoder iterations. Thus, the BCH decoder will see at most one error per BCH codeword and, upon successful decoding, can inform the LDPC decoder (bit-pinning option) or channel detector (trellis-pruning option) that these 256 bits can be assumed to have the highest reliability.

The system diagram with the trellis-pruning option is shown in Fig. 3.14, where the LDPC decoder is followed by a BCH decoder. When the LDPC decoder fails to get past a trapping set, the BCH decoder corrects the bit error due to the trapping set and then feeds back all 256 correct bits value to the channel detector, effectively pruning its trellis. Eventually, the residual errors in the trapping set will be corrected by additional LDPC decoder iterations.

Simulation results of the post-pinning technique on PR1 channel and EPR4 channel are shown in Fig. 3.15 and Fig. 3.16, respectively. Without pinning, with
Figure 3.15: Performance of post-pinning on PR1 channel.
Figure 3.16: Performance of post-pinning on EPR4 channel.
100 local LDPC iterations and 5 global iterations, the error floor shows up at FER \( \simeq 10^{-6} \). With post-pinning and trellis-pruning, the overall code rate is reduced to 0.7734, a rate loss of only 0.04 dB, and the error floor is effectively reduced beyond the range reachable by simulations for both channels. By comparing the performances of trellis-pruning solution in the post-pinning case with that in the pre-pinning case, we see roughly a 0.6 dB to 0.7 dB gain in the water-fall region. This is due to the fact that 5 global iterations between the detector and decoder were performed in the post-pinning simulations.

3.4 Generalized-LDPC Decoder

In [38], an averaged decoding algorithm was proposed to reduce the number of incorrectly decoded frames in the error floor region of the Margulis code. This algorithm is a modified SPA decoding algorithm which averages messages over several iterations to “slow down” the convergence rate of certain variable values, because a sudden magnitude change in the values of certain variable messages or fast convergence to an unreliable estimate is a possible indicator of the emergence of an error trap. In this section, we propose novel SPA decoders which, loosely speaking, are designed by transforming the LDPC code into a generalized LDPC (G-LDPC) code. A G-LDPC code, like an LDPC code, is a code that can be described by a sparse bipartite graph with variable nodes and constraint nodes. However, for G-LDPC codes the constraints may be more general than single parity-check (SPC) constraints. For example, a constraint node can represent an arbitrary \((n', k')\) binary linear code.

To see how the G-LDPC philosophy arises, we remind the reader of the discussion in the Chapter 2 regarding locally optimum versus globally optimum decoders. Each constraint node decoder in an LDPC SPA decoder is locally optimum. It is possible in principle to group all of the SPC constraints into one combined global constraint and design a decoder for that graph. But that would be an ML decoder (for example), which has unacceptable complexity. Our strategy is to instead take
one step toward that ideal and cleverly combine only a few SPC constraints at a time. Specifically, we combine check node processors corresponding to unsatisfied checks in the problematic trapping sets and call the combination a *generalized-constraint processor (GCP)*. Observe that an advantage of doing so is that cycles and other deleterious graphical structures (from the perspective of an iterative decoder) may be removed. See Fig. 3.17 for an illustration of this.

The constituent decoder for the GCP can be any soft-input/soft-output decoder. We use a locally optimal decoder which employs the BCJR algorithm designed to the “BCJR trellis” [44] [45] for the linear code represented by the GCP. We allow both SPCs and GCPs to co-exist in the generalized Tanner graph, *i.e.*, some of the SPCs are not combined to form a GCP. The G-LDPC decoder, essentially the SPA with GCP’s (denoted by SPA-GCP) passes soft information iteratively between VN processors and GCP’s in the same manner as the SPA decoder.

Below we present three different G-LDPC decoders (equivalently, methods for grouping SPCs into GCPs) for both the Margulis code and the QC code. The approaches are based on the knowledge of the dominant trapping sets. The goal of the SPC grouping methods is to eliminate the dominant trapping sets, thus lowering the error floor. In selected cases, we predict the FER performance in the floor region using the method in Chapter 2.

### 3.4.1 G-LDPC Decoder I

**Margulis Code Solution**

At least qualitatively, it is clear why an iterative decoder gets “trapped” by the subgraphs associated with trapping sets: As seen in Fig. 3.1, when all of the bits in a trapping set are incorrect, most of the associated check equations are still satisfied (that is, mis-satisfied), so that a locally operating iterative decoder is incapable of resolving the discrepancies caused by these errors. However, if the unsatisfied CNs can be mutually fortified, the decoder may be able to correct the errors. The Type I G-LDPC (G-LDPC I) decoders are designed based on this idea.
Figure 3.17: An example of combining 3 SPC constraint processors into the GCP $SC_1$: the redundant edges (dashed lines) on the right are deleted after grouping. Conventional iterative decoding updates messages based on the graph on the left, employing SPA constraint processors for each SPC separately. The G-LDPC decoder updates messages based on the generalized graph on the right, using the BCJR algorithm, for example, to calculate extrinsic information to be sent from $SC_1$ to neighboring VNs. Note that we allow SPC constraints to coexist in the generalized graph, whose node processors are the same as those in the original graph.

Because the Margulis code trapping sets are highly overlapped (intersections are non-empty), we chose to require that no two GCPs share any common SPC constraint. We first grouped the four unsatisfied check nodes in the 198 non-overlapping $(12,4)$ trapping sets into 198 GCPs. We also grouped the four unsatisfied check nodes in the remaining 22 non-overlapped $(14,4)$ trapping sets into another 22 GCPs. See Fig. 3.18. In addition to these 220 GCPs (66.7% of all SPCs), each of which is a composition of four standard check nodes, the generalized Tanner graph also has 440 standard SPC nodes. At each GCP, the BCJR algorithm was applied to the BCJR trellis for the four associated check equations. The SPC constraints used standard SPA processing.

We ran Monte Carlo simulations on the generalized Tanner graph and the performance in terms of frame error rate (FER) and bit error rate (BER) are shown in Fig. 3.19. We observe in the figure that the error floor is lowered by an order of magnitude compared to the standard SPA decoder. The simulator collected more than 20 frame errors for the last two simulation points of the G-LDPC I decoder, and it performed 220 iterations in order to obtain the stable trapping sets. We
observed no (12,4) or (14,4) trapping sets by the G-LDPC I decoder. The new trapping sets of the G-LDPC I decoder are transformed from the original ones by missing/appending one VN from/to the (12,4) or (14,4) trapping sets as shown in Fig. 3.20. Given the algebraic construction of the Margulis code, we can find 1320 subgraphs of each configuration in Fig. 3.20 in the original graph. However, from the G-LDPC decoder’s point of view, not all of the 1320 subgraphs are isomorphic. For example, a (13,5) subgraph in the original Tanner graph, which has the same structure as the example in Fig. 3.20, is a dominant trapping set for the G-LDPC I decoder if and only if its five unsatisfied CNs are not involved in any GCP and the VN with two unsatisfied CNs is attached to the rest of the subgraph through a GCP. Similarly for the other configurations in Fig. 3.20.

Table 3.4: Dominant Trapping Set List of Margulis Code with G-LDPC I

<table>
<thead>
<tr>
<th>Trapping Set Class</th>
<th>Multiplicity</th>
<th>Contribution to FER Floor</th>
</tr>
</thead>
<tbody>
<tr>
<td>(11, 5)</td>
<td>22</td>
<td>17.4%</td>
</tr>
<tr>
<td>(13, 5)</td>
<td>77</td>
<td>48.7%</td>
</tr>
<tr>
<td>(14, 6)</td>
<td>77</td>
<td>4.1%</td>
</tr>
<tr>
<td>(15, 5)</td>
<td>99</td>
<td>27.3%</td>
</tr>
<tr>
<td>(15, 7)</td>
<td>22</td>
<td>0.3%</td>
</tr>
<tr>
<td>(16, 6)</td>
<td>33</td>
<td>1.1%</td>
</tr>
<tr>
<td>(17, 5)</td>
<td>22</td>
<td>1.2%</td>
</tr>
<tr>
<td>(21, 5)</td>
<td>11</td>
<td>0.02%</td>
</tr>
</tbody>
</table>
Figure 3.19: Performance of G-LDPC decoders on the Margulis code compared with that of the SPA decoder.
These observations allow us to find a complete list $\mathcal{T}$ of the new trapping sets that are related to the original ones by searching the generalized Tanner graph, or more efficiently by using a two-step importance sampling method which we will describe in Appendix A. Table 3.4 lists the dominant trapping sets and their multiplicities for the G-LDPC I decoder. The contribution of each trapping set class to the error floor performance was evaluated by the IS method discussed in Chapter 2. These results demonstrate that the G-LDPC I decoder eliminates the (12,4) and (14,4) trapping sets as problems. Although new dominant trapping sets arise, the new trapping set configurations are larger in weight and have much smaller multiplicities, and are thus less harmful. The floor prediction curves in Fig. 3.19 are obtained by weighted (by the multiplicity) sums of the error rates of each type in the table.

**Short QC Code Solution**

A similar G-LDPC I decoder is derived for the short QC code by combining the five unsatisfied, non-overlapped SPCs into a GCP (Fig. 3.21). The resulting graph contains 29 GCP and 303 standard SPCs. Through the two-step IS method in Appendix A, we found this decoder has three different dominant (6,8) configurations (Fig. 3.22): one consists of a (5,5) trapping set and a additional VN who is attached to the (5,5) through one of its five unsatisfied CNs; the other two classes are transformed from (5,7) trapping sets by adding a VN through two of the seven unsatisfied
Figure 3.21: The QC code G-LDPC I grouping. $\Box = \supercn$ consists of 5 SPC-CN.

(a) A (6,8) trapping set from (5,5) subgraph  
(b) A (6,8) trapping set from (5,7) subgraph  
(c) A (6,8) trapping set from (5,7) subgraph

Figure 3.22: New trapping sets of the QC code with G-LDPC I decoder.

CNs. The (5,5) and (5,7) SPA trapping sets are eliminated and the FER floor is reduced by around two orders of magnitude relative to the standard SPA decoder. See Fig. 3.23, in which we collected 100 block error events for SPA simulation and 20 block error events in the floor region for the G-LDPC I decoder.

3.4.2 G-LDPC Decoder II

In Section 3.4.1, we presented the G-LDPC I decoder which eliminates the dominant trapping sets of SPA decoder. However, several new trapping sets emerged which dominated the new (lower) floor. On the other hand, only 66.7% of the CNs were combined in the Margulis code G-LDPC I. Further, we can target these new trapping sets and group appropriately selected SPCs into more GCPs. According to Table
Figure 3.23: Performance of G-LDPC decoders on the low-rate QC code compared with that of the SPA decoder.
3.4, three trapping set classes amount to over 90% of the floor of G-LDPC I: (11,5), (13,5) and (15,5). We can thus combine the five non-overlapping unsatisfied SPCs of these trapping sets, arriving at a G-LDPC graph with 229 standard CNs and 258 GCPs. 82.6% of the SPCs in the original graph are combined in this decoder.

The trapping sets observed for the G-LDPC II decoder are also closely related to the SPA trapping sets, and the same two-step IS method can be applied to enumerate the G-LDPC II trapping sets (results listed in Table 3.5). We can see that the three dominant trapping set classes of the G-LDPC I decoder are eliminated by these 38 additional GCPs, and the dominant trapping sets become (14,6) and (16,6) which contribute 90% of the new floor level. A prediction curve is drawn in Fig. 3.19, which is over two orders of magnitude lower than the standard iterative decoder. This procedure of adding more GCPs according to new trapping sets can continue until no SPC CNs remain in the generalized graph or the system’s floor performance requirement is achieved. Of course, the improvement is at the cost of increasing complexity. This realization led us to the G-LDPC III decoder.

Table 3.5: Dominant Trapping Set List of Margulis Code with G-LDPC II

<table>
<thead>
<tr>
<th>Trapping Set Class</th>
<th>Multiplicity</th>
<th>Contribution to FER Floor</th>
</tr>
</thead>
<tbody>
<tr>
<td>(14, 6)</td>
<td>13</td>
<td>54.3%</td>
</tr>
<tr>
<td>(14, 8)</td>
<td>4</td>
<td>1.0%</td>
</tr>
<tr>
<td>(15, 7)</td>
<td>11</td>
<td>8.4%</td>
</tr>
<tr>
<td>(16, 6)</td>
<td>11</td>
<td>26.2%</td>
</tr>
<tr>
<td>(16, 8)</td>
<td>18</td>
<td>5.4%</td>
</tr>
<tr>
<td>(17, 7)</td>
<td>10</td>
<td>1.4%</td>
</tr>
<tr>
<td>(18, 8)</td>
<td>11</td>
<td>3.2%</td>
</tr>
</tbody>
</table>

After discussing the philosophy behind the G-LDPC II decoder in the context of the Margulis code, it is clear that this same obvious approach could be applied to the QC code. However, we felt no need to make this obvious (and tedious) step, particularly since G-LDPC III decoder in the next section provides the best solution.
3.4.3 G-LDPC Decoder III

Margulis Code Solution

It was observed through simulations of many LDPC codes that, in the floor region, a frame error event usually contains a single trapping set. We also confirmed this observation by applying the importance sampling prediction method on a double trapping set which contains two single trapping sets and typically yields error floors several orders lower of magnitude than a single trapping set does. Instead of combining unsatisfied CNs within a trapping set, the G-LDPC III decoder takes trapping sets in pairs and combines an unsatisfied CN of one trapping set with that of the other (see Fig. 3.24). The rationale is when the decoder becomes trapped in one trapping set, reliable information from its companion trapping set will be passed along the GCP, allowing recovery from the trapping set error event.

The G-LDPC III decoder solution for the Margulis code contains 660 GCPs, each consisting of two SPCs. This number came from the 1320 (12,4) trapping sets (660 pairs). Thus there are no SPC CNs in the generalized graph in this case. Due to the overlap of trapping sets (for example, every CN is one of the unsatisfied CNs in four different trapping sets), all the trapping sets form a linked overlapped network. The IS method in Appendix A was applied to this decoder and no trapping sets were observed. The effectiveness of this simple G-LDPC decoder is also confirmed by simulations, which shows no floor down to FER \( \sim 10^{-8} \). We collected 50 error events at 2.2 dB and 2.3 dB, 10 error events at 2.4 dB, and only 5 error events at 2.5 dB (due to weeks-long simulation times).

Short QC Code Solution

Take any (5,5) trapping set in the QC code; its five VNs belong to five VN groups with indices \([0, 1, 2, 3, 4]\); the five unsatisfied CNs belong to four CN groups with indices \([0, 1, 2, 3]\). Thus, we take any two (5,5) trapping sets and group the three unsatisfied CNs which belong to disjoint CN groups (Fig. 3.25). The 64 (5,5) trapping sets can be combined to give \(32 \cdot 3 = 96\) GCPs. The FER and BER performance
curves with this decoder are shown in Fig. 3.23, and no floor or trapping set induced error event is observed down to FER $\sim 10^{-7}$. The curves were drawn with at least 30 block-error occurrences.

### 3.4.4 G-LDPC Decoders for Partial Response Channels

In this section, we extend the G-LDPC decoder technique to LDPC-coded PR channels (as depicted in Fig. 2.12) by simulating the rate-0.78 QC-LDPC code of Section 3.1.3 using a G-LDPC III decoder, which was shown to have the best performance among the three G-LDPC decoders.
Figure 3.25: The QC Code G-LDPC III grouping. □ = super-CN consists of 2 SPC-CNs.

Recall that the (2048, 1600) QC-LDPC code has 10 dominant trapping set classes (640 trapping sets) as listed in Table 3.3. By examining the list of unsatisfied checks of these trapping sets, we found that combining 128 SPCs (28.6% of all SPCs) into 64 GCPs pairwise assures each trapping set has at least one GCP bridging it to another trapping set. The performance of this G-LDPC decoder on the PR1 channel is shown in Fig. 3.26. We have simulated the case of no global iterations and five global iterations between the detector and the decoder, respectively. As shown in the figure, for both the G-LDPC III decoder and the conventional SPA decoder, performing global iterations offers around 0.5 dB FER improvement in the water-fall region. For both the G= 0 and G= 5 cases, the conventional SPA decoder curves quickly run into error floors. We remark that it is a general observation that an increased number of iterations does not help the error floor performance. Fig. 3.26 also shows that the error floors for both cases (G= 0 and G= 5) are substantially reduced by the G-LDPC III decoder, which demonstrates the effectiveness of the G-LDPC III decoders on the PR channel.

3.5 Complexity Discussion

As shown in this chapter, the three proposed decoders efficiently eliminate or lower LDPC error floors given knowledge of the trapping sets that cause the floor. Acquiring this knowledge is an off-line task, which can be done by using extensive
Figure 3.26: Performance of G-LDPC decoder III on the high-rate QC code compared with that of the SPA decoder on PR channel.
simulations, graphical search, and/or importance sampling, as discussed in Chapter 2. In this section, we compare the complexities of the three decoders with that of the conventional SPA decoder based on an LDPC code Tanner graph with only single parity-check nodes.

The bi-mode decoder has the lowest complexity of the three decoders due to the fact it is a post-processing decoder whose erasure flagging/recovering process operates only when the conventional SPA decoder is trapped in one of the known trapping sets. When the decoder enters its second decoding mode, bits in proximity to unsatisfied check nodes are flagged according to a trivial algorithm and then iterative erasure decoding which involves only XOR’s is performed. The erasure flags can be one of the bits in the magnitude portion of the channel messages since in the second decoding mode only the sign bits and the erasure locations (flags) are needed. The XOR operators are already present from the SPA algorithm which handles the sign computations at the check nodes in this way. Thus, the additional hardware beyond standard SPA is truly negligible and, because only a few erasure decoding iterations are necessary, the additional number of computations is negligible as well. Erasure decoding can also be realized using the same decoder hardware by setting the LLRs to zeros. Note that no look-up tables are required as in the case for the algorithm in [37].

The pre-pinning technique (direct pinning without the assistance of out codes) requires no additional computation than the non-pinning detector/decoder solution. For the case of state-pinning on the PR channels, the number of branches in the detector trellis is reduced, thus leading to some savings on the branch metric computations. The error floor improvement is obtained at the cost of a certain amount of rate loss depending on the code’s trapping set spectrum. The additional computational complexity required by the post-pinning decoder/detector is: (1) hard-decision decoder(s) for the outer algebraic code(s) and (2) a few additional SPA iterations (usually less than five) to process the pinned bits fed back from the outer decoder(s) or the trellis-pruned detector. Because algebraic decoders have extremely low complexity, whereas the SPA decoder is quite complex, the percentage
complexity increase can be on the order of 1%, depending on the LDPC code length.

As demonstrated in Section 3.4, the G-LDPC decoders, particularly G-LDPC III, substantially improved the error floor performance and moderately improved the water-fall performance, at the cost of higher decoding complexity. The complexity increase is due to the constituent decoder used to process the extrinsic information of the GCP’s. In our experiments, we used the BCJR algorithm to implement the optimal GCP, though lower-complexity suboptimal algorithms, such as soft-output Chase algorithm of Pyndiah [46] or the soft-output Viterbi algorithm (SOVA) [47] can be used. The BCJR algorithm works on a trellis representing the linear block code [44] that corresponds to the generalized constraint of interest. Its complexity is proportional to the product of the number of states, $S$, trellis length, $L$, and the number of core operations per state, $O$. For the three-stage structure of the BCJR algorithm (forward recursion, backward recursion, and completion stage), $O = 3$ binary max* operations, where $\text{max}^*(x, y) \triangleq \log(e^x + e^y)$ (implementable as lookup table). For example, let us consider the G-LDPC III decoder for the Margulis code. Each GCP is a combination of two degree-6 SPC nodes, i.e., $S = 4$ and $L \leq 12$. Thus, the BCJR algorithm needs $S \cdot L \cdot O = 12L$ max* operations to generate the extrinsic information from this GCP to its $L$ neighboring VNs. This amounts to 12 max* operations per extrinsic message.

As for the SPA decoder, each SPC constraint needs $d_c(d_c - 2)$ log-tanh operations [48] (often denoted by “⊞”) to produce extrinsic information to its $d_c$ neighbors. Thus, the two SPC nodes which are combined in the G-LDPC-III decoder requires $2d_c(d_c - 2)$ log-tanh operations to compute the messages to be sent to their $L$ neighbors. That is, since $d_c = 6$ for the Margulis code, $2d_c(d_c - 2)/2d_c = 4$ log-tanh operations per extrinsic message are required. Further, the following relationship holds:

\[
L_1 \bowtie L_2 = \text{max}^*(0, L_1 + L_2) - \text{max}^*(L_1, L_2) = \max(0, L_1 + L_2) - \max(L_1, L_2) + s(L_1, L_2),
\]

(3.1)
where \( s(x, y) \) is a so-called correction term given by

\[
s(x, y) = \log(1 + e^{-|x+y|}) - \log(1 + e^{-|x-y|}).
\] (3.2)

The correction term can also be implemented using lookup table. This leads us to the conclusion that conventional SPA decoder requires 4 correction-term table-lookup operations and 2 max operations per extrinsic message, compared to 12 max* table-lookup operations for the G-LDPC-III decoder. Hence the computational complexity increase of the G-LDPC-III decoder versus the SPA decoder is 200%.

A similar analysis holds for the G-LDPC-III decoder on the PR channel. The high-rate QC-LDPC code has SPC nodes with degrees 21, 22, 23 and 25, and each GCP is a combination of two SPC nodes, therefore \( L \leq 50 \). The BCJR GCP decoder requires \( S \cdot O = 12 \) max* table-lookup operations per extrinsic message. For comparison, the conventional SPA decoder requires \( d_c - 2 \approx 20 \) log-tanh operations per extrinsic message. Thus, there is a 40% complexity decrease by using G-LDPC III decoder on this QC-LDPC code.

### 3.6 Conclusions

We have presented several decoding strategies for lowering floors in LDPC codes. All of the techniques presented succeeded in lowering the floors of the codes we studied by orders of magnitude and are fully generalizable to other LDPC codes. The techniques have varying levels of complexity and the chosen technique would depend on the performance/complexity requirements of the application. The bimode decoder and its variations should be considered in the early stages of the system design because they have the lowest complexity. On the other hand, as shown in Fig. 3.27 which compares the performances of the three techniques on AWGN channel, the G-LDPC III decoder yields the best performance.
Figure 3.27: Performance comparison of all decoders on AWGN channel: the Margulis code and the low-rate QC code.
Product codes are used in many applications, particularly in magnetic and optical recording systems where errors are isolated and bursty. In the past decade, so-called turbo product codes [46][55] have been shown to have many desirable properties, including near-capacity performance and low-complexity (for a turbo code). Aitsab and Ryndiah presented the performance of RS block turbo code and proposed their use for data storage applications [55]. However, the RS turbo product codes require soft iterative decoding of the component RS codes, which is still quite complex.

We take a smaller, lower-complexity step toward improving the decoder of a product code. The approach is inspired in part by the works of Burshtein and Miller [56] and Jiang and Narayanan [57]. In [56], the authors presented algorithms for the maximum-likelihood (ML) decoding of binary-codes on the binary-erasure channel (BEC). In [57], the authors presented a decoding algorithm for the binary image of a Reed-Solomon (RS) code on the AWGN channel, which is ML when specialized to the BEC as we show in Section 4.2. In the new light of these papers, we seek new decoding approaches for RS-code-based product codes.

Specifically in this chapter, we present a dual-mode decoding technique for product codes in which the rows and columns may comprise either Reed-Solomon (RS) codewords or LDPC codewords. For the sake of discussion here, first consider the case in which the product code comprises only RS codewords. In the first decoding mode, standard algebraic decoding is used on the rows and column codewords. For the second mode, the column code takes on a different character in which “slices” of the row RS codewords are treated as packet-sized symbols within a column codeword, to which an ML packet-erasure decoding algorithm is employed. The ML decoder for this packet-based column code resolves the erroneous packets (erasures)
flagged by the row code (or a CRC code) during the first decoding mode.

This technique improves upon our earlier work in [58] in which we used so-called packet-LDPC codes [59] for error prevention in tape recording systems. We show that packet-based column codes already exist in current tape recording formats, a fact that may be exploited by a dual-mode decoding system. Further, as discussed in [58], decoding of packet-based codes on the packet-erasure channel (PEC) is very simple since it involves only binary XORs, regardless of whether the code is based on an LDPC code or an RS code. The ML decoder also requires a Gaussian elimination preprocessor, requiring $O((n - k)^3)$ binary XORs, as explained below. Still, the additional complexity due to the second decoding mode is manageable, which makes this dual-mode decoding idea promising.

The rest of the chapter is organized as follows. Section 4.1 reviews packet-LDPC codes and packet-based RS codes (or packet-RS codes for short). Section 4.2 discusses maximum-likelihood decoding of linear block codes on the BEC and the PEC. Section 4.3 first discusses dual-mode decoding. It then presents the architecture of two product code designs for use with dual-mode decoding: one employs a packet-RS code as the column code and the other employs a packet-LDPC code as the column code. Included among the results is the performance improvement attainable by dual-mode decoding of the ECMA-319 tape standard. Finally, Section 6.5 makes some concluding remarks.

4.1 Packet-level Codes

The original packet-based codes for the erasure channel were the tornado codes of Luby et al. [59] whose goal was to provide reliable transmission of packets over the Internet. Tornado codes can also undergo dual-mode decoding, but we favor LDPC codes and RS codes here, both of which appear to be more serious candidates for the tape storage application. We discuss packet-LDPC codes and packet-RS codes in the following.
4.1.1 Packet-LDPC Codes

LDPC codes usually refer to bit-level codes (which we will call bit-LDPC codes) where a block of bits is transmitted over a channel that introduces bit-level errors, erasures, noise, etc. In this case, the variable nodes in a code’s Tanner graph correspond to code bits, and encoding and decoding operations are bit-wise. However, inspired by the work reported in [59], we showed in [58] that packet-level LDPC (or packet-LDPC) codes working on the PEC are well suited for a tape channel when a reliable packet-erasure-flagging inner code is available.

The matrix and graphical representations as well as the encoding and decoding procedures for bit-LDPC codes can all be trivially extended to packet-LDPC codes. An \((n, k)\) packet-LDPC code can be fully specified by its parity-check matrix, \(H\), and its packet length, \(L\). The \(H\) matrix is generally one derived for a bit-LDPC code (binary with dimension \((n - k) \times n\)). The size of a packet can be hundreds of bits or several megabytes, depending on the application. In the Tanner graph specified by \(H\), each variable node corresponds to a packet instead of a bit so that each parity check equation requires that the packets associated with it sum to the all-zero packet (“sum” here is bit-wise XOR). For example, a codeword of an \((n, k)\) packet-LDPC code has \(nL\) bits and can be represented by \(\mathbf{c} = [\bar{c}_1 \bar{c}_2 \ldots \bar{c}_n]\), where \(\bar{c}\) is an \(L \times n\) matrix and \(\bar{c}_i\) is a binary \(L\)-tuple \([c_1^i \ c_2^i \ldots c_L^i]^T\) representing a packet. An erased packet is resolved in way analogous to how an erased bit would be solved in a bit-LDPC code. Observe that a packet-LDPC codeword with \(L\) bits in each packet can be viewed as a set of \(L\) independent bit-LDPC codewords and so a packet-LDPC decoder might be deemed as \(L\) bit-LDPC decoders operating in parallel.

As shown in [58], packet-LDPC codes are naturally suited as the column code in product code architectures. A row code, which is decoded first, can be used on an individual packet or several packets, and it generates erasure flags when it fails to decode. The packet-LDPC code, operating as the column code, sees only correct packets or packets erased by the row code, under the assumption of perfect error detection by the row code. Thus, the packet-LDPC code effectively operates on a
PEC. Since the $H$ matrix of a packet-LDPC code is sparse, iterative decoding can
be directly applied to it, which is simply done by iteratively searching for parity
check equations involving a single erased packet and solving for the value of that
packet. Further details of packet-LDPC codes and the decoding algorithm may be
found in [58].

### 4.1.2 Packet-RS Codes

There has been a recent flurry of research activity on the decoding of the binary im-
age of RS codes for various binary-input channels (see [57] and its references). Part
of the appeal of RS codes is that they are known to have good distance properties
(both binary and non-binary) and it is easy to design RS codes for a large range of
lengths and rates. Consider an $(n, k)$ narrow-sense RS code over $\text{GF}(2^m)$, for which
$n = 2^m - 1$, minimum distance $d_{\text{min}} = n - k + 1$, and canonical parity-check matrix

$$H = \begin{bmatrix}
1 & \alpha & \ldots & \alpha^{n-1} \\
1 & \alpha^2 & \ldots & \alpha^{2(n-1)} \\
\vdots & & \ddots & \vdots \\
1 & \alpha^{d_{\text{min}}-1} & \ldots & \alpha^{(d_{\text{min}}-1)(n-1)}
\end{bmatrix}, \quad (4.1)
$$

where $\alpha$ is a primitive element in $\text{GF}(2^m)$. Given a basis of $\text{GF}(2^m)$, denoted by
$\{\gamma_1, \gamma_2, \ldots, \gamma_m\}$, each RS codeword can be expanded in $\text{GF}(2)$ with each symbol in
$\text{GF}(2^m)$ replaced by a binary $m$-tuple, i.e., each RS codeword is mapped to a binary
vector (its binary image) of length $nm$. Correspondingly, the binary parity-check
matrix of this code, can be obtained by replacing each entry in (4.1) with an
$m \times m$ matrix over $\text{GF}(2)$. Thus, the binary image of $H$, denoted by $H_b$, has dimension
$(n - k)m \times nm$.

Just as a binary LDPC code can be used as a packet-LDPC code on the PEC by
replacing bits by packets, the binary image of a RS code can be used as a packet-RS
code, again replacing bits by packets. For example, an RS-based erasure code generated
from $\text{RS}(n, k)$ over $\text{GF}(2^m)$ is a binary linear block code of parameters $(nm, km)$
and parity check matrix $H_b$; while using it as packet-RS code of packet length $L,$
each codeword has \( nmL \) code bits and \( kmL \) information bits. For packet-RS codes, no operations over \( GF(2^m) \) are performed. Rather, the situation is analogous to that of the packet-LDPC code where bit-wise XORs of packets are performed. The only difference is that \( \mathbf{H}_b \) is not low density. This situation is resolved, however, by using the ML decoding algorithm for the BEC or PEC described in the next section.

### 4.2 Maximum-Likelihood Decoding on The Erasure Channel

As indicated in the previous section, the packet-level iterative decoding algorithm over the PEC is essentially identical to that of the bit-wise decoding algorithm over the BEC. Thus, we focus here on ML decoding of a linear block code operating on the BEC. The discussion is heavily influenced by [56] and [57].

Suppose we transmit any codeword in a linear block code \( \mathcal{C} \) over a BEC and we receive the word \( \mathbf{r} \) whose elements are taken from the set \( \{0, 1, e\} \), where \( e \) is the erasure symbol. Let \( J \) represents the index set of unerased positions in \( \mathbf{r} \) and \( J' \) the set of erased positions. It is easy to show that the ML decoder chooses a codeword \( \mathbf{c} \) in \( \mathcal{C} \) satisfying \( c_j = r_j \) for all \( j \) in \( J \) and, further, that this is a solution to the equation

\[
\mathbf{c}_J \mathbf{H}_{J'}^T = \mathbf{c}_J \mathbf{H}_J^T.
\]  

Here, \( \mathbf{H}_J \ (\mathbf{H}_{J'}) \) is the submatrix of the code’s parity-check matrix \( \mathbf{H} \) obtained by taking only the columns of \( \mathbf{H} \) corresponding to \( J \) (\( J' \)), and similarly for \( \mathbf{c}_J \ (\mathbf{c}_{J'}) \). Equation (4.2) follows from the fact that \( \mathbf{0} = \mathbf{c}\mathbf{H}^T = \mathbf{c}_J \mathbf{H}_{J'}^T + \mathbf{c}_J \mathbf{H}_J^T \). There is a unique solution if and only if the rows of \( \mathbf{H}_{J'}^T \) are linearly independent, in which case the elements of the unknown \( \mathbf{c}_{J'} \) may be determined by Gaussian elimination. Because \( \mathbf{H}_{J'}^T \) has \( n - k \) columns, its rows can be linearly independent only if \( |J'| \leq n - k \), giving us a necessary condition for the uniqueness of the solution to (4.2). Also, because any \( d_{\text{min}} - 1 \) rows of \( \mathbf{H}^T \) (hence \( \mathbf{H}_{J'}^T \)) are linearly independent, (4.2) is guaranteed to have a unique solution whenever \( |J'| < d_{\text{min}} \).

One may solve for the unknowns \( \mathbf{c}_{J'} \) as follows. First apply Gaussian elimination
to $\mathbf{H}$, targeting the columns in the index set $J'$ to produce a modified matrix $\tilde{\mathbf{H}}$, which we assume without loss of generality to be of the form $\tilde{\mathbf{H}} = [\tilde{\mathbf{H}}_J \tilde{\mathbf{H}}_{J'}]$. The submatrix $\tilde{\mathbf{H}}_{J'}$ will have the form

$$\tilde{\mathbf{H}}_{J'} = \begin{bmatrix} \mathbf{T} \\ \mathbf{M} \end{bmatrix},$$

(4.3)

where $\mathbf{T}$ is a $|J'| \times |J'|$ lower-triangular matrix with ones along the diagonal and $\mathbf{M}$ is an arbitrary binary matrix. Now one may solve for the unknowns in $\mathbf{c}_{J'}$ by successively solving the $|J'|$ parity-check equations represented by the top $|J'|$ rows of $\tilde{\mathbf{H}}$. This is possible provided none of the top $|J'|$ rows of $\tilde{\mathbf{H}}_J$ is all zeros. However, none of these rows can be all zeros since, for example, if row $p \leq |J'|$ of $\tilde{\mathbf{H}}_J$ is all zeros, then the corresponding row of $\tilde{\mathbf{H}}$ has Hamming weight one. Any word with a one in the same position as the single one in this row of $\tilde{\mathbf{H}}$ will have $\mathbf{c}\tilde{\mathbf{H}}^T \neq \mathbf{0}$ and thus is not a valid codeword. However, half the codewords of any reasonable linear code (i.e., the generator matrix $\mathbf{G}$ has no all-zero columns) have a one in any given position.

We remark that the adaptive belief propagation (ABP) decoding algorithm of [57] for binary-image RS codes on the AWGN channel is ML when applied to the BEC. This is argued as follows. The ABP algorithm in [57] first performs Gaussian elimination on $\mathbf{H}_b$ to reduce the submatrix corresponding to the least reliable bits in the received word to an identity matrix. The sum-product algorithm is then applied to this modified matrix, and then Gaussian elimination and sum-product decoding are both repeated. The full adaptation and decoding process is repeated for several iterations. Upon comparison with the ML decoding algorithm described above, this ABP algorithm is clearly seen to be ML on the BEC. This is because on the BEC the erased bits are the least reliable bits. If the submatrix corresponding to those erased bits (i.e., $\mathbf{H}_{J'}$) is reduced to an identity matrix, it means the columns of $\mathbf{H}_{J'}$ are linearly independent, so that the erased bits can be solved from (4.2). Hence, the ABP algorithm on the BEC can be viewed as a special case of the ML decoder, with $\mathbf{T}$ in (4.3) being an identity matrix. Note that the adaptive decoder on the BEC
need only perform Gaussian elimination once since the locations of the erasures do not change, unlike the AWGN case where the least reliable bits generally change.

Fig. 4.1 shows the performance of several binary-RS codes of different lengths and rates on the BEC with the ML decoding algorithm described above. The performance is characterized in terms of the probability that a bit is still erased after decoding. The capacity of the BEC is $C = 1 - \epsilon$ where $\epsilon$ is the loss (erasure) rate of channel. From Fig. 4.1 we see that sufficiently long RS codes essentially achieve capacity. An LDPC (8192, 6717) code (used in a later section) with rate $R = 0.82$ under iterative decoding is also shown in the figure, and compared with the binary image of the (819, 672) RS code over $\text{GF}(2^{10})$. The LDPC code, designed for a low floor, is described further below. The RS(64,54) code over $\text{GF}(2^8)$ is also included in the figure as a (512, 432) binary code since this code is part of the ECMA-319 standard discussed below.

4.3 Dual-Mode Decoding for Tape Storage

As described above, dual-mode decoding of product codes involves a first round of decoding in standard fashion (e.g., algebraic decoding of the rows and then the columns of a RS-based product code) followed by a second round in which the column code is re-interpreted as a packet-erasure-resolving code, where erasures are flagged by the row code or an auxiliary CRC code. In this section, we make these ideas clearer by considering specific examples. The examples also demonstrate the efficacy of the dual-mode approach. The first two examples consider RS-based product codes and the third example uses an LDPC code for the column code.

4.3.1 Dual-Mode Decoding of RS Codes

Dual-Mode Decoding of ECMA-319

The ECMA-319 standard employs two RS codes concatenated with two levels of interleaving (intra-track and inter-track). As depicted in Fig. 4.2, user data, after being compressed, is grouped into 404, 352-byte chunks which are the ECC-protected
Figure 4.1: Performance of binary-RS codes with ML decoding on the BEC.
entities known as *data sets*. Each data set is then broken into 16 two-dimensional arrays called *sub data sets*, which will be protected by two different RS encodings, $C_1$ and $C_2$, in turn. The structure of each sub data set is shown in Fig. 4.2. The data in each row is protected by an inner code, $C_1$, an even-odd interleave of two RS(240, 234) codewords over GF($2^8$), designed to allow correction of a relatively high background error rate. We note that the bytes are represented vertically in Fig. 4.3 and that bytes are transmitted intact to preserve the burst-error capabilities of both codes. The data in each column is protected by an outer code $C_2$, RS(64, 54) code over GF($2^8$), designed to allow correction of long error bursts (with the aid of the row-column interleaver). After encoding, the size of one encoded sub data set is 30,720 bytes, and the size of one encoded data set is 491,520 bytes (3,932,160 bits); the overall code rate is $R_{\text{total}} = R_1 R_2 = 0.82$.

In addition to the row-column interleaver within a sub data set which facilitates
the correction of moderate-size error bursts, there is also an interleaver within a data set (i.e., among the 16 sub data sets) which facilitates the correction of very long error bursts. This second interleaver distributes the data set uniformly over eight tracks so that, upon playback, the de-interleaver will uniformly distribute the eight tracks among the 16 sub data sets. In the event of a lost track (e.g., due to a head clog or media-defect-induced synchronization loss), the erroneous bytes will be flagged by \( C_1 \) as erasures, and will occupy eight rows in each of the 16 sub data set interleaver blocks (Fig. 4.2). Since \( C_2 \) is capable of correcting 10 erasures, it will correct all eight erased rows within each sub data set and, hence, an entire lost track.

In the dual-mode decoding of each sub data set (i.e., each product code), the decoding of row code \( C_1 \) is the same as in the standard (hard-decision decoding (HDD)). The column code \( C_2 \) has two decoding modes. In the first mode, algebraic HDD is performed on each column (\( C_2 \) codeword) in Fig. 4.3. In the second mode, we view each 480-bit slice along the rows in Fig. 4.3 as a packet, and use a high rate CRC code to declare packet erasures. Then all 480 \( C_2 \) codewords are grouped together as a single packet-erasure codeword derived from the binary image of the RS(64,54) code, where packets are 480 bits long and there are 8 packets along each
Figure 4.4: Performance of ECMA-319 with dual-mode decoding.
sub data set row comprising two $C_1$ codewords. Lastly, ML packet-erasure decoding is performed on this single packet-erasure codeword.

The dual-mode decoder has three steps:

1. $C_1$ RS(240, 234), $m = 8$, algebraic HDD
2. $C_2$ RS(64, 54), $m = 8$, algebraic HDD
3. $C_2$ packet-RS(64, 54), $m = 8$, $L = 480$, ML packet-erasure decoding (erasures from auxiliary CRC code in each packet)

In Fig. 4.4, the performance of the dual-mode decoder is compared with that of the standard algebraic decoder (steps 1 and 2 only) on both the AWGN channel and the AWGN channel conditioned on one erased track (out of eight). For the latter case, we assume that with the help of intra-track and inter-track interleaving, the lost track is distributed over the 16 sub data sets and also within each sub data set. Thus the simulator generates bursts that erase $1/8$ of each sub data set. The performance metric is probability of data set error ($P_{ds}$). Since each sub data set is a packet-RS codeword and there are 16 sub data sets in each data set, we have $P_{ds} = 1 - (1 - P_{cw,packet-RS})^{16}$. The large gains provided by dual-mode decoding are evident. Our results assume a 32-parity-bit CRC code, resulting in a 0.28 dB code rate loss which is incorporated in the figures.

**Dual-Mode Decoding with a Long RS Column Code**

We now consider the dual-mode decoding of a product code architecture which combines all 16 sub data sets into a single product code array, with $C_2$ replaced by a RS codeword 16 times its length. Specifically, the size of a sub data set and the inner code $C_1$ are kept the same as the ECMA-319 standard, while $C_2$ is replaced with a RS code over GF($2^{10}$) with parameters (819, 672). The 16 sub data sets are stacked as in Fig. 4.5, and each thin column is a $C_2$ codeword. Again there are 8 packets in every two $C_1$ codewords, and each packet has 480 bits. Note that each $C_2$ codeword has 819 symbols, i.e., 8190 bits, while there are $64 \times 8 \times 16 = 8192$
Figure 4.5: Structure of a long RS code data set.

bits in each column of a data set, so we are deviating from the ECMA-319 standard slightly. The size of an encoded data set is 3,931,200 bits and the overall code rate is 0.8.

For this scheme, the decoding procedure used is similar to that in Section 4.3.1. We also examine the improvement afforded by repeating Steps 1 and 2 $q$ times ($q = 1, 2, 3$) prior to the second decoding mode which we call Procedure 1. To examine the gain provided simply by the second decoding mode, we present curves corresponding to only Steps 1 and 2 which we call Procedure 2, to be compared with the Procedure 1, $q = 1$ curve. The performance on the AWGN channel is shown in Fig. 4.6 in terms of $P_{ds}$. We can see that $q = 1$ is sufficient and so we choose $q = 1$ hereafter to reduce complexity. The performance for the AWGN channel plus one
Figure 4.6: Performance of long RS data set on AWGN with dual-mode decoding.
Figure 4.7: Performance of long RS data set on AWGN plus one erased track with dual-mode decoding.
erased track case is shown in Fig. 4.7.

In Figs. 4.6 and 4.7 the right-most curve is a RS/RS concatenation we presented in [58] with a very long RS code over GF\( (2^{12}) \) as the outer code. In that scheme, the row decoder does errors-only HDD, and the column decoder does erasures-only algebraic decoding. The structure of a data set with this RS code over GF\( (2^{12}) \) as column code is shown in Fig. 4.8, which has the same data set size as the ECMA-319 standard and its overall code rate is 0.8. It is presented here to illustrate the effectiveness of dual-mode decoding.

4.3.2 Dual-Mode Decoding of LDPC Codes

Although the long RS code system described above provides excellent performance, it is at the expense of high complexity, owing to the Gaussian elimination requirement. In this subsection we consider replacing the long RS code with an LDPC code of the same length. The LDPC code has not been optimized, except it was designed for a low error-rate floor and efficient encoding [58]. Thus, in this RS-code/LDPC-code product code with dual-mode decoding of the LDPC code, we will trade off the erasure correction capability for lower complexity.

In this architecture, the inner code is the same as that of the ECMA-319 standard and the outer code is an LDPC \((8192,6717)\) code. The structure of a data set is almost the same as in Fig. 4.5, except that the number of thin rows is 8192. This
Decoding Procedure 1:
step 1: $C_1$ HDD
step 2: $C_2$ Bit Flipping HDD
step 4: $C_2$ Packet-LDPC erasure decoding with help of CRC

Decoding Procedure 2:
step 1: $C_1$ HDD
step 2: $C_2$ Bit Flipping HDD

Standard long RS-RS:
$C_1$ (120, 111) 8bit/sym
Errors-only decoding
$C_2$ (4096, 3546) 12bit/sym
Erasures-only decoding

Figure 4.9: Performance of RS/LDPC data set on AWGN with dual-mode decoding.
scheme has the same data set size as ECMA-319 and the overall code rate 0.8.

Decoding for this structure also has three steps:

1. $C_1$ RS(240, 234), $m = 8$, algebraic HDD

2. $C_2$ LDPC(8192, 6717), bit flipping HDD

3. $C_2$ Packet-LDPC(8192, 6717), $L = 480$, iterative packet-erasure decoding (erasures from auxiliary CRC code in each packet)

For the first decoding mode of the bit-level LDPC code, we use the bit flipping algorithm [23] with 10 iterations. The performance is simulated on the AWGN channel and the AWGN plus one track erased case in Fig. 4.9 and Fig. 4.10, respectively. The performance is also compared with decoding using Step 1 and Step 2 only (note that this procedure cannot deal with the situation of one lost track out of eight) and the long outer RS code over GF($2^{12}$). In simulation of the second mode, we use iterative packet-erasure decoding instead of ML packet-erasure decoding since LDPC codes have sparse parity check matrices which makes iterative decoding possible. From our simulations we can see that the second mode decoding of LDPC code does provide noticeable performance improvement, even over the very complicated 12-bit RS-RS scheme. Note also that for the AWGN plus one track lost case, this scheme does not perform as well as the RS-RS scheme in the high SNR region.

4.4 Conclusions

A dual-mode decoding method for product code structures is proposed. The performance of three product codes is studied with an eye toward application to tape storage. With this technique, the dual-decoded product code system sacrifices some complexity while enjoying a large performance improvement. The only modification necessary to any existing system is the addition of a CRC code for packet-error detection. Since iterative decoding on the BEC is very simple as it involves only binary XORs on a sparse graph, the proposed scheme using dual-mode decoding of an LDPC code adds only a little complexity. For the dual-mode decoding of a
Figure 4.10: Performance of RS/LDPC data set on AWGN plus one erased track with dual-mode decoding.
RS code, the major complexity introduced is Gaussian elimination (GE) in the ML decoder. In general, GE for a $(n - k) \times n$ binary matrix can be accomplished in time proportional to $O((n - k)^3)$ binary XORs. Note also that we only need to perform GE once to decode an entire sub data set. Thus, for dual-mode decoding of ECMA-319, complexity is on the order of $80^3/480 = 1067$ XORs per $C_2$ codeword. There exists much faster GE algorithms in literature which can perform GE within $O((n - k)^{2.37})$, i.e., about 70 binary XORs per $C_2$ codeword. Hence, the complexity introduced is acceptable.
CHAPTER 5

CONCATENATING A STRUCTURED LDPC CODE AND A CONSTRAINT CODE

Error-control codes (ECCs) and runlength-limited (RLL) codes each play critical roles in maintaining data integrity on storage devices, with ECCs doing so directly by inserting data-dependent redundant bits via some encoding algorithm, and RLL codes doing so indirectly by facilitating timing recovery at the receiver. Their simultaneous presence in a storage system introduces conflicting requirements in the receiver design when soft ECC decoding is a goal. On the one hand, if RLL encoding were to follow ECC encoding, soft ECC decoding would be essentially precluded at the receiver since, for complexity reasons, the RLL decoder must produce hard decisions for the ECC decoder which follows it. On the other hand, if RLL encoding were to precede ECC encoding, the RLL constraint would be lost with the addition of ECC parity bits unless provisions were made to accommodate their addition. Similar comments hold for DC-free codes.

Several recent papers have addressed this dilemma when a low-density parity-check (LDPC) code is used for the ECC code [62]-[64]. Fan [62] considers several such techniques, including a parity-insertion technique which has been used in [65] and [66]. We extend the idea in a manner described in Section 5.1. References [63] and [64] consider “deliberate-error” techniques whereby selected LDPC code bits are inverted in order to ensure runlength-constraint compliance. In [64], this is facilitated by first adding a “guided scrambling” pseudo-random sequence to the LDPC codeword.

We add two other requirements to the RLL-LDPC concatenation problem. First, we require that the LDPC code possess structure to facilitate encoder and decoder implementation. Second, the nominal $H$ matrix of the LDPC code must be modified to accommodate both the RLL constraint and the ability to correct long erasure
bursts. One way to accomplish this is via the deliberate-error techniques of [63] and [64], but we believe these techniques result in substantial performance losses, especially in the floor region of the LDPC code error-rate curve. We propose a simple reverse-concatenation scheme combined with a structured permutation of \( H \) to satisfy these additional requirements, at the expense of negligible complexity increase and only the RLL code-rate loss. We also propose a solution which utilizes a DC-free code and conforms to the above requirements at the same time.

5.1 RLL Code Solution

The components of our design are a rate-50/51 RLL code with a \((0, 14)\) runlength constraint [68] and a rate-0.911(4592, 4182) structured LDPC code [31], corresponding to a combined code rate of 0.89. The nominal parity-check matrix \( H \) for the LDPC code has the form \( H = [H_1 \ H_2] \), where \( H_1 \) consists of an array of square \( Q \times Q \) circulant permutation matrices and \( H_2 \) is an \( m \times m \) “dual-diagonal” matrix \((m \) is the number of parity bits). A block of 4100 bits, comprising 4096 sector data bits plus four “dummy” bits, is RLL encoded systematically to produce 4182 bits at the input to the LDPC encoder, which in turn yields a 4592-bit LDPC codeword (Fig. 5.1). The 410 LDPC parity bits can simply be inserted uniformly among the 4182 systematic bits which relaxes the RLL constraint to \((0, 16)\). This parity-insertion can be realized with no complexity increase by inserting the parity columns of \( H \) among its systematic columns, from which an alternative parity-check matrix is obtained for encoding and decoding.

Such a modification of \( H \) is not sufficient, however, to obtain a code with a large value of \( L_{\text{max}} \), where \( L_{\text{max}} \) is the guaranteed burst-erasure length correction capability corresponding to the code’s parity-check matrix. An algorithm for calculating \( L_{\text{max}} \) can be found in [67]. When \( H \) possesses the structure described above, the maximum burst-erasure length that is guaranteed correctable is only \( L_{\text{max}} = 161 \) bit erasures, an efficiency of only 161/410 = 0.39. \( L_{\text{max}} \) can be greatly improved by randomly permuting the columns of \( H \) matrix. However, an \( H \) matrix with
structure is desired as it facilitates both encoder and decoder implementation.

We may modify $H$ as $H^* = [H_1 \Pi_a \ H_2] \Pi_b$ to simultaneously obtain $L_{\text{max}}$ near the maximum possible for this code, accommodate the reverse concatenation with the RLL code by parity insertion and also keep the structure of the code. In this expression, $\Pi_a$ is a deterministic (structured) permutation of the columns of $H_1$. For the LDPC code we used in this paper, $H_1$ is a $410 \times 4182$ matrix and $Q = 82$, so $H_1$ is a $5 \times 51$ array of $82 \times 82$ circulant permutation matrices. $\Pi_a$ interleaves $H_1$ such that the column index $i^*_c$ of $H_1 \Pi_a$ and the column index $i_c$ of $H_1$ have this relation: $i^*_c = 51 \cdot (i_c \ mod \ 82) + \lfloor i_c/82 \rfloor$. $\Pi_b$ is also a deterministic column permutation of $H$ which corresponds to inserting the columns of $H_2$ evenly among the columns of $H_1 \Pi_a$. Because $\Pi_a$ and $\Pi_b$ are structured, $H^*$ is structured. $H^*$ corresponds to a code with $L_{\text{max}} = 260$ (efficiency 0.63) compared to a maximum of 272 found using random permutations. The RLL constraint corresponding to $H^*$ and the above RLL code is $(0, 16)$.

The performance of this scheme on a binary-input AWGN channel is shown in Fig. 5.2 in terms of frame error rate (FER) and bit error rate (BER). Both FER and BER refer to errors in only the systematic bits. We see that the performance gap between the LDPC code alone and the RLL/LDPC reverse concatenation scheme is very small, and can be largely attributed to the RLL code rate loss: $10 \log_{10}(51/50) = 0.086$ dB. For FER, the performance gap should be attributed to rate loss only,
Figure 5.2: Performance of the RLL solution on the AWGN channel.
since, if the LDPC decoder corrects all of the errors from the channel, the RLL decoder will definitely succeed; if LDPC decoder fails, due to the property of the RLL code design, the residual errors will not propagate outside of the RLL code block. Fig. 5.2 can be compared to Fig. 5 in [63] where the code length (4096), rate (0.83), runlength constraint [(0, 15)], and performance are comparable. However, because the scheme in [63] deliberately adds errors to LDPC codewords prior to recording, we expect our floor to be lower.

5.2 DC-free Code Solution

In this section, we show how to modify the scheme we proposed above to achieve DC-free channel sequences. DC control can be divided into two parts: control on the systematic bits and control on the parity bits.

DC control on the systematic bits modifies the RLL code to suppress DC by adding redundancy so that a bounded running digital sum (RDS) is achieved. To understand how this is done, we briefly describe the structure of the RLL code [68]. As depicted in Fig. 5.3, the encoding is done in several steps: First, each 50-bit data block is divided into five 10-bit symbols (A to E). Second, one of the symbols, say D, is selected for rate 10/11 encoding, leaving the four other symbols uncoded. Third, the 11 encoded bits are partitioned into 4 parts (x_0, x_1, x_2 and x_3) which consist of 3, 3, 3 and 2 bits, respectively. These are then mapped to y_0, y_1, y_2 and y_3 to prohibit all-zeros and all-ones patterns. Next the y_i’s are inserted among the four uncoded symbols. Details about the 10/11 encoding and x_i to y_i mapping can be found in [68].

Since there are 40 uncoded bits in each RLL codeword whose modification will not change the RLL constraint, these uncoded bits together with another three redundant bits will be used to control the RDS, hence, the DC content of the bit stream. The DC control is done by the so-called guided scrambling method [69]. Among the 50 input bits, symbol D is encoded and partitioned as described above. The four uncoded symbols together form a source word X which is then preceded
by two bits to produce the $2^2$ elements of the set $T_X$ (each element has 42 bits).

Scramble all the vectors in $T_X$ with scrambler polynomial $x^7 + x^5 + 1$ to produce set $T'_X$. Further, precede the vectors in $T'_X$ with one bit to obtain the set $T''_X$ which has 8 elements. Then scramble all vectors in this new set with polynomial $x + 1$ to produce the selection set $S_X$, which has 8 43-bit vectors. When transmitting the word $X$, the “best” vector in $S_X$ is selected for transmission, according to the minimum $RDS$ criterion, which is then divided into four sub-words with 10, 11, 11 and 11 bits. The RLL-encoded 11 bits are mapped as before and inserted among the four sub-words. Note that the second scrambler produces a set consisting of pairs of words of opposite disparity, with which the RDS can be bounded. Simulation shows this method yields a rate-50/54 (0, 15) RLL constrained code with $|RDS| \leq 20$. At the receiver, the DC-free RLL decoder de-scrambles the received word followed by the (51, 50) RLL decoding.
A certain fraction of parity bits are selected to control the DC component of the parity bits. Based on the accumulated RDS of all previously recorded codewords, the systematic bits of the current codeword, and the unselected parity bits of the current codeword, the transmitter chooses whether or not to flip the polarity of the selected parity bits. Note that this is not a polarity switch code since no redundancy is added. At the receiver, a hard-decision parity-polarity detector is added, which calculates the syndrome $\bar{s}$ with the polarities of the received parity bits unchanged and the syndrome $\bar{s}'$ with the polarities of the selected parities flipped. If $w_H(\bar{s}') < w_H(\bar{s})$ where $w_H(\cdot)$ is Hamming weight, the parity detector flips the signs of the samples corresponding to the selected parities in the received word; otherwise, the signs are unchanged. In order to have reliable parity recovery, the selected set of parities needs to be carefully chosen. Since the LDPC code used in this paper has a dual-diagonal sub-matrix corresponding to the parity bits, if we label all the parity bits as 1, 2, ..., $M$, the selected parity set is chosen to be all the parity bits with odd labeling. By doing so, the difference between the Hamming weight of the two syndromes is maximized, thus maximizing the correct recovery probability. At the transmitter,
after the entire DC control process is done, we evenly insert the parity bits among the systematic bits as described in Section 5.1.

Fig. 5.4 shows the diagram of this scheme on both an AWGN channel and a PR1 channel with precoder $1/(1 \oplus D)$ and BCJR channel detector. As we know for the PR1 channel case, in addition to the iterations of LDPC decoder, if we exchange soft information between the BCJR detector and the LDPC decoder several times (global iteration, shown in Fig. 5.4 as the loop between BCJR detector and LDPC decoder), the performance will be improved. Notice that there needs to be a polarity flipping between the BCJR detector and the LDPC decoder. The polarity detector will calculate syndromes only in the first global iteration. Later the syndrome information is stored in the detector/decoder and after each iteration the signs of the soft information of the selected parities will be flipped if the syndromes indicate a polarity flipping at the transmitter. The precoder relaxes the RLL constraint to $(0, 17)$ and the channel sequence (after precoder) is still DC-free.

The performances of this scheme on both the AWGN channel and the PR1 channel are shown in Fig. 5.5. On AWGN channel, the performance of DC-free RLL/LDPC concatenation is about 0.4 dB worse than the LDPC-only case. Of this gap, $10\log_{10}(54/50) = 0.33$ dB is attributed to rate loss. We observe the same performance difference on the PR1 channel. Again, we expect the floor of our scheme to be low.
Figure 5.5: Performance of the DC-free RLL solution.
CHAPTER 6

PERFORMANCE ANALYSIS OF LDPC-CODED MAGNETIC RECORDING SYSTEMS

Low-density parity-check (LDPC) codes have been demonstrated to have near-capacity performance on many channels in the past decade \cite{70,9,7}. Because of this, their potential has been studied for many applications, including wireless communication channels \cite{71,72} and magnetic recording channels \cite{62,77}. Their effectiveness for magnetic storage is particularly challenging due to: the binary-input constraint, severe intersymbol interference, colored noise, including pattern-dependent noise, and a requirement for a very low sector error rate. In the present chapter, we examine the applicability of LDPC codes to magnetic storage via computer simulations using a perpendicular recording model with pattern-dependent media noise. The application of LDPC codes to perpendicular magnetic recording has been studied in prior works \cite{74,78}, but none of these papers contain all of the following components:

- a length $N = 4551$, dimension $K = 4096$, rate-0.9 LDPC code that is a structured irregular repeat-accumulate (S-IRA) code
- comparison to a Reed-Solomon (RS) code of the same length and rate, both in terms of SNR gain and user density gain
- a PR1 partial response equalization target
- a pattern-dependent noise-predictive (PDNP) soft-output partial response detector
- a discussion of the need for an interleaver or, equivalently, permuted columns in the code’s parity-check matrix $H$. 
media-noise percentages of 5%, 50%, and 95%

\[ \begin{align*}
a_k & \in \{ \pm 1 \} \\
b_k & \in \{ -1, 0, +1 \} \\
\Delta t_k & \\
\Delta w_k & \\
n(t) & \\
e(t) & : \text{electronics noise}
\end{align*} \]

\[
\begin{array}{c}
\text{Matched Filter} \\
\text{Equalizer}
\end{array}
\]

\[
\begin{array}{c}
\text{Baud-rate (1/T_c) sampling to detector}
\end{array}
\]

Figure 6.1: Continuous-time channel model and receiver front end.

The results in this chapter give insights to the potential gain offered by an LDPC code relative to a RS code. It also demonstrates the importance of the proper ordering of H matrix columns, an issue also considered in [79]. This chapter is organized as follows. In Chapter 6.1, we describe the discrete-time perpendicular recording channel model and the receiver front end. In Chapter 6.2, we discuss the prediction filter design and the soft-output PDNP channel detector that attempts to whiten the noise. In Chapter 6.3, we give some details about the S-IRA code we used and discuss the importance of H matrix column permutations. In Chapter 6.4, we compare the performance of the S-IRA code with the RS code under different media noise conditions. Chapter 6.5 presents some concluding remarks.

6.1 Perpendicular Magnetic Recording Channel Model

The discrete-time channel model used in this chapter is an adaptation to perpendicular recording of the Lorentzian channel model by J. Moon [80]. To derive the discrete-time model, consider the continuous-time model in Fig. 6.1. The transition response is given in [81] by

\[ h(t, w) = V_{\text{max}} \cdot \text{erf}\left\{ \frac{0.954t}{w} \right\} \]

where \( V_{\text{max}} \) is its peak amplitude (\(-V_{\text{max}}\) is its minimum amplitude) and \( w = T_{50} \) is the width of the transition response measured from \(-V_{\text{max}}/2\) to \( V_{\text{max}}/2 \). Thus, \( h(t, w) \) is the response of the channel to the transitions in the recorded bits, representable as

\[ \frac{1}{2}(a_k - a_{k-1}) \in \{ 0, \pm 1 \} \]

where the code bits \( a_k \in \{ -1, 1 \} \); hence, the presence of

\[ \frac{1}{2}(a_k - a_{k-1}) \in \{ 0, \pm 1 \} \]

\[ \text{Figure 6.1: Continuous-time channel model and receiver front end.} \]
the discrete-time differentiator $(1 - D)/2$ in Fig. 6.1. The transitions suffer from random fluctuations in position $(\Delta t_k)$ and width $(\Delta w_k)$, so that the readback signal has the form

$$r(t) = \sum_k \frac{1}{2}(a_k - a_{k-1})h(t - kT_c - \Delta t_k, w + \Delta w_k) + e(t), \quad (6.1)$$

where $T_c$ is the channel bit spacing; $e(t)$ is additive white Gaussian (electronics) noise with power spectral density $N_0$; the transition position deviation ("jitter") $\Delta t_k$ is modeled as truncated Gaussian ($|\Delta t_k| \leq T_c/2$) with variance $\sigma_t^2$; and the width deviation $\Delta w_k$ is single-sided Gaussian, with Gaussian variance $\sigma_w^2$. The normalized linear channel bit density is given by $D_c = T_{50}/T_c$ and the user bit density is given by $D_u = RD_c = RT_{50}/T_c$, for code rate $R$.

By ignoring the width deviation $\Delta w_k$ (jitter is the dominant source of media noise) and approximating $h(t - kT_c - \Delta t_k, w)$ by three Taylor series terms, we may
write

\[ r(t) \approx \sum_k a_k \frac{1}{2} p(t - kT_c) \]

\[ + \sum_k \frac{1}{2} (a_k - a_{k-1}) \Delta t_k \frac{\partial}{\partial t} h(t - kT_c) \]

\[ + \sum_k \frac{1}{2} (a_k - a_{k-1}) \frac{(\Delta t_k)^2}{2} \frac{\partial^2}{\partial t^2} h(t - kT_c) + e(t), \]  

(6.2)

where \( p(t) = h(t) - h(t - T_c) \) is the dibit response and where we dropped the dependency of \( h(t, w) \) on \( w \). Noting that the last three terms in (6.2) represent noise, the corresponding second-order channel model is shown in Fig. 6.2.

Let \( R_p(D) \) denote the overall noiseless transition response from the channel input up to the sampler. The matched filter (MF) is matched to \( \frac{1}{2} p(t) \) and the discrete-time partial response equalizer \( Q(D) \) shapes the overall noiseless transition response into the prescribed target response \( F(D) = 1 + D \). The equalizer is designed using the MMSE criterion, i.e., \( Q(D) = F(D) / [R_p(D) + N_0] \). Let the combined discrete-time response of the first-order derivative filter and the MF be denoted by \( R_{hp}^{(1)}(D) \) and let
$R_{hp}^{(2)}(D)$ represent the second-order derivative filter counterpart. Then the discrete-time model for the equalizer output $r_k$ given channel input $a_k$ and electronics noise $e_k$ is given in Fig. 6.3.

The signal-to-noise ratio (SNR) is defined with respect to the electronics noise: $SNR = \frac{V_{max}^2}{N_0}$. A given simulation data point will be specified as a function of this SNR and the jitter percentage, defined as

$$JP = \frac{\sigma_t^2 E_t/2}{N_0 + \sigma_t^2 E_t/2} \cdot 100\%,$$

(6.3)

where $E_t = \int [\frac{\partial h}{\partial t}]^2 dt$. The jitter percentage is measured at the input to the equalizer. Because the second-order jitter power is much smaller than the first-order jitter power, it is not included in the JP definition.

### 6.2 Channel Detector

Using a standard BCJR detector designed under the AWGN assumption is far from optimal due to the presence of equalizer-colored noise and pattern-dependent noise. One straightforward way to improve the performance is to whiten the noise using noise-predictive filters which modify branch metric computations with the help of PDNP filters. A PDNP maximum-likelihood (ML) detector based on linear prediction theory for the underlying signal-dependent noise was proposed in [82]. Also in [83], the PDNP technique was applied to a belief propagation-based sub-optimal detector. In this section we review some of the principles of the PDNP-ML detector, particularly how they apply to our situation.
Denote the channel input bit sequence by \( a \), the equalized noiseless channel output sequence by \( s(a) \), and the noise sequence by \( n(a) \), all of length \( N \). The equalized channel output sequence is \( r(a) = s(a) + n(a) \) where the \( i \)th signal sample is given by \( s_i = \sum_{j=0}^{I} f_j a_{i-j} \) and the equalization target \( F(D) \) has \( I + 1 \) coefficients \( f_0, f_1, \ldots, f_I \). The ML detector finds the bit sequence \( a \) that maximizes the conditional pdf

\[
\prod_{i=1}^{N} p(r_i|r_{i-1}, \ldots, r_1; a) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_p(a)}} \exp\left\{ -\frac{(r_i - s_i - \hat{n}_i(a))^2}{2\sigma_p^2(a)} \right\}, \tag{6.4}
\]

where \( \hat{n}_i(a) \) is the optimal linear prediction of the \( i \)th noise sample and \( \sigma_p^2(a) \) is the prediction error variance. From linear prediction theory, if the noise is Markov of order \( Q \), it can be perfectly modeled using a \( Q \)th-order autoregressive (AR) process and perfectly whitened using an optimal prediction filter with \( Q \) taps. For the media noise model we are considering, though the underlying Markov order is unknown, it has been shown that the optimal predictors can be approximated using finite-length predictors of order \( L \) \[82\]. Assuming the prediction filter taps are given by \( q(a) = [q_1(a), \ldots, q_L(a)] \) for each bipolar \( N \)-sequence \( a \), the \( i \)th predicted noise sample is

\[
\hat{n}_i(a) = \sum_{l=1}^{L} q_l(a) \cdot n_{i-l}(a) = \sum_{l=1}^{L} q_l(a) \cdot [r_{i-l} - s_{i-l}(a)]. \tag{6.5}
\]

\( q(a) \) and \( \sigma_p^2(a) \) can be obtained by using the Yule-Walker equation from prediction theory.

Accommodating all bipolar \( N \)-sequences \( a \) to realize the above algorithm in a PDNP-BCJR detector is not feasible. Thus, in practice a finite segment of \( a \) is utilized. Let \( \tilde{a} = \{a_{i-P}, \ldots, a_i, \ldots, a_{i+F}\} \) be the finite segment of \( a \) at time index \( i \), with \( P \) past bits and \( F \) future bits. Therefore, there are \( 2^{P+F+1} \) input patterns, prediction filters, and prediction error variances. \( q(\tilde{a}) \) and \( \sigma_p^2(\tilde{a}) \) can be obtained as described above.

The resulting PDNP-BCJR detector operates on a trellis expanded from the trellis of the PR target, using (6.4) to calculate the branch metrics. The conventional
BCJR detector has a trellis with $2^I$ states with each state labeled by $[a_{i-I}, \ldots, a_{i-1}]$.

For the PDNP-BCJR detector, the number of states is expanded to $2^{\max(P,I+L)+F}$ and each state is labeled by $[a_{i=\max(P,I+L)}, \ldots, a_i, \ldots, a_{i+F-1}]$.

By solving the Yule-Walker equation for AR modeling which involves inversion of the correlation matrices obtained from real data through sample statistics, the predictor tap weights and the error variances can be obtained. But without a complete knowledge of the noise statistics and a confident noise model, matrix inversion may cause stability problems. In [82], the authors proposed a method which applies the LMS algorithm to train the predictor taps and error variances from real data.

In the LMS algorithm, the filter taps are trained by the following recursive
equation:

\[ q_{k+1}(\tilde{a}) = q_k(\tilde{a}) + 2\mu \cdot e_k(\tilde{a}) \cdot n_k(\tilde{a}), \quad (6.6) \]

where \( n_k(\tilde{a}) = [n_{k-1}(\tilde{a}), \ldots, n_{k-L}(\tilde{a})] \) and \( \mu \) is the step size that controls the convergence speed. The block diagram of filter training is shown in Fig. 6.5. Throughout this chapter, the channel is equalized to a PR1 target \((I = 1)\) and \(F = 1, P = I + L\).

For each channel condition characterized by \( SNR \) and \( JP \), a proper \( \mu \) is chosen to make the filters converge within 400 training samples. The length of prediction filters \( L \) is determined by training the filters under various channel conditions with \( \mu = 0.02, D_c = 1.67 \) and \( L \in \{0, 1, \cdots, 6\} \). We found that \( L = 3 \) is large enough to achieve a sufficiently low MSE. Therefore, for the rest of the chapter, we set \( L = 3 \) and use \( 2^{I+L+F} = 2^5 \) predictors.

### 6.3 The S-IRA LDPC Code

The LDPC code-based simulation model is depicted in Fig. 6.4. The LDPC code we use in this chapter is a rate-0.9 \((4551, 4096)\) structured irregular repeat-accumulate (S-IRA) code [31]. The nominal parity-check matrix \( H \) for the LDPC code has the form \( H = [H_1 \ H_2] \), where \( H_1 \) consists of an array of circulant permutation matrices and \( H_2 \) is an \( M \times M \) “dual-diagonal” matrix \((M \) is the number of parity bits):

\[
H_2 = \begin{bmatrix}
1 & 1 \\
1 & 1 \\
& \cdots \\
1 & 1 \\
1 & 1 \\
\end{bmatrix}.
\]

For this code, \( H_1 \) has column weight 5.

The interleaver \( \Pi \) and de-interleaver \( \Pi^{-1} \) in Fig. 6.4 are necessary to mitigate the bursts nature of the PDNP-BCJR detector output. This characteristic can be seen in the auto-covariance function of the log likelihood ratio (LLR) magnitudes at the input to the LDPC decoder. Fig. 6.6 shows three interleaver situations for the
Figure 6.6: Covariance functions of the LDPC decoder inputs.
worst-case recorded sequence, \( \mathbf{a} = [+1 -1 +1 -1 ...] \), on the 95% JP channel. Our simulations used a uniform parity insertion interleaver, which spreads the parity bits evenly among the information bits with the purpose of reducing the correlations of the LDPC code’s parity bits (which correspond to low weight columns). Therefore, to show the effect of this interleaver on the parity bits, Fig. 6.6 plots the covariance of detector output LLR magnitudes corresponding to the parity bits after de-interleaving.

The interleaver and de-interleaver are conceptual quantities and can in practice be realized by re-ordering the columns of the LDPC code parity-check matrix. It is possible to select the column permutation (equivalently, select \( \Pi \)) to accommodate both the presence of a modulation code and soft decoding [79].

A reliable way of characterizing the ability of a code to combat correlated noise or correlated fading is via the parameter \( L_{\text{max}} \) [84]. Given an LDPC code’s \( \mathbf{H} \) matrix, \( L_{\text{max}} \) is defined to be the longest erasure burst length that is guaranteed to be correctable by the iterative erasure decoding algorithm based on that matrix. For the code described above, we found that \( L_{\text{max}} = 181 \) erasures, an efficiency of only \( 181/455 = 0.39 \). \( L_{\text{max}} \) can be greatly improved by permuting the columns of \( \mathbf{H} \) so that the “weak” parity columns (i.e., the submatrix \( \mathbf{H}_2 \)) do not all occur together. For example, the uniform parity insertion interleaver yields \( L_{\text{max}} = 239 \) (efficiency 0.52). We also considered a random interleaver for which \( L_{\text{max}} = 300 \) (efficiency 0.66).

### 6.4 Performance Results

In this section, we present performance results on the PMR channel with recording density \( D_s = 1.67 \) by simulating the second-order discrete-time model described in Section 6.1. To make a fair comparison between the LDPC code in Section 6.3 with RS codes, we simulate a rate-0.9 RS code (455, 410) with symbols over \( GF(2^{10}) \). Since the code rate is fixed to \( R = 0.9 \) for both codes, the user density \( D_u = 1.5 \).

The LDPC decoder is a sum-product algorithm (SPA) decoder which updates
Table 6.7: Performances on PMR channel with 5% jitter noise.

<table>
<thead>
<tr>
<th>SNR [dB]</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-9}</td>
<td></td>
</tr>
<tr>
<td>10^{-8}</td>
<td></td>
</tr>
<tr>
<td>10^{-7}</td>
<td></td>
</tr>
<tr>
<td>10^{-6}</td>
<td></td>
</tr>
<tr>
<td>10^{-5}</td>
<td></td>
</tr>
<tr>
<td>10^{-4}</td>
<td></td>
</tr>
<tr>
<td>10^{-3}</td>
<td></td>
</tr>
<tr>
<td>10^{-2}</td>
<td></td>
</tr>
<tr>
<td>10^{-1}</td>
<td></td>
</tr>
<tr>
<td>10^{0}</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.7: Performances on PMR channel with 5% jitter noise.
soft information iteratively based on the modified $H$ matrix corresponding to the uniform parity insertion interleaver. One can also use a more carefully designed structured column permutation for S-IRA codes [79] to achieve better performance. Although most of our results focus on the parity-insertion interleaver, later in this section we compare the performance of this interleaver to the random interleaver and to no interleaving.

As indicated in the system model of Fig. 6.4, we do not feed soft information back to the PDNP BCJR detector in order to maintain detector/decoder complexity to a manageable level. An additional gain on the order of 0.5 dB is possible if such global iterations are used, but instead we use 50 local iterations at the S-IRA SPA decoder. The performance measures used are frame error rate (FER) and bit error rate (BER), but further clarification is necessary since FER can correspond to both systematic bits and parity bits, or to only systematic bits. We denote by $FER_{all}$ the probability that a decoded code block contains one or more bit errors anywhere in the block and by $FER_{sys}$ the probability that a decoded code block contains one or more errors among the systematic bits. The latter definition is applicable to recording devices for which an outer CRC code is available so that the LDPC information word is a CRC word. Whereas the parity bits for IRA codes are more susceptible to error than the systematic bits, this is not the case for most regular LDPC codes or for RS codes, for which $FER_{all} \approx FER_{sys}$. The parity bits are more susceptible to error because they correspond to the weight-two columns of $H$ which have the effect also of improving the “decoding threshold” (allowing operation in lower SNR).

Figures 6.7, 6.8, and 6.9 present the performance results for the S-IRA and RS codes on the PMR channel with $D_u = 1.5$ and with jitter percentages of 5%, 50% and 95%, respectively. The parity-insertion column permutation for the S-IRA code was used in all cases in these figures. For 5% jitter noise, we see that the gain from the PDNP-BCJR detector over the conventional non-predictive BCJR detector for the S-IRA code is about 3.7 dB at $FER_{sys} = 10^{-5}$, and the S-IRA code provides a gain of 1 dB over the RS code. With 50% jitter noise, the PDNP-BCJR detector
Figure 6.8: Performances on PMR channel with 50% jitter noise.
provides a 2.4 dB gain at $\text{FER}_{\text{sys}} \sim 10^{-5}$ over the traditional BCJR detector for the S-IRA code, and the S-IRA code provides a gain of about 1.3 dB over the RS code at $\text{FER}_{\text{sys}} = 10^{-4}$. For 95% jitter noise, the gain of PDNP detector is 2.5 dB at $\text{FER}_{\text{sys}} \sim 10^{-4}$ and the S-IRA code offers a 1.1 dB gain at $\text{FER}_{\text{sys}} \sim 10^{-4}$.

As observed from the figures, the PDNP-BCJR detector achieves the largest gain over the non-predictive BCJR detector with 5% jitter noise. This shows that the PDNP detector is more effective at whitening the colored noise due to electronics noise than that from jitter noise. In support of this statement, in Fig. 6.10 the correlation function at the detector input for the $JP = 5\%$ case has larger tails than that for the $JP = 95\%$ case. We note that the plot is based on the worse-case data sequence and the same total noise power, with electronics noise dominating in the $JP = 5\%$ case. We conjecture that increasing the detector parameters $F$ and $P$ would improve the performance of the $JP = 95\%$ case, but this will require further work.

Notice in all three figures the $\text{FER}_{\text{all}}$ curves suffer from a floor. Most of the floor-inducing frame errors are attributable to errors in the parity bits. The $\text{FER}_{\text{all}}$ error-rate floor can be lowered by using a more random $H$ column permutation. In Fig. 6.11, we compare the performance of a computer-generated random interleaver (equivalently, column permutation) with the parity-insertion permutation and with no permutation on the 95% jitter channel. It is observed that the random permutation does indeed lower the floor significantly. Further research would be required to understand if an optimal permutation can be designed.

We remark that the S-IRA code used in this chapter has a very low error-rate floor on the binary-input AWGN channel: the $\text{FER}_{\text{all}}$ curve starts to floor at $10^{-7}$ [32]. It is the presence of data-dependent correlated media noise and the bursts nature of the detector that raises the floors. The error-rate floor has been one of the major concerns regarding the application of LDPC codes in magnetic recording. However, when an outer CRC code is available so that $\text{FER}_{\text{sys}}$ is the appropriate performance measure, then it appears that there is no floor down to at worst $\text{FER}_{\text{sys}} \sim 10^{-6}$. (Simulating very much below this level would take many weeks.)
Figure 6.9: Performances on PMR channel with 95\% jitter noise.
Additionally, the floors can be improved by using a very high-rate outer RS or BCH code since the frame errors in the floor region usually have only a few bit errors [32]. Another approach to improve the LDPC floor is to design codes and/or permutations which yield improved performance in the presence of correlated noise.

While the results in Figures 6.7, 6.8 and 6.9 demonstrate that a coding gain of at least 1 dB is attainable by the S-IRA code relative to the RS code, it is not clear how this SNR gain translates to user density gain. Toward this end, for the $JP = 95\%$ case, we increased the user density in our S-IRA simulator until its FER$_{sys}$ curve coincided with that of the RS code. Specifically, we increased the user density $D_u$ from 1.5 to 1.62 and simulated the same S-IRA code again under the same channel conditions as the 95% jitter noise simulations in Fig. 6.9. In attempting to make a fair comparison between the two user densities, we fix the jitter noise variance $\sigma^2_t$ for the two densities for a given SNR. Since the transition response is fixed (so is $E_t$) and $JP = 95\%$, for every given SNR point, $\sigma^2_t$ can be calculated from (6.3). Fig. 6.12 shows that the S-IRA code with $D_u = 1.62$ has comparable performance to the RS code with $D_u = 1.5$. Hence, the 1 dB gain in error rate performance translates to a $(1.62 - 1.5)/1.5 = 8\%$ increase in user density.

### 6.5 Conclusions

In this chapter, we evaluated the performance of a S-IRA LDPC code on perpendicular magnetic recording channel using a pattern-dependent noise-predictive BCJR detector. From simulations, we show that the S-IRA code provides an SNR gain of at least 1 dB which amounts to a user density gain of 8\%.
Figure 6.10: Correlation functions of detector inputs.

Figure 6.11: The impact of various interleavings on FER_{all}.
Figure 6.12: The user density gain of LDPC code over RS code with 95% jitter noise.
CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

In this dissertation, we have provided some insights into the error floor problem by studying its failure mechanisms and fundamental cause - trapping sets. We have discussed existing trapping set search algorithms and customized them for studying the unconventional decoders proposed in this dissertation. The importance sampling method for evaluating the low-floor performance of iterative decoders was first reviewed for memoryless channels and then generalized to channels with memory. To demonstrate the effectiveness of this prediction method, we simulated the Margulis code on a binary-input AWGN channel and the EPR4 channel. Intuition gained from this work enabled the low-floor decoding algorithm designs.

We have proposed three novel decoder designs, which include: a bi-mode decoder based on erasure decoding and its generalizations; bit-pinning/state-pinning decoders with/without outer algebraic codes; and three generalized-LDPC decoders. All of the techniques presented succeeded in lowering the floors of the codes we studied by orders of magnitude, beyond our ability to measure them. Because these algorithms directly target the dominant trapping sets, they are fully generalizable to other LDPC codes given the knowledge of their dominant trapping sets.

We have presented two system designs for magnetic storage. The dual-mode decoder is proposed for product code structures. It provides large performance improvement with only a little complexity increase. The only modification necessary to any existing system is the addition of a CRC code for packet-error detection. We have also proposed a simple reverse-concatenation scheme of a structured LDPC code and a RLL code, which was combined with a structured permutation of $H$ to satisfy the following requirements: soft information is available to the LDPC decoder, the LDPC code’s structure is preserved, and the system has the capability
of correcting long erasure bursts. A solution that utilized a DC-free code was also derived.

We have also evaluated the performance of a structured IRA LDPC code on the perpendicular magnetic recording channel using a pattern-dependent noise-predictive BCJR detector. We demonstrated the importance of a properly designed interleaver in such systems. From simulations, we showed that the S-IRA code provides an SNR gain of at least 1 dB which amounts to a user density gain of 8%.

Several interesting research topics were opened up during the course of this work. Future research includes the following:

1. **Study the relationship of trapping sets on different types of channels.** Trapping set configurations on the AWGN channel are relatively well-understood compared to those on other channels. One observation we made is, for the regular LDPC codes we have studied in this work, their dominant trapping sets on the AWGN channel have the same configurations as on the PR channel we considered. It would be interesting if we can understand the connections of trapping sets on different channels. Also the rankings of dominant trapping sets on different channels may vary, so the intuition gained by understanding how different channels affect the ranking will help with error floor prediction and mitigation.

2. **Study the relationship of trapping sets for a given code and different iterative decoders.** It is known that trapping sets are dependent on the decoding algorithms. For example, a general observation is that the min-sum algorithm [85] typically has lower floors than the sum-product algorithm [86]. Therefore, it is interesting to understand when, why, and how do trapping sets change when we change the iterative decoders.

3. **LDPC code design through trapping set optimizations.** Many existing LDPC code design algorithms, such as PEG and ACE, focus on the optimization of short cycles of the code. Given the tools of trapping set enumeration, we can perform trapping set optimizations (for example, a PEG-like algorithm targeting short trapping sets) on any well-designed LDPC codes to further
eliminate the most harmful trapping sets, thus achieving lower floor levels. However, most likely the improvement is limited to 1 or 2 orders of magnitudes.

4. *Importance Sampling on channels with jitter noise.* The IS method can be generalized to jitter channels starting from our work for the AWGN and PR channels.

5. *Lower-complexity GCP designs and optimal scheduling.* In the G-LDPC decoder designs, we utilized the BCJR algorithm for the GCPs, which can be replaced with a soft-output Chase-like processor to reduce complexity overhead. Also, the scheduling between the GCPs and SPCs can be optimized for lower floors.

6. *Determining the critical number for bit-pinning and state-pinning.* Reference [34] introduced the definition of critical number on the binary symmetric channel (BSC) as the minimum number of nodes in a trapping set that have to be initially in error for the decoder to end in that trapping set. Similarly for the pinning techniques, a critical number also exists for reverting the errors on the trapping set.

7. *Study the error floor performance with a limited number of iterations.* Throughout this work, we have been assuming a sufficiently large number of LDPC iterations were performed. However, trapping sets behave differently with a few number of iterations as in most practical systems. It is an interesting topic to characterized such behaviors.

8. *Error floor mitigation through noise biasing.* This is inspired by the idea of partial response channel, where ISI is intentionally introduced and controlled to a desired target response to increase recording density. Similarly, during the decoding process, noise samples can be intentionally biased to force the decoder converge to certain known trapping sets instead of unknown failures, and thus can be easily corrected.
This appendix describes the two-step importance sampling method that is custom-tailored for finding the dominant trapping sets of G-LDPC decoders (which we call “G-LDPC trapping sets”), starting with the trapping sets obtained from SPA decoder (“SPA trapping sets”). Section 2.4 discusses the method of injecting error impulses to locally interconnected variable nodes, for the purpose of finding dominant trapping sets. This concept can be easily adopted here. As shown in Section 3.4.1, the G-LDPC trapping sets are closely related to the SPA trapping sets in the way such that the new trapping sets are extended/extracted from the original SPA trapping sets, which are reasonable local structures for deterministic biasing. The new trapping sets can be found in two steps:

(a) **Identify which of the SPA trapping sets lead to G-LDPC trapping sets:** Assume a random codeword is transmitted. Apply a bias of $-2$ to each SPA trapping set VN with value $+1$; apply a bias of $+2$ to trapping set VNs with value $-1$. Then observe the G-LDPC decoder failures (no noise is added, only the biases). If the decoder clears all of the errors or an unstable error event is obtained, then it is assumed that the SPA trapping set leads to no G-LDPC trapping sets. Otherwise, record the stable patterns, and accumulate their multiplicities.

(b) **Find all relevant trapping sets:** The recorded patterns from Step (a) are not in the exact forms of G-LDPC trapping sets on AWGN channel yet. They are used as initial conditions in this step. Based on these patterns, we bias each location with a Gaussian random variable (RV) with mean -2 and variance corresponding to an SNR value in the floor region. We then observe the
G-LDPC decoder outputs. The stable low-weight error events are the trapping sets of interest. Their multiplicities are also obtained in this way. Note that one pattern from Step (a) may lead to several stable trapping sets, thus multiple realizations of the RV need to be observed.

The advantage of applying the two-step IS instead of Step (b) alone is: When biasing with Gaussian noise, multiple G-LDPC decoding instances are performed. It could be very time-consuming because every SPA trapping set has to be tested and the cardinality may be large (for example, there are 2640 SPA trapping sets to be tested for the Margulis code).
REFERENCES


