

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Improved Coding Techniques for Digital Recording Systems

Permalink

<https://escholarship.org/uc/item/78b64372>

Author

Bhatia, Aman

Publication Date

2015

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Improved Coding Techniques for Digital Recording Systems

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

in

Electrical Engineering
(Communication Theory and Systems)

by

Aman Bhatia

Committee in charge:

Professor Paul H. Siegel, Chair
Professor Alon Orlitsky
Professor Steven Swanson
Professor Alexander Vardy
Professor Kenneth Zeger

2015

Copyright
Aman Bhatia, 2015
All rights reserved.

The dissertation of Aman Bhatia is approved, and it is acceptable in quality and form for publication on micro-film and electronically:

Chair

University of California, San Diego

2015

TABLE OF CONTENTS

Signature Page	iii
Table of Contents	iv
List of Figures	vi
List of Tables	ix
Acknowledgements	x
Vita	xiii
Abstract of the Dissertation	xiv
Chapter 1 Introduction	1
1.1 Background	1
1.2 System Model	1
1.3 Dissertation Overview	8
Chapter 2 Lattice-Based WOM Codes for Multilevel Flash Memories	13
2.1 WOM Codes and Flash Memories	14
2.2 Lattice-based WOM Codes	16
2.3 Optimal Codebooks	21
2.4 Message Assignment for Codes with Discrete Support	33
2.5 Concluding Remarks	41
2.6 Appendix	42
Chapter 3 Precoding Mapping Optimization for Magnetic Recording Channels	56
3.1 Introduction	56
3.2 System Model	57
3.3 Precoder Design	59
3.4 Block-precoders in Magnetic Recording	60
3.5 Simulation Results	62
3.6 Concluding Remarks	66
Chapter 4 Polar Codes for Magnetic Recording Channels	71
4.1 Introduction	72
4.2 Channel Model	74
4.3 Turbo-Equalization Architecture	74
4.4 Multistage Decoding Architecture	77
4.5 Simulation Results	80
4.6 Conclusions	91

Chapter 5	Analysis of Stochastic Decoding of LDPC Codes	94
5.1	Introduction	94
5.2	Sum Product Algorithm	96
5.3	Stochastic Decoding Algorithm	96
5.4	Alternative Descriptions of SD	98
5.5	Asymptotic Analysis for SD	102
5.6	Conclusions	114
Chapter 6	Enhancing Binary Images of Non-Binary LDPC Codes	118
6.1	Introduction	118
6.2	$\text{GF}(2^b)$ -LDPC Codes	119
6.3	Binary Images	121
6.4	Superiority of $\text{BP}(\mathbf{H}_q)$ over $\text{BP}(\mathbf{H}_B)$	126
6.5	Enhancing $\mathcal{C}_B$ using Redundant Parities	128
6.6	Concluding Remarks	132

LIST OF FIGURES

Figure 1.1:	A simplified block diagram representing a generic communication system.	2
Figure 1.2:	A block diagram showing the encoder and decoder in a magnetic recording system.	5
Figure 2.1:	A 2-dimensional rectangular hyperbola $\mathbb{H}(u)$ in region $\mathbb{A} = [0, \ell] \times [0, \ell]$. The region under the hyperbola and the region accessible from a given point $\mathbf{x}$ on the hyperbola are shaded in blue and red, respectively, and their volumes are equal to $\Delta(u) \cdot \ell^n$ and $u \cdot \ell^n$, respectively.	23
Figure 2.2:	Optimal boundaries for 3 writes over 2 cells. We can see that the boundaries for the second write are independent of the points written on the first write.	27
Figure 2.3:	Optimal hyperbola parameter, u_2^* , computed numerically and bounds from Section 2.3.3.	31
Figure 2.4:	Ratio of z_n^* and n , where $u_2^* = e^{-z_n^*}$ is the optimal hyperbola parameter for $t = 2$ writes on n cells. The ratio z_n^*/n converges monotonically to 0.5 as n increases.	32
Figure 2.5:	4-write code for 2 cells with 8 levels each that achieves rate 6.085 bits/cell/erase. Points in the i^{th} codebook are assigned messages $\{\mathbf{A}_i, \mathbf{B}_i, \dots\}$ according to message assignment function defined in Algorithm 2.2 such that after any $i - 1$ writes, M_i messages may be stored on the next write.	37
Figure 2.6:	Diagram for Example 2.4.1 showing points of second-write region accessible from points of first write region	40
Figure 2.7:	Real branches of Lambert W function.	45
Figure 3.1:	Equivalent channel from source $\mathbf{X}$ to detected sequence $\mathbf{Y}$	57
Figure 3.2:	Channel model with block-precoder π	58
Figure 3.3:	Reverse concatenation architecture with separate block-precoders, π_{MTR} and π_{Parity} , for MTR-constrained user bits and unconstrained parity bits, respectively.	61
Figure 3.4:	Schematic diagrams showing the effect of the precoders in the RC architecture on (a) the MTR-constrained user bits, (b) the unconstrained parity bits.	62
Figure 3.5:	Raw Bit Error Rate at the output of the detector with various precoders for an i.u.d. source $\mathbf{X}$	63
Figure 3.6:	Raw Symbol Error Rate at the output of the detector with various precoders for an i.u.d. source $\mathbf{X}$ with alphabet $\mathbb{F}_4$	64
Figure 3.7:	Raw Bit Error Rate at the output of the detector with a 6-bit precoder for a MTR-3 constrained sequence.	65

Figure 3.8:	Raw Symbol Error Rate at the output of the detector with a 3-symbol precoder for MTR-3 constrained sequence where each symbol consists of 2 bits.	66
Figure 3.9:	Bit Error Rate on MTR-constrained user bits with various precoders and a rate 0.9 binary LDPC code decoded using turbo equalization.	67
Figure 4.1:	Encoder and decoder for turbo-equalization.	75
Figure 4.2:	Performance of a rate 0.7 blocklength 4096 polar code on AWGN channel when decoded using BP, SCAN, SC and SC-list decoders. The polar code was designed for AWGN channel with $E_b/N_o = 2.14$ dB. Concatenation with an outer 16-bit CRC code is used for SC-list with $L = 8$	75
Figure 4.3:	Performance of polar codes optimized for various AWGN and dicode channels with rate 0.83 (corresponding to design SNR 2.5 dB) and blocklength 16384 when decoded using SCAN on the dicode channel. Maximum 10 iterations of SCAN and 20 iterations of turbo equalization are used.	76
Figure 4.4:	Encoder for interleaved coding scheme.	78
Figure 4.5:	Multi-stage decoding scheme.	78
Figure 4.6:	Equivalent sub-channel for i^{th} stage with $C^{(1)}, \dots, C^{(i-1)}$ known to the decoder.	80
Figure 4.7:	Achievable rates for the 2 stages and the average rate for the 2-stage MSD scheme on the dicode channel.	81
Figure 4.8:	Achievable rates for the 4 stages and the average rate for the 4-stage MSD scheme on the dicode channel	82
Figure 4.9:	Overall rate for a MLC-MSD scheme on the dicode channel for various number of stages.	83
Figure 4.10:	Overall rate for a MLC-MSD scheme on the EPR4 channel for various number of stages.	84
Figure 4.11:	BER versus SNR for the 2-stage MSD scheme with various block lengths, N . The dashed and solid lines show the BER for the systematic and non-systematic encodings, respectively.	86
Figure 4.12:	BER for the MSD scheme using SC decoder for polar codes ($N = 2^{16}$) constructed at different SNRs.	87
Figure 4.13:	BER versus SNR for the MSD scheme for polar codes with rates $R^{(1)} = 0.69$, $R^{(2)} = 0.85$ and blocklength $N = 1024$ decoded with SC-list decoder for various list sizes without CRC-concatenation.	88
Figure 4.14:	FER for the 2-stage MSD scheme using SC-List decoder for polar codes ($N = 2^{14}$), with a concatenated 16-bit CRC. FER for the 2-stage MSD scheme with LDPC codes and TE scheme with LDPC and polar codes is also shown.	89
Figure 5.1:	Hardware implementation for stochastic decoding updates.	97

Figure 5.2:	Markov chain representing the variable node update.	98
Figure 5.3:	Alternative way to represent variable node update for a stochastic decoder used in density evolution in Section 5.5.	102
Figure 5.4:	Bayesian network for a stochastic decoder. Variables in red, green and blue are channel APPs, binary and ternary variables, respectively. Variables with dashed incoming links are stochastic functions of the incoming variables; all other variables are a deterministic function of the incoming variables.	104
Figure 5.5:	$\mathbb{P}(N^{(\ell)} = 1)$ for different iterations, ℓ , computed for the $(3, 6)$ -regular LDPC code ensemble using approximate density evolution with memory order $K \in \{3, 4, \dots, 8\}$ when transmission takes place over a binary symmetric channel with crossover probability $\epsilon = 0.0338$. . .	112
Figure 5.6:	$\mathbb{P}(N^{(\ell)} = 1)$ for different iterations, ℓ , computed for the $(3, 6)$ -regular LDPC code ensemble using approximate density evolution with memory order $K = 7$ when transmission takes place over a binary symmetric channel with crossover probability ϵ	113
Figure 6.1:	Bit (dashed curves) and frame (solid curves) error rates for the random non-binary code $\mathcal{C}_4$ with parameters $q = 2^2, n_q = 96, k_q = 32, d_l = 2, d_r = 3$ and its binary images considered in Table 6.2. . . .	133
Figure 6.2:	Bit (dashed curves) and frame (solid curves) error rates for the PEG constructed non-binary code $\mathcal{C}_8$ with parameters $q = 2^3, n_q = 100, k_q = 50, d_l = 2, d_r \simeq 4$ and its binary images considered in Table 6.3.	134

LIST OF TABLES

Table 2.1:	Rivest-Shamir 2-write WOM code	14
Table 2.2:	Worst-Case Sum-Rates R_t in bits per cell per erase achieved by t -write codes on 2 cells with q levels.	38
Table 3.1:	SNR gains for the proposed block-precoders for MTR-constrained bits and parity bits at various channel bit densities compared to no-precoding scheme.	65
Table 4.1:	Code parameters for the LDPC Code used as component code for the first stage in the 2-stage MSD scheme in Figure 4.14.	90
Table 4.2:	Code parameters for the LDPC Code used as component code for the second stage in the 2-stage MSD scheme in Figure 4.14.	90
Table 4.3:	Code parameters for the LDPC Code used for the turbo equalization scheme in Figure 4.14.	91
Table 5.1:	Thresholds for various LDPC code ensembles under stochastic decoding, Gallager's algorithm A, and the sum-product algorithm, along with the corresponding Shannon capacity for BSC.	114
Table 6.1:	Mapping $\Phi_B : \text{GF}(8) \mapsto \text{GF}(2)^3$	124
Table 6.2:	Stopping set weights for a $\text{GF}(4)$ code, $\mathcal{C}_4$, and its binary images . . .	131
Table 6.3:	Stopping set weights for a $\text{GF}(8)$ code, $\mathcal{C}_8$, and its binary images . . .	132

ACKNOWLEDGEMENTS

When I started the Ph.D. program in 2009, I did not know what challenges lay ahead let alone how to overcome them. These past years have been immensely stimulating intellectually and personally rewarding. All these experiences, and this dissertation research, would not have been possible without the support and contribution of many people who I thank sincerely.

First and foremost, I would like to express my gratitude to my advisor, Prof. Paul Siegel, for his unwavering encouragement and endless patience. With his vast knowledge and experience, Prof. Siegel has steered me towards elegant solutions on many occasions. While his guidance and support has been invaluable in my research, his kind attitude and attention to detail have been a constant source of inspiration for me. I would also like to thank him as well as Prof. Alon Orlitsky, Prof. Ken Zeger, Prof. Jack Wolf, and Prof. Laurence Milstein, for teaching a number of stimulating courses on communications, coding and information theory. Their courses have been a joy to attend and have taught me extensive topics that were helpful in this dissertation. I am sincerely thankful to Prof. Ken Zeger, Prof. Alexander Vardy, Prof. Alon Orlitsky, and Prof. Steve Swanson for their time and effort to serve on my doctoral committee. I also appreciate the help of my B. Tech. advisors, Prof. R.K. Bansal and Prof. Adrish Banerjee, who encouraged me to study the theory of communications and pursue a Ph.D. degree.

A special thanks goes to my manager during my visit to LSI Corporation, Dr. Shaohua Yang. I appreciate all his help and guidance before, during and after my visit. I would also like to thank Victor Krachkovsky, Xuebin Wu, Razmik Karabed, Shu Li and Yang Han for enlightening discussions during my visit.

I would like to thank the CMRR, OGS and ECE department staff especially Iris Villanueva, Betty Manoulian, Ray Descoteaux, Julie Matsuda, Gennie Miranda, Shana Slebiada, and Jennifer Kim for their dedicated administrative support during my Ph.D. program. They have helped me with friendly smiles and diligent efforts to meet all my needs at work for which I am grateful.

I wish to thank the many members of the STAR group who have provided a delightful atmosphere to collaborate and conduct research. I thank my office-mates, Aravind Iyengar and Bing Fan for their numerous stimulating interactions at work.

Aravind deserves a special thanks for his guidance on many parts of this dissertation. I wish to thank Brian Butler, Brian Kurkoski, Ido Tal, Saeed Sharifi Tehrani, Eitan Yaakobi, and Xiaojie Zhang, for providing me a sound background in coding as I started this research. Thanks to Minghai Qin for being a great colleague and friend. My thanks to Jing Guo, Pengfei Huang, Dustin Hudson, Vasu Kanekal, Seyhan Karakulak, Scott Kayser, Uday Khandelwal, Veeresh Taranalli, Hironori Uchikawa, who have taught me a lot through their discussions. I also wish to thank Dr. Hemant Thapar for his invaluable guidance and encouragement.

I would thank the many people whose friendship has supported me through these five years. They have provided me memories that I will cherish throughout my life. Finally I express my thanks to my wife Swati and my family back home in India. Without their constant love and support, it would not have been feasible for me to achieve much.

This research was supported in part by the National Science Foundation grants CCF-0829865, CCF-1405119 and CCF-1116739, UC Lab Fees Research Program Award No. 09-LR-06-118620-SIEP and the Center for Magnetic Recording Research, Western Digital Corporation, and LSI Corporation.

The material in this dissertation is based in part on the following papers. Chapter 2 contains material from the papers: “Multilevel 2-Cell t-Write Codes,” Aman Bhatia, Aravind R. Iyengar, Paul H. Siegel; published in Proc. IEEE Information Theory Workshop, Lausanne, Switzerland, September 3-7, 2012, pp. 247 – 251, and “Lattice-based WOM Codes for Multilevel Flash Memories,” Aman Bhatia, Minghai Qin, Aravind R. Iyengar, Brian M. Kurkoski, Paul H. Siegel; published in the IEEE Journal on Selected Areas in Communications, vol. 32, no. 5, pp. 933 – 945, May 2014.

Chapter 3 is in part a reprint of the paper “Precoding Mapping Optimization for Magnetic Recording Channels,” Aman Bhatia, Shaohua Yang, Paul H. Siegel; published in the IEEE Transactions on Magnetics, vol. 50, no. 11, pp. 1–4, November 2014.

Chapter 4 is part of the paper “Polar Codes for Magnetic Recording Channels,” Aman Bhatia, Paul H. Siegel; which is being prepared for submission to the IEEE Transactions on Communications.

Chapter 5 contains material from the paper “On Stochastic Decoding for LDPC

Codes,” Aman Bhatia, Aravind R. Iyengar, Paul H. Siegel; which is being prepared for submission to the IEEE Transactions on Communications.

Chapter 6 is a reprint of the paper “Enhancing Binary Images of Non-Binary LDPC Codes,” Aman Bhatia, Aravind R. Iyengar, Paul H. Siegel; published in Proc. IEEE Globecom, Houston, Texas, December 5-9, 2011, pp. 1 – 6.

The dissertation author was the primary investigator and author of these papers, and co-authors have approved the use of the material for this dissertation.

VITA

2009	Bachelor of Technology, Indian Institute of Technology, Kanpur
2009-2014	Graduate Student Researcher, University of California, San Diego
2011	Master of Science, University of California, San Diego
2015	Doctor of Philosophy, University of California, San Diego

PUBLICATIONS

A. Bhatia, A. R. Iyengar, and P. H. Siegel, “Enhancing Binary Images of Non-Binary LDPC Codes,” *Proc. IEEE Globecom*, Houston, Texas, pp. 1 – 6, December 5-9, 2011.

A. Bhatia, A. R. Iyengar, and P. H. Siegel, “Multilevel 2-Cell t-Write Codes,” *Proc. IEEE Information Theory Workshop*, Lausanne, Switzerland, pp. 247 – 251, September 3-7, 2012.

A. Bhatia, M. Qin, A. R. Iyengar, B. M. Kurkoski, and P. H. Siegel, “Lattice-based WOM Codes for Multilevel Flash Memories,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 933 – 945, May 2014.

A. Bhatia, S. Yang, and P. H. Siegel, “Precoding Mapping Optimization for Magnetic Recording Channels,” *IEEE Transactions on Magnetics*, vol. 50, no. 11, pp. 1–4, November 2014.

A. Bhatia and P. H. Siegel, “Polar Codes for Magnetic Recording Channels,” *to be submitted to the IEEE Transactions on Communications*.

A. Bhatia, A. R. Iyengar, and P. H. Siegel, “On Stochastic Decoding of LDPC Codes,” *to be submitted to the IEEE Transactions on Communications*.

ABSTRACT OF THE DISSERTATION

Improved Coding Techniques for Digital Recording Systems

by

Aman Bhatia

Doctor of Philosophy in Electrical Engineering
(Communication Theory and Systems)

University of California, San Diego, 2015

Professor Paul H. Siegel, Chair

This dissertation addresses various problems related to data encoding and error-correction techniques to design more reliable and higher density digital data storage technologies.

In the first chapter, we give an overview of the existing data storage technologies and highlight the relevance of the techniques discussed in this dissertation to these technologies.

In the second chapter, the problem of rewriting for multilevel flash memories is considered and a novel construction method for WOM codes based on lattices is proposed. Using the continuous approximation, hyperbolic write-regions are shown to be sum-rate optimal for arbitrary number of writes. An algorithm that determines an

optimal encoding scheme is proposed for the case of two cells. Using these ideas, WOM codes are proposed that achieve high sum-rates.

In the third chapter, the problem of designing a block-precoder for a magnetic recording channel is considered with the objective of minimizing the error rate. The precoder design problem is equivalent to solving a quadratic assignment problem which is NP-complete in general. Precoders are constructed using a branch-and-bound technique with a reduced search space, as well as using a sub-optimal local search algorithm. Simulation results show that these block-precoders out-perform existing precoding techniques.

The fourth chapter discusses a novel technique for using polar codes for partial response channels. Multiple polar codes of various rates are designed for memoryless channels estimated by removing the effects of intersymbol interference from the partial response channel. Data is encoded using these codes and the codewords are interleaved before transmission. Decoding is done sequentially using a multi-stage decoding technique. Simulation results demonstrate that the performance of these codes is comparable to LDPC codes.

The fifth chapter studies the performance of stochastic decoding on LDPC code ensembles in the limit of infinite blocklength. Two methods to perform the density evolution for stochastic decoding are provided. Alternative descriptions of the variable node update in stochastic decoding are given and connections to other decoding algorithms are highlighted.

The sixth chapter explores the performance of binary images of non-binary LDPC codes. For the binary erasure channel, it is shown that the collection of stopping sets for a non-binary LDPC code are a subset of the collection of stopping sets of its basic binary image. An efficient algorithm is proposed that eliminates these additional stopping sets by adding redundant parity checks to the basic binary images. Simulation results confirm that the few redundant checks are sufficient to match the performance of the original non-binary LDPC codes with lower complexity.

Chapter 1

Introduction

1.1 Background

The ubiquitous use of computers in our lives and increasing broadband availability has caused digital data to be generated at a rapidly increasing rate. The fast-growing global data volume leads to an ever-growing demand from storage systems. Digital storage technologies have evolved progressively, starting from mechanical devices like punch cards to magnetic storage including tape and spinning disk drives, to optical disks and semiconductor based random-access memory devices including flash memories, and many more technologies are in development.

With increasing demands of higher densities, higher transmission speeds and lower costs from these storage systems, communication theory plays an important role in their design. The research in this dissertation focuses on several coding techniques that are important in contemporary storage and communication technologies.

1.2 System Model

The concept of a *communication channel* is typically associated with the exchange of information between two agents at separate physical locations, with one agent transmitting and the other agent receiving data through some medium. This concept of information exchange is also incorporated in the data storage and retrieval process. While a wireless connection to a cell phone, or a TV broadcast allows communication in

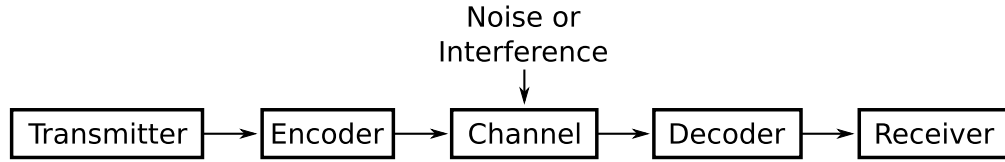


Figure 1.1: A simplified block diagram representing a generic communication system.

space, a data storage device enables communication in time.

Typical communication channels are imperfect - the received data can be corrupted by random noise and interference due to various sources present in the channel - while accurate transmission is often vital. Claude Shannon in his seminal 1948 paper “A Mathematical Theory of Communication” [1] set up the theoretical foundation of modern communications. Shannon argued that all information can be represented digitally, for e.g. as binary digits (bits) 0 and 1. He showed that information sources have a *source rate* associated with them and channels have a *capacity*. He proved that *reliable* communication is possible over noisy channels as long as the transmission rate does not exceed the capacity of the channel. That is, by encoding the information source suitably, one could transmit it over an imperfect channel so that the receiver can then decode with a high probability the intended message based on the output of the channel. Figure 1.1 shows a simplified block diagram of such a communication system. It is assumed that the encoding scheme is known to both the transmitter and the receiver, while the channel noise realization is unknown to both.

1.2.1 Coding for noisy channels

While Shannon’s work set up the theoretical limits to the rate of communication, his proof relies on a coding scheme which cannot be implemented efficiently. Several efforts have been made since then in the coding theory literature to develop efficient methods that achieve reliable communication over such noisy channels at rates close to their Shannon limit but using low-complexity encoding and decoding processes. A simple scheme, known as the *repetition code*, is as follows. To communicate any bit, the encoder transmits multiple copies of the message bit. For example, if the bit to be communicated

is 1, the encoder transmits the codeword

1111111.

The noise in the channel may cause some of these transmitted bits to be received incorrectly due to random noise so that the sequence received may, for example, be

1011011.

The receiver can then employ a majority vote rule to decode the message bit as 1. This scheme allows transmission at a rate of $1/7$ bits per use of the channel and the message bit can be communicated correctly if no more than 3 bits are in error.

However, this simple repetition coding scheme cannot be used to decrease the probability of error to an arbitrary small number without simultaneously decreasing the transmission rate to zero. Many error correcting codes that performed well including Bose-Chaudhuri-Hocquenghem (BCH) codes [2, 3, 4], Reed-Solomon codes [5], and convolutional codes [6, 7] have been studied. Still, error-correcting codes that allow the error probability to vanish while achieving rates close to the Shannon limit were not known till the discovery of *Turbo codes* [8] and the rediscovery of *low-density parity check (LDPC) codes* in the 1990s [9]. Both these classes of codes have random-like constructions with large blocklengths. They are specified by simpler constituent codes that interact with each other to define the larger code. This allows them to be decoded using low-complexity sub-optimal decoding algorithms that iteratively exchange their local information regarding the transmitted bit. Since these codes achieve rates close to the capacity of the channel, there has been a large body of work to analyze and develop lower-complexity decoding algorithms for these codes. In this work, we analyze a low-complexity decoding algorithm for LDPC codes, and also develop a low-complexity decoding technique for LDPC codes based on non-binary elements that have better performance than those based on binary elements.

More recently, a family of linear block codes called *polar codes* has been proven to achieve the capacity of arbitrary symmetric binary-input discrete memoryless channels (B-DMC) with low-complexity encoding and decoding algorithms [10]. These codes

are constructed based on a technique called *polarization* that transforms a B-DMC to create channels that are either noiseless or completely noisy, by performing a sequence of recursive transformations. An important difference between polar codes and previously known codes is that the construction of polar codes is based on the channel on which transmission occurs. As we will see in Section 1.2.2, magnetic recording systems are modeled as intersymbol interference channels that are not memoryless. This implies that, as described in literature, it is not known if polar codes are capacity-achieving for magnetic recording and the known polar code construction methods do not apply. In this dissertation, we will propose a novel technique that allows the use of known construction, encoding and decoding methods to achieve performance comparable to LDPC codes on models of magnetic recording channels.

1.2.2 Magnetic Recording Systems

The magnetic disk drive has improved rapidly in the past through developments in heads, media, servo and signal processing technologies that have improved the areal densities from 2 Kb/inch² in 1957 [11] to over 800 Gb/inch² in 2014 [12]. However, with the areal density approaching its physical limits and its CAGR in the last 5 years slowing down to $\sim 10\%$ from a historical 40% [13], improvements in signal processing and coding techniques have become more significant. As the worldwide need for storage continues to grow, hard disk drives remain the only cost effective option to meet this demand.

A block diagram showing a model of the contemporary magnetic recording system is shown in Figure 1.2. The user data is protected by an outer *modulation code* and an inner systematic error-correcting code. The parity bits of the ECC codeword are then precoded and the coded sequence is recorded on the magnetic disk. The read-back signal is sampled and equalized to a partial-response (PR) target. A sequence detector is applied to the equalized target and a-posteriori probabilities are generated for each bit of the coded sequence. These are then passed to the ECC decoder which tries to estimate the transmitted codeword. After the ECC decoder converges to a valid codeword, the decoded codeword is passed to the modulation decoder which generates the decoded user data.

The modulation code is used to encode arbitrary input sequences into *constrained*

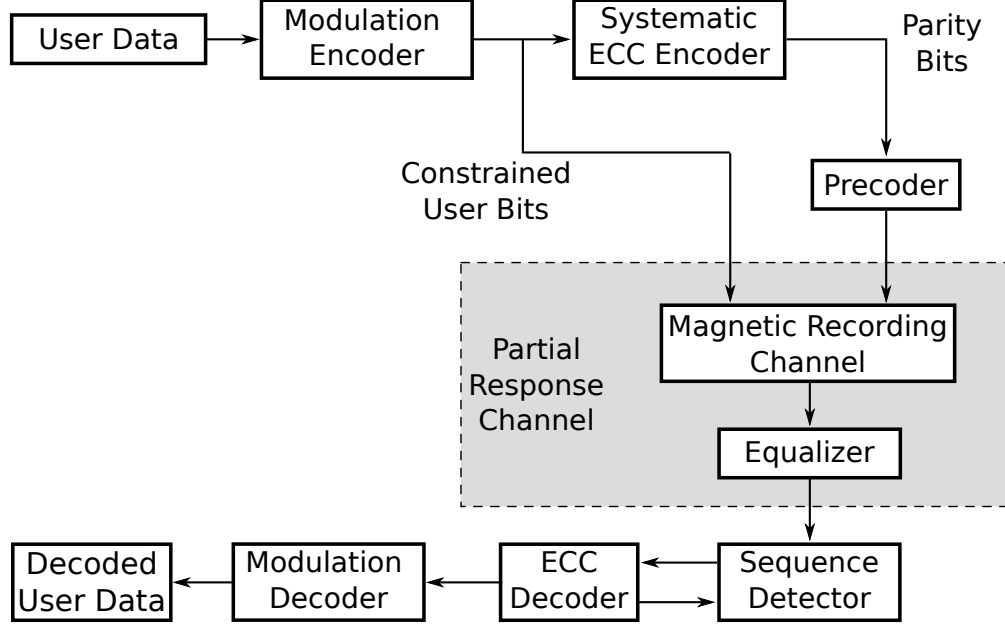


Figure 1.2: A block diagram showing the encoder and decoder in a magnetic recording system.

sequences that satisfy certain desirable properties. These constraints could be designed to disallow patterns that may be more prone to errors. For example, a (d, k) run-length-limited code ensures that there are at least d and at most k 0 bits between two 1 bits in the output sequence, which can help the timing recovery algorithm at the decoder. Other examples include modulation codes that increase the minimum distance properties of the sequence detector [14, 15, 16] by eliminating sequences that the detector cannot distinguish very well. In comparison, the error-correcting code is used to correct any errors after they occur at the detector output.

Early disk drives used the ECC as the outer code and the modulation code as the inner code. This results in severe error propagation from the output of the modulation decoder when a high-rate modulation code is used. Current disk drives therefore use the reverse concatenation (RC) architecture [17], as depicted in Figure 1.2, that does not suffer from the error propagation problem. The RC architecture also allows the use of soft-decision decoding of the error correcting code that is required for turbo-equalization. In this architecture, the user bits are encoded with a high-rate modulation code and stored on the magnetic recording channel, while the parity-bits of the ECC codeword are encoded by a precoder such as the $1/(1 \oplus D)$ precoder. In this dissertation

we consider designing better precoders for the parity bits, as well as for the constrained bits.

The read-back signal from the magnetic recording channel includes intersymbol interference (ISI) from a large number of adjacent bits. The amount of ISI increases with increasing recording densities as the bits are recorded closer to each other. On one hand, the complexity of detecting the sequence increases exponentially with the memory. On the other hand, removing the ISI completely through a zero-forcing equalizer enhances the noise present in the read-back signal. Thus, to limit the complexity of the detector while avoiding noise enhancement, the read-back signal is equalized to a partial response (PR) target which has a short impulse response. The output of a PR channel with target $h(D) = \sum_{i=0}^{\nu} h_i D^i$ is given by

$$y_t = \sum_{i=0}^{\nu} h_i x_{t-i} + n_t$$

where x_t is the input bit at time t and n_t is an i.i.d. Gaussian noise sequence. The PR channel impulse response is often of the form

$$h(D) = (1 - D)(1 + D)^n.$$

When $n = 0$, the equalized channel, or the PR channel, is called the dicode channel; when $n = 1$, the PR4 channel; when $n = 2$, the extended PR4 (EPR4) channel. Generalized PR channels which have arbitrary coefficients rather than integer coefficients are also used in magnetic recording systems [18].

Given the output of the PR channel, a sequence detector like soft-output Viterbi detector [19] or BCJR [20] estimates the likelihood of each input bit being 0 or 1. These likelihoods, also known as soft-information, are then input to the ECC decoder. This soft-input decoding scheme performs better than maximum likelihood (ML) detection which only provides hard-decisions as inputs to the decoding algorithm.

Turbo-equalization is often used where the ECC decoder and the sequence detector iteratively generate soft-information to be used as priors by each other. For this purpose, an ECC decoder that produces soft-outputs is required. LDPC codes have iterative message-passing decoding algorithms that use soft-information as input and produce

soft-information as outputs. Due to their superior performance and low-complexity decoding algorithms, they have replaced Reed-Solomon codes as the ECC codes in magnetic recording applications.

In this dissertation, we explore using polar codes for PR channels and compare their performance to the use of LDPC codes decoded using turbo-equalization.

1.2.3 Flash Memories

While the storage industry has been dominated by traditional magnetic storage technologies, recent advances in semiconductor technology have enabled solid-state memories that offer many advantages that make them attractive for many applications. These benefits, which include higher throughputs, smaller form-factors, lower power consumption, and no moving parts, have made solid-state memories the preferred non-volatile storage device in most consumer electronics and high-performance applications. However, these technologies present unique challenges in terms of cost, capacity, reliability and endurance. Even the most developed and cheapest of these technologies, flash memory technology, has a per-gigabyte cost that is higher than that offered by hard disk drives. Flash memories have a limited lifetime in which they can be used reliably. Moreover, increasing the data density to increase the capacity of flash memory devices decreases their lifetime further. Improved error-correction techniques and better coding methods may allow these issues to be mitigated.

Flash memory devices store information in the form of charge in a floating gate transistor, referred to as a *cell*. Using hot-electron injection charge is injected into a cell where the electrons get trapped in the floating gate of the transistor. When charge is present in the floating gate, the voltage at which the transistor becomes conductive, known as the *threshold voltage*, rises. Therefore, the presence or absence of charge can be sensed by testing whether the threshold voltage is large or small. This process is used to read the cell in flash memory. The cell is said to be in the *programmed state* if the cell stores charge, or in the *erased state* if it does not. For this thesis, we map the programmed and erased states to bits 1 and 0 respectively.

The state of each cell can change from 0 to 1 by the hot-electron injection method as mentioned above. However, changing the state from 1 to 0, or *erasing* the cell, requires

moving the charge from the floating gate through a tunnel oxide layer using high electric field Fowler-Nordheim tunneling. Since erasing a cell is a slow process, it is performed together on a large array of cells called *blocks* that may contain $\sim 10^6$ cells. Moreover, each flash cell can tolerate a limited number of program-erase cycles before the tunnel oxide layer is depleted and the cell can no longer store charge reliably.

Cells in early flash devices only supported two levels of charge and could therefore store a single bit each. Newer flash memories have cells that can support multiple levels of charge through tighter control of charge injection during programming and by testing for multiple threshold voltages to determine the charge level. Depending on the number of levels supported, flash memory devices are categorized as SLC (2 levels), MLC (4 levels), or TLC (8 levels). With increasing number of charge levels, the storage density increases. However, smaller changes in the charge of the floating gate may lead to an error while writing or reading data from a cell. Thus, the number of program-erase cycles that a device can reliably perform decreases.

In this dissertation, we investigate data encoding techniques that overcome the asymmetry in program and erase operations in a flash memory device. In particular, we consider codes that allow data to be rewritten on the cells multiple times without requiring an erase operation. We propose a novel construction of codes that allow rewriting the multilevel flash memories that support a large number of charge levels.

1.3 Dissertation Overview

This dissertation is divided into self-contained chapters that focus first on data encoding techniques for flash memories, then on coding techniques for magnetic recording systems, and finally on reduced complexity decoding algorithms for LDPC codes.

Chapter 2 considers the problem of endurance for flash memories and proposes rewriting on multilevel flash cells using WOM codes. A novel construction method for WOM codes based on lattices is proposed. Using the continuous approximation for dense lattices, it is shown that hyperbolic write-regions are sum-rate optimal for arbitrary number of writes and arbitrary number of cells. Motivated by these hyperbolic write-regions, we discuss construction and encoding of codebooks for cells with discrete

support. We present a polynomial-time algorithm to assign messages to the codebooks and show that it achieves the optimal sum-rate for any given codebook when the number of cells is equal to 2. Using this approach, we construct codes that achieve high-sum rate. We describe an alternative formulation of the message assignment problem when the number of cells is greater than or equal to 3, a problem which remains open.

In Chapter 3, the problem of designing a rate-1 block-precoder for a magnetic recording channel is considered with the objective of minimizing the bit/symbol error rate. It is shown that the precoder design problem is equivalent to solving a quadratic assignment problem which is NP-complete in general. Precoders for various blocklengths are constructed using a branch-and-bound technique with a reduced search space, as well as using a sub-optimal local search algorithm. Simulation results show that these block-precoders out-perform existing precoding techniques.

Chapter 4 discusses a novel technique for using polar codes for magnetic recording channels, modeled as partial response channels with additive white Gaussian noise. The proposed scheme uses interleaved polar codes and a sequential decoding technique as follows. Multiple polar codes of various rates are designed for memoryless channels estimated by removing the effects of intersymbol interference from the partial response channel. Data is partitioned and encoded using these codes. The codewords are interleaved before transmission and decoded sequentially using a multi-stage decoding technique that allows the use of a hard decision decoder. Simulation results demonstrate that the performance of these codes is comparable to LDPC codes.

Chapter 5 studies the stochastic decoding algorithm for LDPC codes. The performance of stochastic decoding on LDPC code ensembles is analyzed in the limit of infinite blocklength. We provide two methods to perform the density evolution for stochastic decoding. We first provide a method to perform exact density evolution and show that its complexity grows with the number of iterations. We then propose an approximation to the exact method which assumes that the memory of the decoding process is truncated. The approximation allows efficient computation of thresholds for LDPC code ensembles. We also provide alternative descriptions of the variable node update in stochastic decoding and highlight connections to other decoding algorithms.

Chapter 6 explores the idea of decoding non-binary LDPC codes by using message

passing algorithms on binary graphs. For the binary erasure channel, it is shown that the collection of stopping sets for a non-binary LDPC code are a subset of the collection of stopping sets of its basic binary image. An efficient algorithm is proposed that eliminates these additional stopping sets by adding redundant parity checks to the basic binary images. Simulation results confirm that the few redundant checks are sufficient to match the performance of the original non-binary LDPC codes with lower complexity.

Bibliography

- [1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, Jul. , Oct. 1948.
- [2] A. Hocquenghem, "Codes correcteurs derreurs," *Chiffres*, vol. 2, pp. 147–156, 1959.
- [3] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information Control*, vol. 3, pp. 68–79, Mar. 1960.
- [4] W. W. Peterson, "Encoding and error-correction procedures for the Bose-Chaudhuri codes," *IRE Transactions on Information Theory*, vol. 6, no. 5, pp. 459–470, Sept. 1960.
- [5] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, Jun. 1960.
- [6] G. D. Forney, "Convolutional codes I: Algebraic structure," *IEEE Transactions on Information Theory*, vol. 16, no. 11, pp. 720–738, Nov. 1970.
- [7] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*, New York: IEEE Press, 1999.
- [8] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding," *Proc. International Conference on Communications*, Geneva, Switzerland, pp. 1064–1070, May 1993.
- [9] D. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457458, Mar. 1997.
- [10] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
- [11] IBM Archives. (2014, August). *Magnetic head development* [Online]. Available: http://www.ibm.com/ibm/history/exhibits/storage/storage_magnetic.html
- [12] Seagate. (2014, Aug.). *Seagate ST5000DM000 Desktop HDD Product Manual* [Online]. Available: <http://www.seagate.com/www-content/product-content/desktop-hdd-fam/en-us/docs/100743772a.pdf>
- [13] D. Anderson. (2013). *HDD Opportunities & Challenges, Now to 2020* [Online]. Available: https://www.dtc.umn.edu/resources/bd2013_anderson.pdf
- [14] R. Karabed, P. H. Siegel, and E. Soljanin, "Constrained coding for binary channels with high intersymbol interference," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1777–1797, Sept. 1999.

- [15] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Transactions on Magnetics*, vol. 32, no. 5, pp. 3992–3994, Sept. 1996.
- [16] R. Karabed and P. H. Siegel, "Matched spectral-null codes for partial-response channels," *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 818–855, May 1991.
- [17] J. L. Fan and A. R. Calderbank, "A modified concatenated coding scheme, with applications to magnetic data storage," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1565–1574, Jul. 1998.
- [18] P. Kovintavewat, I. Ozgunes, E. Kurtas, J. R. Barry, and S. W. McLaughlin, "Generalized partial-response targets for perpendicular recording with jitter noise," *IEEE Transactions on Magnetics*, vol. 38, no. 5, pp. 2340–2342, Sep. 2002.
- [19] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," *IEEE Globecom*, vol. 3, pp. 1680–1686, Dallas, TX, Nov. 27–30, 1989.
- [20] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

Chapter 2

Lattice-Based WOM Codes for Multilevel Flash Memories

As introduced in Chapter 1, flash memory stores information in the form of charge in a floating gate transistor, referred to as a *cell*. In the case of single-level-cell (SLC) flash memory, each cell has two levels of charge and thus stores a single bit. Multilevel cells (MLCs) or triple-level cells (TLCs) can store 2 or 3 bits by storing 4 or 8 different levels of charge, respectively. With rising demand for increasing the capacity of flash memories, the number of levels stored in a cell is expected to increase further [1]. While increasing the charge level of an individual cell is a simple *program* operation with low latency, decreasing the level of a cell requires a complex *erase* operation on a large block of cells ($\sim 10^6$ cells) that also decreases the lifetime of the device.

In this chapter we treat multilevel flash memory cells as noiseless write-once memory (WOM) elements. We then aim to design WOM-codes that allow t -writes and maximize the amount of information that can be stored in a given number of cells in the worst-case before an erase operation is needed. When the number of levels, denoted by q in this chapter, is large, we assume that the cells can store a continuous range of levels. For cells that satisfy this *continuous approximation*, we prove that hyperbolic write-regions are optimal for maximizing the worst-case sum-rate. Motivated by these hyperbolic write-regions, we discuss construction and encoding of codebooks for cells with discrete support. We present a polynomial-time algorithm to assign messages to the codebooks and show that it achieves the optimal sum-rate for any given codebook

when $n = 2$. Using this approach, we construct codes that achieve high-sum rate. We describe an alternative formulation of the message assignment problem for $n \geq 3$, a problem which remains open.

2.1 WOM Codes and Flash Memories

The model of *write-once memories* was first proposed in [2]. WOMs are storage devices consisting of memory elements called *cells* each of which can change state from 0 to 1 but not vice-versa. Though such memories seem to be capable of only storing information once, *WOM codes* allow data to be ‘re-written’ on these WOMs without changing the state of any cell from 1 to 0. We present the construction from [2] in Table 2.1, which is referred to as the Rivest-Shamir code in literature. The code allows storing 2 bits of information twice in two writes using 3 WOM cells. All cells are assumed to be in state 0 before the first write. Let the data to be stored on the first write be 01, so the state of the cells after the first write is 100 as indicated by the first-write codebook. Suppose, the data is updated to 00, so according to the codebook for the second write, the state of the cells is changed to 111. It is clear from the table that, regardless of the data to be stored, no cell changes state from 1 to 0. The code achieves rates $2/3$ bits/cell in each of the two writes and therefore has an overall *sum-rate* of $4/3$ bits/cell.

Table 2.1: Rivest-Shamir 2-write WOM code

Data	First-Write Codebook	Second-Write Codebook
00	000	111
01	100	011
10	010	101
11	001	110

The WOM model introduced in [2] was generalized to memories with q -ary cells in [3]. In such a memory, each cell is in one out of q states and the legal state transitions are described by an arbitrary directed acyclic graph. The simplest such generalization allows the states $0, 1, \dots, q-1$ with the restriction that the state of a cell cannot decrease. These memories will be referred to as WOMs with *multilevel cells*. Designing high sum-

rate WOM codes for such memories are the topic of study in this chapter.

The study of write-once memories (WOMs) in the field of data storage was first motivated by applications like punch cards and optical disks [4]. More recently, WOM codes have been proposed as a lifetime-enhancing method for flash memories. WOM codes can be used to write information multiple times before an erase operation is required, thereby enhancing the lifetime of the device [5, 6, 7].

Considerable progress has been made in the last few years in the study of binary WOM code constructions for single-level cell (SLC) flash memory devices where each cell supports $q = 2$ levels [5, 7, 8]. However, to increase storage densities, future flash memory devices are expected to support a large number of cell levels, continuing the trend seen with the use of MLC and TLC devices that support 4 and 8 levels, respectively. This has motivated a body of work on the construction of WOM codes for multilevel cells that support $q \geq 3$ levels [9, 10, 11, 12, 13]. The capacity of WOM codes for t writes on q -ary cells is known to be $\log_2 \binom{q+t-1}{q-1}$ [14], although explicit characterization of the capacity region remains an open problem.

In this chapter, we consider the construction of lattice-based WOM codes for t writes on q -level cells. Lattice-based 2-write WOM codes over n cells in the asymptotic setting of continuous cell levels were derived in [13] for the fixed-rate scenario, where the cardinality of the message set is the same on each write. Allowing the cardinality of the message set on each write to be different can increase the sum-rate. Using a continuous approximation approach, it was hypothesized in [13] that the hyperbolic shaping regions were optimal for maximizing sum-rates of two writes over lattices in arbitrary dimensions. Optimality of hyperbolic shaping regions was proven in [15] for lattices in $n = 2$ dimensions when the number of writes, t , is arbitrary. A proof of optimality was provided in [16] for the case of an arbitrary number of cells, n , and $t = 2$ writes.

Here, we consider the most general case of an arbitrary number of writes on an arbitrary number of cells where each cell supports a large number of levels. Using the continuous approximation approach we prove that hyperbolic shaping regions are optimal for maximizing the sum-rate. We also discuss the problem of encoding for these codes. Encoding requires a consistent assignment of messages to cell levels that

does not depend on the state of the memory after the previous write. A consistent message assignment algorithm is optimal if, for any given codebook, the message set cardinality is the largest possible. We show that a consistent assignment scheme for a given codebook and message set cardinality may not always exist when $n > 2$. The problem of determining the existence of a consistent assignment for arbitrary n is shown to be an instance of an NP-complete problem, but with additional structure introduced by the geometry of the shaping regions. We exploit this structure to find an optimal linear-time algorithm for the case where $n = 2$. For $n \geq 3$, we use a sub-optimal algorithm for finding consistent assignments. Using these results, we construct codes that achieve high sum-rates for memories with multilevel cells.

The rest of the chapter is organized as follows. Section 2.2 formulates the code design problem. In Section 2.3, we extend ideas presented in [13] and invoke the continuous approximation to obtain an upper bound on the worst-case sum-rate optimal t -write regions for n cells and consider its asymptotic behavior as the number of cells grows large. In Section 2.4, we discuss the case where the cells support a finite number of levels and propose methods for assignment of messages to cell levels. Finally, we present our conclusions in Section 2.5. Proofs of the lemmas and propositions appearing in Sections 2.3 and 2.4 are given in the Appendix 2.6.

2.2 Lattice-based WOM Codes

We now give a precise definition of lattices and WOM codes that were introduced less formally in the previous section.

2.2.1 Lattices, Lattice Codes and Flash Memories

An n -dimensional lattice Λ is defined by an n -by- n generator matrix G . The lattice consists of the discrete set of points $\mathbf{x} = (x_1, x_2, \dots, x_n)$ for which

$$\mathbf{x} = \mathbf{b} \cdot G,$$

where $\mathbf{b} = (b_1, b_2, \dots, b_n) \in \mathbb{Z}^n$ is any n -dimensional integer-valued vector.

The Voronoi region of a point $\mathbf{x} \in \Lambda$ is the set of points of $\mathbb{R}^n$ which are closer to $\mathbf{x}$ than to any other point $\mathbf{x}' \in \Lambda$. The volume of a Voronoi region $\text{Vol}(\Lambda)$ is

$$\text{Vol}(\Lambda) \triangleq |\det \mathbf{G}|.$$

A lattice code $\mathcal{L}$ is a finite subset of a lattice Λ , described by a shaping region $\mathbb{A} \subset \mathbb{R}^n$,

$$\mathcal{L} = \Lambda \cap \mathbb{A}. \quad (2.1)$$

For a thorough treatment of lattices, refer to [17].

The coordinate values of an n -dimensional lattice code may be stored in n cells of a flash memory. In the most general case, cell j stores a continuous value between 0 and ℓ_j , so that the stored values in n cells are represented by $\mathbf{x} \in [0, \ell_1] \times [0, \ell_2] \times \dots \times [0, \ell_n] \triangleq \mathbb{A}$ where $x_j \in \mathbb{R}, \forall j = 1, 2, \dots, n$. The volume of region $\mathbb{A}$ is $\|\mathbb{A}\| = \prod_{j=1}^n \ell_j$. Allowing arbitrary ℓ_j proves to be helpful in the sequel. However, in the typical case of a q -ary flash memory, we have $\ell_j = q - 1$ for all j , so $\mathbb{A} = [0, q - 1]^n$ and the codebook is $\mathcal{L} = \mathbb{Z}_q^n$.

2.2.2 WOM Codebooks

Consider a flash memory device with n cells storing values $(x_1, x_2, \dots, x_n) \in \mathcal{L}$ such that the level of a cell can only be increased during a write operation. We consider writing information in these cells t times before there is a need for a block erasure.

A t -write WOM code stores M_i messages in n cells in the worst case at write i , $i = 1, \dots, t$. The instantaneous rate for write i is

$$R_{i,t} = \frac{1}{n} \log_2 M_i \text{ bits per cell per write.}$$

The *worst-case sum-rate* for the t -write code is

$$R_t = \sum_{i=1}^t R_{i,t} \text{ bits per cell per erase.}$$

A lattice-based t -write WOM codebook is defined by a partition of a lattice code $\mathcal{L}$ into

t subsets, denoted as $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_t$. The subset $\mathcal{L}_i$ is the codebook for write i and has cardinality $|\mathcal{L}_i|$. Note that since these codes have disjoint codebooks for each write, they are a special case of WOM codes referred to as *synchronized WOM codes* in [2].

A point $\mathbf{x} \in \mathcal{L}$ is said to be *accessible* from another point $\mathbf{s}$, denoted as $\mathbf{x} \succ \mathbf{s}$, if $x_j \geq s_j$ for all $j = 1, 2, \dots, n$ and $\mathbf{s} \neq \mathbf{x}$. Suppose the point stored at write $i - 1$ is $\mathbf{s}$; then the set of points that may be stored at write i is

$$\mathcal{L}_i(\mathbf{s}) \triangleq \{\mathbf{x} \in \mathcal{L}_i : \mathbf{x} \succ \mathbf{s}\}.$$

Here, $\mathcal{L}_i(\mathbf{s})$, the subset of $\mathcal{L}_i$ accessible from $\mathbf{s}$, may be a proper subset of the codebook $\mathcal{L}_i$. Since the worst-case rate is of interest, define *codebook cardinality*, denoted by C_i , as the minimum number of points in $\mathcal{L}_i$ that are accessible from any point in $\mathcal{L}_{i-1}$,

$$C_i \triangleq \min_{\mathbf{s} \in \mathcal{L}_{i-1}} |\mathcal{L}_i(\mathbf{s})|. \quad (2.2)$$

Also define the *total codebook cardinality*, Π_t , as

$$\Pi_t \triangleq \prod_{i=1}^t C_i.$$

The state of the memory before the first write is $\mathbf{s} = \mathbf{0}$ and all points in the codebook $\mathcal{L}_1$ are accessible. Thus, $M_1 = C_1$. However, the set of points that may be stored at any other write $i > 1$ depends on the point stored on the previous write. As a result, there may not exist a scheme which can consistently map C_i messages to points in the codebook $\mathcal{L}_i$. In some cases, then, M_i may be smaller than C_i ,

$$M_i \leq C_i,$$

and accordingly each rate $R_{i,t}$ is upper bounded as,

$$R_{i,t} \leq \frac{1}{n} \log_2 C_i,$$

and the worst-case sum-rate is upper bounded as,

$$R_t \leq \frac{1}{n} \log_2 \Pi_t. \quad (2.3)$$

Parts of this work concentrate on maximizing C_i and Π_t because they provide upper bounds on $R_{i,t}$ and R_t , respectively. The matter of consistent encoding-decoding is introduced in the next subsection and is discussed again in Section 2.4.

2.2.3 WOM Encoding and Decoding

The set of messages that can be stored on write i is $\mathcal{M}_i \triangleq \{1, 2, \dots, M_i\}$. For given codebooks $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_t$, the encoder-decoder pair (ϕ, ψ) is

$$\phi : \bigcup_{i=1}^t (\mathcal{M}_i \times \mathcal{L}_{i-1}) \mapsto \mathcal{L}, \quad \psi : \mathcal{L} \mapsto [t] \times \bigcup_{i=1}^t \mathcal{M}_i$$

where $[t] \triangleq \{1, 2, \dots, t\}$ which satisfy

$$\phi(m, \mathbf{x}) \in \mathcal{L}_i(\mathbf{x}) \quad \forall m \in \mathcal{M}_i, \mathbf{x} \in \mathcal{L}_{i-1}, \quad (2.4)$$

$$\psi(\mathbf{x}) \in \{i\} \times \mathcal{M}_i \quad \forall \mathbf{x} \in \mathcal{L}_i \quad (2.5)$$

for $i = 1, \dots, t$. The condition in (2.4) implies that on write i , the t -write code encodes a message by only increasing the cell levels and the condition in (2.5) implies that the decoder maps every point in $\mathcal{L}_i$ to a message in $\mathcal{M}_i$. For a consistent encoder-decoder pair, we will further require

$$\psi(\phi(m, \mathbf{x})) = (i, m) \quad \forall m \in \mathcal{M}_i, \mathbf{x} \in \mathcal{L}_{i-1} \quad (2.6)$$

for $i = 1, \dots, t$. The condition in (2.6) requires that the cell levels after write i are decoded to the correct message without the knowledge of any of the previous $i - 1$ writes. Thus, an n -cell t -write code $(\{\mathcal{L}_i\}, \phi, \psi)$ that satisfies conditions (2.4), (2.5) and (2.6) achieves worst-case sum-rate $R_t = \frac{1}{n} \sum_{i=1}^t \log M_i$.

Section 2.3 only considers the problem of partitioning the code $\mathcal{L}$ into t codebooks $\{\mathcal{L}_i\}$. Since the value of M_i cannot be determined without giving a consistent encoder-

decoder scheme, the codebook cardinality, C_i , will be used as the figure of merit to define sum-rates in Section 2.3.

2.2.4 Continuous Approximation

According to the *continuous approximation* principle for dense lattices [18], [19], the number of points in a codebook $\mathcal{L}$ formed using (2.1) can be approximated as

$$|\mathcal{L}| \approx \frac{\|\mathbb{A}\|}{\text{Vol}(\Lambda)},$$

where $|\mathcal{L}|$ denotes the cardinality of the discrete set $\mathcal{L}$ and $\|\mathbb{A}\|$ denotes the volume of the shaping region $\mathbb{A}$. This approximation becomes increasingly accurate as the density of the lattice increases. The use of the continuous approximation principle for WOM codes was introduced in [13].

WOM codebooks $\mathcal{L}_1, \dots, \mathcal{L}_t$ may be constructed by partitioning $\mathbb{A}$ into t write regions, $\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_t$. To construct codebooks for cells that support discrete levels, let

$$\mathcal{L}_i = \mathcal{L} \cap \mathbb{A}_i. \quad (2.7)$$

Applying the continuous approximation to the individual write regions, the codebook cardinality for the first write is approximated by

$$C_1 = |\mathcal{L}_1| \approx \frac{\|\mathbb{A}_1\|}{\text{Vol}(\Lambda)} \triangleq V_1.$$

If the state of the memory after write $i - 1$ is $\mathbf{s} \in \mathbb{A}_{i-1}$, then the set of possible levels that can be written on write i is

$$\mathbb{A}_i(\mathbf{s}) \triangleq \{\mathbf{x} \in \mathbb{A}_i : \mathbf{x} \succ \mathbf{s}\}.$$

Applying the continuous approximation for writes $2, 3, \dots, t$, the codebook cardinality is approximated by

$$C_i \approx \frac{1}{\text{Vol}(\Lambda)} \inf_{\mathbf{s} \in \mathbb{A}_{i-1}} \|\mathbb{A}_i(\mathbf{s})\| \triangleq V_i,$$

and the total codebook cardinality in t writes is approximated by

$$\Pi_t \approx \prod_{i=1}^t V_i \triangleq S_t.$$

In the following sections, the quantities V_i and S_t are also referred to as the codebook cardinality and the total codebook cardinality, respectively. Since both the lattice and the maximum cell values ℓ_i may be scaled arbitrarily, in Section 2.3 we assume that $\text{Vol}(\Lambda) = 1$.

2.3 Optimal Codebooks

In this section and the next section, we do not design WOM codebooks directly. Rather, we select shaping regions $\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_t$ and maximize the total codebook cardinality S_t . Under the continuous approximation principle, this corresponds to maximizing the upper bound on the worst-case sum rate (2.3).

Subsection 2.3.1 shows that for $t = 2$ writes, the optimal shaping region $\mathbb{A}_1$ has a hyperbolic shape. Subsection 2.3.2 extends this result to an arbitrary number of writes t . Subsection 2.3.3 considers the parameters defining the shaping regions which provide the optimal total codebook cardinality. Subsection 2.3.4 finds the value of S_t asymptotic in n , and gives conditions under which S_t approaches capacity.

2.3.1 Optimal 2-Write Regions

This subsection describes the case where n cells are used to store information twice before being erased, that is, $t = 2$. We will assume that the levels of the cell can only be increased from a previously written level.

Let $\mathbb{B}_1$ be the manifold that forms the boundary between $\mathbb{A}_1$ and $\mathbb{A}_2$. Alternatively, for a given boundary $\mathbb{B}_1$, $\mathbb{A}_1(\mathbb{B}_1)$ is defined as the closed region bounded by $\mathbb{B}_1$ and hyperplanes $x_j \geq 0$. The explicit dependence of the region $\mathbb{A}_1$ on the given boundary $\mathbb{B}_1$ will be suppressed when the meaning is clear from the context. With this notation, the

codebook cardinality in the first write is

$$V_1(\mathbb{B}_1) \triangleq \|\mathbb{A}_1\|.$$

Suppose the state of the memory after the first write is $\mathbf{s}$. Then, the region that remains available for writing is

$$\mathbb{A}_2(\mathbf{s}) = \{\mathbf{x} \in \mathbb{A}_2 : \mathbf{x} \succ \mathbf{s}\}.$$

This is a rectangular region with volume $\prod_{j=1}^n (\ell_j - s_j)$, that is

$$\|\mathbb{A}_2(\mathbf{s})\| = \prod_{j=1}^n (\ell_j - s_j).$$

In the worst case, the total codebook cardinality with two writes using n cells is

$$\begin{aligned} S_2(\mathbb{B}_1) &= V_1(\mathbb{B}_1) \cdot V_2(\mathbb{B}_1) \\ &= \|\mathbb{A}_1\| \cdot \inf_{\mathbf{s} \in \mathbb{A}_1} \|\mathbb{A}_2(\mathbf{s})\|. \end{aligned}$$

This quantity is a function of the boundary $\mathbb{B}_1$ alone. We refer to

$$\mathbb{B}_1^* = \arg \max_{\mathbb{B}_1 \subset \mathbb{A}} S_2(\mathbb{B}_1),$$

as the *optimal* boundary for the first write, assuming that the cells are to be used twice before each erase. The optimal boundary between the two write regions is known to be a rectangular hyperbola [16].

Definition 2.3.1. Define $\mathbb{H}(u)$ as the region in $\mathbb{A}$ enclosed by an n -dimensional rectangular hyperbola with parameter u , i.e.,

$$\mathbb{H}(u) \triangleq \left\{ \mathbf{x} \in \mathbb{A} : \prod_{j=1}^n (\ell_j - x_j) \geq u \cdot \|\mathbb{A}\| \right\},$$

with $0 \leq u \leq 1$.

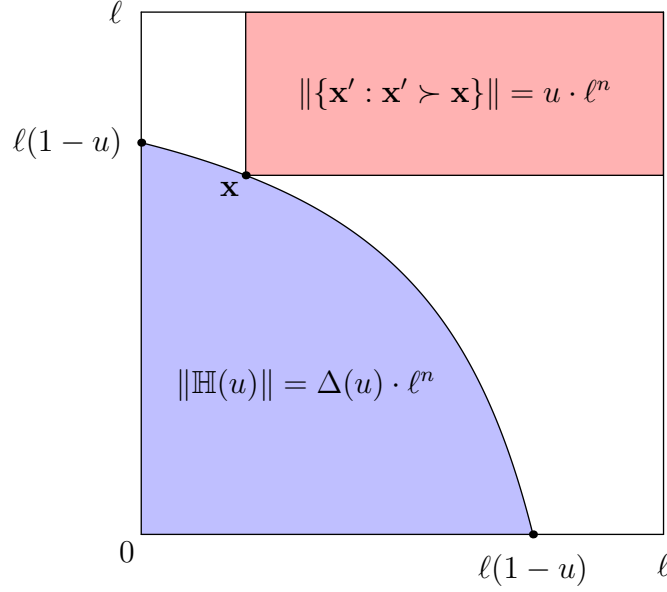


Figure 2.1: A 2-dimensional rectangular hyperbola $\mathbb{H}(u)$ in region $\mathbb{A} = [0, \ell] \times [0, \ell]$. The region under the hyperbola and the region accessible from a given point $\mathbf{x}$ on the hyperbola are shaded in blue and red, respectively, and their volumes are equal to $\Delta(u) \cdot \ell^n$ and $u \cdot \ell^n$, respectively.

Definition 2.3.2. *The normalized volume of the region $\mathbb{H}(u)$ is denoted as*

$$\Delta(u) \triangleq \|\mathbb{A}\|^{-1} \cdot \|\mathbb{H}(u)\|.$$

Geometrically, the parameter u , given in Definition 2.3.1, characterizes the point where the hyperbola touches the axes, and is also equal to the normalized volume of the region accessible from any point on the boundary of the hyperbola. $\Delta(u)$ is equal to the normalized volume of the region under the hyperbola. As will be shown in Lemma 2.3.5, $\Delta(u)$ can be expressed in closed form. Figure 2.1 shows a 2-dimensional rectangular hyperbola $\mathbb{H}(u)$.

The next lemma, first stated in [16], shows the optimality of the hyperbolic boundary.

Lemma 2.3.3 (Optimal boundary for 2 writes). *The optimal boundary for the first*

write-region for a 2-write WOM code is an n -dimensional rectangular hyperbola, i.e.

$$\mathbb{B}_1^* = \left\{ \mathbf{x} \in \mathbb{A} : \prod_{j=1}^n (\ell_j - x_j) = u_2^* \cdot \|\mathbb{A}\| \right\}$$

where

$$u_2^* = \arg \max_{u \in [0,1]} u \cdot \Delta(u).$$

Proof. Let $\mathbb{B}_1^*$ denote the optimal boundary for the first write. Then, the codebook cardinality for the first write is

$$V_1(\mathbb{B}_1^*) = \|\mathbb{A}_1(\mathbb{B}_1^*)\|$$

and for the second write is

$$\begin{aligned} V_2(\mathbb{B}_1^*) &= \inf_{\mathbf{x} \in \mathbb{A}_1(\mathbb{B}_1^*)} \|\mathbb{A}_2(\mathbf{x})\| \\ &= \inf_{\mathbf{x} \in \mathbb{B}_1^*} \prod_{j=1}^n (\ell_j - x_j). \end{aligned}$$

Define another boundary $\mathbb{B}'_1 \subset \mathbb{A}$ such that

$$\mathbb{B}'_1 = \left\{ \mathbf{x} \in \mathbb{A} : \prod_{j=1}^n (\ell_j - x_j) = V_2(\mathbb{B}_1^*) \right\}.$$

Then, for any $\mathbf{x} \in \mathbb{B}'_1$, $\|\mathbb{A}_2(\mathbf{x})\| = V_2(\mathbb{B}_1^*)$, and therefore, the boundaries $\mathbb{B}'_1$ and $\mathbb{B}_1^*$ achieve the same codebook cardinality on the second write, i.e.

$$V_2(\mathbb{B}'_1) = V_2(\mathbb{B}_1^*).$$

We claim that $\mathbb{A}_1(\mathbb{B}'_1) \supseteq \mathbb{A}_1(\mathbb{B}_1^*)$. Suppose the opposite is true, i.e., there exists some $\mathbf{x}' \in \mathbb{A}_1(\mathbb{B}_1^*)$ such that $\mathbf{x}' \notin \mathbb{A}_1(\mathbb{B}'_1)$ which implies that $\mathbf{x}' \notin \mathbb{H}\left(V_2(\mathbb{B}_1^*) \cdot \|\mathbb{A}\|^{-1}\right)$.

Then,

$$\prod_{j=1}^n (\ell_j - x'_j) < V_2(\mathbb{B}_1^*),$$

which contradicts the assumption that the codebook cardinality from any point in $\mathbb{A}_1(\mathbb{B}_1^*)$ is

$$V_2(\mathbb{B}_1^*) = \inf_{\mathbf{x} \in \mathbb{B}_1^*} \prod_{j=1}^n (\ell_j - x_j).$$

Hence, $\mathbb{A}_1(\mathbb{B}'_1) \supseteq \mathbb{A}_1(\mathbb{B}_1^*)$ and $\|\mathbb{A}_1(\mathbb{B}'_1)\| = V_1(\mathbb{B}'_1) \geq V_1(\mathbb{B}_1^*) = \|\mathbb{A}_1(\mathbb{B}_1^*)\|$; that is, the codebook cardinality on the first write when using the boundary $\mathbb{B}'_1$ is at least as large as the codebook cardinality when using the optimal boundary.

Suppose $V_1(\mathbb{B}'_1) > V_1(\mathbb{B}_1^*)$. Thus, using $\mathbb{B}'_1$ as the boundary for the first write increases the codebook cardinality for the first write, while ensuring that the worst-case codebook cardinality on the second write is still the same. This contradicts the assumption that $\mathbb{B}_1^*$ is optimal, and therefore, $\|\mathbb{A}_1(\mathbb{B}'_1)\| = V_1(\mathbb{B}'_1) = V_1(\mathbb{B}_1^*) = \|\mathbb{A}_1(\mathbb{B}_1^*)\|$. We have shown that the first-write regions corresponding to boundaries $\mathbb{B}_1^*$ and $\mathbb{B}'_1$ have the same volume, and one is a subset of the other. It follows from Lemma 2.6.1 in Appendix 2.6.1 that $\mathbb{A}_1(\mathbb{B}_1^*) = \mathbb{A}_1(\mathbb{B}'_1)$, and $\mathbb{B}_1^* = \mathbb{B}'_1$. The optimal total codebook cardinality in two writes is given by

$$\begin{aligned} S_2(\mathbb{B}_1^*) &= V_1(\mathbb{B}_1^*) \cdot V_2(\mathbb{B}_1^*) \\ &= V_1(\mathbb{B}'_1) \cdot V_2(\mathbb{B}'_1) \\ &= (\Delta(u_2^*) \cdot \|\mathbb{A}\|) \cdot (u_2^* \cdot \|\mathbb{A}\|). \end{aligned}$$

Since $\mathbb{B}_1^*$ is optimal, it follows that $u_2^* = \arg \max_{u \in [0,1]} u \cdot \Delta(u)$. This completes the proof. $\square$

In the next subsection, we extend Lemma 2.3.3 to the case of $t > 2$ writes.

2.3.2 Optimal t -Write Regions

Consider n cells that can store levels $[0, \ell_1] \times [0, \ell_2] \times \dots \times [0, \ell_n] = \mathbb{A}$. Let the subset $\mathbb{A}_i \subset \mathbb{A}$ denote the set of points that may be used on write i , for $i = 1, \dots, t$. If $\mathbf{s}_i \in \mathbb{A}_i$ is written on write i , the set of points, $\mathbb{A}_{i+1}(\mathbf{s}_i)$, that can be written in write $i + 1$ is

$$\mathbb{A}_{i+1}(\mathbf{s}_i) = \{\mathbf{x} \in \mathbb{A}_{i+1} : \mathbf{x} \succ \mathbf{s}_i\}.$$

Let $\mathbb{B}_i$ denote the boundary for write i , V_i denote the codebook cardinality for write i , and $S_t(\ell_1, \ell_2, \dots, \ell_n)$ denote the total codebook cardinality.

The following theorem gives the sum-rate optimal write-regions for t writes on n cells.

Theorem 2.3.4 (Sum-rate optimal t -writes). *The boundary for the i^{th} write when storing information t times on n cells such that the total codebook cardinality is maximized is given by*

$$\mathbb{B}_i^* = \left\{ \mathbf{x} \in \mathbb{A} : \prod_{j=1}^n (\ell_j - x_j) = \prod_{m=t-i+1}^t u_m^* \cdot \|\mathbb{A}\| \right\}$$

for all $i = 1, \dots, t-1$ where $u_1^* = 0$ and

$$u_k^* \triangleq \arg \max_{u \in [0,1]} \Delta(u) \cdot u^{k-1} \quad (2.8)$$

for all $k = 2, \dots, t$. The codebook cardinality on write i , for $i = 1, \dots, t$, is

$$V_i^* = \left(\prod_{m=t-i+2}^t u_m^* \right) \cdot \Delta(u_{t-i+1}^*) \cdot \|\mathbb{A}\|,$$

and the total codebook cardinality in t writes is

$$S_t^*(\ell_1, \dots, \ell_n) = \left(\prod_{m=2}^t (u_m^*)^{m-1} \Delta(u_m^*) \right) \cdot (\|\mathbb{A}\|)^t. \quad (2.9)$$

Proof. The theorem is proved by induction. From Lemma 2.3.3, the claim is true for

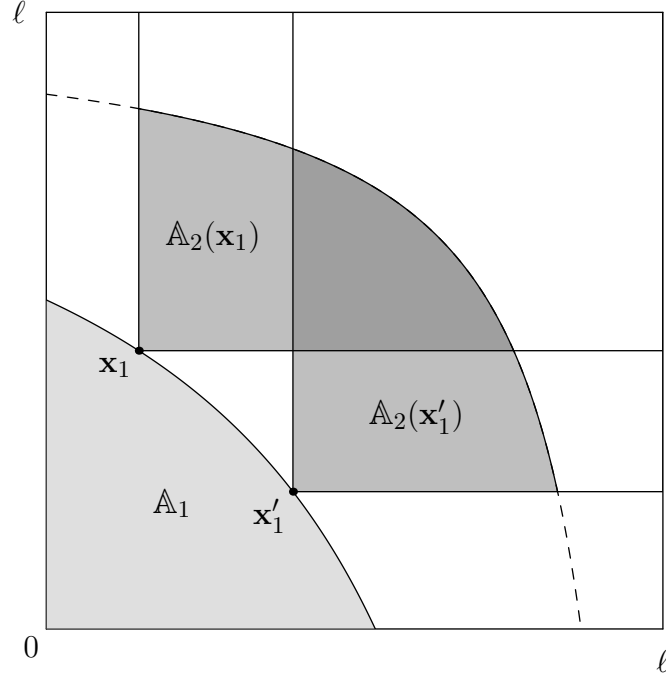


Figure 2.2: Optimal boundaries for 3 writes over 2 cells. We can see that the boundaries for the second write are independent of the points written on the first write.

$t = 2$. Suppose the claim is true for k writes. Consider the case for $k + 1$ writes. Let the optimal boundary for the first write be $\mathbb{B}_{1,k+1}^*$. Here the second subscript denotes that the total number of writes is $k + 1$. The codebook cardinality on the first write is

$$V_{1,k+1}(\mathbb{B}_{1,k+1}^*) = \|\mathbb{A}_{1,k+1}(\mathbb{B}_{1,k+1}^*)\|.$$

Suppose that a point $\mathbf{x} \in \mathbb{A}_{1,k+1}(\mathbb{B}_{1,k+1}^*)$ is written on the first write; then

$$S_k^*(\ell_1 - x_1, \ell_2 - x_2, \dots, \ell_n - x_n)$$

is an upper bound on the total codebook cardinality in k subsequent writes. Let $\Omega_2^{k+1}(\mathbb{B}_{1,k+1}^*)$ denote the product of the codebook cardinalities for writes $i = 2, 3, \dots, k+1$ after an arbitrary first write with $\mathbb{B}_{1,k+1}^*$ as the boundary of the first write-region. Thus, we can upper bound this product of codebook cardinalities as

$$\Omega_2^{k+1}(\mathbb{B}_{1,k+1}^*) \leq \inf_{\mathbf{x} \in \mathbb{A}_{1,k+1}(\mathbb{B}_{1,k+1}^*)} S_k^*(\ell_1 - x_1, \dots, \ell_n - x_n)$$

$$\begin{aligned}
&= \inf_{\mathbf{x} \in \mathbb{A}_{1,k+1}(\mathbb{B}_{1,k+1}^*)} c_k^* \cdot [(\ell_1 - x_1)(\ell_2 - x_2) \dots (\ell_n - x_n)]^k \quad (2.10) \\
&= c_k^* \left[\inf_{\mathbf{x} \in \mathbb{A}_{1,k+1}(\mathbb{B}_{1,k+1}^*)} \prod_{j=1}^n (\ell_j - x_j) \right]^k,
\end{aligned}$$

where $c_k^* = \prod_{m=2}^k (u_m^*)^{m-1} \Delta(u_m^*)$ is a constant independent of $\mathbf{x}$ or $\mathbb{B}_{1,k+1}^*$. Here the equality labeled (2.10) follows from the induction hypothesis. Let $p \in [0, 1]$ satisfy

$$p \triangleq \|\mathbb{A}\|^{-1} \cdot \inf_{\mathbf{x} \in \mathbb{A}_{1,k+1}(\mathbb{B}_{1,k+1}^*)} \prod_{j=1}^n (\ell_j - x_j).$$

Define $\mathbb{B}'$ to be an n -dimensional hyperbola such that

$$\prod_{j=1}^n (\ell_j - y_j) = p \cdot \|\mathbb{A}\|$$

for all $\mathbf{y} \in \mathbb{B}'$. Then, using the definition of p , one can easily show that

$$\mathbb{A}_{1,k+1}(\mathbb{B}_{1,k+1}^*) \subseteq \mathbb{A}_{1,k+1}(\mathbb{B}') = \mathbb{H}(p)$$

and therefore,

$$V_{1,k+1}(\mathbb{B}_{1,k+1}^*) \leq V_{1,k+1}(\mathbb{B}') = \Delta(p) \cdot \|\mathbb{A}\|.$$

By the induction hypothesis, for a point $\mathbf{x}' \in \mathbb{B}'$, the optimal boundary for the i^{th} subsequent write in $[x'_1, \ell_1] \times [x'_2, \ell_2] \times \dots \times [x'_n, \ell_n]$, is

$$\begin{aligned}
&\mathbf{x}' + \mathbb{B}_{i,k}^*(\ell_1 - x'_1, \dots, \ell_n - x'_n) \quad (2.11) \\
&= \left\{ \mathbf{x} \in \mathbb{A} : \prod_{j=1}^n ((\ell_j - x'_j) - (x_j - x'_j)) = \prod_{m=k-i+1}^k u_m^* \cdot \prod_{j=1}^n (\ell_j - x'_j) \right\} \\
&= \left\{ \mathbf{x} \in \mathbb{A} : \prod_{j=1}^n (\ell_j - x_j) = \prod_{m=k-i+1}^k u_m^* \cdot p \cdot \|\mathbb{A}\| \right\}
\end{aligned}$$

where $\mathbb{B}_{i,k}^*(a_1, \dots, a_n)$ is the optimal boundary for cells that can store levels $[0, a_1] \times$

$\dots \times [0, a_n]$ and $\mathbf{x} + \mathbb{B}$ denotes the set $\{\mathbf{x} + \mathbf{y} : \mathbf{y} \in \mathbb{B}\}$. Note that the boundary in (2.11) is independent of the point $\mathbf{x}'$ written on the first write, for all $i = 1, \dots, k-1$. This implies that

$$\begin{aligned}\Omega_2^{k+1}(\mathbb{B}') &= c_k^* \cdot p^k \|\mathbb{A}\|^k \\ &\geq \Omega_2^{k+1}(\mathbb{B}_{1,k+1}^*).\end{aligned}$$

Therefore the optimal boundary $\mathbb{B}_{1,k+1}^*$ is same as the hyperbolic boundary $\mathbb{B}'$, and

$$V_{1,k+1}(\mathbb{B}_{1,k+1}^*) = \|\mathbb{A}_{1,k+1}(\mathbb{B}')\| = \Delta(p) \cdot \|\mathbb{A}\|$$

for some $p \in [0, 1]$. Then the total codebook cardinality in $k+1$ writes is

$$\begin{aligned}S_{k+1}(\mathbb{B}_{1,k+1}^*) &= V_{1,k+1}(\mathbb{B}') \cdot \Omega_2^{k+1}(\mathbb{B}') \\ &= c_k^* \|\mathbb{A}\|^{k+1} \cdot \Delta(p) \cdot p^k\end{aligned}$$

By (2.8), the value of p that maximizes $p^k \cdot \Delta(p)$ is u_{k+1}^* , which implies that

$$\mathbb{B}_{i,k+1}^* = \left\{ \mathbf{x} \in \mathbb{A} : \prod_{j=1}^n (\ell_j - x_j) = \prod_{m=k-i+2}^{k+1} u_m^* \cdot \|\mathbb{A}\| \right\}$$

for $i = 1, \dots, k$. Hence the claim is true for $k+1$ writes. This completes the induction step, proving the theorem. $\square$

2.3.3 Computing the optimal hyperbola parameters, u_k^*

The expression defining u_k^* in (2.8) can be interpreted as follows. When the first write region is $\mathbb{H}(u_k^*)$, the total codebook cardinality for k writes, S_k , is given by the product of the codebook cardinality in the first write, V_1 , and the total codebook cardinality in the last $k-1$ writes, S_{k-1} . V_1 is proportional to $\Delta(u_k^*)$, the normalized volume enclosed by the hyperbola. In the worst case, the volume accessible for the last $k-1$ writes is equal to the volume accessible from a point on the boundary of $\mathbb{H}(u_k^*)$ and, thus, proportional to u_k^* . Then, replacing t by $k-1$ and $\|\mathbb{A}\|$ by a constant times u_k^* in

(2.9), the total codebook cardinality in the last $k - 1$ writes is proportional to $(u_k^*)^{k-1}$. Therefore, the total codebook cardinality in k writes is proportional to $(u_k^*)^{k-1} \cdot \Delta(u_k^*)$ and the optimal hyperbola parameter is clearly given by the expression in (2.8). We now show how to compute this optimal parameter. We make use of the following lemma, which gives an expression for the normalized volume of the region $\mathbb{H}(u)$.

Lemma 2.3.5. *For $n \geq 2$,*

$$\Delta(u) = 1 - u \sum_{i=0}^{n-1} \frac{1}{i!} \left[\ln \left(\frac{1}{u} \right) \right]^i$$

Proof. See Appendix 2.6.2. □

First consider the case of $n = 2$ cells. The following proposition from [15] gives the value of the optimal hyperbola parameter, u_k^* .

Proposition 2.3.6. *For $n = 2$ cells and for any $k \geq 2$, let*

$$\begin{aligned} f_k(u) &\triangleq \Delta(u) \cdot u^{k-1} \\ &= (1 - u + u \ln u) \cdot u^{k-1} \end{aligned}$$

for all $u \in [0, 1]$ and let $\tau_k \triangleq -\frac{k-1}{k}$. Then, f_k has one local maximum in the interval $[0, 1]$ at

$$u_k^* = \arg \max_{u \in (0, 1)} f_k(u) = \frac{\tau_k}{W_{-1}(\tau_k e^{\tau_k})}$$

where W_{-1} is the real branch of the Lambert W function satisfying $W(x) < -1$ [20].

Proof. See Appendix 2.6.3. □

For $n = 2$, the proposition shows that the optimal hyperbola parameter, u_k^* , can be expressed in closed form using the Lambert W function. In contrast, for arbitrary n , the parameter u_k^* can only be found numerically. For the case of $t = 2$ writes on n cells, let $z_n^* \in [0, \infty)$ satisfy $u_2^* = e^{-z_n^*}$, where u_2^* is the optimal hyperbola parameter. We computed the value of z_n^* for different values of n and plot the results in Figure 2.4.

We now bound u_t^* as a function of t and n .

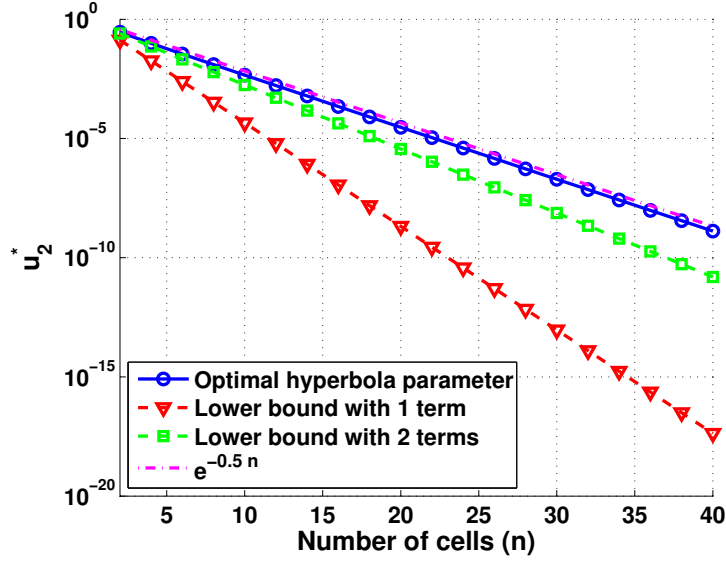


Figure 2.3: Optimal hyperbola parameter, u_2^* , computed numerically and bounds from Section 2.3.3.

Proposition 2.3.7. For $t \geq 2$,

$$e^{-\frac{n}{t-1}} < u_t^* < e^{-\frac{n}{t}}$$

Proof. See Appendix 2.6.4. □

Remark 2.3.8. Note that tighter bounds may be obtained by setting multiple terms from the summation in (2.20) equal to $\frac{1}{t-1} \cdot \frac{1}{(n-1)!}$, as done in Proposition 2.3.7 with one term. For example, Figure 2.3 shows the lower bounds obtained on u_2^* with one and two terms in the summation in (2.20), compared to actual values obtained numerically.

2.3.4 Asymptotic Rates

In this subsection, we determine the rates that the continuous approximation achieves asymptotically as the number of cells, n , becomes large, that is, as $n \rightarrow \infty$, when cells can store fixed discrete levels in $\mathbb{Z}_q^n$. We consider $t = 2$ writes on n cells in the region $[0, q-1]^n$ in this section.

Let the first write region be the hyperbolic region $\mathbb{H}(u)$. Then the total codebook cardinality in two writes is $S_2(u) = u \cdot \Delta(u) \cdot \|\mathbb{A}\|^2$. We consider $u = e^{-z_n}$ for some non-

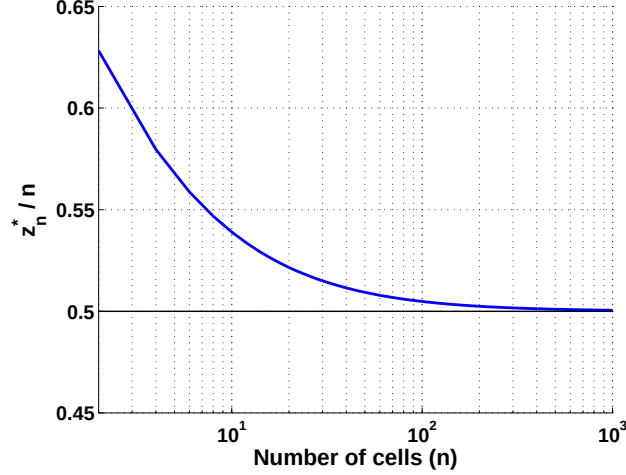


Figure 2.4: Ratio of z_n^* and n , where $u_2^* = e^{-z_n^*}$ is the optimal hyperbola parameter for $t = 2$ writes on n cells. The ratio z_n^*/n converges monotonically to 0.5 as n increases.

negative $z_n \in \mathbb{R}$ so that $0 \leq u \leq 1$. Then, by Lemma 2.3.5,

$$\begin{aligned} S_2(e^{-z_n}) &= e^{-z_n} \cdot \Delta(e^{-z_n}) \cdot \|\mathbb{A}\|^2 \\ &= e^{-z_n} \left(1 - e^{-z_n} \sum_{i=0}^{n-1} \frac{(z_n)^i}{i!} \right) \cdot (q-1)^{2n} \end{aligned}$$

where $\|\mathbb{A}\| = \prod_{i=1}^n \ell_i = (q-1)^n$. Continuing, the sum-rate upper bound in (2.3) is

$$\begin{aligned} R_2 &\leq \frac{1}{n} \log_2 \Pi_2 \approx \frac{1}{n} \log_2 S_2(e^{-z_n}) \\ &= \log_2 (q-1)^2 - \log_2 e \cdot \frac{z_n}{n} + \log_2 [P(n, z_n)]^{\frac{1}{n}} \end{aligned} \quad (2.12)$$

where $P(n, x)$ is the lower normalized incomplete Gamma function [21].

The value of z_n that maximizes the expression in (2.12), denoted by z_n^* , can be found numerically. Figure 2.4 shows that z_n^*/n approaches 0.5 monotonically as n increases. The following proposition confirms this numerical behavior.

Proposition 2.3.9. *Consider $t = 2$ writes on n cells with hyperbola parameter $u_2 = e^{-\alpha n}$, where α depends on n . For increasing n , the optimal value of α , in the sense of maximizing the total codebook cardinality, $S_2(e^{-\alpha n})$, approaches 0.5, that is*

$$\lim_{n \rightarrow \infty} \arg \max_{\alpha} \frac{1}{n} \log_2 S_2(e^{-\alpha n}) = 0.5,$$

and

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log_2 S_2(e^{-0.5n}) = \log_2(q^2 - 2q + 1) - 1.$$

Proof. See Appendix 2.6.5. □

For comparison, the capacity for t writes on cells with q discrete levels was shown in [14] to be

$$C_{q,t} \triangleq \log_2 \binom{q+t-1}{q-1}.$$

For $t = 2$ writes,

$$\begin{aligned} R_2 &\leq \lim_{n \rightarrow \infty} \frac{1}{n} \log_2 \Pi_2 \\ &\approx \lim_{n \rightarrow \infty} \frac{1}{n} \log_2 S_2(e^{-z_n^*}) \\ &< C_{q,2} = \log_2(q^2 + q) - 1. \end{aligned}$$

That is, for any fixed q , even if the optimal value z_n^* is selected, the sum-rate under the assumptions of the continuous approximation is strictly less than the capacity. But, as q becomes large, their ratio goes to 1.

2.4 Message Assignment for Codes with Discrete Support

In this section we consider cells that support discrete levels $\mathbf{x} \in \mathcal{L}$ and discuss the construction of an encoder-decoder pair (ϕ, ψ) for a given codebook $\{\mathcal{L}_i\}_{i=1}^t$ as described in Section 2.2.2. We associate with the encoder-decoder pair a function

$$\Phi : \mathcal{L} \mapsto \cup_i \mathcal{M}_i,$$

such that $\Phi(\mathbf{x})$ denotes the message associated with any point $\mathbf{x} \in \mathcal{L}$. Then, the encoder and decoder are defined in Algorithm 2.1 using look-ups of the message assignment Φ . We will refer to the function Φ as the *message assignment*. The encoder-decoder pair (ϕ, ψ) that satisfy the conditions (2.4), (2.5) and (2.6) can be defined in this manner if

and only if Step 7 in Algorithm 2.1 can be performed successfully, that is, Φ satisfies the following property for all $j = 1, \dots, t$ and for all $\mathbf{x} \in \mathcal{L}_{j-1}$:

$$\exists \mathbf{x}' \in \mathcal{L}_j(\mathbf{x}) : \Phi(\mathbf{x}') = m \quad \forall m \in \mathcal{M}_j. \quad (2.13)$$

The condition in (2.13) implies that from any point, $\mathbf{x}$, there exists at least one accessible point in the next write-region, $\mathbf{x}'$, that encodes the message m , for all messages m .

Algorithm 2.1 Encoder-decoder pair (ϕ, ψ) based on matrix Φ

```

1: // Inputs are message to be written and current cell levels
2: // Output is next cell levels
Require:  $(m, \mathbf{x}) \in \cup_{j=1}^t (\mathcal{M}_j \times \mathcal{L}_{j-1})$ 
3: function  $\phi(m, \mathbf{x})$ 
4:   // Determine the number of writes already completed
5:    $i \leftarrow \sum_{j=1}^t j \cdot \mathbf{1}\{\mathbf{x} \in \mathcal{L}_{j-1}\}$ 
6:   // Find an accessible point that encodes message  $m$ 
7:   Find  $\mathbf{x}' \in \mathcal{L}_i(\mathbf{x}) : \Phi(\mathbf{x}') = m$ 
8:   return  $\mathbf{x}'$ 
9: end function


---


10: // Input is current cell levels
11: // Outputs are number of writes and message
12: function  $\psi(\mathbf{x})$ 
13:   // Determine the number of writes done
14:    $i \leftarrow \sum_{j=1}^t j \cdot \mathbf{1}\{\mathbf{x} \in \mathcal{L}_j\}$ 
15:   // Determine the message that point  $\mathbf{x}$  encodes
16:    $m \leftarrow \Phi(\mathbf{x})$ 
17:   return  $(i, m)$ 
18: end function

```

In the following subsections, we discuss the construction of a message assignment function Φ for the case of $n = 2$ cells, and then for an arbitrary number of cells. Note that [12] uses a lattice to define the message assignment function where codebooks are defined using tilings of the lattice. Here, we consider message assignment problem for an arbitrary codebook $\{\mathcal{L}_i\}$. For numerical results in this section, the codebooks were constructed using (2.7) with the proposed hyperbolic shaping regions as $\mathbb{A}_i$.

2.4.1 Message assignment for 2 cells

We first consider the case of $n = 2$ cells with cell levels $(x_1, x_2) \in \mathcal{L} = \mathbb{Z}_q^2$. In this case, we propose an algorithm to find a function Φ that satisfies (2.13) for a given codebook $\{\mathcal{L}_i\}_{i=1}^t$ such that the number of messages on write i , M_i , is equal to the codebook cardinality, C_i . The implementation details are given in Algorithm 2.2. We now describe how this iterative algorithm works and prove its correctness.

For each write i , on iteration 0, we start at the point in the codebook $\mathcal{L}_{i-1}$ that can access the least number of points,

$$(\hat{x}_1, \hat{x}_2) = \arg \min_{(x_1, x_2) \in \mathcal{L}_{i-1}} |\mathcal{L}_i(x_1, x_2)|.$$

Let us call this point the *pivot point* at iteration 0. Note that $|\mathcal{L}_i(\hat{x}_1, \hat{x}_2)|$ is equal to C_i by the definition of codebook cardinality in (2.2). We assign messages $\mathcal{M}_i = \{1, 2, \dots, C_i\}$ to the points in $\mathcal{L}_i$ accessible from this pivot point. On the first iteration, the algorithm considers as the pivot point a point $(x_1'', x_2'') \in \mathcal{L}_{i-1}$ such that $x_1'' = \hat{x}_1 - 1$ and x_2'' is the largest possible. On subsequent iterations, the algorithm will consider pivot points with decreasing levels on the first cell. The pivot point in the previous iteration is denoted as (x_1', x_2') .

At every iteration, the algorithm finds messages previously assigned to the points in $\mathcal{L}_i(x_1', x_2') \setminus \mathcal{L}_i(x_1'', x_2'') \triangleq \mathcal{L}_{\text{lost}}(x_1'', x_2'')$. Let $\mathcal{M}_{\text{lost}}$ be the set of these messages that were encodable in the previous iteration but are not encodable in the current iteration. The algorithm then assigns $\mathcal{M}_{\text{lost}}$ to the points $\mathcal{L}_i(x'', y'')$ that have not yet been assigned any messages, a set we denote as $\mathcal{L}_{\text{spare}}(x'', y'')$. The algorithm repeats these steps, with a different pivot point at every iteration. We show in Appendix 2.6.6 that $|\mathcal{L}_{\text{spare}}(x'', y'')| \geq |\mathcal{M}_{\text{lost}}|$ at every iteration, so that a message assignment function that satisfies the condition in (2.13) with $M_i = C_i$ can be defined for any given codebook.

A 4-write code for $n = 2$ cells with $q = 8$ levels each is shown in Figure 2.5. For this code, the codebook $\{\mathcal{L}_i\}_{i=1}^4$ is constructed using Equation (2.7) where the shaping regions $\{\mathbb{A}_i\}$ are as defined in Theorem 2.3.4 with $\ell = q - 1 = 7$ and the message assignment function Φ is defined using Algorithm 2.2. The code allows 8, 8, 9 and 8

Algorithm 2.2 Algorithm to construct Φ given $\{\mathcal{L}_i\}_{i=1}^t$

```

1: procedure DEFINEPHI ( $\{\mathcal{L}_i\}_{i=1}^t$ )
2:   for all  $i \in \{1, \dots, t\}$  do
3:     // Initialize  $\Phi$  to Unassigned (denoted by  $\epsilon$ )
4:     for all  $(x_1, x_2) \in \mathcal{L}_i$  do
5:        $\Phi(x_1, x_2) \leftarrow \epsilon$ 
6:     end for
7:     // Start with point that achieves the min. volume
8:      $(\hat{x}_1, \hat{x}_2) \leftarrow \arg \min_{(x_1, x_2) \in \mathcal{L}_{i-1}} |\mathcal{L}_i(x_1, x_2)|$ 
9:      $\mathcal{M}_i \leftarrow \{1, 2, \dots, |\mathcal{L}_i(\hat{x}_1, \hat{x}_2)|\}$ 
10:    // Assign messages to points  $(\hat{x}_1, \hat{x}_2)$  can access
11:    ASSIGNPHI ( $\mathcal{L}_i(\hat{x}_1, \hat{x}_2), \mathcal{M}_i$ )
12:    // Assign messages from other points in  $\mathcal{L}_{i-1}$ 
13:     $(x'_1, x'_2) \leftarrow (\hat{x}_1, \hat{x}_2)$ 
14:    while  $x'_1 > 1 \wedge x'_2 < q$  do
15:      // Find new pivot point and lost messages
16:       $(x''_1, x''_2) \leftarrow (x'_1 - 1, x_2) \in \mathcal{L}_{i-1} : x_2 \text{ is largest}$ 
17:       $\mathcal{M}_{\text{lost}} \leftarrow \mathcal{M}_i \setminus \Phi(\mathcal{L}_i(x''_1, x''_2))$ 
18:      ASSIGNPHI ( $\mathcal{L}_i(x''_1, x''_2), \mathcal{M}_{\text{lost}}$ )
19:       $(x'_1, x'_2) \leftarrow (x''_1, x''_2)$ 
20:    end while
21:     $(x'_1, x'_2) \leftarrow (\hat{x}_1, \hat{x}_2)$ 
22:    while  $x'_1 < q \wedge x'_2 \geq 1$  do
23:      // Find new pivot point and lost messages
24:       $(x''_1, x''_2) \leftarrow (x'_1 + 1, x_2) \in \mathcal{L}_{i-1} : x_2 \text{ is largest}$ 
25:       $\mathcal{M}_{\text{lost}} \leftarrow \mathcal{M}_i \setminus \Phi(\mathcal{L}_i(x''_1, x''_2))$ 
26:      ASSIGNPHI ( $\mathcal{L}_i(x''_1, x''_2), \mathcal{M}_{\text{lost}}$ )
27:       $(x', y') \leftarrow (x'', y'')$ 
28:    end while
29:  end for
30: end procedure

```

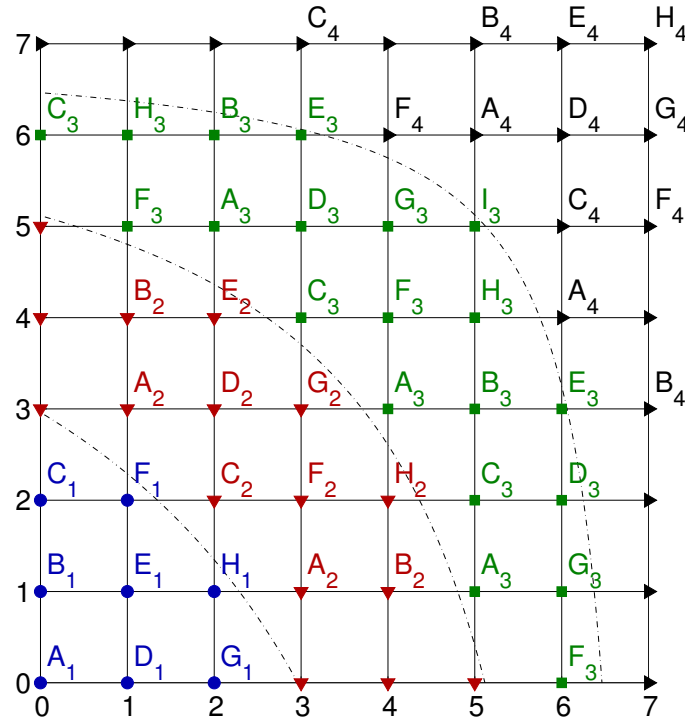


Figure 2.5: 4-write code for 2 cells with 8 levels each that achieves rate 6.085 bits/cell/erase. Points in the i^{th} codebook are assigned messages $\{A_i, B_i, \dots\}$ according to message assignment function defined in Algorithm 2.2 such that after any $i - 1$ writes, M_i messages may be stored on the next write.

Algorithm 2.3 Procedure to assign messages $\mathcal{M}'$ to points in set $\mathcal{L}'$

Require: $|\mathcal{M}'| \leq |\{(x_1, x_2) \in \mathcal{L}' : \Phi(x_1, x_2) = \epsilon\}|$

```

1: procedure ASSIGNPHI ( $\mathcal{L}', \mathcal{M}'$ )
2:    $\mathcal{L}_{\text{spare}} \leftarrow \{(x_1, x_2) \in \mathcal{L}' : \Phi(x_1, x_2) = \epsilon\}$ 
3:   for all  $m \in \mathcal{M}'$  do
4:      $(x_{\text{spare}}, y_{\text{spare}}) \leftarrow (x, y) : (x, y) \in \mathcal{L}_{\text{spare}}$ 
5:      $\Phi(x_{\text{spare}}, y_{\text{spare}}) \leftarrow m$ 
6:      $\mathcal{L}_{\text{spare}} \leftarrow \mathcal{L}_{\text{spare}} \setminus \{(x_{\text{spare}}, y_{\text{spare}})\}$ 
7:   end for
8: end procedure

```

Table 2.2: Worst-Case Sum-Rates R_t in bits per cell per erase achieved by t -write codes on 2 cells with q levels.

q	4	8	12	16	32
$\downarrow t \setminus \log_2 q \rightarrow$	2	3	3.59	4	5
2	2.70	4.55	5.63	6.44	8.40
3	2.95	5.48	7.11	8.25	11.13
4	2.59	6.09	8.17	9.71	13.46
5	2.09	6.55	9.07	10.90	15.40
6	1.79	6.61	9.63	11.80	17.21
7	—	6.70	10.10	12.54	18.72
8	—	6.42	10.26	13.15	20.22
9	—	6.38	10.55	13.73	21.43
10	—	5.88	10.78	14.19	22.57

messages on the sequence of four writes to give a worst-case sum-rate of

$$R_4 = \frac{1}{2} \log_2(8 \cdot 8 \cdot 9 \cdot 8) = 6.085 \text{ bits per cell per erase.}$$

In Table 2.2, we list the worst-case sum rate R_t achieved by the proposed codes over $n = 2$ cells that support levels $\{0, 1, \dots, q - 1\}$, for various values of t and q . With no coding, these cells achieve a sum-rate equal to $\log_2 q$ bits/cell/erase. As is clear from Table 2.2, the proposed WOM codes achieve a significant gain in sum-rate even when coding is done over only 2 cells.

2.4.2 Message assignment for arbitrary number of cells

In the case of $n = 2$ cells above, we could always define an encoder-decoder pair that, on any write, could store a set of messages with cardinality equal to the codebook cardinality, that is, $M_i = C_i$ for all i . However, when $n > 2$, a consistent encoder-decoder with this property may not exist for some codebooks over n cells as we show in the following example.

Example 2.4.1. *Consider a code over 3 cells that support 2 levels, $\{0, 1\}$, with codebooks*

$$\begin{aligned}\mathcal{L}_1 &= \{(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1)\} \\ &= \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}\end{aligned}$$

and

$$\begin{aligned}\mathcal{L}_2 &= \{(1, 1, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)\} \\ &= \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{c}\}.\end{aligned}$$

Then, there are 3 points in the second write-region that are accessible from points $\mathbf{x}_i$ for $i = 1, 2, 3$. For example, points $\mathbf{b}_1, \mathbf{b}_2$ and $\mathbf{c}$ are accessible from $\mathbf{x}_1$, but $\mathbf{b}_3$ is not. This is depicted schematically in Figure 2.6 where points $\mathcal{L}_1$ are represented as circular regions and accessible points in $\mathcal{L}_2$ for each of the points in $\mathcal{L}_1$ are represented as subregions of the respective regions. Clearly $C_2 = 3$. However, there does not exist any assignment of 3 messages to points in $\mathcal{L}_2$ such that all points in $\mathcal{L}_1$ satisfy the condition in (2.13) required for a valid Φ function. Therefore, $M_2 < C_2$ for the given codebooks.

It is easy to determine that the maximum number of messages that can be stored in the second write for the case discussed in Example 2.4.1 is 2; that is, $M_2 = 2$. However, in general, given codebooks $\mathcal{L}_{i-1}$ and $\mathcal{L}_i$ and a positive integer $M \geq 2$, the problem to determine whether there exists an encoder-decoder pair that guarantees M messages on write i is an instance of a combinatorial problem, referred to as *Set M -Coloring* in [22], and defined as follows.

Definition 2.4.1 (Set M -Coloring). *Given a collection of finite subsets $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m$*

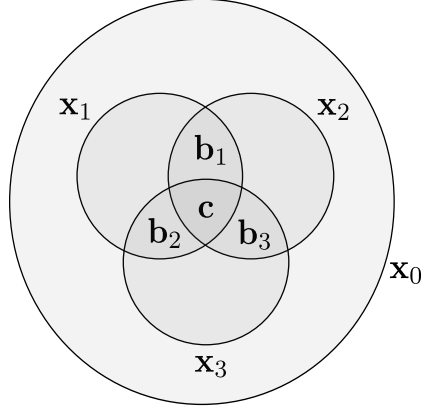


Figure 2.6: Diagram for Example 2.4.1 showing points of second-write region accessible from points of first write region

of a set $\mathcal{U}$, does there exist a coloring

$$I : \bigcup_{k=1}^m \mathcal{S}_k \mapsto \{1, 2, \dots, M\}$$

such that, for each $k = 1, \dots, m$, and for each color $j = 1, \dots, M$, the set $\mathcal{S}_k$ has at least one element of color j ?

One can see that the existence of a message assignment with M messages can be interpreted as an instance of the Set M -Coloring problem. Specifically suppose $\mathcal{L}_{i-1} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$. Then sets $\mathcal{U} = \mathcal{L}_i$ and $\mathcal{S}_k = \mathcal{L}_i(\mathbf{x}_k)$ for $k = 1, \dots, m$. It was shown in [22, Theorem 5.5] that the Set M -Coloring problem is NP-complete [23] for any fixed $M \geq 2$. It is important to note that this does not prove hardness for the message assignment problem. Indeed, we show that Algorithm 2.2 is a polynomial-time algorithm that solves the message assignment problem for $n = 2$.

For a given codebook, the problem of finding a message assignment for a given codebook has been interpreted as finding the disjoint coverings for a corresponding bipartite graph, as shown in [24]. This bipartite covering problem is an optimization version of the Set- M Coloring problem and therefore is NP-hard. Solving this covering problem sub-optimally [25, Algorithm 1], we have found message assignments for some codebooks that achieve higher sum-rates with 3 cells than achieved with 2 cells in Table 2.2. For e.g., for $t = 2$ writes on $n = 3$ cells where $q = 8$, we could construct a message assignment function with $M_1 = 136$ and $M_2 = 101$, giving a sum-rate 4.582 bits/cell/erase;

the $n = 2$ cells case achieves 4.554 bits/cell/erase for the same t and q .

2.5 Concluding Remarks

This chapter described the design of lattice-based WOM codebooks using minimal assumptions of the continuous approximation and a worst-case rate. The optimal shaping regions were shown to be bounded by rectangular hyperbolas, and we defined a total codebook cardinality Π_t which gave an upper bound on the sum-rate. For large q , corresponding to the condition of the continuous approximation, this upper bound approaches the capacity, for the case of $t = 2$ writes.

In this chapter, we derived worst-case sum-rate optimal write-regions. In practice, one may require the rates on all writes to be equal. In this case of fixed-rate for all writes, it has been optimal write-regions for t writes on an arbitrary number of cells for memories that support continuous levels are also bounded by rectangular hyperbolas with a suitable parameter [26].

Note that the analysis only considered maximizing the size of codebooks. Indeed, an example showed that a consistent encoder-decoder pair that achieves rates equal to the codebook cardinality may not exist for all codebooks when $n \geq 3$. That is, it has not been shown that hyperbolic shaping regions have a consistent encoder-decoder that can achieve capacity. But for $n = 2$ cells, we proposed an algorithm to find a message assignment for codes and proved that the algorithm achieves a message set cardinality equal to the codebook cardinality and is therefore optimal.

Determining the existence of a message assignment with $M \geq 2$ messages for codes over multiple cells for a given codebook is a special case of the Set M -Coloring problem. An important open problem is finding low-complexity algorithms to determine message assignments that can achieve high rates for a given codebook.

2.6 Appendix

2.6.1 Lemma for Section 2.3

The following lemma [27] is used to prove the uniqueness of the optimal boundary in Section 2.3.

Lemma 2.6.1. *Let $\mathbb{C}$ and $\mathbb{D}$ be closed subsets of $\mathbb{A}$. Let $\mathbb{E}_{\mathbf{x},\delta} = \{\mathbf{y} \in \mathbb{A} : \|\mathbf{y} - \mathbf{x}\| < \delta\}$ be an open ball with radius δ and center $\mathbf{x}$ and let $\|\mathbf{x} - \mathbf{y}\|$ denote the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. Suppose $\mathbb{C} \subseteq \mathbb{D}$ and $\|\mathbb{C}\| = \|\mathbb{D}\| > 0$. If, $\forall \delta > 0$ and $\forall \mathbf{x} \in \mathbb{D}$, $\|\mathbb{D} \cap \mathbb{E}_{\mathbf{x},\delta}\| > 0$, then $\mathbb{C} = \mathbb{D}$.*

Proof. Suppose the opposite is true; that is, suppose $\mathbb{C}$ is strictly contained in $\mathbb{D}$. Let $\mathbf{x}' \in \mathbb{A}$ be a point in $\mathbb{A}$ such that $\mathbf{x}' \in \mathbb{D}$, but $\mathbf{x}' \notin \mathbb{C}$. Since $\mathbb{C}$ is closed, there exists an open ball $\mathbb{E}_{\mathbf{x}',\delta'}$ such that $\mathbb{E}_{\mathbf{x}',\delta'} \cap \mathbb{C} = \emptyset$. $\|\mathbb{D} \cap \mathbb{E}_{\mathbf{x}',\delta'}\| > 0$ implies that $\|\mathbb{D}\| \geq \|\mathbb{C}\| + \|\mathbb{D} \cap \mathbb{E}_{\mathbf{x}',\delta'}\| > \|\mathbb{C}\|$, which contradicts the assumption that $\|\mathbb{C}\| = \|\mathbb{D}\|$. Therefore, $\mathbb{C} = \mathbb{D}$. $\square$

2.6.2 Proof of Lemma 2.3.5

$$\begin{aligned} \Delta^{\langle n \rangle}(u) &= \|\mathbb{A}\|^{-1} \cdot \int_{\mathbf{x} \in \mathbb{H}(u)} d\mathbf{x} \\ &= \int_0^{a_{n-1}} \int_0^{a_{n-2}} \cdots \int_0^{a_1} \left(1 - \frac{u}{\prod_{j=1}^{n-1} (1 - x_j)}\right) dx_1 \cdots dx_{n-1} \end{aligned} \quad (2.14)$$

where

$$a_i \triangleq 1 - \frac{u}{\prod_{j=i+1}^{n-1} (1 - x_j)} \text{ for all } i = 1, 2, \dots, n-1,$$

is the normalized volume of region $\mathbb{H}(u)$ when the number of cells is n . One can verify the claim for $n = 2$, namely

$$\begin{aligned}\Delta^{\langle 2 \rangle}(u) &= \int_0^{1-u} \left(1 - \frac{u}{1-x_1}\right) dx_1 \\ &= 1 - u \left(1 + \ln\left(\frac{1}{u}\right)\right).\end{aligned}$$

Now, let the hypothesis be true for some $n - 1$, $n \geq 3$. Then from (2.14),

$$\begin{aligned}\Delta^{\langle n \rangle}(u) &= \int_0^{1-u} \Delta^{\langle n-1 \rangle} \left(\frac{u}{1-x_{n-1}}\right) dx_{n-1} \\ &= \int_0^{1-u} \left(1 - \left(\frac{u}{1-x}\right) \sum_{i=0}^{n-2} \frac{1}{i!} \left[-\ln\left(\frac{u}{1-x}\right)\right]^i\right) dx \quad (2.15)\end{aligned}$$

$$= (1-u) - \sum_{i=0}^{n-1} \frac{(-1)^i}{i!} \Psi_i(u) \quad (2.16)$$

where the equality labeled (2.15) is true from the induction hypothesis and

$$\Psi_i(u) \triangleq \int_0^{1-u} \frac{u}{1-x} \left[\ln\left(\frac{u}{1-x}\right)\right]^i dx.$$

If $1 - x = t$, then $dx = -dt$. Thus,

$$\Psi_i(u) = \int_u^1 \frac{u}{t} \left[\ln\left(\frac{u}{t}\right)\right]^i dt.$$

Now, note that

$$\frac{d}{dt} \left[\ln\left(\frac{u}{t}\right)\right]^{i+1} = -(i+1) \frac{1}{t} \left[\ln\left(\frac{u}{t}\right)\right]^i.$$

Therefore,

$$\int_u^1 \frac{u}{t} \left[\ln\left(\frac{u}{t}\right)\right]^i dt = -\frac{u}{i+1} \left[\ln\left(\frac{u}{t}\right)\right]^{i+1} \Big|_{t=u}^1,$$

which implies

$$\Psi_i(u) = -\frac{u}{i+1} (\ln u)^{i+1}. \quad (2.17)$$

Then, replacing (2.17) in (2.16),

$$\begin{aligned} \Delta^{(n)}(u) &= (1-u) - \sum_{i=0}^{n-2} \frac{(-1)^i}{i!} \left(-\frac{u}{i+1} (\ln u)^{i+1} \right) \\ &= 1 - u \sum_{i=0}^{n-1} \frac{1}{i!} \left[\ln \left(\frac{1}{u} \right) \right]^i, \end{aligned}$$

which is the required result for n . Hence, the lemma is true for all $n \geq 2$ by induction. $\square$

2.6.3 Proof of Proposition 2.3.6

First, let us show that $f_k(u) = \Delta(u) \cdot u^{k-1} \geq 0$ for all $u \in [0, 1]$. Since $f_k(0) = 0$ and $u \geq 0$ in the interval, it is sufficient to show

$$\frac{1-u}{u} + \ln u \geq 0 \quad \forall u \in (0, 1].$$

Let $y = 1 - u$. Then it is sufficient to show

$$\frac{y}{1-y} + \ln(1-y) \geq 0 \quad \forall y \in [0, 1).$$

Replacing each term on the left by its Taylor series, we get

$$\begin{aligned} &\sum_{n=1}^{\infty} y^n - \sum_{n=1}^{\infty} \frac{y^n}{n} \\ &= \sum_{n=1}^{\infty} \left(\frac{n-1}{n} \right) y^n \geq 0, \end{aligned} \quad (2.18)$$

where the summation in (2.18) is non-negative because each term in the summation above is non-negative. Thus $f_k(u) \geq 0$ for all $u \in [0, 1]$.

Now, to prove the proposition, we determine the local extrema for f_k . We set

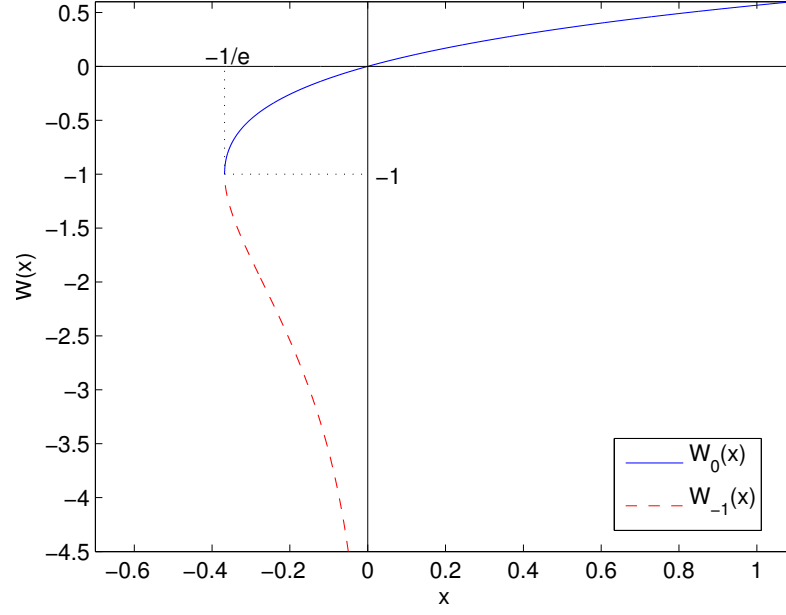


Figure 2.7: Real branches of Lambert W function.

the derivative of $f_k(u)$ with respect to u to 0 and solve for u to get,

$$\frac{d}{du} [f_k(u)] = (t-1) u^{k-2} \left(1 - u + \frac{k}{k-1} u \ln u \right) = 0$$

which implies

$$u = 0 \quad \text{or} \quad \left(1 - u - \frac{1}{\tau_k} u \ln u \right) = 0 \quad (2.19)$$

where $\tau_k = -\frac{k-1}{k}$ as defined in the statement of the proposition. For $u = 0$, $f_k(0) = 0$; thus $u_k^* \neq 0$. The other expression in (2.19) is equivalent to

$$e^{\frac{\tau_k}{u} \tau_k} = \tau_k e^{\tau_k}.$$

But, by the definition of the Lambert W function, $e^{W(\tau_k e^{\tau_k})} W(\tau_k e^{\tau_k}) = \tau_k e^{\tau_k}$; therefore,

$$u = \frac{\tau_k}{W(\tau_k e^{\tau_k})}$$

is a root of (2.19). Since $\tau_k e^{\tau_k} \in \left(-\frac{1}{e}, -\frac{1}{2\sqrt{e}}\right] \subset \left(-\frac{1}{e}, 0\right)$, $W(\tau_k e^{\tau_k})$ is multi-valued and takes two values $W_0(\tau_k e^{\tau_k})$ and $W_{-1}(\tau_k e^{\tau_k})$ corresponding to the two branches of the

Lambert W function, as shown in Figure 2.7. One can check that the root $\frac{\tau_k}{W_0(\tau_k e^{\tau_k})}$ equals 1 and is a local minimum. The other root, denoted as

$$u_k^* = \frac{\tau_k}{W_{-1}(\tau_k e^{\tau_k})},$$

corresponds to the local maximum. □

2.6.4 Proof of Proposition 2.3.7

Before we prove Proposition 2.3.7, first let us prove the following lemma.

Lemma 2.6.2. *For $t \geq 2$, u_t^* is the solution to the equation*

$$\Delta(u) = \frac{u}{(t-1)} \cdot \frac{[\ln(1/u)]^{n-1}}{(n-1)!}$$

Proof. Since $u_t^* = \arg \max_u u^{t-1} \cdot \Delta(u)$, it is the solution to

$$\frac{d}{du} [u^{t-1} \cdot \Delta(u)] = 0,$$

or

$$(t-1)u^{t-2} \left[1 - u \sum_{i=0}^{n-1} \frac{[\ln(1/u)]^i}{i!} - \frac{u}{(t-1)} \cdot \frac{[\ln(1/u)]^{n-1}}{(n-1)!} \right] = 0.$$

Note that $u \neq 0$ because

$$u^{t-1} \Delta(u) \big|_{u=0} = 0 = \min_u u^{t-1} \Delta(u).$$

Therefore u_t^* is the solution to

$$1 - u \sum_{i=0}^{n-1} \frac{[\ln(1/u)]^i}{i!} - \frac{u}{(t-1)} \cdot \frac{[\ln(1/u)]^{n-1}}{(n-1)!} = 0. \quad \square$$

We now prove Proposition 2.3.7.

Proof of Proposition 2.3.7. From Lemma 2.6.2, u_t^* is the solution to

$$1 - u \sum_{i=0}^{n-1} \frac{[\ln(1/u)]^i}{i!} - \frac{u}{(t-1)} \cdot \frac{[\ln(1/u)]^{n-1}}{(n-1)!} = 0.$$

Let $\ln \frac{1}{u} = z$, that is, $u = e^{-z}$. Therefore,

$$1 - e^{-z} \sum_{i=0}^{n-1} \frac{z^i}{i!} - \frac{e^{-z}}{t-1} \cdot \frac{z^{n-1}}{(n-1)!} = 0.$$

Using the Taylor series for e^z at $z = 0$,

$$1 - e^{-z} \left(e^z - \sum_{i=n}^{\infty} \frac{z^i}{i!} \right) - \frac{e^{-z}}{t-1} \cdot \frac{z^{n-1}}{(n-1)!} = 0,$$

or

$$\sum_{i=n}^{\infty} \frac{z^{i-(n-1)}}{i!} = \frac{1}{t-1} \cdot \frac{1}{(n-1)!}$$

Define

$$g_{t,n}(z) = \sum_{i=n}^{\infty} \frac{z^{i-(n-1)}}{i!} - \frac{1}{t-1} \cdot \frac{1}{(n-1)!}. \quad (2.20)$$

Then,

$$\begin{aligned} g_{t,n} \left(\frac{n}{t} \right) &= \sum_{j=0}^{\infty} \frac{\left(\frac{n}{t} \right)^{j+1}}{(n+j)!} - \frac{1}{t-1} \cdot \frac{1}{(n-1)!} \\ &= \frac{t^{-1}}{(n-1)!} \sum_{j=0}^{\infty} \left(t^{-j} \cdot \frac{n^{j+1}}{\prod_{k=0}^j (n+k)} \right) - \frac{1}{t-1} \cdot \frac{1}{(n-1)!} \\ &< \frac{t^{-1}}{(n-1)!} \sum_{j=0}^{\infty} t^{-j} - \frac{1}{t-1} \cdot \frac{1}{(n-1)!} \\ &= \frac{t^{-1}}{(n-1)!} \cdot \frac{t}{t-1} - \frac{1}{t-1} \cdot \frac{1}{(n-1)!} \\ &= 0 \end{aligned}$$

and

$$\begin{aligned} g_{t,n} \left(\frac{n}{t-1} \right) &= \frac{1}{n!} \left(\frac{n}{t-1} \right) + \sum_{i=2}^{\infty} \frac{\left(\frac{n}{t-1} \right)^i}{(i+n-1)!} - \frac{1}{(t-1)(n-1)!} \\ &= \sum_{i=2}^{\infty} \frac{\left(\frac{n}{t-1} \right)^i}{(i+n-1)!} > 0. \end{aligned}$$

Also, $g_{t,n}(z)$ is a continuous function of z , increasing for $z > 0$. Therefore $g_{t,n}(z)$ must have all its roots in the interval $\left(\frac{n}{t}, \frac{n}{t-1} \right)$. Since $-\log(u_t^*)$ is a root of $g_{t,n}(z)$,

$$e^{-\frac{n}{t-1}} < u_t^* < e^{-\frac{n}{t}}.$$
 □

2.6.5 Proof of Proposition 2.3.9

An asymptotic expression is needed for the last term in (2.12) as it depends on n . From [21], the lower normalized incomplete Gamma function P can be expressed as

$$P(n, \alpha n) = \frac{1}{2} \operatorname{erfc} \left(-\zeta \sqrt{n/2} \right) - R(n, \zeta),$$

where $\zeta = -(2(\alpha - 1 - \log_e \alpha))^{\frac{1}{2}}$ for $\alpha < 1$, erfc is the complementary error function and function R has the following asymptotic expansion that is valid for all $\alpha \geq 0$:

$$R(n, \zeta) \sim \frac{e^{-\frac{1}{2}n\zeta^2}}{\sqrt{2\pi n}} \sum_{k=0}^{\infty} d_k(\zeta) n^{-k}.$$

where d_k is a constant that depends only on ζ and k . The following asymptotic expression for the complementary error function holds for $x \rightarrow \infty$:

$$\operatorname{erfc}(x) \sim \frac{e^{-x^2}}{x\sqrt{\pi}} \left[1 + \sum_{k=1}^{\infty} (-1)^k \frac{(2k-1)!!}{(2x^2)^k} \right],$$

where $(2k-1)!!$ is the double factorial, that is, the product of all odd numbers up to $(2k-1)$. Thus,

$$P(n, \alpha n) = \frac{e^{-\frac{\zeta^2 n}{2}}}{\sqrt{2\pi n}} \sum_{k=0}^{\infty} \left((-1)^k \frac{(2k-1)!!}{\zeta^{2k+1}} - d_k(\zeta) \right) n^{-k},$$

$$\triangleq \frac{e^{-\frac{\zeta^2 n}{2}}}{\sqrt{2\pi n}} \sum_{k=0}^{\infty} D_k(\zeta) n^{-k}.$$

Therefore,

$$\frac{1}{n} \log_2 [P(n, \alpha n)] \sim -\frac{\zeta^2}{2} \log_2(e) + \frac{\log_2(2\pi n)}{2n} + \frac{\log_2 \left(\sum_{k=0}^{\infty} D_k(\zeta) n^{-k} \right)}{n},$$

and from (2.12),

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log_2 S_2(e^{-\alpha n}) = \log_2(q-1)^2 - (2\alpha - 1) \log_2(e) + \log_2(\alpha). \quad (2.21)$$

The total codebook cardinality in (2.21) is maximized when $\alpha = 0.5$ and is equal to

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log_2 S_2(e^{-0.5n}) = \log_2(q-1)^2 - 1.$$

Similarly, it can be shown that for $\alpha > 1$, $\zeta = +(2(\alpha - 1 - \log_e \alpha))^{\frac{1}{2}}$ so that

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log_2 [P(n, \alpha n)] = 0.$$

This completes the proof of the proposition. $\square$

2.6.6 Proof of correctness of Algorithm 2.2

Consider stage i where the algorithm assigns messages to points in the codebook for write i . Let $\mathbf{x}_k$ denote the k^{th} pivot point in the codebook $\mathcal{L}_{i-1}$ and $\tilde{V}_k = |\mathcal{L}_i(\mathbf{x}_k)|$ denote the number of points in codebook $\mathcal{L}_i$ accessible from $\mathbf{x}_k$. In this proof, we will show that at any iteration k of the i^{th} stage, the number of points in $\mathcal{L}_i(\mathbf{x}_k)$ that have not been assigned a message at any iteration $\ell < k$ is larger than the number of messages that point $\mathbf{x}_k$ cannot encode at the start of the k^{th} iteration. Then, the call to the procedure **AssignPhi** from **DefinePhi** has valid arguments; Thus, the algorithm is able to assign messages such that the k^{th} pivot point, $\mathbf{x}_k$, satisfies the condition in (2.13) at the end of the k^{th} iteration, assuming the previous pivot point satisfied the condition in

(2.13) after the previous iteration.

We need to keep track of the following variables for the purpose of this proof. Let S_k^- be the number of points in $\mathcal{L}_i(\mathbf{x}_k)$ that were not assigned any messages up to any iteration $\ell < k$. Similarly, let S_k^+ be the number of points in $\mathcal{L}_i(\mathbf{x}_k)$ that were not assigned any messages up to any iteration $\ell \leq k$. The set $\mathcal{L}_{\text{lost}}(\mathbf{x}_{k+1})$ denotes the points in $\mathcal{L}_i$ that were accessible from $\mathbf{x}_k$ but are not accessible from $\mathbf{x}_{k+1}$. Partition $\mathcal{L}_{\text{lost}}(\mathbf{x}_{k+1})$ into sets of points that were assigned (or were not assigned) messages up to some iteration $\ell \leq k$. Denote the cardinality of these sets as $A_{k,k+1}$ and $S_{k,k+1}$, respectively. That is, $A_{k,k+1}$ is the number of points in $\mathcal{L}_{\text{lost}}(\mathbf{x}_{k+1})$ that were assigned messages up to some iteration $\ell \leq k$. and $S_{k,k+1}$ is the number of points in $\mathcal{L}_{\text{lost}}(\mathbf{x}_{k+1})$ that were not assigned messages up to any iteration $\ell \leq k$. Finally, let $G_{k,k+1}$ be the number of points in $\mathcal{L}_i$ accessible from $\mathbf{x}_{k+1}$ but were not accessible from $\mathbf{x}_k$. By the definitions above, for all ℓ ,

$$\tilde{V}_\ell = \tilde{V}_{\ell-1} - (A_{\ell-1,\ell} + S_{\ell-1,\ell}) + G_{\ell-1,\ell}, \quad (2.22)$$

$$S_\ell^- = S_{\ell-1}^+ - S_{\ell-1,\ell} + G_{\ell-1,\ell}, \quad (2.23)$$

$$S_\ell^+ = S_\ell^- - A_{\ell-1,\ell}, \quad (2.24)$$

$$S_\ell^- = |\mathcal{L}_{\text{spare}}(\mathbf{x}_\ell)| \geq S_{\ell,\ell+1}, \quad (2.25)$$

and

$$A_{\ell-1,\ell} = |\mathcal{M}_{\text{lost}}(\mathbf{x}_\ell)|. \quad (2.26)$$

Here, (2.22) follows from the property that the number of points accessible from the ℓ^{th} pivot point, $\mathbf{x}_\ell$, are equal to the sum of the number of points accessible from $\mathbf{x}_{\ell-1}$ and the number of points accessible from $\mathbf{x}_\ell$ but not $\mathbf{x}_{\ell-1}$, minus the number of points accessible from $\mathbf{x}_{\ell-1}$ but not $\mathbf{x}_\ell$. Similarly, (2.23) follows by partitioning and counting the elements of the set of spare points accessible from $\mathbf{x}_\ell$ into the set of spare points accessible from $\mathbf{x}_\ell$ and $\mathbf{x}_{\ell-1}$ and the set of all points accessible from $\mathbf{x}_\ell$ but not $\mathbf{x}_{\ell-1}$. (2.24) holds because the set of spare points accessible from $\mathbf{x}_\ell$ after the ℓ^{th} iteration consist of all spare points that were accessible before the ℓ^{th} iteration except the ones that are assigned a message at that iteration. (2.25) and (2.26) follow from the definitions of $\mathcal{L}_{\text{spare}}$ and $\mathcal{M}_{\text{lost}}$, from Section 2.4.1, respectively.

Now, from (2.23) and (2.24),

$$S_{\ell+1}^- = S_\ell^- - A_{\ell-1,\ell} - S_{\ell,\ell+1} + G_{\ell,\ell+1},$$

which implies

$$S_{k+1}^- - S_0^- = \sum_{\ell=0}^k (-A_{\ell-1,\ell} - S_{\ell,\ell+1} + G_{\ell,\ell+1}). \quad (2.27)$$

However, from (2.22),

$$\tilde{V}_{k+1} - \tilde{V}_0 = \sum_{\ell=0}^k (-A_{\ell,\ell+1} - S_{\ell,\ell+1} + G_{\ell,\ell+1}). \quad (2.28)$$

So, combining (2.27) and (2.28),

$$S_{k+1}^- - S_0^- = (\tilde{V}_{k+1} - \tilde{V}_0) + A_{k,k+1} - A_{-1,0}.$$

Since $S_0^- = A_{-1,0} = \tilde{V}_0 = M_i$ and $\tilde{V}_{k+1} \geq M_i$ by definition of the first pivot point, we have

$$S_{k+1}^- \geq A_{k,k+1},$$

So,

$$|\mathcal{L}_{\text{spare}}(\mathbf{x}_{k+1})| \geq |\mathcal{M}_{\text{lost}}(\mathbf{x}_{k+1})|.$$

and $S_{k+1}^+ \geq 0$. Thus, at every iteration, the algorithm has enough spare points that have not yet been assigned a message to assign all the missing messages. $\square$

Acknowledgements

This chapter contains material from the papers: “Multilevel 2-Cell t-Write Codes,” Aman Bhatia, Aravind R. Iyengar, Paul H. Siegel; published in Proc. IEEE

Information Theory Workshop, Lausanne, Switzerland, September 3-7, 2012, pp. 247–251, and “Lattice-based WOM Codes for Multilevel Flash Memories,” Aman Bhatia, Minghai Qin, Aravind R. Iyengar, Brian M. Kurkoski, Paul H. Siegel; published in the IEEE Journal on Selected Areas in Communications, vol. 32, no. 5, pp. 933–945, May 2014. The dissertation author was the primary investigator and author of these papers, and co-authors have approved the use of the material for this dissertation.

Bibliography

- [1] N. Shibata, H. Maejima, K. Isobe, K. Iwasa, M. Nakagawa, M. Fujiu, T. Shimizu, M. Honma, S. Hoshi, T. Kawaai, K. Kanebako, S. Yoshikawa, H. Tabata, A. Inoue, T. Takahashi, T. Shano, Y. Komatsu, K. Nagaba, M. Kosakai, N. Motohashi, K. Kanazawa, K. Imamiya, H. Nakai, "A 70 nm 16 Gb 16-level-cell NAND flash memory," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 929–937, Apr. 2008.
- [2] R. L. Rivest and A. Shamir, "How to reuse a write-once memory," *Information and Control*, vol. 55, Dec. 1982.
- [3] A. Fiat and A. Shamir, "Generalized write-once memories," *IEEE Transactions on Information Theory*, vol. IT-30, pp. 470–480, May, 1984.
- [4] J. K. Wolf, A. D. Wyner, J. Ziv, and J. Körner, "Coding for a write-once memory," *AT&T Bell Labs. Technical Journal*, vol. 63, no. 6, pp. 1089–1112, 1984.
- [5] A. Jiang, V. Bohossian, and J. Bruck, "Rewriting Codes for joint information storage in flash memories," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5300–5313, Oct. 2010.
- [6] S. Kayser, E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "Multiple-write WOM-codes," *Proc. 48th Ann. Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sep. 29 - Oct. 1, 2010, pp. 1062–1068.
- [7] Y. Wu and A. Jiang, "Position modulation code for rewriting write-once memories," *IEEE Transactions on Information Theory*, vol. 57, no. 6, pp. 3692–3697, Jun. 2011.
- [8] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Efficient two-write WOM-codes," *Proc. IEEE Information Theory Workshop*, Dublin, Ireland, Aug. 20 - Sep. 3, 2010, pp. 1–5.
- [9] Q. Huang, S. Lin, and K. A. S. Abdel-Ghaffar, "Error-correcting codes for flash coding," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6097–6108, Sep. 2011.
- [10] R. Gabrys, E. Yaakobi, L. Dolecek, P. H. Siegel, A. Vardy, and J. K. Wolf, "Non-binary WOM-codes for multilevel flash memories," *Proc. IEEE Information Theory Workshop*, Paraty, Brazil, Oct. 16 - 20, 2011, pp. 40–44.
- [11] A. Shpilka, "Capacity achieving two-write WOM codes," *Latin American Symposium on Theoretical Informatics*, Peru, Apr. 2012, pp. 631–642.
- [12] Y. Cassuto and E. Yaakobi, "Short q-ary WOM codes with hot/cold write differentiation," *Proc. IEEE International Symposium on Information Theory*, Cambridge, MA, Jul. 1 - 6, 2012, pp. 1391–1395.

- [13] B. M. Kurkoski, "Notes on a lattice-based WOM construction that guarantees two writes," *Proc. 34th Symp. Information Theory and Applications*, Ousyuku, Iwate, Japan, Nov. 29 - Dec. 2, 2011, pp. 520–524.
- [14] F. Fu and A. J. Han Vinck, "On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 308–313, Jan. 1999.
- [15] A. Bhatia, A. R. Iyengar, and P. H. Siegel, "Multilevel 2-cell t -write codes," *Proc. IEEE Information Theory Workshop*, Lausanne, Switzerland, Sep. 3 - 7, 2012, pp. 247–251.
- [16] B. Kurkoski, "Lattice-based WOM codebooks that allow two writes," *International Symposium on Information Theory and its Applications*, Honolulu, Hawaii, Oct. 28 - 31, 2012, pp. 101–105.
- [17] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, New York, NY, Springer-Verlag, 3rd ed., 1999.
- [18] G. D. Forney, R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. H. Qureshi, "Efficient modulation for band-limited channels," *IEEE Journal on Selected Areas in Communications*, JSAC-2, pp. 632–647, Sep. 1984.
- [19] G. D. Forney and L.-F. Wei, "Multidimensional constellations – Part 1: Introduction, figures of merit, and generalized cross constellations," *IEEE Journal on Selected Areas in Communications*, JSAC-7, pp. 877–892, Aug. 1989.
- [20] R. L. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. Knuth, "On the Lambert W function," *Advances in Computational Mathematics*, vol. 5, no. 4, pp. 329–359, 1996.
- [21] N. M. Temme, "Uniform asymptotic expansions of the incomplete Gamma functions and the incomplete Beta function," *Mathematics of Computation*, vol. 29, no. 132, pp. 1109–1114, Oct. 1975.
- [22] J. J. Ashley, R. Karabed, and P. H. Siegel, "Complexity and sliding-block decodability," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1925–1947, Nov. 1996.
- [23] M. Garey and D. Johnson, *Computers and Intractability: A guide to the theory of NP-Completeness*, San Fransisco, CA: W. H. Freeman, 1979.
- [24] Y. Wu, "Low complexity codes for writing a write-once memory twice," *Proc. IEEE International Symposium on Information Theory*, Austin, TX, June 13 - 18, 2010, pp. 1928–1932.

- [25] M. Qin, E. Yaakobi, and P. H. Siegel, “Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 836–846, May 2014.
- [26] A. Bhatia, M. Qin, A. R. Iyengar, B. M. Kurkoski, and P. H. Siegel, “Lattice-based WOM Codes for Multilevel Flash Memories,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 933–945, May 2014.
- [27] M. Qin, University of California San Diego, La Jolla, USA, Personal Communication, 2012.

Chapter 3

Precoding Mapping Optimization for Magnetic Recording Channels

We study the problem of designing a rate-1 block-precoder to minimize bit/symbol error rate when storing a given source on a magnetic recording channel. A block-precoder of length b -bits is defined by a permutation π on 2^b blocks. We show that the problem of finding a permutation for the block-precoder that minimizes bit/symbol error rate is equivalent to solving a known combinatorial optimization problem that is NP-complete in general. We exploit the symmetry group of the b -dimensional hypercube to reduce the search space, allowing a branch-and-bound technique to find the optimal 5-bit precoders. We also implement a local search algorithm that can find good precoders for larger block-lengths. We design precoders for MTR-constrained user bits and unconstrained parity bits with a reverse-concatenation architecture, and we evaluate the resulting SNR gains in a turbo equalization scheme.

3.1 Introduction

In many magnetic recording applications, low-density parity check (LDPC) codes are used as an outer code in a turbo equalization scheme, where the partial response channel is treated as the inner code. In high linear density regimes, the performance of iterative detection-decoding schemes has been shown to be improved by the use of precoders [1] and modulation codes [2]. In this work, we study the problem of designing

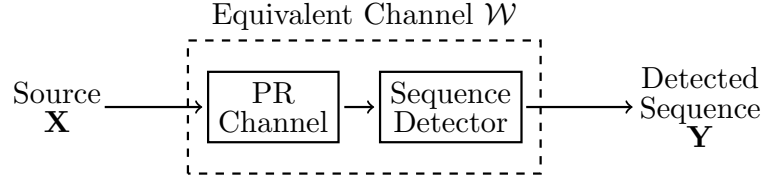


Figure 3.1: Equivalent channel from source $\mathbf{X}$ to detected sequence $\mathbf{Y}$.

rate-1 block-precoders that minimize the error-rate as seen by the outer code. We show that designing optimal precoders is equivalent to solving a combinatorial optimization problem. For small block lengths, we derive good precoders that give noticeable SNR gains over the $1/(1 \oplus D)$ precoder studied in [1] with negligible increase in the complexity. Our setup allows us to design precoders that improve raw bit/symbol error rates for both parity bits as well as the MTR-constrained bits [3].

This chapter of the dissertation is organized as follows. In Section 3.2, we formulate the system model that is used to define a block-precoder and quantify its performance. In Section 3.3 we describe the various approaches we take to find good precoders. In Section 3.4 we show how the proposed precoders may be incorporated in the encoding and decoding architectures used for magnetic recording applications. Section 3.5 shows the performance of the proposed precoders over magnetic recording channels through simulation results.

3.2 System Model

Let $\mathbf{X} = \{X_k\}_{k \in \mathbb{N}}$ denote the outputs of a source that has alphabet $\mathbb{F}_2^b$, for some $b \in \mathbb{N}$ such that X_k are independent and identically distributed (i.i.d.) according to the probability distribution function $\mathbf{p}_X$ for all k . This source is transmitted on a magnetic recording channel, modeled as a partial response (PR) channel with data-dependent colored noise, and the received sequence is passed to the bit-wise MAP sequence detector with noise prediction. Let the output from the sequence detector be denoted by $\mathbf{Y} = \{Y_k\}_{k \in \mathbb{N}}$, with $Y_k \in \mathbb{F}_2^b$. We denote the channel from $\mathbf{X}$ to $\mathbf{Y}$ as the *equivalent channel* $\mathcal{W}$ as shown in Fig. 3.1.

Let $\mathbf{P} = [p_{i,j}]$ denote the *equivalent channel transition probability matrix*, where

$$p_{i,j} = \mathbf{Pr}(Y_k = j \mid X_k = i) \cdot \mathbf{p}_X(i),$$

for $1 \leq i, j \leq 2^b$. Although $\mathbf{P}$ is often not available in closed form, it can be calculated numerically using Monte Carlo techniques. Let $\mathbf{H} = [h_{i,j}]$ denote the *Hamming distance matrix*, where $h_{i,j}$ denotes the Hamming distance between binary sequences i and j , for $i, j \in \mathbb{F}_2^b$. Then the bit error rate, $d(\mathbf{X}, \mathbf{Y})$, can be expressed as

$$d(\mathbf{X}, \mathbf{Y}) \triangleq \frac{1}{b} \langle \mathbf{H}, \mathbf{P} \rangle = \frac{1}{b} \sum_{i,j} h_{i,j} p_{i,j}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius product of two matrices.

The block-precoder is defined by a permutation $\pi : \mathbb{F}_2^b \mapsto \mathbb{F}_2^b$. Fig. 3.2 shows the system model with the precoder. The data $\mathbf{X}$ is grouped into b -bit blocks and each block is precoded using a rate-1 encoder given by π . At the decoder, the output of the equivalent channel, denoted by $\mathbf{Z}$, is then inverse precoded to get $\mathbf{Y} = \{Y_k\}$, where each $Y_k = \pi^{-1}(Z_k)$. Note that the precoder is different from an interleaver; the former replaces b -bit blocks with other blocks, while the latter shuffles the bit sequence. The bit error rate of the scheme with the precoder, $d_\pi(\mathbf{X}, \mathbf{Y})$, is given by

$$\begin{aligned} d_\pi(\mathbf{X}, \mathbf{Y}) &\triangleq \frac{1}{b} \sum_{i,j} h_{i,j} p_{\pi(i), \pi(j)} \\ &= \frac{1}{b} \langle \mathbf{H}, \mathbf{\Pi P \Pi}^\top \rangle \end{aligned} \tag{3.1}$$

where $\mathbf{\Pi}$ will be used to denote the permutation matrix associated with the permutation π .



Figure 3.2: Channel model with block-precoder π .

3.3 Precoder Design

The objective of the precoder design is to find a precoder that minimizes the bit error rate. The optimal precoder, denoted by π^* , is given by

$$\pi^* \triangleq \arg \min_{\pi \in \mathcal{S}_{2^b}} d_\pi(\mathbf{X}, \mathbf{Y}) = \arg \min_{\pi \in \mathcal{S}_{2^b}} \sum_{i,j} h_{i,j} p_{\pi(i), \pi(j)}, \quad (3.2)$$

where $\mathcal{S}_{2^b}$ is the symmetric group of permutations on 2^b elements. Searching for the optimal precoder by enumeration of all possible $(2^b)!$ permutations is infeasible, even for small values of b . In fact, the optimization problem in (3.2) is a known combinatorial optimization problem, called the Koopmans-Beckmann *Quadratic Assignment Problem* (QAP) [4, 5, 6]. QAP is in general an NP-complete problem. Moreover, any method that gives an ϵ -approximate solution for QAP is NP-complete [7]. This problem has been studied in the context of communication theory earlier in the study of index assignment for vector-quantization systems that operate on noisy channels [8, 9, 10, 11].

Methods for solving the QAP problem can be divided into two classes: exact algorithms and suboptimal algorithms. We discuss these in the following subsections.

3.3.1 Exact Algorithms

Naive enumeration of all $(2^b)!$ permutations gives us the exact solution, but it is only feasible to perform when $b \leq 3$. For $b = 4$ the search space consists of over 20 trillion permutations as candidates for the solution. This search space can be reduced by noting that for our problem the matrix $\mathbf{H}$ has a special structure since it consists of the Hamming distance between all b -bit blocks. The symmetry group of a b -dimensional hypercube defines a set of permutations $\mathcal{I} \subset \mathcal{S}_{2^b}$ with cardinality $|\mathcal{I}| = b! \cdot 2^b$ such that for all $\pi \in \mathcal{I}$, $\mathbf{\Pi} \cdot \mathbf{H} \cdot \mathbf{\Pi}^T = [h_{\pi(i), \pi(j)}] = \mathbf{H}$. Then, for any permutation $\pi'' \in \mathcal{S}_{2^b}$ there are $|\mathcal{I}|$ distinct permutations $\pi' \in \mathcal{I}$ such that $d_{\pi'' \circ \pi'^{-1}}(\mathbf{X}, \mathbf{Y}) = d_{\pi''}(\mathbf{X}, \mathbf{Y})$. This reduces the search space for the optimization problem in (3.2) from $(2^b)!$ to $\frac{(2^b)!}{b! \cdot 2^b}$.

After reducing the search space, we used a branch-and-bound technique where a given QAP was divided into multiple QAPs with fewer degrees of freedom by partial assignment, and the Gilmore-Lawler lower bounds on the achievable cost were obtained

for these reduced QAPs by solving a relaxation to a linear assignment problem [12, 13]. Details of the techniques used are available in [5].

3.3.2 Heuristic Algorithms

The branch-and-bound algorithm described above lets us find optimal block-precoders for up to $b = 5$ bits. QAP problems of higher dimensions are not practical to solve optimally because of the higher running time requirements. This necessitates the use of heuristic algorithms that provide good solutions in a reasonable time, although they do not provide any guarantees of optimality. Many types of heuristic algorithms have been proposed for solving QAPs [5]. We used the tabu search algorithm, first introduced in [14, 15]. Tabu search is a local search algorithm that starts with an initial permutation and iteratively tries to improve it by finding a better permutation in its neighborhood. The algorithm maintains a *tabu list* where it lists forbidden transpositions on the current solution. Transpositions are moved out of the list after some iterations. Tabu search was used to find good precoders for up to $b = 6$ bits. For $b \leq 5$, these precoders were confirmed to be optimal using exact algorithms.

3.4 Block-precoders in Magnetic Recording

We now relate the system model with the block-precoder introduced in Section 3.2 to the system model used in current magnetic recording applications. We also generalize the system model in Section 3.2 to allow the precoder to minimize the symbol error rate when a non-binary LDPC code is used for error-correction.

3.4.1 Precoding in reverse concatenation architecture

We consider the reverse concatenation (RC) architecture [16] of magnetic recording channels, where the user bits are encoded using an MTR-constrained code [3] and parity bits are calculated on the MTR-constrained bits using a systematic error correction code. We design separate block-precoders for the MTR-constrained bit sequence and unconstrained parity bits that may be used independently. Fig. 3.3 shows the block diagram of the RC architecture with the block-precoders. For error-correction, we con-

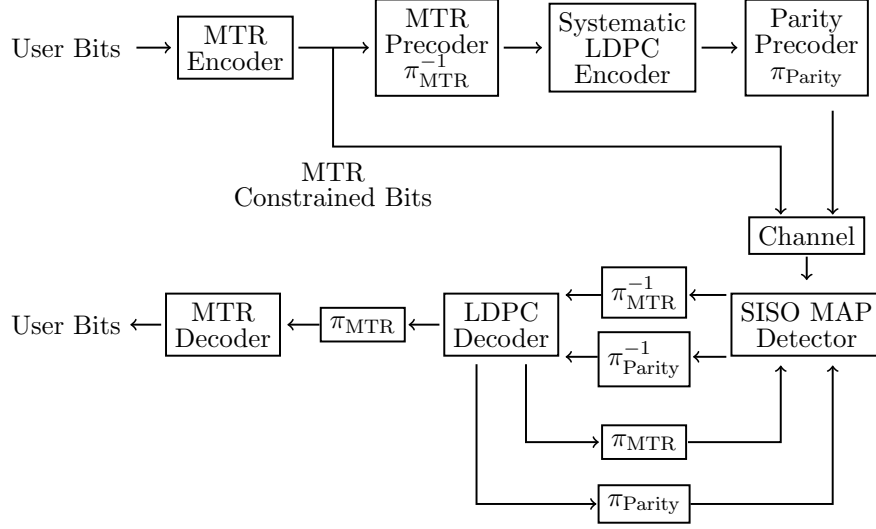


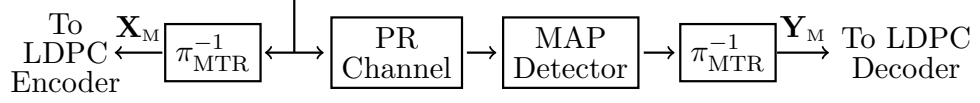
Figure 3.3: Reverse concatenation architecture with separate block-precoders, π_{MTR} and π_{Parity} , for MTR-constrained user bits and unconstrained parity bits, respectively.

sider LDPC codes decoded using sum-product algorithm decoding. For the detector, a soft-input soft-output MAP detector like SOVA [17] is used with a trellis that represents the possible MTR-constrained sequences at the output of a PR channel. The SOVA is modified to appropriately encode the survivor and loser paths using the chosen block-precoder. Figs. 3.4a and 3.4b show the channel models observed by the MTR-constrained bits and the parity bits between the LDPC encoder and the LDPC decoder in the RC architecture. These diagrams can be compared to the generic channel model presented earlier in Fig. 3.2. For all simulations, we used $h(D) = 6 + 11D + 5D^2$ as the PR target and assume 70% jitter noise.

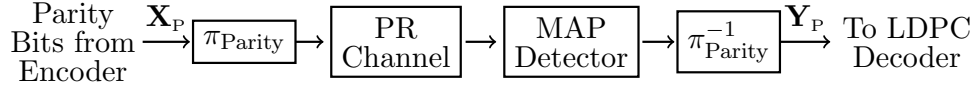
3.4.2 Minimizing symbol error rate

An extension of the system model presented in Section 3.2 is one where sequences $\mathbf{X}, \mathbf{Y}$ have alphabet $(\mathbb{F}_q)^m$ where q is some power of a prime. Such a system is of importance for the use of non-binary LDPC codes in magnetic recording applications [18] where minimizing the raw symbol error rate is of concern. For such a model the elements of matrix $\mathbf{H} = [h_{i,j}]$ are defined as the Hamming distance between non-binary sequences i and j , for $i, j \in (\mathbb{F}_q)^m$. Using this $\mathbf{H}$ matrix, the cost function $d_\pi(\mathbf{X}, \mathbf{Y})$, defined according to (3.1), is proportional to the symbol error rate. The optimal precoder

MTR-constrained Bits
from MTR Encoder
 $\pi(\mathbf{X}_M)$



(a) Encoding and decoding the MTR-constrained bits using precoder π_{MTR} .



(b) Encoding and decoding the unconstrained parity bits using precoder π_{Parity} .

Figure 3.4: Schematic diagrams showing the effect of the precoders in the RC architecture on (a) the MTR-constrained user bits, (b) the unconstrained parity bits.

that minimizes the symbol error rate is then given by the optimization problem in (3.2) and was found to be different from the block-precoder optimized for minimizing the bit error rate.

3.5 Simulation Results

We now present some experimental performance results obtained for the precoders we designed in the previous sections on magnetic recording channels and compare them to those obtained for the $1/(1 \oplus D)$ precoder. We first consider the raw error rates between the transmitted codeword and the word received by the LDPC decoder and show the improvement obtained using the proposed precoders. We then discuss the improvement in the SNR gains obtained by these precoders as a function of the channel bit density. Finally we present the simulation results obtained by the precoders when LDPC coded sequences are transmitted and decoded in the RC architecture along with turbo equalization between the MAP detector and the LDPC code decoder.

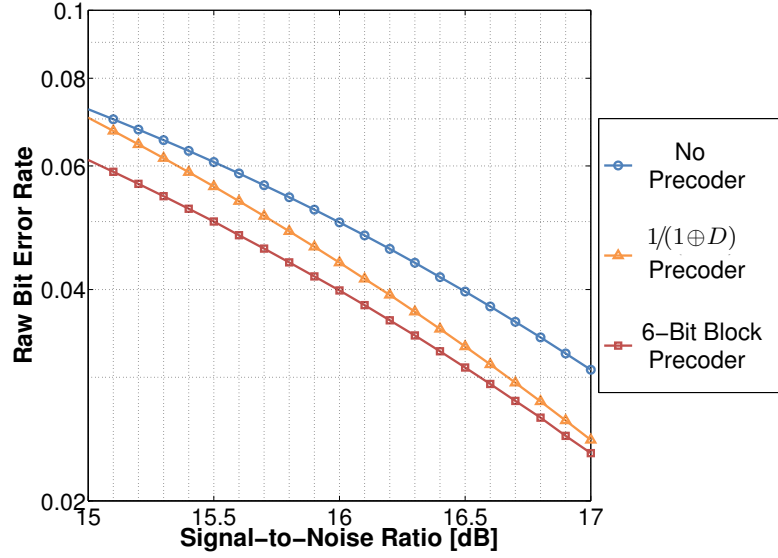


Figure 3.5: Raw Bit Error Rate at the output of the detector with various precoders for an i.u.d. source $\mathbf{X}$.

3.5.1 Precoders for parity bits

Consider a bit sequence $\mathbf{X}_P$ generated using an independent uniformly distributed (i.u.d.) source with alphabet $(\mathbb{F}_2)^b$. This models the distribution of the parity bits of a linear block code. This bit sequence is transmitted through the channel shown in Fig. 3.4b and the average bit error rate, $d(\mathbf{X}_P, \mathbf{Y}_P)$, is calculated by simulation. Fig. 3.5 shows the raw bit error rates for the 6-bit block-precoder and the $1/(1 \oplus D)$ precoder, where $+ - +$ is the dominant error event without any precoding. The proposed block-precoder has a gain of 0.5 dB over the scheme with no precoding and 0.2 dB compared to the $1/(1 \oplus D)$ precoder at low SNRs. Though the $1/(1 \oplus D)$ precoder reduces the effect of the $+ - +$ error, the proposed block-precoder can reduce the effect of other high weight error events as well.

As discussed in Section 3.4.2, a non-binary source $\mathbf{X}_P$ may be of interest. We consider block-precoders that map 3 symbols of $\mathbb{F}_4$ to other symbols. The precoder that minimizes the symbol error rate is found using the techniques described in Section 3.3. Fig. 3.6 shows the symbol error rates for the best 3-symbol block-precoder we found and for the bit-wise use of the $1/(1 \oplus D)$ precoder, where the source $\mathbf{X}_P$ randomly generates i.u.d. symbols from $\mathbb{F}_4$. As shown in the Fig. 3.6, the $1/(1 \oplus D)$ precoder increases the

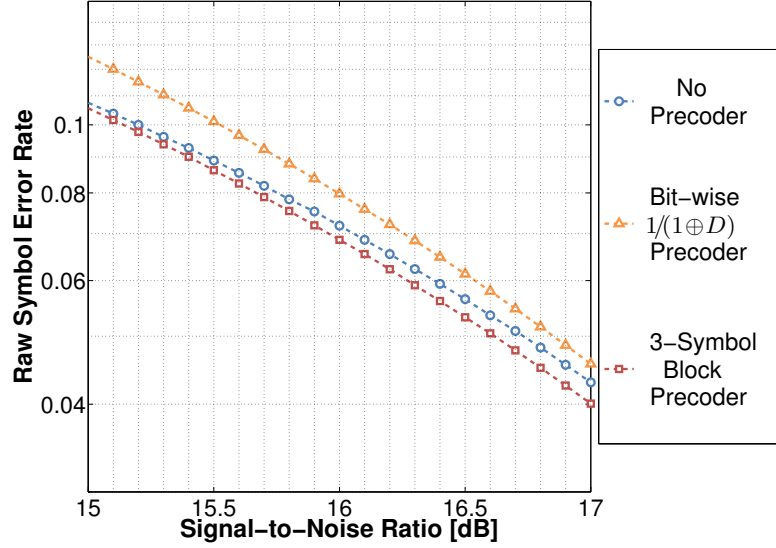


Figure 3.6: Raw Symbol Error Rate at the output of the detector with various precoders for an i.u.d. source $\mathbf{X}$ with alphabet $\mathbb{F}_4$.

symbol error rate compared to the scheme with no precoding.

3.5.2 Precoders for MTR-constrained bits

We now consider optimizing a different precoder for the MTR-constrained bits. MTR ($j = 3; k = \infty$) constrained bits are generated using a high-rate finite-state encoder [1]. These bit sequences represent $\pi(\mathbf{X}_M)$ in the channel model in Fig. 3.4a. We find good block-precoders that have low error rates. Figs. 3.7 and 3.8 show the bit error rate for the best 6-bit precoder and the symbol error rate for the best 3-symbol precoder, respectively. Note that the gain for the MTR-constrained bits is lower than the gains for the parity bits since there are no error events with high weight that are dominant in this case, unlike the $+-+$ error events for the latter. However, the MTR-constrained bits form a large fraction of the codeword bits for a high-rate code. Therefore even smaller gains for them using precoding are useful.

3.5.3 SNR gain versus channel bit density

Table 3.1 shows the SNR gains for the proposed block-precoders at various channel bit densities, for the channel models given in Figs. 3.4a and 3.4b for MTR-constrained

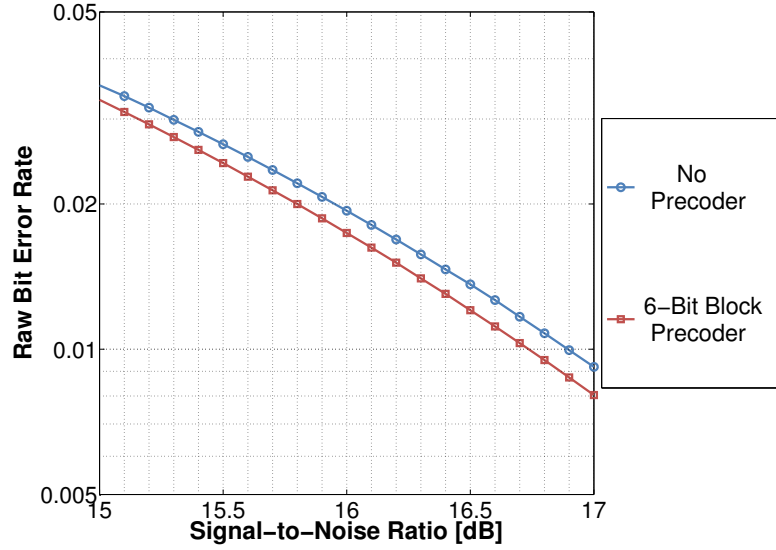


Figure 3.7: Raw Bit Error Rate at the output of the detector with a 6-bit precoder for a MTR-3 constrained sequence.

bits and parity bits, respectively. These SNR gains are relative to no-precoding scheme and are measured at raw bit/symbol error rates 0.01. It is clear that the SNR gains increase as the channel bit density increases.

Table 3.1: SNR gains for the proposed block-precoders for MTR-constrained bits and parity bits at various channel bit densities compared to no-precoding scheme.

Channel Bit Density	SNR Gains (in dB)			
	MTR Bits		Parity Bits	
	BER	SER	BER	SER
1.151	0	0	0.043	0
1.319	0.001	0	0.130	0
1.532	0.052	0.007	0.514	0.052

3.5.4 Precoders in RC architecture

We consider transmission of a coded sequence over the channel model described in Section 3.4.1. A rate 0.9 (3,31)-regular binary LDPC code of blocklength 4818 is used in the reverse concatenation architecture along with turbo equalization at channel bit density 1.532. We performed up to 5 global iterations, with 20 iterations of sum-

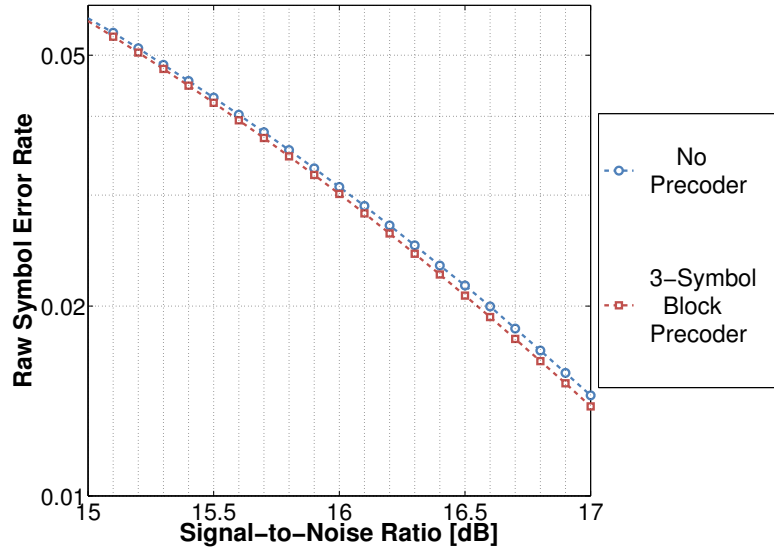


Figure 3.8: Raw Symbol Error Rate at the output of the detector with a 3-symbol precoder for MTR-3 constrained sequence where each symbol consists of 2 bits.

product algorithm decoding in each global iteration. We measured the bit error rate on the MTR-constrained bits with and without a precoder on the MTR-constrained bits and/or parity bits.

Fig. 3.9 shows the bit error rates for each of these cases, showing that the SNR gain is approximately 0.1 dB for each precoder. This confirms that the proposed precoders give gains similar to those observed in raw bit error rates, even when multiple rounds of the detector and decoder are performed in turbo equalization.

3.6 Concluding Remarks

In this chapter we studied the design of precoders for magnetic recording channels that minimize the raw bit/symbol error rates. We found optimal precoders by using a branch-and-bound method with a reduced search space. For the branch-and-bound algorithm, the Gilmore-Lawler lower bound was used which is easy to compute but deteriorates fast for increasing problem sizes. Tighter bounds based on semidefinite programming relaxations have been proposed in [19]. These bounds have been improved for QAPs with a distance matrix that has suitable symmetry properties [20, 21] which also hold for the problem in (3.2). Using these techniques may result in an overall faster

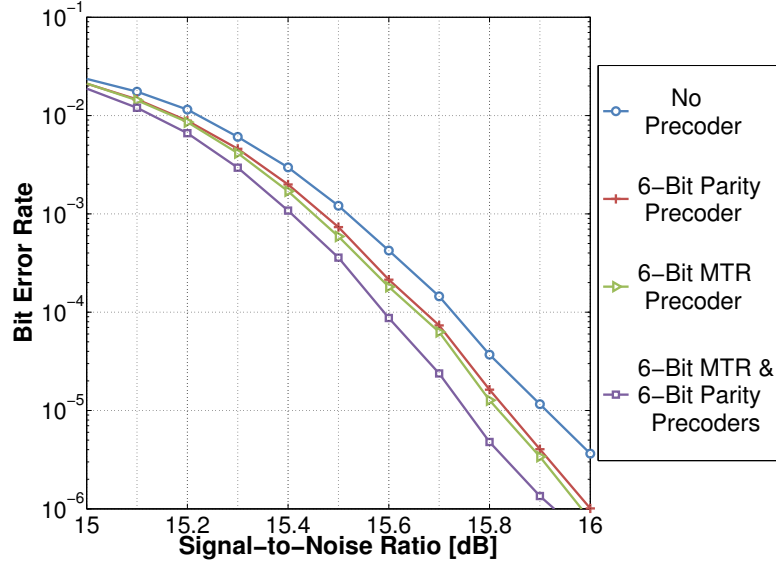


Figure 3.9: Bit Error Rate on MTR-constrained user bits with various precoders and a rate 0.9 binary LDPC code decoded using turbo equalization.

search for the optimal precoder. This should allow us to prove the optimality of the larger block length precoders found by the tabu search.

The precoder design discussed in this chapter can be generalized in various ways. Instead of using rate-1 block-precoders that map b input bits to b output bits, one may use $m : n$ precoders that have rate $\frac{m}{n} < 1$ when $m < n$. The rate loss may be compensated by ensuring that the precoder does not produce patterns that cause frequent errors as outputs by satisfying a weak constraint like MTR with $j \geq 4$. The use of guided-scrambling [22] has been proposed [23] to encode the unconstrained parity bit sequences such that the encoded sequences also satisfy the MTR-constraint with a high probability. This approach enables the trade-off between the rate-loss of using the precoder and the probability that the MTR-constraint is satisfied. Further study is needed to determine which of these approaches provide the highest SNR gain when the rate-loss is compensated for by using a higher channel bit density.

Another generalization is using block-precoders that have memory. These precoders may use a different permutation based on the previous encoded block. Such a scheme increases the complexity associated with the design process, since a joint search over multiple permutations is needed. However, such precoders may provide lower average error rates since, unlike the precoders proposed in this chapter, they do not ignore

the effects of the memory in the magnetic recording channel.

Acknowledgements

This chapter is in part a reprint of the paper “Precoding Mapping Optimization for Magnetic Recording Channels,” Aman Bhatia, Shaohua Yang, Paul H. Siegel; published in the IEEE Transactions on Magnetics, vol. 50, no. 11, pp. 1–4, Nov. 2014. The dissertation author was the primary investigator and author of this paper, and co-authors have approved the use of the material for this dissertation.

Bibliography

- [1] T. Mittelholzer and E. Eleftheriou, "Channel precoding and low-density parity check codes for magnetic recording," *Proc. IEEE International Symposium on Information Theory*, Yokohama, Japan, Jun. 29 - Jul. 4, 2003, pp. 47.
- [2] J. Kim, J. Lee, and J. Lee, "Performance of low-density parity check codes with parity encoded by (1,7) run-length limited code for perpendicular magnetic recording," *IEEE Transactions on Magnetics*, vol. 48, no. 11, pp. 4610–4613, Nov. 2012.
- [3] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Transactions on Magnetics*, vol. 32, no. 5, pp. 3992–3994, Sep. 1996.
- [4] T. C. Koopmans and M. J. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, no. 1, pp. 53–76, 1957.
- [5] R. E. Burkard, E. Cela, P. M. Pardalos, and L. Pitsoulis, "The quadratic assignment problem," *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, pp. 241–338, 1998.
- [6] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. M. Hahn, and T. Querido, "A survey for the quadratic assignment problem," Invited Review, *European Journal of Operational Research*, vol. 176, no. 2, pp. 657–690, Jan. 2006.
- [7] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the Association of Computing Machinery*, vol. 23, no. 3, pp. 555–565, Jul. 1976.
- [8] K. Zeger and A. Gersho, "Pseudo-Gray coding," *IEEE Transactions on Communications*, vol. 38, no. 12, pp. 2147–2158, Dec. 1990.
- [9] G. Ben-David and D. Malah, "Bounds on the performance of vector-quantizers under channel errors," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 2227–2235, Jun. 2005.
- [10] X. Wu, H. D. Mittelmann, X. Wang, J. Wang, "On computation of performance bounds on optimal index assignment," *IEEE Transactions on Communications*, vol. 59, no. 12, pp. 3229–3233, Dec. 2011.
- [11] P. Zhu, L. Tang, Y. Wang, X. You, and J. Hua, "Pseudo-Gray coding for beamforming systems," *Proc. IEEE International Conference on Communications*, Beijing, China, May 19 - 23, 2008, pp. 4689–4695.
- [12] P. C. Gilmore, "Optimal and suboptimal algorithms for the quadratic assignment problem," *SIAM Journal on Applied Mathematics*, vol. 10, no. 2, pp. 305–313, Jun. 1962.
- [13] E. Lawler, "The quadratic assignment problem," *Management Science*, vol. 9, no. 4, pp. 586–599, Jul. 1963.

- [14] F. Glover, "Tabu search - Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.
- [15] F. Glover, "Tabu search - Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.
- [16] W. G. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Technical Disclosure Bulletin*, vol. 23, no. 10, pp. 4633–4634, Mar. 1981.
- [17] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," *Proc. IEEE Globecom*, Dallas, TX, Nov. 27 - 30, 1989, vol. 3, pp. 1680–1686.
- [18] H. Song and J. R. Cruz, "Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording," *IEEE Transactions on Magnetics*, vol. 39, no. , pp. 1081–1087, Mar. 2003.
- [19] Q. Zhao, S. E. Karisch, F. Rendl, and H. Wolkowicz, "Semidefinite programming relaxations for the quadratic assignment problem, *Journal of Combinatorial Optimization*, vol. 2, no. 1, pp. 71-109, 1998.
- [20] E. de Klerk and R. Sotirov, "Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry," *Mathematical Programming*, vol. 133, no. 1-2, pp. 75–91, Jun. 2012.
- [21] H. Mittelmann and J. Peng, "Estimating bounds for quadratic assignment problems associated with Hamming and manhattan distance matrices based on semidefinite programming," *SIAM Journal on Optimization*, vol. 20, no. 6, pp. 3408–3426, Aug. 2010.
- [22] I. J. Fair, W. D. Grover, W. A Krzymien, and R. I. MacDonald, "Guided scrambling: a new line coding technique for high bit rate fiber optic transmission systems," *IEEE Transactions on Communications*, vol. 39, no. 2, pp. 289–297, Feb. 1991.
- [23] V. Krachkovsky, LSI Corporation, USA, Personal Communication, 2012.

Chapter 4

Polar Codes for Magnetic Recording Channels

Polar codes provably achieve the capacity of binary memoryless symmetric (BMS) channels with low complexity encoding and decoding algorithms, and their finite-length performance on these channels, when combined with suitable decoding algorithms (such as list decoding) and code modifications (such as a concatenated CRC code), has been shown in simulation to be competitive with that of LDPC codes. However, magnetic recording channels are generally modeled as binary-input intersymbol interference (ISI) channels, and the design of polar coding schemes for these channels remains an important open problem. Current magnetic hard disk drives use LDPC codes incorporated into a turbo-equalization (TE) architecture that combines a soft-output channel detector with a soft-input, soft-output sum-product algorithm (SPA) decoder. An interleaved coding scheme with a multistage decoding (MSD) architecture with LDPC codes as component codes has been proposed as an alternative to TE for ISI channels. In this work, we investigate the use of polar codes as component codes in the TE and MSD architectures. In the TE scheme, soft-output decoders based upon variations of belief-propagation decoding are considered, and modified polar code construction algorithms are used to improve performance. In the MSD scheme, the equivalent channels for each codeword may be assumed to be memoryless, so encoding/decoding methods developed for polar codes on memoryless channels may be used. It is shown that the achievable rate of the MSD scheme converges to the symmetric information rate of the ISI channel when the

number of interleaves is large. Simulations results comparing the performance of LDPC codes and polar codes in TE and MSD architectures are presented.

4.1 Introduction

Channel polarization was originally proposed in [1] for binary input discrete memoryless channels. This technique allows transforming multiple copies of an arbitrary symmetric binary-input discrete memoryless channel (B-DMC) to create ‘bit-channels’ that are either noiseless or completely noisy, such that the fraction of bit-channels that are essentially noiseless is close to the capacity of the original B-DMC. Identifying which bit-channels are noiseless after this polarization transformation allows the construction of *polar codes*, a family of codes that provably achieve the capacity of B-DMC with low-complexity encoding and decoding algorithms. Since the seminal paper by Arikan, polar codes have been generalized in a wide range of directions. General conditions for a transformation to polarize a binary-input memoryless channel have been obtained in [2]. Polar codes have found use in the context of MAC channels [3], wiretap channels [4] and source coding [5].

Since construction of polar codes has complexity exponential in blocklength for channels other than the binary erasure channel, a number of approximate construction methods that have linear complexity have been proposed [6], [7], [8]. While polar codes achieve the capacity with a successive cancellation (SC) decoder at infinite blocklengths, the performance of finite-length polar codes can be improved by the use of better decoding algorithms. Successive cancellation list decoder was proposed in [9] for hard-decision decoding and the performance was comparable to that of LDPC codes with belief propagation (BP) decoding. SC-list decoding operates like SC decoding, but whenever a decision on an unfrozen bit is needed, it splits the decoding path into two to try both 0 and 1. When the number of paths grows beyond the prescribed threshold L , the decoder keeps only the L most likely paths. BP decoding over the polar code factor graph was proposed in [10] as a soft-output decoder for polar codes. Soft cancellation (SCAN) decoder, proposed in [11], uses a message passing schedule motivated by the SC decoder and improves the soft-output decoding performance further.

Polar codes are closely related to Reed-Muller (RM) codes, a connection that was identified in [1]. The rows of the generator matrix for both codes are chosen from the rows of the matrix $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n}$ where $\otimes$ denotes the Kronecker product. For RM codes, the rows with the largest Hamming weight are chosen. However, for polar codes, rows corresponding to the noiseless channels obtained by the recursive channel-polarization transforms are chosen. Thus the choice depends on the channel and requires that the recursive transforms polarize the channel. Magnetic recording channels are modeled as a binary-input channel with finite memory due to intersymbol interference. The polarization phenomenon for stationary, ergodic, finite-order Markov processes through the polarization transform has been shown in [12]. However, it is only conjectured that the polarization result extends to intersymbol interference channels. Also, code construction for channels with memory remains an open problem as the code construction techniques in [6, 7, 8] only apply for memoryless channels. Thus, the construction of capacity-achieving polar codes for magnetic recording channels is a challenging problem.

Current magnetic recording technologies use LDPC codes as error-correction codes (ECC) incorporated into a turbo-equalization (TE) architecture that combines a soft-output channel detector with a soft-input soft-output (SISO) LDPC decoder. An interleaved coding with multistage decoding (MSD) architecture with LDPC codes as component codes has been proposed in [13] as an alternative to TE for ISI channels. In this scheme, M independent codewords are interleaved at the encoder. At the receiver, the M codewords are decoded sequentially, with each decoded codeword being used for decoding subsequent interleaves. It is known that the achievable rate of this MSD scheme converges to the symmetric information rate (SIR) of the ISI channel when $M \rightarrow \infty$.

In this work, we investigate the use of polar codes in system architectures based on turbo-equalization and interleaved coding with multistage decoding. First we study the incorporation of polar codes for use with the TE architecture. Since this requires a soft-output polar decoder, neither SC decoding nor SC-list decoding can be used. Instead, we need a soft-output variant of the SC decoder, such as the BP decoder, or the SCAN decoder. We compare some methods to construct polar codes for use in this architecture.

We also propose the use of an interleaved coding scheme with multistage decoding

with polar codes as component codes. The MSD architecture has several attractive features when polar codes are used as component codes. First, this architecture does not require soft outputs from the component polar decoders, so SC and SC-list decoders may be used. Second, the component polar code for each stage can be designed using techniques developed for memoryless channels. Finally, common hardware designs for encoding and decoding can be shared among the interleaves.

The rest of the paper is organized as follows. The channel model and the TE and MSD architectures are presented in Sections 4.2, 4.3, and 4.4, respectively. Simulation results are presented in Section 4.5. Conclusions are discussed in Section 4.6.

4.2 Channel Model

Magnetic recording channels are modeled as binary-input intersymbol-interference channels with additive white Gaussian noise (AWGN). More specifically, if $\underline{X} = \{x_k\}$ is the input sequence drawn from a binary alphabet $\{\pm 1\}$, the channel output at time n is given by $y_n = \sum_{i=0}^{\nu} h_i x_{n-i} + a_n$. Here $\{h_i\}$ is a known channel impulse response associated with the transfer function $h(D) = \sum_{i=0}^{\nu} h_i D^i$ and a_n is i.i.d. Gaussian noise with zero mean and variance σ^2 . Such a channel is referred to as a partial-response (PR) channel. A well known example is the dicode channel which has transfer function $h(D) = 1 - D$. We note here that the results presented in [12] regarding polarization in the presence of memory do not apply to PR channels.

4.3 Turbo-Equalization Architecture

Current magnetic storage drives use a decoding architecture based upon turbo-equalization [14] that combines a SISO sequence detector and a SISO decoder exchange information iteratively as shown in Figure 4.1. Here, the user data sequence is denoted by $\underline{W}$, the output of the ECC encoder is denoted by $\underline{X}$, and the channel output sequence is denoted by $\underline{Y}$. To decode the user data, the BCJR algorithm [15] is used to determine the log-a-posteriori probability (APP) ratios $\underline{L}$. An ECC decoder feeds back soft-information to the sequence detector until it can decode to the codeword $\hat{\underline{X}}$.

There are two issues in using polar codes as ECC in this architecture. First, we

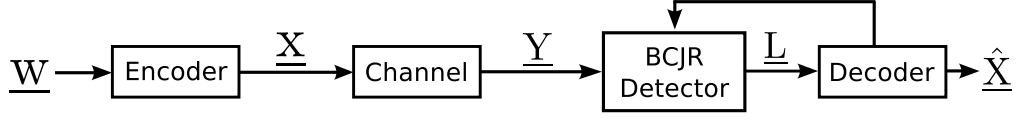


Figure 4.1: Encoder and decoder for turbo-equalization.

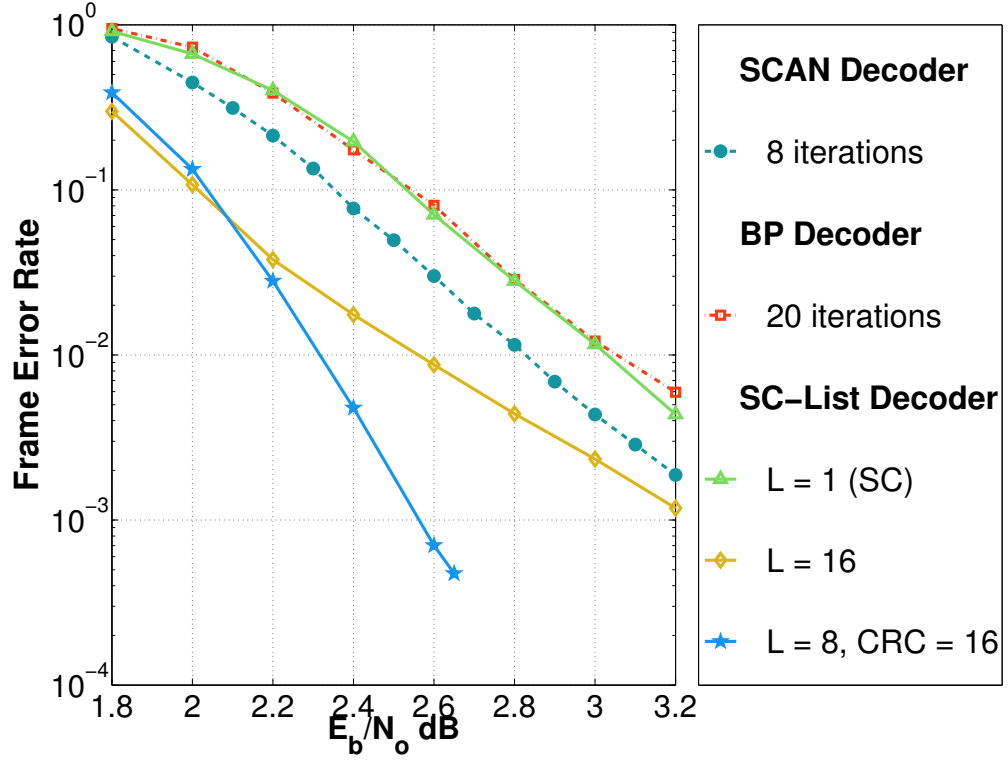


Figure 4.2: Performance of a rate 0.7 blocklength 4096 polar code on AWGN channel when decoded using BP, SCAN, SC and SC-list decoders. The polar code was designed for AWGN channel with $E_b/N_o = 2.14$ dB. Concatenation with an outer 16-bit CRC code is used for SC-list with $L = 8$.

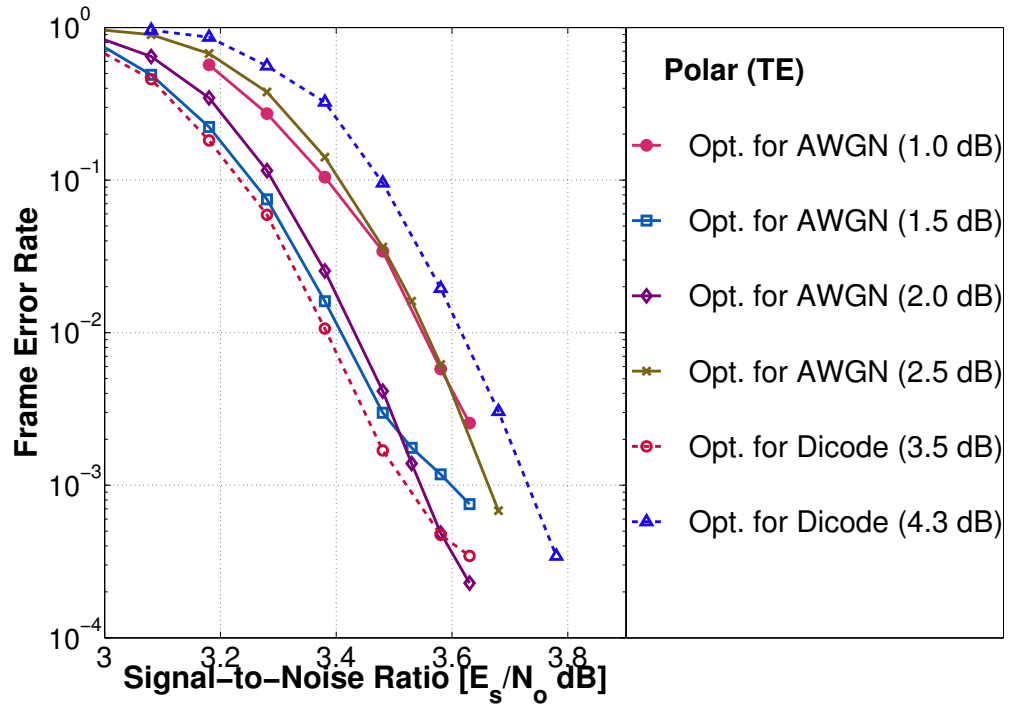


Figure 4.3: Performance of polar codes optimized for various AWGN and dicode channels with rate 0.83 (corresponding to design SNR 2.5 dB) and blocklength 16384 when decoded using SCAN on the dicode channel. Maximum 10 iterations of SCAN and 20 iterations of turbo equalization are used.

need to use a soft-output decoder to be able to feed back soft information to the sequence detector as priors. While BP and SCAN decoders can provide soft-outputs, a larger coding gain is obtained when polar codes are decoded by the SC-list decoder which only provides hard decisions. This is demonstrated in Figure 4.2 which shows the performance of the BP, SCAN, SC, and SC-list for a rate 0.7 polar code on a AWGN channel. The second issue is that the choice of unfrozen bit-channels during polar code construction depends on the underlying channel. The equivalent channel seen by the ECC decoder is not memoryless and there is no method known for making the optimal choice. Therefore, we use the following two sub-optimal approaches. In the first approach, we construct polar codes using the method of [7] for AWGN channels with different noise variances. In the second approach, the conditional output density of the equivalent channel seen by the ECC decoder on the first iteration of TE, that is $\mathbb{P}(L_j|C_j)$, is estimated using Monte-Carlo simulation. Polar codes are constructed for this equivalent channel by assuming it to be memoryless (not necessarily true). The simulation results obtained for codes constructed by these two approaches are shown in Figure 4.3.

4.4 Multistage Decoding Architecture

An *interleaved code* design with a *multi-stage decoding* (MSD) algorithm is considered as an alternative to turbo equalization for binary ISI channels in [13]. In this scheme, M independent codewords of potentially different rates are interleaved at the encoder. At the receiver, each of the M codewords are decoded sequentially, with each decoded codeword being used for decoding subsequent interleaves. Figure 4.4 shows the block diagram for the encoder. The user data, $\underline{W}$, is partitioned into M blocks, $\{\underline{U}^{(i)}\}_{i=1}^M$ of lengths $K^{(1)}, K^{(2)}, \dots, K^{(M)}$ respectively. Each of these M blocks are encoded using codes $\mathcal{C}_1, \dots, \mathcal{C}_M$ of rates $R_i = \frac{K^{(i)}}{N}$ such that the codewords $\{\underline{C}^{(i)}\}_{i=1}^M$ are all of length N bits, denoted as $(C_1^{(i)}, \dots, C_N^{(i)})$ for each i . These M codewords are then randomly interleaved into a sequence $\underline{X}$ which is transmitted.

Figure 4.5 presents the decoder architecture for multi-stage decoding. Given received word $\underline{Y}$, the decoding proceeds to recover the interleaved codewords in M stages. In stage i , let $\hat{\underline{C}}^{(k)}$ denote the estimate for the k^{th} codeword obtained by the decoder at

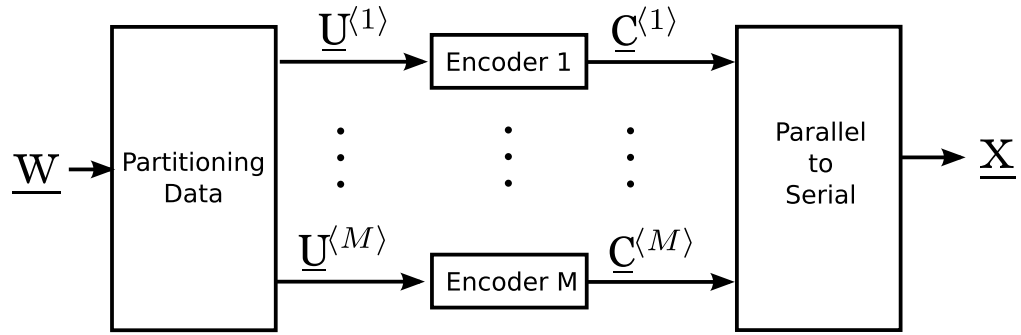


Figure 4.4: Encoder for interleaved coding scheme.

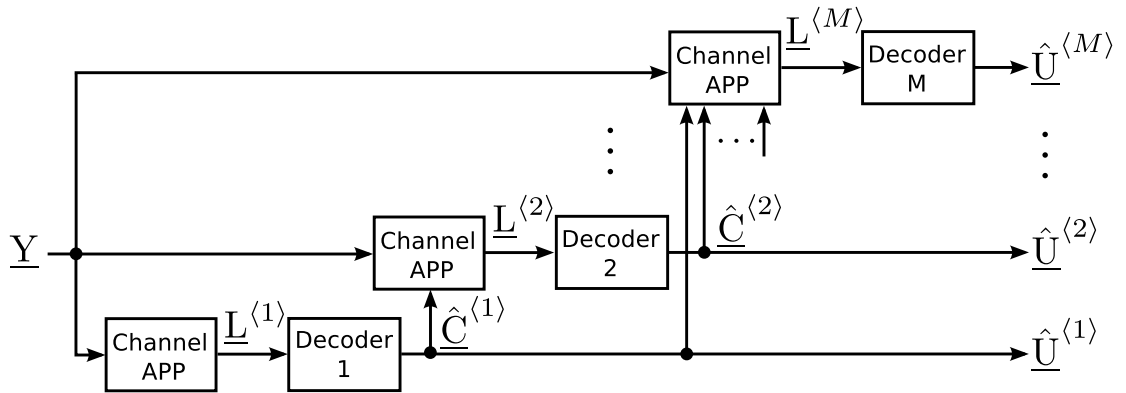


Figure 4.5: Multi-stage decoding scheme.

stage $k = 1, 2, \dots, i-1$. Then, the channel detector determines the log-APP ratio vector $\underline{L}^{(i)} = (L_1^{(i)}, L_2^{(i)}, \dots, L_N^{(i)})$ according to

$$L_j^{(i)} = \log \left(\frac{1 - P_j^{(i)}}{P_j^{(i)}} \right),$$

where,

$$P_j^{(i)} = \mathbb{P} \left(C_j^{(i)} = 1 \mid \underline{Y}, \underline{C}^{(1)} = \hat{\underline{C}}^{(1)}, \dots, \underline{C}^{(i-1)} = \hat{\underline{C}}^{(i-1)} \right).$$

$\underline{L}^{(i)}$ can be computed efficiently using the BCJR algorithm on the entire received sequence while assuming that the estimates for codeword at previous stages are correct and assuming the codewords belonging to the future stages are distributed independently and uniformly. These log-APP ratios are then used by the i^{th} component decoder to estimate the i^{th} codeword as $\hat{\underline{C}}^{(i)}$.

The channel, as seen by the code at the i^{th} stage, can be represented by the *equivalent sub-channel* shown in Figure 4.6. It is shown in [13] that a rate

$$R^{(i)} \triangleq I \left(C_j^{(i)}; L_j^{(i)} \right) = 1 - \mathbb{E} \left[\log_2 \left(1 + e^{-L_j^{(i)}} \right) \mid C_j^{(i)} \right] \quad (4.1)$$

is an achievable information rate for the i^{th} stage, even when the decoder has only the knowledge of the marginal channel law $\mathbb{P}(C_j^{(i)}, L_j^{(i)})$. It is shown in [13] that random codes achieve these bounds with decoders that assume that the equivalent sub-channels are memoryless and that use only partial knowledge about the channel in the form of the marginal output densities. The overall rate, $R_{\text{av},M} = \frac{1}{M} \sum_{i=1}^M R^{(i)}$, is shown to converge to the SIR of the ISI channel as $M \rightarrow \infty$. The paper then optimizes component LDPC codes for each sub-channel to achieve thresholds close to the achievable rates.

In this work, we propose the use of polar codes in place of LDPC codes as interleaved codes in the MSD scheme described in [13]. One can simulate the equivalent sub-channel assuming perfect decision feedback from the component decoders to obtain a set of input and output sequences. The densities $\mathbb{P}(C_j^{(i)}, L_j^{(i)})$ can then be estimated using the histograms for each of the M stages. A different polar code is constructed for each stage by choosing the frozen bits for a memoryless channel according to the method

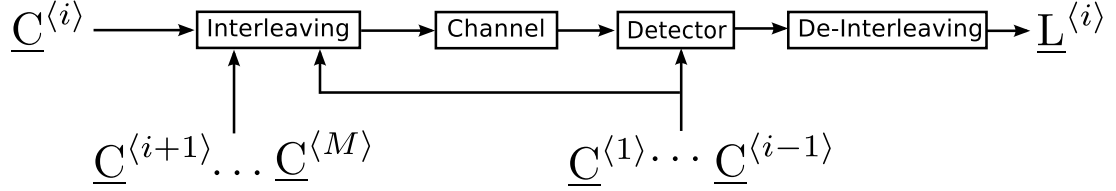


Figure 4.6: Equivalent sub-channel for i^{th} stage with $C^{(1)}, \dots, C^{(i-1)}$ known to the decoder.

described in [7], starting with the channel law given by the densities estimated by the Monte-Carlo simulations.

We now explain how interleaved polar codes can achieve the symmetric information rate of the magnetic recording channel when the number of interleaves is large. First, consider the use of a windowed APP-detector that determines the APP based on a fixed window of channel observations with side-length ω . For large enough ω and codeword length N , the window-APP detector achieves the performance of the BCJR algorithm [16]. For such a windowed detector, the output log-APP ratios that are sufficiently separated in time are conditionally independent. Therefore, with large enough M , the equivalent sub-channel seen by each stage is memoryless. Once these equivalent sub-channels have been estimated with sufficient accuracy using the Monte-Carlo simulations described above, the methods known for construction and decoding of polar codes for memoryless channels may be used to achieve rates given by (4.1). The overall rate $R_{\text{av},M}$ has been shown to converge to the SIR of the ISI channel asymptotically [13], which completes the argument.

4.5 Simulation Results

We demonstrate the performance of the proposed MSD scheme on the dicode channel with $h(D) = 1 - D$. First, we simulate the equivalent subchannels for different number of levels to determine the achievable rates given by (4.1) at various SNR points. The results with $M = 2$ and with $M = 4$ stages are presented in Figure 4.7 and 4.8, respectively. As expected, the rates for successive interleaved stages increase and saturate at the capacity of the AWGN channel with the same noise variance as the ISI channel.

Figures 4.9 and 4.10 show the overall rate for the M -stage MSD system for the

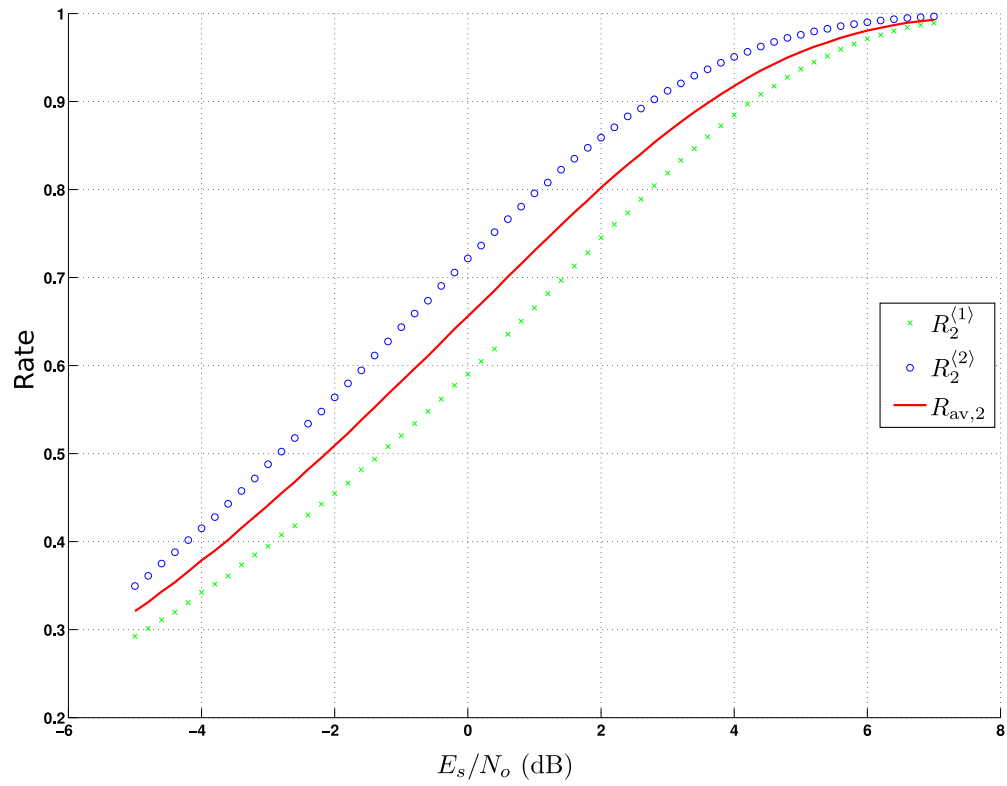


Figure 4.7: Achievable rates for the 2 stages and the average rate for the 2-stage MSD scheme on the decode channel.

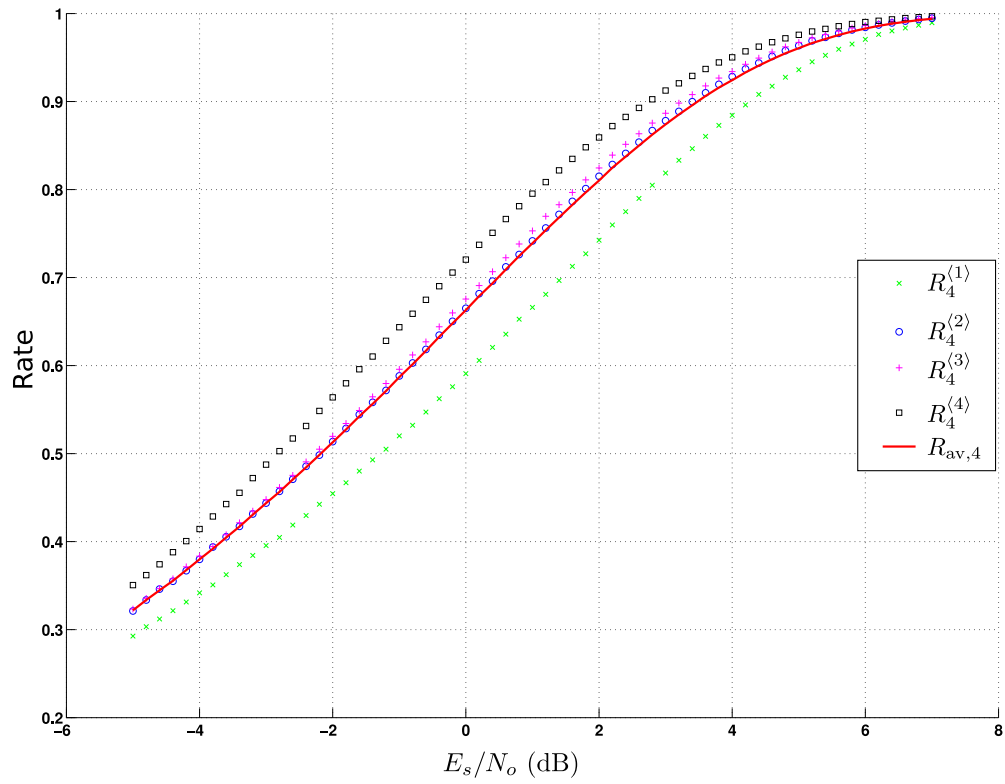


Figure 4.8: Achievable rates for the 4 stages and the average rate for the 4-stage MSD scheme on the decode channel

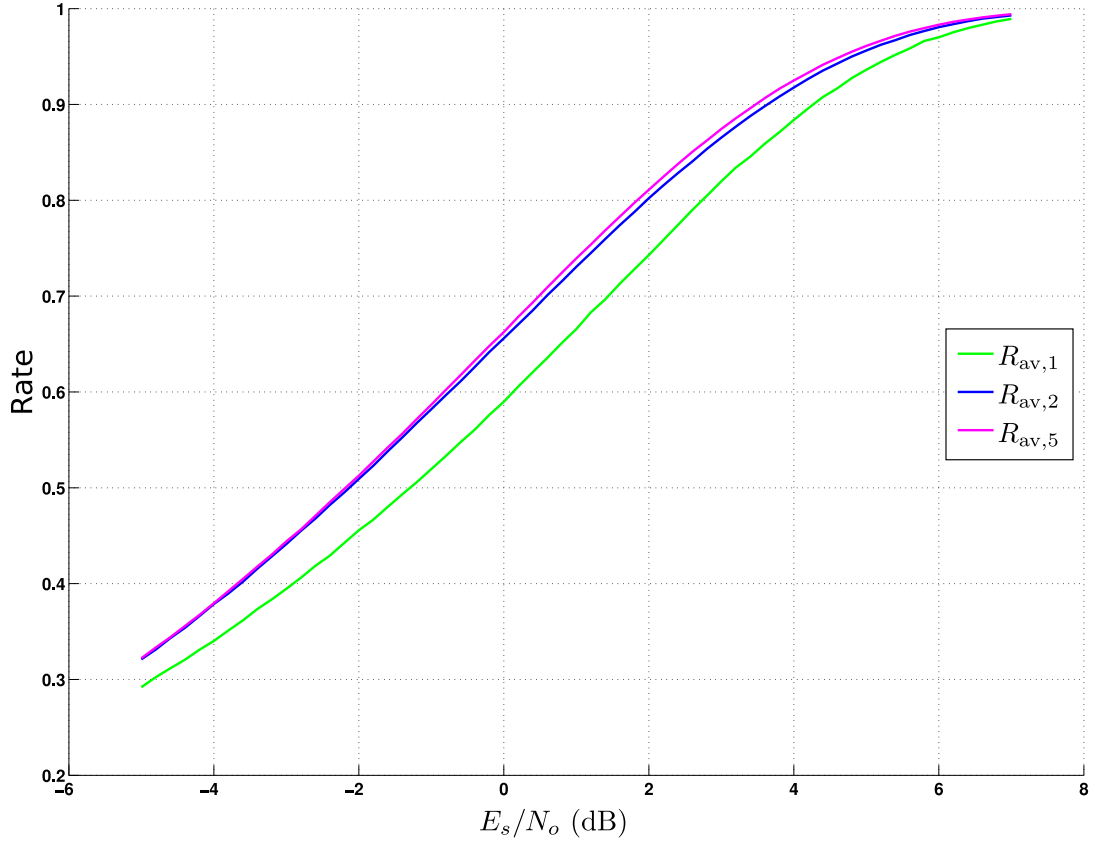


Figure 4.9: Overall rate for a MLC-MSD scheme on the dicode channel for various number of stages.

dicode and EPR4 channels, respectively, for various values of M . It can be seen that a small value of M is sufficient to achieve a rate close to SIR of the channel when random codebooks are used.

We use the histograms obtained by the Monte-Carlo simulations to estimate the densities $\mathbb{P}(L_j^{(i)} = l_j^{(i)} \mid C_j^{(i)} = c)$ at various values of channel SNR and construct polar codes of the rates given by (4.1). The choice of frozen bits for these polar codes was done using the algorithm proposed in [7]. The codes were decoded using a SC decoder in the probability domain using double precision floating point numbers. The following results have been obtained by simulations of a 2-stage MSD scheme for the dicode channel. We constructed pairs of polar codes of lengths $N = 2^{10}, 2^{12}, 2^{14}, 2^{16}$ such that $R^{(1)} = 0.85$ and $R^{(2)} = 0.93$ (as given by the plot in Figure 4.7 at $E_s/N_o = 3.5$ dB). We examined the performance of systematic and non-systematic encoding of

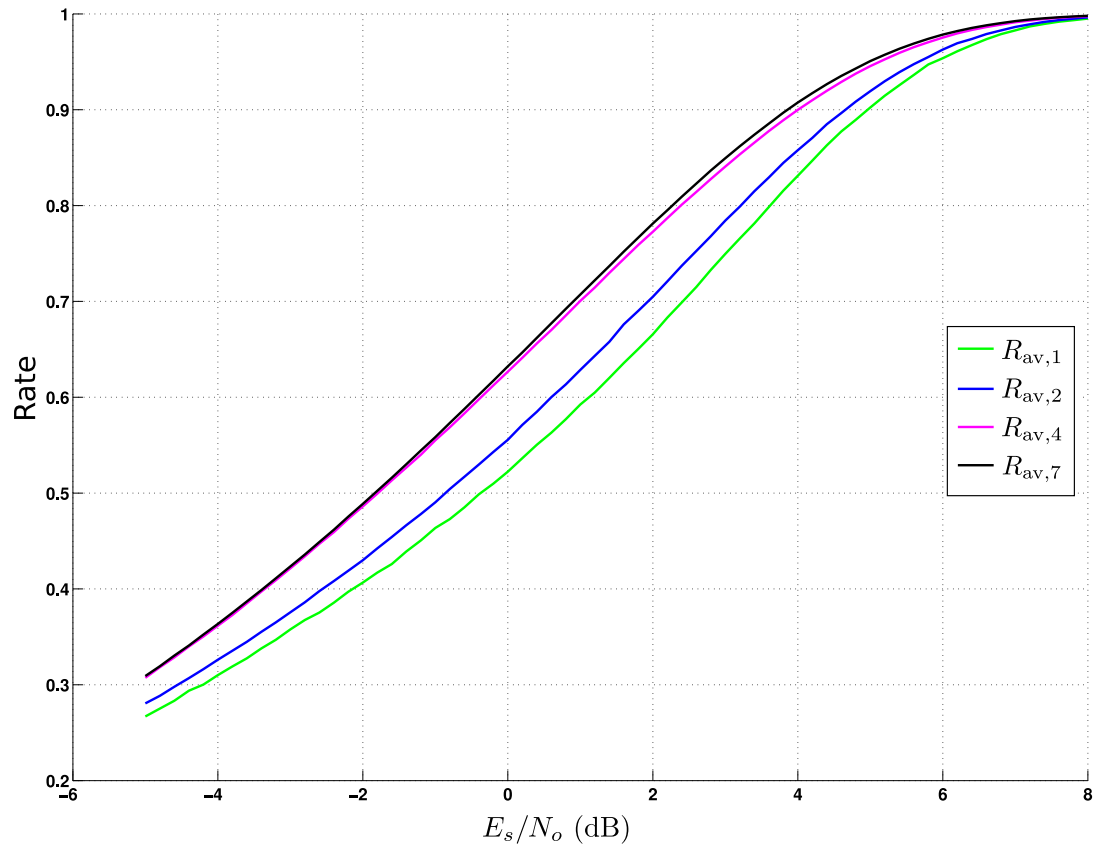


Figure 4.10: Overall rate for a MLC-MSD scheme on the EPR4 channel for various number of stages.

polar codes [17]. Figure 4.11 shows the BER curves for systematic and non-systematic encoding of polar codes that have been designed for the 2 stages of MSD estimated at channel SNR $E_s/N_o = 3.5$ dB. Clearly, the systematic encoding has much better BER performance compared to non-systematic encoding. These results are surprising, given that the SC decoder computes the codeword bits indirectly by first making decisions on the information bits and then encoding the information word back to the codeword. One would expect any decoding errors in the information word to be amplified in the re-encoding process. However, simulation results shown here and those presented earlier in the literature for AWGN channels [17] show that this is not the case. An analytic justification for this phenomenon is still lacking and this represents an important open problem when considering reverse-concatenation with constrained codes, an architecture that is favored in magnetic recording systems.

Since polar codes are constructed for the underlying sub-channels which are obtained by Monte-Carlo simulations at a certain SNR of the channel, we also examined how the choice of the design SNR affects the performance of the polar code. Figure 4.12 shows the BER performance for three pairs of polar codes of the same rate but with different frozen-bit sets which have been determined using simulations at different SNRs ($E_s/N_o = 3.5$ dB, 4.3 dB and 5.0 dB). It can be seen that the codes optimized at a slightly lower noise variance have a reasonable waterfall performance while ensuring that the error floor is low enough. A similar result has been reported for polar codes on memoryless channels in [18]. Also, if the decoders for all polar codes of the same length share the same SC decoding architecture, the knowledge of the operating SNR will optionally allow one to exploit the difference in the performance of these codes of the same rate.

Next we consider the use of the more powerful SC-list decoder. Figure 4.13 shows the BER improvement as the list size increases for a 2-stage MSD scheme with polar codes of rates 0.69 and 0.85 constructed at design SNR 3.5 dB. The list decoder with $L = 32$ performs 0.65 dB better than the SC decoder, which is plotted as list size = 1. These results match those reported for memoryless channels where at higher SNRs the SC-list decoder quickly approaches the performance of the ML decoder with increasing list sizes.

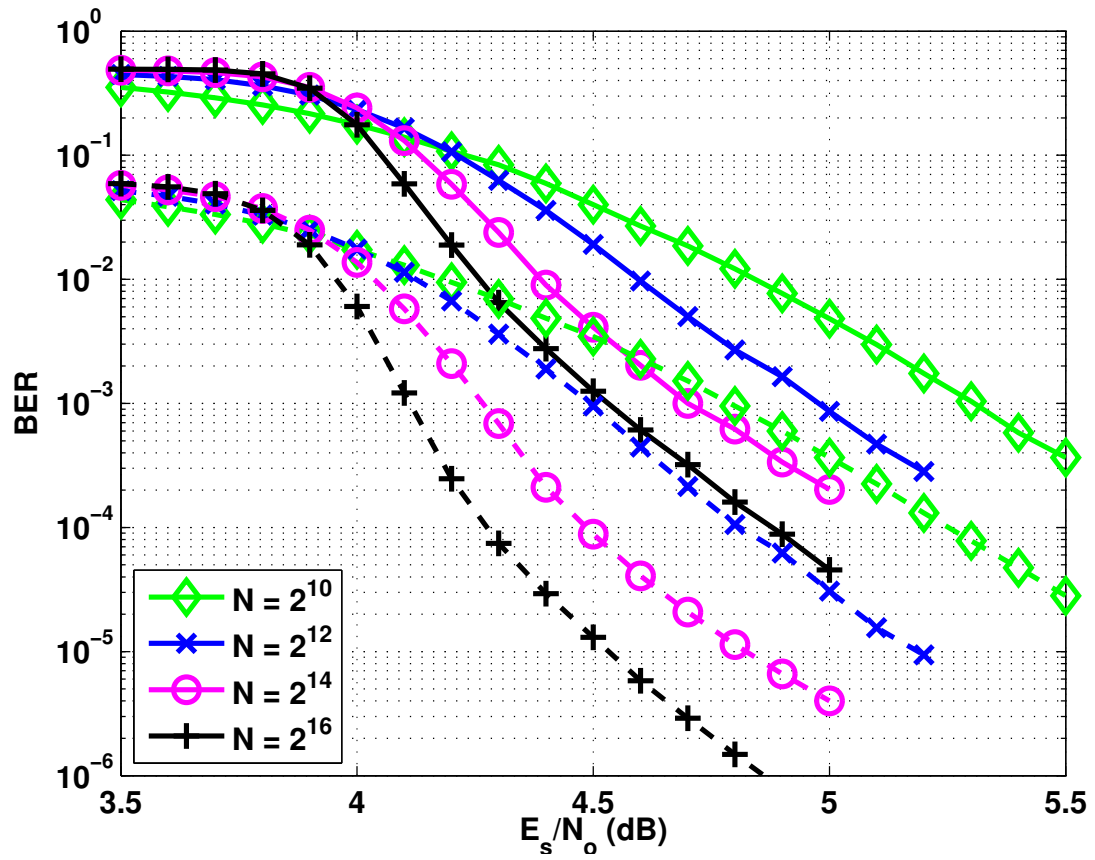


Figure 4.11: BER versus SNR for the 2-stage MSD scheme with various block lengths, N . The dashed and solid lines show the BER for the systematic and non-systematic encodings, respectively.

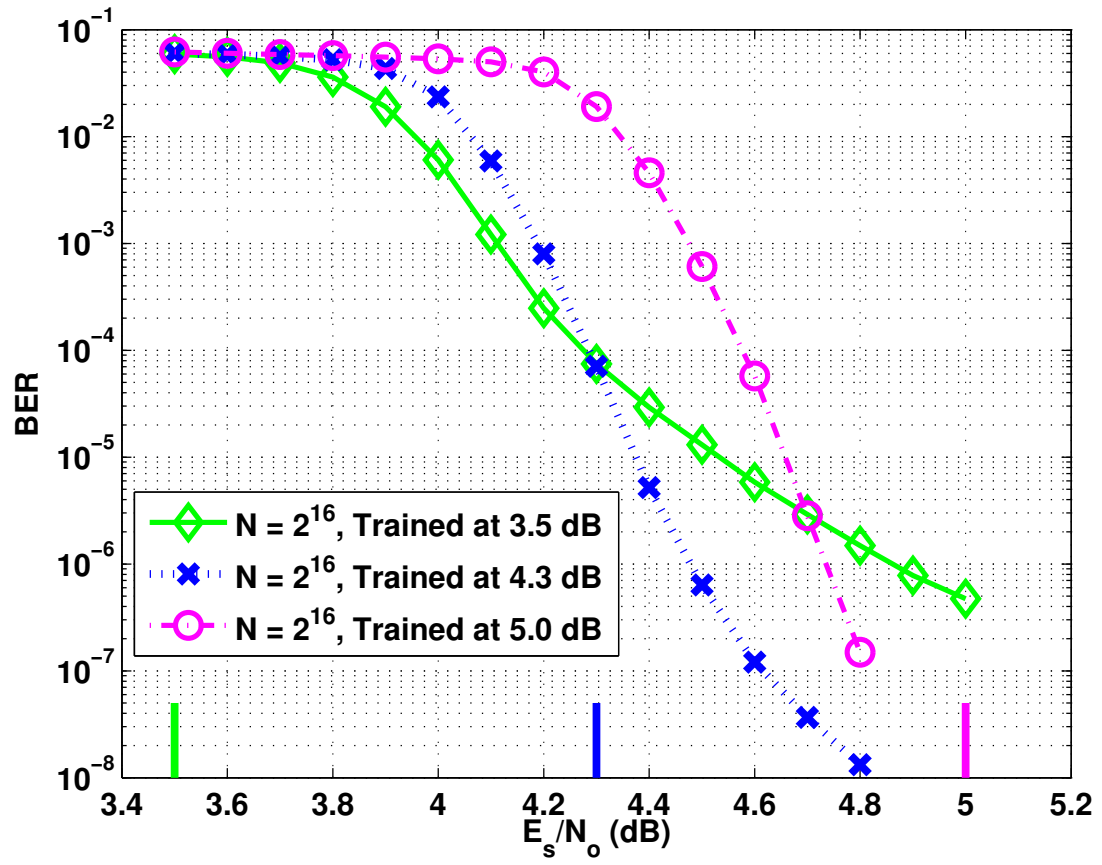


Figure 4.12: BER for the MSD scheme using SC decoder for polar codes ($N = 2^{16}$) constructed at different SNRs.

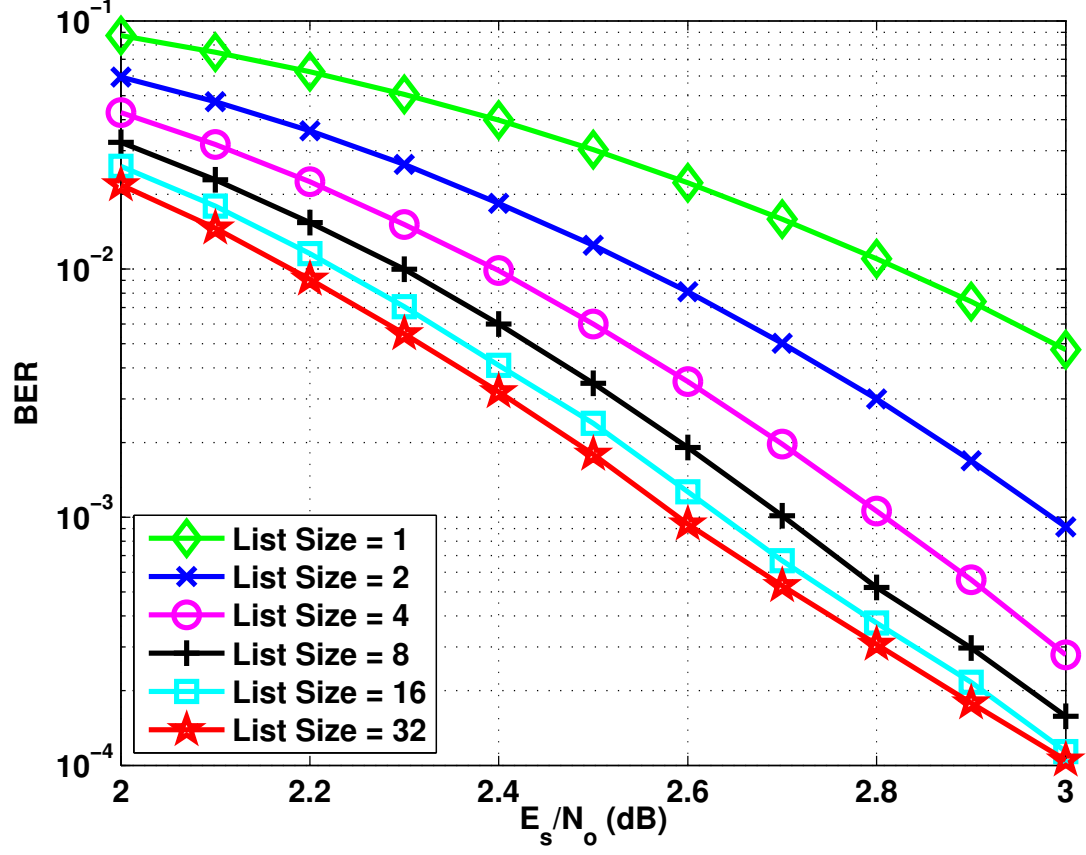


Figure 4.13: BER versus SNR for the MSD scheme for polar codes with rates $R^{(1)} = 0.69$, $R^{(2)} = 0.85$ and blocklength $N = 1024$ decoded with SC-list decoder for various list sizes without CRC-concatenation.

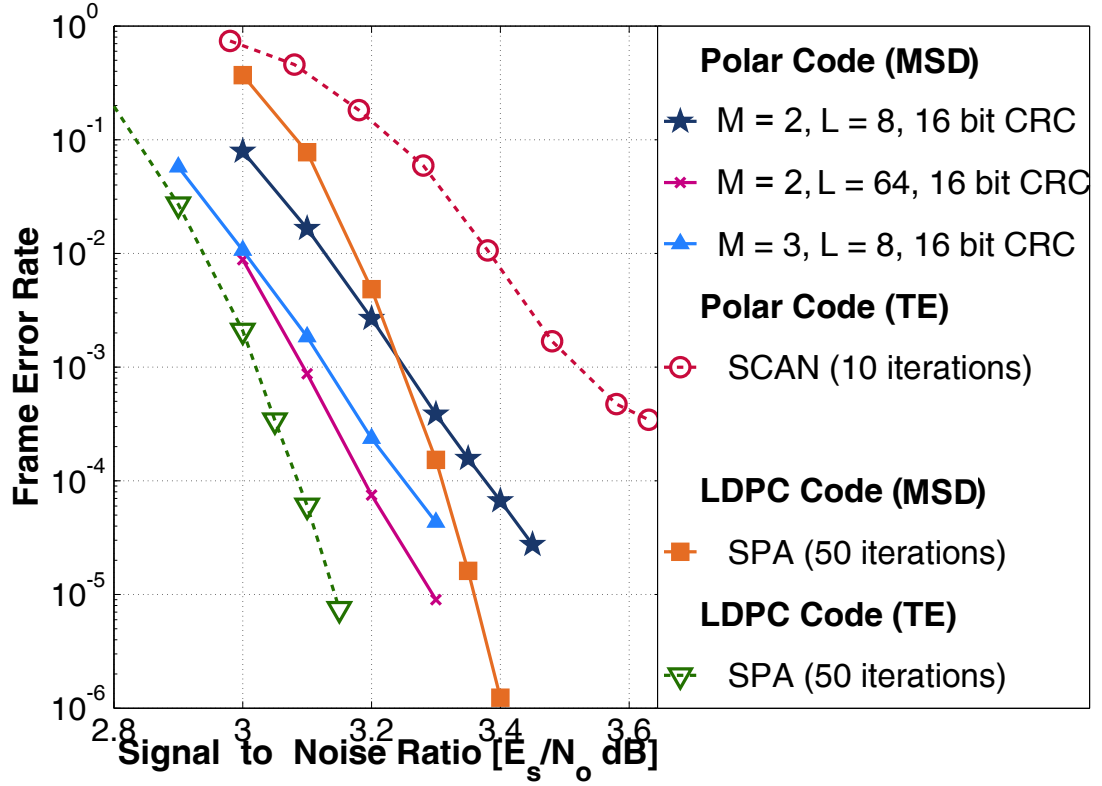


Figure 4.14: FER for the 2-stage MSD scheme using SC-List decoder for polar codes ($N = 2^{14}$), with a concatenated 16-bit CRC. FER for the 2-stage MSD scheme with LDPC codes and TE scheme with LDPC and polar codes is also shown.

In order to improve the SC-list decoder, concatenation with an outer CRC code was proposed in [7]. During decoding, the decoder searches for a survivor path which satisfies the CRC. Results in [19] show that this concatenation improves the minimum distance of the polar codes by eliminating low-weight codewords. We obtained a list of minimum weight codewords for two polar codes of length $N = 2^{14}$ of rates $R^{(1)} \approx 0.77$ and $R^{(2)} \approx 0.88$ with design SNR 2.4 dB. We found that a 16-bit CRC could eliminate all weight-16 minimum Hamming weight codewords we could find for those codes, using $L \leq 10000$. Figure 4.14 shows the FER performance of these polar codes. We also plot the performance of the MSD scheme using optimized LDPC codes of equal rates and blocklengths (design parameters in Table 4.1, 4.2 and 4.3) and the performance obtained using the TE scheme, for comparison. The performance of concatenated polar codes can be improved further by increasing the list size as was noted in [19], or by increasing the number of interleaves.

Table 4.1: Code parameters for the LDPC Code used as component code for the first stage in the 2-stage MSD scheme in Figure 4.14.

Design Rate	0.7812
Number of Variable Nodes	16384
Number of Check Nodes	3584
Number of Degree-2 Variable Nodes	1536
Number of Degree-3 Variable Nodes	8192
Number of Degree-4 Variable Nodes	3584
Number of Degree-5 Variable Nodes	512
Number of Degree-6 Variable Nodes	512
Number of Degree-7 Variable Nodes	1536
Number of Degree-9 Variable Nodes	512
Number of Degree-12 Check Nodes	512
Number of Degree-13 Check Nodes	512
Number of Degree-16 Check Nodes	512
Number of Degree-17 Check Nodes	1024
Number of Degree-20 Check Nodes	512
Number of Degree-28 Check Nodes	512

Table 4.2: Code parameters for the LDPC Code used as component code for the second stage in the 2-stage MSD scheme in Figure 4.14.

Design Rate	0.8750
Number of Variable Nodes	16384
Number of Check Nodes	2048
Number of Degree-2 Variable Nodes	1024
Number of Degree-3 Variable Nodes	10752
Number of Degree-4 Variable Nodes	2048
Number of Degree-5 Variable Nodes	512
Number of Degree-6 Variable Nodes	1024
Number of Degree-7 Variable Nodes	512
Number of Degree-8 Variable Nodes	512
Number of Degree-26 Check Nodes	512
Number of Degree-28 Check Nodes	512
Number of Degree-29 Check Nodes	512
Number of Degree-32 Check Nodes	512

Table 4.3: Code parameters for the LDPC Code used for the turbo equalization scheme in Figure 4.14.

Design Rate	0.8281
Number of Variable Nodes	16384
Number of Check Nodes	2816
Number of Degree-2 Variable Nodes	1024
Number of Degree-3 Variable Nodes	14592
Number of Degree-9 Variable Nodes	256
Number of Degree-13 Variable Nodes	256
Number of Degree-14 Variable Nodes	256
Number of Degree-20 Check Nodes	1792
Number of Degree-19 Check Nodes	512
Number of Degree-21 Check Nodes	512

4.6 Conclusions

The MSD scheme allows the decoder to assume that the equivalent sub-channels are memoryless, allowing us to use encoding/decoding methods designed for polar codes for binary memoryless channels. For long blocklengths and large number of interleaved stages, polar codes are able to achieve the SIR of the ISI channels. For small number of interleaves, simulation results show that the MSD scheme outperforms the TE scheme when polar codes are used as component codes and are comparable to LDPC codes with the TE scheme.

Acknowledgements

This chapter is part of the paper “Polar Codes for Magnetic Recording Channels,” Aman Bhatia, Paul H. Siegel; which is being prepared for submission to the IEEE Transactions on Communications. The dissertation author was the primary investigator and author of this paper.

Bibliography

- [1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
- [2] S. B. Korada and E. Sasoglu, "A class of transformations that polarize binary-input memoryless channels," *Proc. IEEE International Symposium on Information Theory*, pp. 1478–1482, Jun. 28 - Jul. 3, 2009.
- [3] E. Abbe and E. Telatar, "Polar codes for the m-user MAC and matroids," *CoRR*, Aug. 2010, available from <http://arxiv.org/pdf/1002.0777>.
- [4] H. MahdaviFar and A. Vardy, "Achieving the secrecy capacity of wiretap channels using polar codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6428–6443, Oct. 2011.
- [5] S. B. Korada and R. L. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1751–1768, Apr. 2010.
- [6] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," *Proc. IEEE International Symposium on Information Theory*, Jun. 28 - Jul. 3, 2009, pp. 1496–1500.
- [7] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.
- [8] R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," *Proc. IEEE International Symposium on Information Theory*, Jul. 31 - Aug. 5, 2011, pp. 11–15.
- [9] I. Tal and A. Vardy, "List decoding of polar codes," *Proc. IEEE International Symposium on Information Theory*, Jul. 31 - Aug. 5, 2011, pp. 1–5.
- [10] E. Arikan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Communication Letters*, vol. 12, pp. 447–449, Jun. 2008.
- [11] U. U. Fayyaz and J. R. Barry, "Polar codes for partial response channels," *Proc. IEEE International Conference on Communications*, Budapest, Hungary, June 9 - 13, 2013, pp. 4337–4341.
- [12] E. Sasoglu, "Polarization in the presence of memory," *Proc. IEEE International Symposium on Information Theory*, Jul. 31 - Aug. 5, 2011, pp. 189–193.
- [13] J. B. Soriaga, H. D. Pfister, and P. H. Siegel, "Determining and approaching achievable rates of binary intersymbol interference channels using multistage decoding," *IEEE Transactions on Information Theory*, vol. 53, no. 4, pp. 1416–1429, Apr. 2007.

- [14] T. Souvignier, M. Oberg, P. H. Siegel, R. Swanson, and J. K. Wolf, "Turbo decoding for partial response channels," *IEEE Transactions on Communications*, vol. 48, no. 8, pp. 1297–1308, Aug. 2000.
- [15] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
- [16] J. B. Soriaga, "On near-capacity code design for partial-response channels," Ph.D. dissertation, Univ. Calif., San Diego, La Jolla, Mar. 2005.
- [17] E. Arıkan, "Systematic Polar Coding," *IEEE Communication Letters*, vol. 15, no. 8, pp. 860–862, Aug. 2011.
- [18] M. Mondelli, S. Hassani, and R. Urbanke, "From polar to Reed-Muller codes: A technique to improve the finite-length performance," *Proc. IEEE International Symposium on Information Theory*, Jun. 29 - Jul. 4, 2014, pp. 131–135.
- [19] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Communication Letters*, vol. 16, no. 12, pp. 2044–2047, Dec. 2012.

Chapter 5

Analysis of Stochastic Decoding of LDPC Codes

While stochastic decoding of LDPC codes has been shown to perform well with low hardware complexity, an analysis of the decoding algorithm has been lacking. In this chapter, we study the performance of stochastic decoding on LDPC codes ensembles in the limit of infinite blocklength and provide a method to perform density evolution for stochastic decoding. We also provide alternative descriptions of the variable node update in stochastic decoding: as a Markov process, as a probabilistic and-or gate, and as a successively relaxed sum-product algorithm.

5.1 Introduction

Low-density parity-check (LDPC) codes, introduced by Gallager [1] and rediscovered in the 1990's by Davey and MacKay have been shown to be among the best known codes for an AWGN channel. Out of the various message-passing soft-decoding schemes proposed for LDPC codes, the sum-product algorithm (SPA) is the most widely studied decoding algorithm. However its high complexity has been a bottleneck in many applications. This complexity is two-fold. First, the update rules at each node in the SPA require considerable hardware. Second, passing the messages between nodes while retaining the parallelism inherent in the decoding requires many parallel wires, leading to routing congestion. In order to reduce the complexity of the hardware involved in

the decoder implementation, several ways to approximate the SPA using finite precision messages have been proposed [2, 3, 4, 5].

The aim of stochastic decoding (SD) is to simplify the update rules of the SPA while retaining its superior performance. While SD has been shown to perform as well as the SPA for certain codes and with a few modifications to take care of dependencies in the Tanner graphs (e.g. see [8, 6, 7, 9] and references therein), an analysis of the process showing the reason for the observed performance has been lacking. In this chapter, we aim to analyze the performance of ensembles of LDPC codes using SD in a process that parallels density evolution analysis for SPA. The introduction of memory in the variable node update for SD makes density evolution more complicated because the outgoing messages on an edge at iteration ℓ are no longer independent of the messages on the same edge at the previous iteration, an assumption that is required for the density evolution for SPA. Similar to the work on density evolution for deterministic decoding algorithms with memory [10], we derive a Bayesian network [13] for SD which is then used to analyze the performance of SD using density evolution. We also provide novel descriptions for the variable node update used in SD, connecting it to other known decoding algorithms. First, we present the variable node update as producing output bits according to a Markov process and study the stationary distribution and rate of convergence of this process. Second, we show that the variable node update can be interpreted as a probabilistic AND-OR logic. Finally, we cast SD as a successively relaxed SPA, where the relaxation parameter depends on the incoming messages at each iteration.

This chapter is organized as follows. A review of the SPA decoder is provided in Section 5.2 for the case where messages are in the probability domain. Section 5.3 introduces the update rules for SD and analyzes their connection to those used in SPA decoding. Alternative descriptions for the variable node update for SD are provided in Section 5.4. Density evolution for SD is discussed in Section 5.5. Finally, conclusions are presented in Section 5.6.

5.2 Sum Product Algorithm

SPA works iteratively with exchange of soft information from variable nodes to check nodes and vice-versa. Though the SPA decoder is typically implemented in the log-likelihood-ratio (LLR) domain, one may represent the update rules in the probability domain where messages are the a-posteriori probability (APP) of the sent bit being a 1 (i.e. -1 after BPSK modulation). Assuming equal priors, the LLR (μ) and APP (p) are related as follows:

$$\mu = \ln \left(\frac{\mathbb{P}(x = +1|y)}{\mathbb{P}(x = -1|y)} \right) = \ln \left(\frac{1-p}{p} \right). \quad (5.1)$$

Then, the update rules are represented in the APP-domain as follows. At a check node of degree d_c ,

$$q_j = \frac{1}{2} \left(1 - \prod_{\substack{i=1 \\ i \neq j}}^{d_c} (1 - 2p_i) \right) \quad \forall j = 1, \dots, d_c \quad (5.2)$$

and at a variable node of degree d_v ,

$$p_j = \frac{\prod_{\substack{i=0 \\ i \neq j}}^{d_v} q_i}{\prod_{\substack{i=0 \\ i \neq j}}^{d_v} q_i + \prod_{\substack{i=0 \\ i \neq j}}^{d_v} (1 - q_i)} \quad \forall j = 1, \dots, d_v \quad (5.3)$$

where q_0 is the APP received from the channel and p_i 's and q_i 's denote messages from variable nodes and check nodes respectively.

5.3 Stochastic Decoding Algorithm

The idea behind stochastic decoding is to communicate the APP between a pair of nodes not by passing quantized versions of the probability as in (5.2) and (5.3), but through the statistics of a bitstream passed between these nodes. SD is an iterative message passing algorithm where random bits are passed along the edges of the Tanner graph. These bits are distributed according to a Bernoulli distribution with parameter p where the parameter p is the a-posteriori probability in equation (5.1). This alleviates

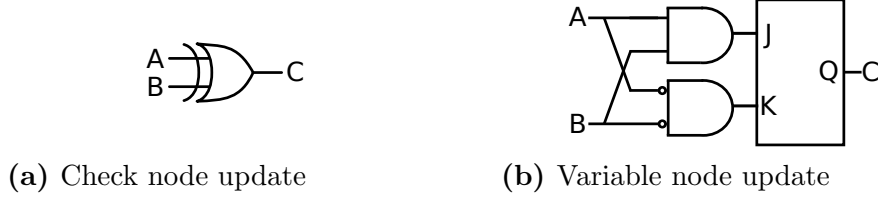


Figure 5.1: Hardware implementation for stochastic decoding updates.

the routing congestion problem mentioned before as only a single wire is required to pass the ‘messages’ as bits. Since SD aims at achieving the performance of the SPA, the update rules defined for the messages are such that the outgoing bits on any edge have a Bernoulli distribution with parameter p given by the equations (5.2) and (5.3). We start with updates for a node with two incoming bits A and B , and an outgoing bit C , and then generalize the implementation to other node degrees. In this section, we assume that incoming bits are independent.

The check node update, as given by equation (5.2) for $d_c = 2$, can be written as

$$q_c = p_a(1 - p_b) + p_b(1 - p_a)$$

where $\mathbb{P}(A = 1) = p_a$, $\mathbb{P}(B = 1) = p_b$ and $\mathbb{P}(C = 1) = q_c$. Then the hardware implementation shown in Figure 5.1a, where A and B are fed as inputs to an *XOR*-gate and the output is fed to C , gives us exactly the required update rule. A check node update for an arbitrary node degree is obtained by *XOR*ing all the input streams.

For a degree-2 variable node, from equation (5.3) we have

$$p_c = \frac{q_a q_b}{q_a q_b + (1 - q_a)(1 - q_b)}.$$

Consider the hardware setup shown in Figure 5.1b. Here we use a *JK* flip-flop, whose output is given as

$$Q = (J \wedge \neg Q') \vee (\neg K \wedge Q')$$

where the $\neg$ denotes negation and Q' denotes the value of Q at the previous cycle. Then the outgoing message is 1 with probability

$$p_c = \mathbb{P}(C = 1)$$

$$\begin{aligned}
&= \mathbb{P}(A = B = 1) + \mathbb{P}(A = 0, B = 1, C' = 1) \\
&\quad + \mathbb{P}(A = 1, B = 0, C' = 1) \\
&= q_a q_b + (1 - q_a) q_b p'_c + q_a (1 - q_b) p'_c \\
&= p'_c + \beta_{ab} (p_{\text{SPA}} - p'_c)
\end{aligned} \tag{5.4}$$

where $\beta_{ab} = q_a q_b + (1 - q_a)(1 - q_b)$ and

$$p_{\text{SPA}} = \frac{q_a q_b}{q_a q_b + (1 - q_a)(1 - q_b)}.$$

Equation (5.4) has the form of a gradient update rule that converges to p_{SPA} , the required update. A generalization to degree d_v at iteration ℓ is straightforward and the outgoing message is 1 with probability given by

$$p^{(\ell)} = \prod_{i=0}^{d_v-1} q_i^{(\ell)} + p^{(\ell-1)} \left(1 - \prod_{i=0}^{d_v-1} q_i^{(\ell)} - \prod_{i=0}^{d_v-1} (1 - q_i^{(\ell)}) \right) \tag{5.5}$$

Note that although the variable node operation is deterministic, it has memory.

5.4 Alternative Descriptions of SD

In this section, we provide some alternative descriptions of the stochastic decoding algorithm and highlight its some novel connections to other decoding algorithms.

5.4.1 Variable node update as a Markov chain

The two-input variable node can also be analyzed as a Markov chain since the outputs of the node depend on the present inputs and the previous output. The state diagram for this Markov chain is shown in Figure 5.2. This chain is aperiodic and

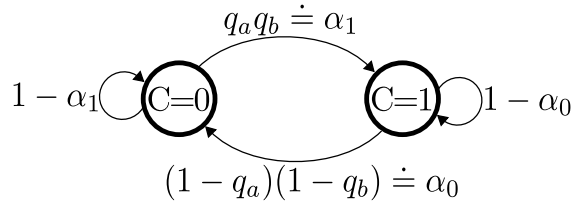


Figure 5.2: Markov chain representing the variable node update.

irreducible and therefore has a stationary distribution with probability of C being 1

given by

$$\pi_1 = \frac{\alpha_1}{\alpha_0 + \alpha_1} = p_{\text{SPA}}.$$

Although the variable update rule for SD does converge to the update given by SPA, the rate of this convergence is also important and we consider it in the following. The statistic of the output of the variable node at the l^{th} iteration of the update, given by equation (5.4), can be written as

$$\begin{aligned} p^{(\ell)} &= p^{(\ell-1)} + \beta_{ab} (p_{\text{SPA}} - p^{(\ell-1)}) \\ &= p^{(0)} (1 - \beta_{ab})^\ell + p_{\text{SPA}} (1 - (1 - \beta_{ab})^\ell). \end{aligned}$$

Therefore,

$$\left| p^{(\ell)} - p_{\text{SPA}} \right| = \left| p^{(0)} - p_{\text{SPA}} \right| e^{-\eta \ell} \quad (5.6)$$

where

$$\eta = \ln \left(\frac{1}{1 - \beta_{ab}} \right) \geq 0.$$

From equation (5.6), the output statistic converges to the target value, p_{SPA} , exponentially in ℓ , the number of iterations of the update rules. However, this is true when the statistics of the inputs do not change with the iterations, which is not true for SD as the statistics of the input streams are also being updated. Nevertheless, the analysis above indicates that if incoming statistics have converged to the correct value, the output stream statistic should converge exponentially fast in iterations.

5.4.2 Variable node update as a probabilistic AND-OR gate

We now look at another interpretation of the variable node update used in stochastic decoding. We attempt to find a simple update rule for the bit-messages that would require less hardware for implementation and would update the statistics of the bit-messages in a logical way. Using the notation from equation (5.5), trivial upper and

lower bounds on the statistic of the outgoing bit-stream from a variable node are

$$\prod_{i=0}^{d-1} q_i^{(\ell)} \leq p_d^{(\ell)} \leq 1 - \prod_{i=0}^{d-1} (1 - q_i^{(\ell)}).$$

These lower and upper bounds are achieved when all incoming bits are 1 and 0, respectively. These bounds correspond to statistics obtained with *AND* and *OR* gate operations on incoming bit-streams. While the *AND* gate update drives the statistic towards 0, the *OR* gate update drives it towards 1. Thus, the variable node update should be an *AND* gate update or an *OR* gate update according to whether the variable node is likely to be 0 or 1, respectively. The current estimate regarding whether the bit corresponding to the variable node is 0 or 1 is the value of the update on the previous iteration, $p_d^{(\ell-1)}$. Therefore, it seems reasonable to use the *AND* gate update and the *OR* gate update in the proportion of $1 - p_d^{(\ell-1)} : p_d^{(\ell-1)}$. Then, the statistic of the outgoing bitstream is given by

$$p_d^{(\ell)} = \left(1 - p_d^{(\ell-1)}\right) \prod_{i=0}^{d-1} q_i^{(\ell)} + p_d^{(\ell-1)} \left(1 - \prod_{i=0}^{d-1} (1 - q_i^{(\ell)})\right)$$

and is the same as the SD variable node update in equation (5.5). Thus, the above analysis shows that the variable update used in SD is a simple sequential logic that one can use to update the bitstream at the variable node.

5.4.3 Variable node update as successively relaxed SPA

We now look at yet another interpretation of the variable node update used in stochastic decoding. The outgoing bitstream statistic for the variable node update can be written as

$$p_d^{(\ell)} = p_{\text{SPA}} \beta_{q^{(\ell)}} + p_d^{(\ell-1)} (1 - \beta_{q^{(\ell)}})$$

$$\text{where } p_{\text{SPA}} = \frac{\prod_i q_i^{(\ell)}}{\prod_i q_i^{(\ell)} + \prod_i (1 - q_i^{(\ell)})}$$

is the SPA variable update with the same incoming messages, and

$$\beta_{q^{(\ell)}} \triangleq \prod_i q_i^{(\ell)} + \prod_i (1 - q_i^{(\ell)}) \quad (5.7)$$

is a *relaxation parameter* in the range $[0, 1]$. Thus, the outgoing statistic is a weighted sum of the SPA update p_{SPA} and the statistic used on the edge in the previous iteration $p_d^{(\ell-1)}$. The variable node update used in SD may therefore be treated as a successively relaxed version of the variable node update in SPA, where the relaxation factor is given by equation (5.7). The relaxation parameter $\beta_{q^{(\ell)}}$ is a function of the incoming bitstream statistics only and as the consensus in the incoming bitstreams increases, $\beta_{q^{(\ell)}}$ increases and the weight for p_{SPA} increases; otherwise, when incoming messages do not agree, $p_d^{(\ell)}$ has larger weight.

There are some other forms of relaxed SPA variable node update known [14, 15]. In [14], relaxed SPA is used as an approximation to analog iterative decoding of LDPC codes. Analog decoding is modeled by first-order non-linear differential equations and the system of differential equations is approximated by a successive relaxation method as

$$p^{(\ell)} = p_{\text{SPA}} \beta + p^{(\ell-1)} (1 - \beta)$$

where β is a small constant. It is shown through simulation that given a large number of iterations (10 000), this version of relaxed SPA ($\beta = 0.05$) performs better than conventional SPA ($\beta = 1$) by 0.4 dB. [15] uses a randomized message passing similar to SD for belief propagation for the purpose of inference over graphs called Stochastic Belief Propagation (SBP). A variable node update corresponding to the SBP algorithm is given by

$$p^{(\ell)} = p_{\text{SPA}} \beta_\ell + p^{(\ell-1)} (1 - \beta_\ell)$$

where β_ℓ depends on iteration number, ℓ , and not on incoming statistics, $\{q_i\}$. It was shown through analysis that SBP converges almost surely to the BP fixed point.

It is interesting to note that relaxed versions of SPA have been shown to have good performance and asymptotic analysis has been provided. However, it is difficult

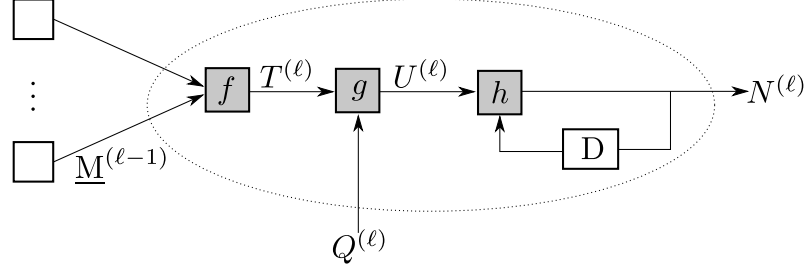


Figure 5.3: Alternative way to represent variable node update for a stochastic decoder used in density evolution in Section 5.5.

to generalize the analysis to the case of SD, where the relaxation parameter depends on the incoming messages.

5.5 Asymptotic Analysis for SD

Following the extensive work on asymptotic analysis for decoding LDPC codes from a given ensemble using various message passing algorithms, in this section we explore an approach for computing the densities of the messages in stochastic decoding as they evolve during the decoding process. Assuming decoding is being performed for an infinitely long code, the computation graph of the message over any edge is a tree. Since we are considering BMS channels, it also suffices to analyze the performance under the assumption of transmission of the all-zero codeword.

Upper case variables will denote random variables, while their lower case counterparts denote a value from their alphabet. Vectors will be denoted by a bar underneath. Let M and N denote outgoing the messages (bits) from check nodes and variable nodes, respectively, and Q denote the channel messages (bits) generated randomly according to the APP value received from the channel,

$$\tilde{Q} \triangleq \mathbb{P}(X_{ch} = 1 | Y_{ch}).$$

In the following, $\underline{M}_d^{(\ell)}$ represents a random vector of bits corresponding to the outgoing messages from a check node at depth d in the computation tree for ℓ iterations. The subscript denoting depth in the tree will be dropped whenever it is clear from the context to simplify the notation. To simplify the density evolution, we introduce ternary variables

T and U in the variable node update in stochastic decoding, as depicted pictorially in Figure 5.3. T and U take values from the alphabet $\{0, 1, ?\}$ and are defined in terms of the incoming messages to a variable node, $\underline{M}^{(\ell-1)}$ and $Q^{(\ell)}$, as follows:

$$T^{(\ell)} = f(\underline{M}^{(\ell-1)}) \triangleq \begin{cases} m, & \text{if } \underline{M}^{(\ell-1)} = (m, \dots, m) \\ ?, & \text{otherwise,} \end{cases}$$

$$\text{and } U^{(\ell)} = g(T^{(\ell)}, Q^{(\ell)}) \triangleq \begin{cases} ?, & \text{if } T^{(\ell)} \neq Q^{(\ell)} \\ T^{(\ell)}, & \text{otherwise.} \end{cases}$$

Then, the outgoing message from the variable node, $N^{(\ell)}$ at iteration ℓ is given by

$$N^{(\ell)} = h(U^{(\ell)}, N^{(\ell-1)}) \triangleq \begin{cases} N^{(\ell-1)}, & \text{if } U^{(\ell)} = ? \\ U^{(\ell)}, & \text{otherwise.} \end{cases}$$

5.5.1 Bayesian Network for the Stochastic Decoder

A Bayesian network is a graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). Formally, a DAG $G(\mathcal{V}, \mathcal{E})$ is a Bayesian network with respect to a set of random variables $\mathcal{V} = \{V_1, V_2, \dots, V_N\}$ if the joint probability distribution of $\mathcal{V}$ can be factored as

$$\mathbb{P}(\mathcal{V}) = \mathbb{P}(V_1, \dots, V_N) = \prod_{i=1}^N \mathbb{P}(V_i \mid pa(V_i))$$

where $pa(V_i) = \{W \in \mathcal{V} : (W, V_i) \in \mathcal{E}\}$ denotes the set of parents of V_i .

The dependencies between messages passed in stochastic decoding are depicted in the Bayesian network in Figure 5.4, where one dimension represents the iteration number ℓ and the other represents the depth in the computation tree d .

5.5.2 Exact Density Evolution

Using the Bayesian network in Figure 5.4, we can compute the exact densities of the random variables $N^{(\ell)}$ by recursively computing the joint densities of $U^{(0:\ell)}$, $T^{(1:\ell)}$, $M^{(0:\ell-1)}$, and $N^{(0:\ell-1)}$ for each ℓ . The steps required are presented next in this subsection.

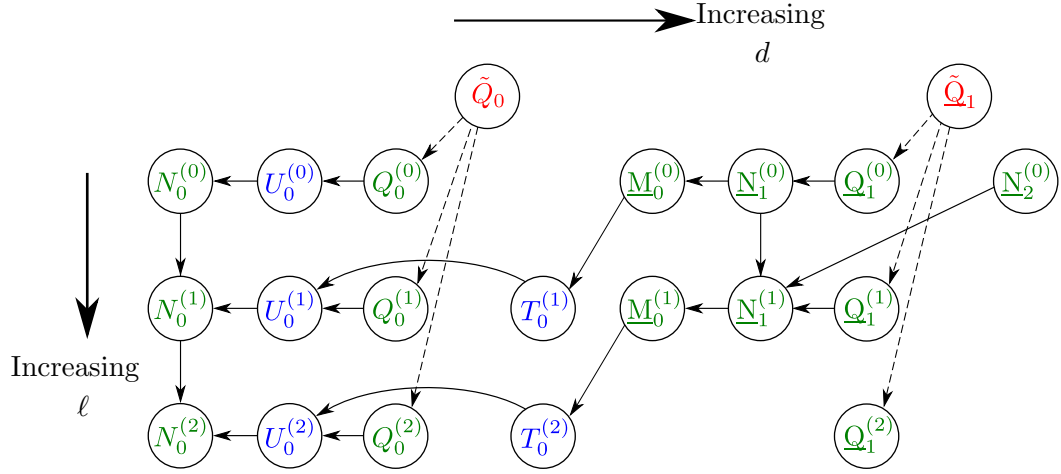


Figure 5.4: Bayesian network for a stochastic decoder. Variables in red, green and blue are channel APPs, binary and ternary variables, respectively. Variables with dashed incoming links are stochastic functions of the incoming variables; all other variables are a deterministic function of the incoming variables.

Computing $\mathbb{P}(N_0^{(\ell)})$

From the Bayesian network in Figure 5.4, we have

$$\begin{aligned} \mathbb{P}(N^{(\ell)} = n') &= \sum_{n,u} \mathbb{P}(N^{(\ell)} = n', N^{(\ell-1)} = n, U^{(\ell)} = u) \\ &= \sum_{n,u} \mathbb{P}(N^{(\ell-1)} = n, U^{(\ell)} = u) \mathbb{1}\{n' = h(u, n)\} \end{aligned}$$

We therefore consider the joint distribution $\mathbb{P}(N^{(\ell-1)} = n, U^{(\ell)} = u)$. From the Bayesian network in Figure 5.4, $N^{(\ell-1)}$ and $U^{(\ell)}$ are independent given $U^{(0:\ell-1)}$ so that

$$\begin{aligned} &\mathbb{P}(N^{(\ell-1)} = n, U^{(\ell)} = u) \\ &= \sum_{u^{(0:\ell-1)}} \mathbb{P}(N^{(\ell-1)} = n, U^{(0:\ell)} = u^{(0:\ell-1)} \cdot u) \\ &= \sum_{u^{(0:\ell-1)}} \underbrace{\mathbb{P}(N^{(\ell-1)} = n \mid U^{(0:\ell-1)} = u^{(0:\ell-1)})}_{\triangleq A^{(\ell)}(n, u^{(0:\ell-1)})} \underbrace{\mathbb{P}(U^{(0:\ell)} = u^{(0:\ell-1)} \cdot u)}_{\triangleq B^{(\ell)}(u^{(0:\ell-1)} \cdot u)}. \quad (5.8) \end{aligned}$$

In (5.8), $A^{(\ell)}$ can be computed as

$$\begin{aligned} A^{(\ell)}(n, u^{(0:\ell-1)}) &= \mathbb{P}\left(N^{(\ell-1)} = n \mid U^{(0:\ell-1)} = u^{(0:\ell-1)}\right) \\ &= \sum_{j=0}^{\ell-1} \mathbb{1}\{u^{(j)} = n\} \mathbb{1}\{u^{(j+1:\ell-1)} = (?, \dots, ?)\}. \end{aligned}$$

Computing $\mathbb{P}(U_0^{(0:\ell)})$

To evaluate Equation (5.8), we need to compute

$$\begin{aligned} B^{(\ell)}(u^{(0:\ell)}) &= \mathbb{P}(U^{(0:\ell)} = u^{(0:\ell)}) \\ &= \int_{\tilde{q} \in [0,1]} d\mathbb{P}(\tilde{Q} = \tilde{q}) \underbrace{\mathbb{P}(U^{(0:\ell)} = u^{(0:\ell)} \mid \tilde{Q} = \tilde{q})}_{\triangleq C^{(\ell)}(u^{(0:\ell)}, \tilde{q})} \end{aligned} \quad (5.9)$$

which can be computed via

$$\begin{aligned} C^{(\ell)}(u^{(0:\ell)}, \tilde{q}) &= \mathbb{P}(U^{(0:\ell)} = u^{(0:\ell)} \mid \tilde{Q} = \tilde{q}) \\ &= \mathbb{P}(U^{(0)} = u^{(0)} \mid \tilde{Q} = \tilde{q}) \sum_{t^{(1:\ell)}} \left(\mathbb{P}(T^{(1:\ell)} = t^{(1:\ell)}) \times \right. \\ &\quad \left. \prod_{i=1}^{\ell} \mathbb{P}(U^{(i)} = u^{(i)} \mid T^{(i)} = t^{(i)}, \tilde{Q} = \tilde{q}) \right). \end{aligned} \quad (5.10)$$

According to the update rules of the stochastic decoder,

$$\begin{aligned} \mathbb{P}(U^{(0)} = u^{(0)} \mid \tilde{Q} = \tilde{q}) \\ = \tilde{q} \cdot \mathbb{1}\{u^{(0)} = 1\} + (1 - \tilde{q}) \cdot \mathbb{1}\{u^{(0)} = 0\}. \end{aligned} \quad (5.11)$$

For any $i \geq 1$,

$$\begin{aligned} \mathbb{P}(U^{(i)} = u^{(i)} \mid T^{(i)} = t^{(i)}, \tilde{Q} = \tilde{q}) \\ = \tilde{q} \cdot \mathbb{1}\{u^{(i)} = g(t^{(i)}, 1)\} + (1 - \tilde{q}) \cdot \mathbb{1}\{u^{(i)} = g(t^{(i)}, 0)\}. \end{aligned} \quad (5.12)$$

Equations (5.11) and (5.12) are then used to compute (5.10).

Computing $\mathbb{P}\left(T_0^{(1:\ell)}\right)$

Next, we need to compute

$$\begin{aligned} D^{(\ell)}\left(t^{(1:\ell)}\right) &\triangleq \mathbb{P}\left(T^{(1:\ell)} = t^{(1:\ell)}\right) \\ &= \sum_{\underline{\mathbf{m}}^{(0:\ell-1)}} \mathbb{P}\left(T^{(1:\ell)} = t^{(1:\ell)} \mid \underline{\mathbf{M}}^{(0:\ell-1)} = \underline{\mathbf{m}}^{(0:\ell-1)}\right) \mathbb{P}\left(\underline{\mathbf{M}}^{(0:\ell-1)} = \underline{\mathbf{m}}^{(0:\ell-1)}\right) \\ &= \sum_{\underline{\mathbf{m}}^{(0:\ell-1)}} \left(\prod_{i=1}^{\ell} \mathbb{1}\left\{t^{(i)} = f\left(\underline{\mathbf{m}}^{(i-1)}\right)\right\}\right) \mathbb{P}\left(\underline{\mathbf{M}}^{(0:\ell-1)} = \underline{\mathbf{m}}^{(0:\ell-1)}\right) \end{aligned}$$

where $\underline{\mathbf{M}}^{(i)} = \left\{M_{[1]}^{(i)}, M_{[2]}^{(i)}, \dots, M_{[d_v-1]}^{(i)}\right\}$ denote the $d_v - 1$ incoming messages from neighboring check nodes. Now, we note that

$$\mathbb{P}\left(\underline{\mathbf{M}}_0^{(0:\ell-1)} = \underline{\mathbf{m}}_0^{(0:\ell-1)}\right) = \prod_{i=1}^{d_v-1} \mathbb{P}\left(M_{0,[i]}^{(0:\ell-1)} = m_{0,[i]}^{(0:\ell-1)}\right)$$

because messages from different parts of the computation tree are independent. Then, we need to calculate $\mathbb{P}\left(M_0^{(0:\ell-1)}\right)$ where the subscript $[i]$ has been dropped because $M_{0,[i]}^{(0:\ell-1)}$ are identically distributed for all i .

Computing $\mathbb{P}\left(M_0^{(0:\ell-1)}\right)$

According to the check node operation of the stochastic decoder,

$$\begin{aligned} E^{(\ell)}\left(m^{(0:\ell-1)}\right) &\triangleq \mathbb{P}\left(M^{(0:\ell-1)} = m^{(0:\ell-1)}\right) \\ &= \sum_{\underline{\mathbf{n}}_1^{(0:\ell-1)}} \left(\prod_{i=0}^{\ell-1} \mathbb{1}\left\{m^{(i)} = \bigoplus_{j=1}^{d_c-1} n_{1,[j]}^{(i)}\right\}\right) \mathbb{P}\left(\underline{\mathbf{N}}_1^{(0:\ell-1)} = \underline{\mathbf{n}}_1^{(0:\ell-1)}\right) \quad (5.13) \\ \text{where } \underline{\mathbf{N}}_1^{(i)} &= \left\{N_{1,[1]}^{(i)}, N_{1,[2]}^{(i)}, \dots, N_{1,[d_c-1]}^{(i)}\right\}. \end{aligned}$$

Again,

$$\mathbb{P}\left(\underline{\mathbf{N}}_1^{(0:\ell-1)} = \underline{\mathbf{n}}_1^{(0:\ell-1)}\right) = \prod_{j=1}^{d_c-1} \mathbb{P}\left(N_{1,[j]}^{(0:\ell-1)} = n_{1,[j]}^{(0:\ell-1)}\right)$$

by independence of messages from different subtrees of the computation tree.

Computing $\mathbb{P}\left(N_1^{(0:\ell-1)}\right)$

To evaluate Equation (5.13) we need to compute $\mathbb{P}\left(N_1^{(0:\ell-1)}\right)$; however, the density of $N_1^{(0:\ell-1)}$ is the same as the density of $N_0^{(0:\ell-1)}$ which can be computed as

$$\begin{aligned} F^{(\ell)}\left(n^{(0:\ell-1)}\right) &\triangleq \mathbb{P}\left(N^{(0:\ell-1)} = n^{(0:\ell-1)}\right) \\ &= \sum_{u^{(0:\ell-1)}} B^{(\ell-1)}\left(u^{(0:\ell-1)}\right) \mathbb{1}\left\{n^{(0)} = u^{(0)}\right\} \left(\prod_{i=1}^{\ell-1} \mathbb{1}\left\{n^{(i)} = h\left(n^{(i-1)}, u^{(i)}\right)\right\}\right) \end{aligned}$$

where $B^{(\ell-1)}$ has been computed using equation (5.9) in the previous iteration.

Remark 5.5.1. *The analysis presented here is easily extended to incorporate various improvements proposed for stochastic decoding, including noise-scaling, edge-memories and tracking-forecast memories [6, 7].*

5.5.3 Approximate Density Evolution

The support of the densities computed in Section 5.5.2 grows exponentially in the number of iterations. For example, $U^{(0:\ell)}$ can take all $3^{\ell+1}$ possible values from the set $\{0, 1, ?\}^{\ell+1}$. This makes it infeasible to compute these densities when the number of iterations is large. In [10], the authors face a similar problem while performing the density evolution analysis of a differential decoding with binary message passing (DD-BMP) algorithm [11, 12]. The authors in [10] proposed an approximation by truncating the memory of the decoding process, and estimating the densities assuming a Markov process of order K , where K is a constant independent of the number of iterations ℓ . This is achieved by removing some edges in the Bayesian network, or equivalently, ignoring some dependencies between certain random variables.

In this subsection we use a similar approach to define an approximate density evolution technique for stochastic decoding that can be computed efficiently. In particular, we assume that at iteration ℓ the edge between $N^{(\ell-K)}$ and $N^{(\ell-K+1)}$ is removed from the Bayesian network, for some $K > 0$, so that

$$N^{(\ell-1)} \perp\!\!\!\perp U^{(\ell)} \mid U^{(\ell-K+1:\ell-1)}. \quad (5.14)$$

The parameter K is called the *memory order* of the approximate density evolution.

Approximate density evolution proceeds as follows.

Computing $\mathbb{P}(N^{(\ell-1)}, U^{(\ell)})$

Instead of using Equation (5.8) for computing $\mathbb{P}(N^{(\ell-1)}, U^{(\ell)})$, we use the following

$$\begin{aligned}
& \mathbb{P}(N^{(\ell-1)}, U^{(\ell)}) \\
&= \sum_{u^{(\ell-K+1:\ell-1)}} \int_{\tilde{q} \in [0,1]} \mathbb{P}(N^{(\ell-1)}, U^{(\ell-K+1:\ell)}, \tilde{Q}) \, d\mathbb{P}(\tilde{Q}) \\
&= \sum_{u^{(\ell-K+1:\ell-1)}} \int_{\tilde{q} \in [0,1]} \mathbb{P}(N^{(\ell-1)} \mid U^{(\ell-K+1:\ell)}, \tilde{Q}) \mathbb{P}(U^{(\ell-K+1:\ell)} \mid \tilde{Q}) \, d\mathbb{P}(\tilde{Q}) \\
&= \sum_{u^{(\ell-K+1:\ell-1)}} \int_{\tilde{q} \in [0,1]} \tilde{A}^{(\ell)}(n^{(\ell-1)}, u^{(\ell-K+1:\ell)}, \tilde{q}) \tilde{B}^{(\ell)}(u^{(\ell-K+1:\ell)}, \tilde{q}) \, d\mathbb{P}(\tilde{Q}) \quad (5.15)
\end{aligned}$$

where

$$\tilde{A}^{(\ell)}(n^{(\ell-1)}, u^{(\ell-K+1:\ell)}, \tilde{q}) \triangleq \mathbb{P}(N^{(\ell-1)} \mid U^{(\ell-K+1:\ell)}, \tilde{Q})$$

and

$$\tilde{B}^{(\ell)}(u^{(\ell-K+1:\ell)}, \tilde{q}) \triangleq \mathbb{P}(U^{(\ell-K+1:\ell)} \mid \tilde{Q}).$$

Computing $\tilde{A}^{(\ell)}$

Using the approximation in (5.14),

$$\begin{aligned}
& \tilde{A}^{(\ell)}(n^{(\ell-1)}, u^{(\ell-K+1:\ell)}, \tilde{q}) \\
&= \mathbb{P}(N^{(\ell-1)} \mid U^{(\ell-K+1:\ell)}, \tilde{Q}) \\
&\approx \mathbb{P}(N^{(\ell-1)} \mid U^{(\ell-K+1:\ell-1)}, \tilde{Q}) \\
&= \sum_{n^{(\ell-2)}} \mathbb{P}(N^{(\ell-1)} \mid N^{(\ell-2)}, U^{(\ell-1)}) \mathbb{P}(N^{(\ell-2)} \mid U^{(\ell-K+1:\ell-1)}, \tilde{Q}) \\
&= \sum_{n^{(\ell-2)}} \mathbb{1}\{n^{(\ell-1)} = h(n^{(\ell-2)}, u^{(\ell-1)})\} \tilde{C}^{(\ell)}(n^{(\ell-2)}, u^{(\ell-K+1:\ell-1)}, \tilde{q}) \quad (5.16)
\end{aligned}$$

where

$$\tilde{C}^{(\ell)}\left(n^{(\ell-2)}, u^{(\ell-K-1:\ell-1)}, \tilde{q}\right) \triangleq \mathbb{P}\left(N^{(\ell-2)} \mid U^{(\ell-K+1:\ell-1)}, \tilde{Q}\right).$$

Computing $\tilde{B}^{(\ell)}$

$\tilde{B}^{(\ell)}$ needed for (5.15) is computed as

$$\begin{aligned} & \tilde{B}^{(\ell)}\left(u^{(\ell-K+1:\ell)}, \tilde{q}\right) \\ &= \mathbb{P}\left(U^{(\ell-K+1:\ell)} \mid \tilde{Q}\right) \\ &= \sum_{t^{(\ell-K+1:\ell)}} \underbrace{\mathbb{P}\left(T^{(\ell-K+1:\ell)}\right)}_{\triangleq \tilde{D}^{(\ell)}} \prod_{i=\ell-K+1}^{\ell} \mathbb{P}\left(U^{(i)} \mid T^{(i)}, \tilde{Q}\right) \end{aligned} \quad (5.17)$$

where

$$\tilde{D}^{(\ell)}\left(t^{(\ell-K+1:\ell)}\right) \triangleq \mathbb{P}\left(T^{(\ell-K+1:\ell)}\right).$$

Computing $\tilde{C}^{(\ell)}$

For evaluating (5.16), $\tilde{C}^{(\ell)}$ should be evaluated first using

$$\begin{aligned} & \tilde{C}^{(\ell)}\left(n^{(\ell-2)}, u^{(\ell-K-1:\ell-1)}, \tilde{q}\right) \\ &= \mathbb{P}\left(N^{(\ell-2)} \mid U^{(\ell-K+1:\ell-1)}, \tilde{Q}\right) \\ &= \sum_{u^{(\ell-K)}} \mathbb{P}\left(N^{(\ell-2)} \mid U^{(\ell-K:\ell-1)}, \tilde{Q}\right) \mathbb{P}\left(U^{(\ell-K)} \mid U^{(\ell-K+1:\ell-1)}, \tilde{Q}\right) \\ &\approx \frac{\sum_{u^{(\ell-K)}} \mathbb{P}\left(N^{(\ell-2)} \mid U^{(\ell-K:\ell-2)}, \tilde{Q}\right) \mathbb{P}\left(U^{(\ell-K:\ell-1)} \mid \tilde{Q}\right)}{\sum_{u^{(\ell-K)}} \mathbb{P}\left(U^{(\ell-K:\ell-1)} \mid \tilde{Q}\right)} \\ &= \frac{\sum_{u^{(\ell-K)}} \tilde{A}^{(\ell-1)}\left(n^{(\ell-2)}, u^{(\ell-K:\ell-2)}, \tilde{q}\right) \tilde{B}^{(\ell-1)}\left(u^{(\ell-K:\ell-1)}, \tilde{q}\right)}{\sum_{u^{(\ell-K)}} \tilde{B}^{(\ell-1)}\left(u^{(\ell-K:\ell-1)}, \tilde{q}\right)}. \end{aligned} \quad (5.18)$$

$\tilde{A}^{(\ell-1)}$ and $\tilde{B}^{(\ell-1)}$ in Equation (5.18) are known from the previous iteration.

Computing $\tilde{D}^{(\ell)}$

The following steps are needed to compute $\tilde{D}^{(\ell)}$ required for (5.17),

$$\begin{aligned} \tilde{D}^{(\ell)}\left(t^{(\ell-K+1:\ell)}\right) &= \mathbb{P}\left(T^{(\ell-K+1:\ell)}\right) \\ &= \sum_{\underline{m}^{(\ell-K:\ell-1)}} \left(\prod_{i=\ell-K+1}^{\ell} \mathbb{P}\left(T^{(i)} \mid \underline{M}^{(i-1)}\right) \right) \left(\prod_{j=1}^{d_o-1} \mathbb{P}\left(M_{[j]}^{(\ell-K:\ell-1)}\right) \right). \end{aligned}$$

As in Section 5.5.2,

$$M_{[j_1]}^{(\ell-K:\ell-1)} \perp\!\!\!\perp M_{[j_2]}^{(\ell-K:\ell-1)} \text{ for } j_1 \neq j_2$$

and identically distributed. Therefore, we only need to compute

$$\tilde{E}^{(\ell)}\left(m^{(\ell-K:\ell-1)}\right) \triangleq \mathbb{P}\left(M^{(\ell-K:\ell-1)}\right).$$

Computing $\tilde{E}^{(\ell)}$

$\tilde{E}^{(\ell)}$ needed above is computed as

$$\begin{aligned} \tilde{E}^{(\ell)}\left(m^{(\ell-K:\ell-1)}\right) &= \mathbb{P}\left(M^{(\ell-K:\ell-1)}\right) \\ &= \sum_{\underline{n}^{(\ell-K:\ell-1)}} \left(\prod_{i=\ell-K}^{\ell-1} \mathbb{P}\left(M^{(i)} \mid \underline{N}^{(i)}\right) \right) \prod_{j=1}^{d_c-1} \mathbb{P}\left(N_{[j]}^{(\ell-K:\ell-1)}\right) \end{aligned} \quad (5.19)$$

As before, $N_{[j]}^{(\ell-K:\ell-1)}$ are i.i.d. for all $j = 1, 2, \dots, d_c - 1$. Therefore, we only need to compute

$$\tilde{F}^{(\ell)}\left(n^{(\ell-K:\ell-1)}\right) \triangleq \mathbb{P}\left(N^{(\ell-K:\ell-1)}\right).$$

Computing $\tilde{F}^{(\ell)}$

$\tilde{F}^{(\ell)}$, required for (5.19), is computed as

$$\tilde{F}^{(\ell)}\left(n^{(\ell-K:\ell-1)}\right) = \mathbb{P}\left(N^{(\ell-K:\ell-1)}\right)$$

$$= \sum_{u^{(\ell-K+1:\ell-1)}} \left(\prod_{i=\ell-K+1}^{\ell-1} \mathbb{P} \left(N^{(i)} \mid N^{(i-1)}, U^{(i)} \right) \right) \underbrace{\mathbb{P} \left(N^{(\ell-K)}, U^{(\ell-K+1:\ell-1)} \right)}_{\triangleq \tilde{G}^{(\ell)}}$$

where

$$\tilde{G}^{(\ell)} \left(n^{(\ell-K)}, u^{(\ell-K+1:\ell-1)} \right) \triangleq \mathbb{P} \left(N^{(\ell-K)}, U^{(\ell-K+1:\ell-1)} \right).$$

Computing $\tilde{G}^{(\ell)}$

$\tilde{G}^{(\ell)}$ needed above is computed as

$$\begin{aligned} \tilde{G}^{(\ell)} \left(n^{(\ell-K)}, u^{(\ell-K+1:\ell-1)} \right) &= \mathbb{P} \left(N^{(\ell-K)}, U^{(\ell-K+1:\ell-1)} \right) \\ &= \sum_{u^{(\ell-K)}, n^{(\ell-K-1)}} \int_{\tilde{q} \in [0,1]} d\mathbb{P} \left(\tilde{Q} \right) \mathbb{P} \left(N^{(\ell-K-1:\ell-K)}, U^{(\ell-K:\ell-1)} \mid \tilde{Q} \right) \\ &= \sum_{u^{(\ell-K)}, n^{(\ell-K-1)}} \int_{\tilde{q} \in [0,1]} d\mathbb{P} \left(\tilde{Q} \right) \mathbb{P} \left(N^{(\ell-K)} \mid N^{(\ell-K-1)}, U^{(\ell-K)} \right) \times \\ &\quad \mathbb{P} \left(N^{(\ell-K-1)}, U^{(\ell-K:\ell-1)} \mid \tilde{Q} \right) \\ &= \sum_{u^{(\ell-K)}, n^{(\ell-K-1)}} \int_{\tilde{q} \in [0,1]} d\mathbb{P} \left(\tilde{Q} \right) \mathbb{1} \left\{ n^{(\ell-K)} = h \left(n^{(\ell-K-1)}, u^{(\ell-K)} \right) \right\} \times \\ &\quad \mathbb{P} \left(U^{(\ell-K:\ell-1)} \mid \tilde{Q} \right) \mathbb{P} \left(N^{(\ell-K-1)} \mid U^{(\ell-K:\ell-1)}, \tilde{Q} \right) \\ &\approx \sum_{u^{(\ell-K)}, n^{(\ell-K-1)}} \int_{\tilde{q} \in [0,1]} d\mathbb{P} \left(\tilde{Q} \right) \mathbb{1} \left\{ n^{(\ell-K)} = h \left(n^{(\ell-K-1)}, u^{(\ell-K)} \right) \right\} \times \\ &\quad \underbrace{\mathbb{P} \left(U^{(\ell-K:\ell-1)} \mid \tilde{Q} \right)}_{=\tilde{B}^{(\ell-1)}} \underbrace{\mathbb{P} \left(N^{(\ell-K-1)} \mid U^{(\ell-K)}, \tilde{Q} \right)}_{\triangleq \tilde{H}^{(\ell)}} \end{aligned}$$

Computing $\tilde{H}^{(\ell)}$

$\tilde{H}^{(\ell)}$ needed above is computed as

$$\begin{aligned} \tilde{H}^{(\ell)} \left(n^{(\ell-K-1)}, u^{(\ell-K)}, \tilde{q} \right) &= \mathbb{P} \left(N^{(\ell-K-1)} \mid U^{(\ell-K)}, \tilde{Q} \right) \\ &= \sum_{u^{(\ell-2K+1:\ell-K-1)}} \mathbb{P} \left(N^{(\ell-K-1)} \mid U^{(\ell-2K+1:\ell-K)}, \tilde{Q} \right) \frac{\mathbb{P} \left(U^{(\ell-2K+1:\ell-K)} \mid \tilde{Q} \right)}{\mathbb{P} \left(U^{(\ell-K)} \mid \tilde{Q} \right)} \end{aligned}$$

$$= \frac{\sum_{u^{(\ell-2K+1:\ell-K-1)}} \tilde{A}^{(\ell-K)} \tilde{B}^{(\ell-K)}}{\sum_{u^{(\ell-2K+1:\ell-K-1)}} \tilde{B}^{(\ell-K)}} \quad (5.20)$$

$\tilde{A}^{(\ell-K)}$ and $\tilde{B}^{(\ell-K)}$ in Equation (5.20) are known from the iteration $\ell - K$. Thus, following the sequence of steps described in this subsection, one can perform the approximate density evolution for a given LDPC ensemble.

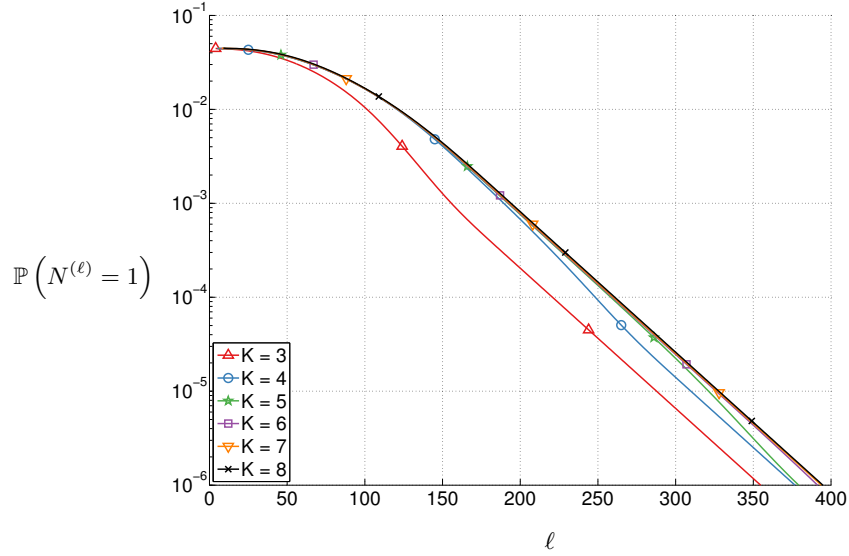


Figure 5.5: $\mathbb{P}(N^{(\ell)} = 1)$ for different iterations, ℓ , computed for the $(3, 6)$ -regular LDPC code ensemble using approximate density evolution with memory order $K \in \{3, 4, \dots, 8\}$ when transmission takes place over a binary symmetric channel with crossover probability $\epsilon = 0.0338$.

Figure 5.5 shows the evolution of $\mathbb{P}(N^{(\ell)} = 1)$ with iterations ℓ for a $(3, 6)$ -regular LDPC code ensemble computed using the approximate density evolution method described above with different values of the memory order K . Clearly, the results converge even with small values of the parameter K , which indicates that the approximation is good.

5.5.4 Threshold for stochastic decoding

Figure 5.6 shows $\mathbb{P}(N^{(\ell)} = 1)$ versus ℓ for a $(3, 6)$ -regular LDPC code ensemble when transmitted over binary symmetric channels with various crossover probabili-

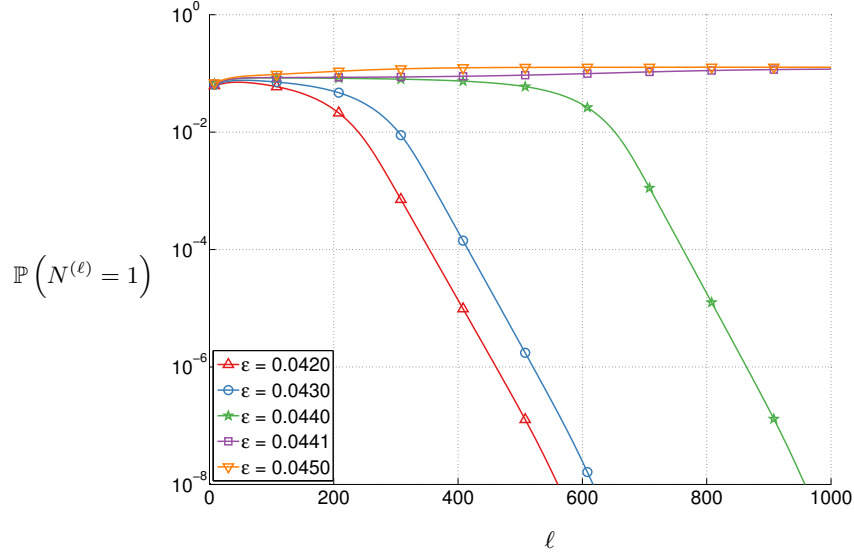


Figure 5.6: $\mathbb{P}(N^{(\ell)} = 1)$ for different iterations, ℓ , computed for the $(3, 6)$ -regular LDPC code ensemble using approximate density evolution with memory order $K = 7$ when transmission takes place over a binary symmetric channel with crossover probability ϵ .

ties. The graph demonstrates a thresholding behavior, that is, $\mathbb{P}(N^{(\ell)} = 1)$ goes to zero rapidly for crossover probabilities below 0.0440 with increasing iterations ℓ ; however, $\mathbb{P}(N^{(\ell)} = 1)$ does not decrease when the crossover probability is larger than 0.0440 even for large number of iterations. Note that $\mathbb{P}(N^{(\ell)} = 1)$ denotes the probability that an edge carries an incorrect message to the check node since the all-zero codeword is assumed to be transmitted. Thus, this allows us to define an approximate threshold for the stochastic decoder that can be computed efficiently using the approximate density evolution technique described in the above subsection.

Definition 5.5.2 (Approximate threshold). *Consider a family of BMS channels parameterized by channel parameter ϵ . Then the threshold, ϵ_K^{SD} , is defined as*

$$\epsilon_K^{SD} = \sup \left\{ \epsilon : \mathbb{P}(N^{(\ell)} = 1) \xrightarrow{\ell \rightarrow \infty} 0 \right\}$$

where memory truncation of order K is used to compute $\mathbb{P}(N^{(\ell)})$.

Table 5.1 shows the threshold computed for various values of the parameter K for some ensembles with regular degree nodes. The thresholds computed with different

values of the memory order K are close to each other and are comparable to those obtained for the Gallager A algorithm.

Table 5.1: Thresholds for various LDPC code ensembles under stochastic decoding, Gallager’s algorithm A, and the sum-product algorithm, along with the corresponding Shannon capacity for BSC.

d_v	d_c	$\epsilon_{K=4}^{\text{SD}}$	$\epsilon_{K=5}^{\text{SD}}$	$\epsilon_{K=6}^{\text{SD}}$	$\epsilon_{K=7}^{\text{SD}}$	$\epsilon_{K=8}^{\text{SD}}$	ϵ^{Gal}	ϵ^{SPA}	ϵ^{Sha}
3	6	0.0451	0.0448	0.0445	0.0440	0.0435	0.039	0.084	0.110
4	8	0.0393	0.0394	0.0395	-	-	0.047	0.076	0.110
5	10	0.0324	0.0326	-	-	-	0.027	0.068	0.110
3	5	0.0631	0.0629	0.0626	0.0621	-	0.061	0.113	0.146
4	6	0.0619	0.0622	0.0625	0.0626	-	0.066	0.116	0.174
3	4	0.0972	0.0974	0.0973	0.0968	0.0959	0.106	0.167	0.215

5.6 Conclusions

Novel descriptions of the variable node update in stochastic decoding of LDPC codes are presented. We represent the variable node update as producing output bits according to a Markov process and study the stationary distribution and rate of convergence of this process. We show that the variable node update can be interpreted as a probabilistic AND-OR logic. We also represent SD as a successively relaxed SPA, where the relaxation parameter depends on the incoming messages at each iteration unlike other successively relaxed sum-product algorithms. We propose a method for performing the density evolution for stochastic decoding. However, the proposed method for performing exact density evolution is shown to be computationally infeasible. A method for performing approximate density evolution efficiently is demonstrated and thresholds for some regular LDPC codes are calculated. The proposed method should allow optimizing LDPC code design for stochastic decoding.

Acknowledgements

This chapter contains material from the paper “On Stochastic Decoding for LDPC Codes,” Aman Bhatia, Aravind R. Iyengar, Paul H. Siegel; which is being prepared for submission to the IEEE Transactions on Communications. The dissertation author was the primary investigator and author of this paper, and co-authors have approved the use of the material for this dissertation.

Bibliography

- [1] R. G. Gallager, *Low Density Parity Check Codes*, Cambridge, Massachusetts: MIT Press, 1963.
- [2] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. Thesis,
- [3] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, Xiao-Yu Hu, “Reduced-Complexity Decoding of LDPC Codes,” *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.
- [4] S. K. Planjery, D. Declercq, S. K. Chilappagari, and B. Vasic, “Multilevel decoders surpassing belief propagation on the binary symmetric channel,” in *Proc. IEEE International Symposium on Information Theory*, pp. 769–773, Jun. 13-18 2010.
- [5] X. Zhang and P. H. Siegel, “Quantized min-sum decoders with low error floor for LDPC codes,” *Proc. IEEE International Symposium Information Theory*, Jul. 2 - 5, 2012, pp. 2871–2875.
- [6] S. S. Tehrani, W. J. Gross, and S. Mannor, “Stochastic decoding of LDPC codes”, *IEEE Communications Letters*, vol. 10, no. 10, pp. 716–718, Oct. 2006.
- [7] S. S. Tehrani, A. Naderi, G.-A. Kamendje, S. Mannor, and W. J. Gross, “Tracking forecast memories in stochastic decoders”, in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 19 - 24, 2009, pp. 561–564.
- [8] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, “Delayed stochastic decoding of LDPC codes,” *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5617–5626, Nov. 2011.
- [9] S. S. Tehrani, “Stochastic decoding of low-density parity-check codes”, *Ph.D. Dissertation*, McGill University, Montreal, Canada, Jan. 2011.
- [10] E. Janulewicz and A. H. Banihashemi, “Performance analysis of iterative decoding algorithms with memory over memoryless channels,” *IEEE Transactions on Communications*, vol. 60, no. 12, pp. 3556–3566, Dec. 2012.
- [11] N. Mobini, A. H. Banihashemi, and S. Hemati, “A differential binary message-passing LDPC decoder,” *Proc. 2007 IEEE Global Telecommunication Conference*, Nov. 26 - 27, 2007, pp. 1561–1565.
- [12] N. Mobini, A. H. Banihashemi, and S. Hemati, “A differential binary message-passing LDPC decoder,” *IEEE Transactions on Communications*, pp. 2518–2523, Sep. 2009.
- [13] S. L. Lauritzen, “Graphical Models,” *Oxford University Press*, 1996.

- [14] S. Hemati and A. H. Banihashemi, “Dynamics and performance analysis of analog iterative decoding for low-density parity-check codes”, *IEEE Transactions on Communications*, vol. 54, no. 1, pp. 61–70, Jan. 2006.
- [15] N. Noorshams and M. J. Wainwright, “Stochastic belief propagation: A low-complexity alternative to the sum-product algorithm”, *IEEE Transactions on Information Theory*, vol. 59, no. 4, pp. 1981–2000, Apr. 2013.

Chapter 6

Enhancing Binary Images of Non-Binary LDPC Codes

In this chapter, we investigate the reasons behind the superior performance of belief propagation decoding of non-binary LDPC codes over their binary images when the transmission occurs over the binary erasure channel. We show that although decoding over the binary image has lower complexity, it has worse performance owing to its larger number of stopping sets relative to the original non-binary code. We propose a method to find redundant parity-checks of the binary image that eliminate these additional stopping sets, so that we achieve performance comparable to that of the original non-binary LDPC code with lower decoding complexity.

6.1 Introduction

Low-density parity-check (LDPC) codes were introduced by Gallager [1] and rediscovered in the 1990's. Davey and MacKay showed that non-binary LDPC (NB-LDPC) codes perform better than binary codes for the same bit length [2]. Despite their better performance, higher decoding complexity has limited the implementation of non-binary LDPC codes in real-world applications. Reduced complexity decoding is possible using an equivalent binary Tanner graph, called the binary image of the code. There are many ways to construct binary images of non-binary codes - one way, referred to as basic binary image, is to represent each non-binary codeword by a binary vector of the same

bit length. A recent result [3] defines an extended binary image of longer bit length and establishes the equivalence of belief propagation (BP) decoding over it to BP decoding over the non-binary codes for the binary erasure channel (BEC). Based on this image, we compare NB-LDPC code to their basic binary images and suggest an algorithm to find redundant parity-checks to improve the performance of the basic binary images.

The remainder of the chapter is organized as follows. In Section 6.2, we briefly describe the BP decoding of NB-LDPC codes. We define the basic and extended binary images in Section 6.3. The superiority of BP decoding over NB-LDPC code for BEC compared to its basic binary image is shown in Section 6.4 and the algorithm to bridge this gap in performance through addition of redundant parity-checks to the basic binary image is given in Section 6.5. We summarize our findings in Section 6.6.

6.2 GF(2^b)-LDPC Codes

An LDPC code $\mathcal{C}_q$ over GF(q) where $q = 2^b$ is specified by a *parity-check matrix* $\mathbf{H}_q$ of dimension $m_q \times n_q$ whose elements are from GF(q), where the number of non-zero elements in $\mathbf{H}_q$ is proportional to n_q . The code is then defined as the *nullspace* of $\mathbf{H}_q$, i.e.,

$$\mathcal{C}_q = \{\underline{X} \in \text{GF}(q)^{n_q} : \mathbf{H}_q \otimes_q \underline{X}^\top = \underline{0}^\top\}$$

where $\underline{X}$ is assumed to be a row-vector, $\underline{0}$ the all-zero row-vector, $\otimes_q$ denotes matrix multiplication when matrix elements belong to GF(q), and $^\top$ denotes transposition. When the parity-check matrix $\mathbf{H}_q$ is full-rank, the code $\mathcal{C}_q$ is a vector-subspace of $\text{GF}(q)^{n_q}$ of *dimension* $k_q = n_q - m_q$, has a *blocklength* n_q symbols, and is therefore of *rate* $R = \frac{k_q}{n_q} = 1 - \frac{m_q}{n_q}$.

We associate with the matrix $\mathbf{H}_q$ a bipartite graph, called the *Tanner graph* $\mathcal{G}_q$, as follows. Corresponding to the j^{th} column of $\mathbf{H}_q$ is a *variable* node V_j^q , and corresponding to the i^{th} row is a *check* node C_i^q , $i \in [m_q] \triangleq \{1, 2, \dots, m_q\}, j \in [n_q]$. Every non-zero entry $h_{i,j}$ of $\mathbf{H}_q$ corresponds to an edge between C_i^q and V_j^q with label $h_{i,j}$. We denote by $\mathcal{N}^v(j)$ the set of check nodes connected to a given variable node $j \in [n_q]$, i.e., the neighbors of the j^{th} variable node, and by $\mathcal{N}^c(i)$ the set of variable nodes connected to a given check node $i \in [m_q]$. When the degree of every variable node and every check

node in $\mathcal{G}_q$ is d_l and d_r respectively, i.e., $|\mathcal{N}^v(j)| = d_l$, $|\mathcal{N}^c(i)| = d_r \forall i \in [m_q], j \in [n_q]$, the LDPC code $\mathcal{C}_q$ is said to be (d_l, d_r) -regular. It is easy to see that for a (d_l, d_r) -regular code, the rate satisfies $R = 1 - \frac{m_q}{n_q} = 1 - \frac{d_l}{d_r}$.

Belief Propagation Decoding

The messages passed around on the Tanner graphs in the BP decoder represent the *a posteriori* probabilities of the symbols of $\text{GF}(q)$. We will briefly describe the BP decoding of non-binary LDPC codes over the binary erasure channel (BEC) using the analogue of the *peeling decoder*, which for the BEC is the same as the BP decoder over $\mathcal{G}_q$, which we denote as $\text{BP}(\mathbf{H}_q)$.

We assume a fixed isomorphism $\Phi_B : \text{GF}(q) \mapsto \text{GF}(2)^b$ that preserves the addition operation defined on $\text{GF}(q)$. This isomorphism is used to map codewords $\underline{X} \in \mathcal{C}_q \subset \text{GF}(q)^{n_q}$ to binary vectors $\underline{x} \in \text{GF}(2)^{bn_q}$. The vector $\underline{x}$ is transmitted over a BEC with erasure probability ϵ and let $\mathcal{E}$ be the index set of the erasures in the received word. At the receiver, the *a priori* set of eligible symbols $\mathcal{Q}_j^{(0)}$ for the j^{th} variable node consists of symbols which fit the received binary sequence corresponding to the j^{th} transmitted symbol. Thus $\mathcal{Q}_j^{(0)}$ has 2^a symbols if a out of the b bits in the received binary sequence corresponding to the j^{th} transmitted symbol are erased. The peeling decoder updates these sets iteratively by exchanging messages between variable and check nodes. For the BEC, the messages passed around iteratively can be represented by sets of eligible symbols. Let $\mathcal{Q}_{i,j}^{(\ell)}$ denote the set of eligible symbols sent by the i^{th} check node to the j^{th} variable node in the ℓ^{th} iteration. Then, the peeling decoder performs the following steps iteratively for $\ell \geq 1$

- *Check node processing:* For check node $C_i^q, i \in [m_q]$,

$$\mathcal{Q}_{i,j}^{(\ell)} = \sum_{j' \in \mathcal{N}^c(i) \setminus \{j\}} h_{i,j'} \mathcal{Q}_{j'}^{(\ell-1)} \quad \forall j \in \mathcal{N}^c(i).$$

- *Variable node processing:* For variable node $V_j^q, j \in [n_q]$,

$$\mathcal{Q}_j^{(\ell)} = \mathcal{Q}_j^{(\ell-1)} \cap \left(\bigcap_{i \in \mathcal{N}^v(j)} h_{i,j}^{-1} \mathcal{Q}_{i,j}^{(\ell)} \right).$$

where, if $h \in \text{GF}(q)$ and $\mathcal{Q}, \mathcal{Q}_1, \mathcal{Q}_2 \subseteq \text{GF}(q)$, we define

$$h\mathcal{Q} \triangleq \{ha \mid a \in \mathcal{Q}\}$$

$$\mathcal{Q}_1 + \mathcal{Q}_2 \triangleq \{a_1 + a_2 \mid a_1 \in \mathcal{Q}_1, a_2 \in \mathcal{Q}_2\}.$$

The decoder stops when $|\mathcal{Q}_j^{(\ell)}| = |\mathcal{Q}_j^{(\ell-1)}| \forall j \in [n_q]$. Decoding is successful if $|\mathcal{Q}_j^{(\ell)}| = 1 \forall j \in [n_q]$ for some $\ell \in \mathbb{N}$. It should be noted that any set of eligible symbols ($\mathcal{Q}_j^{(\ell)}$ or $\mathcal{Q}_{i,j}^{(\ell)}$) is a coset of a vector subspace of $\text{GF}(q)$ [4].

6.3 Binary Images

Since codes defined over $\text{GF}(2^b)$ can be written as a collection of vectors over $\text{GF}(2)$, we can consider the *binary images* of these codes. Further, since BP decoding for $\text{GF}(q)$ has a high computational complexity, we can make use of the binary images to decode the non-binary code.

6.3.1 Basic Binary Image

The isomorphism Φ_B defined in Section 6.2 maps any row-vector of symbols

$$\underline{X} = (X_1, \dots, X_{n_q}) \in \text{GF}(q)^{n_q}$$

to a binary row-vector

$$\Phi_B(\underline{X}) \triangleq (\Phi_B(X_1), \dots, \Phi_B(X_{n_q}))$$

of length bn_q . The *basic binary image* of $\mathcal{C}_q$ is defined as the set $\mathcal{C}_B \subset \text{GF}(2)^{bn_q}$ where $\mathcal{C}_B = \{\Phi_B(\underline{X}) : \underline{X} \in \mathcal{C}_q\}$. As described earlier, the codewords of $\mathcal{C}_B$ are the ones that are transmitted over the BEC. Note that there are $\prod_{l=0}^{b-1} (2^b - 2^l)$ possible choices for the isomorphism Φ_B .

Once fixed, the isomorphism Φ_B identifies a mapping $\Psi_m : \text{GF}(q) \mapsto \mathcal{M}_b$, where $\mathcal{M}_b$ is the collection of all invertible matrices over $\text{GF}(2)$ of size $b \times b$, defined below. For

any $h \in \text{GF}(q)$,

$$\Psi_m(h) = \begin{pmatrix} \Phi_B(h \times \Phi_B^{-1}(\underline{u}_{1,b})) \\ \Phi_B(h \times \Phi_B^{-1}(\underline{u}_{2,b})) \\ \vdots \\ \Phi_B(h \times \Phi_B^{-1}(\underline{u}_{b,b})) \end{pmatrix}^\top$$

where $\underline{u}_{i,b}$ denotes the binary row-vector of length b with the element i as 1 and other elements equal to 0, i.e., the *unit vector* along the i^{th} dimension. The set $\{\Psi_m(h) : h \in \text{GF}(q)\} \subset \mathcal{M}_b$ constitutes a field under matrix addition and matrix multiplication operations over $\text{GF}(2)$, i.e., for $h_1, h_2 \in \text{GF}(q)$, $\Psi_m(h_1 + h_2) = \Psi_m(h_1) \oplus_2 \Psi_m(h_2)$ and $\Psi_m(h_1 h_2) = \Psi_m(h_1) \otimes_2 \Psi_m(h_2)$. Also, it is easy to see that for any $h, x \in \text{GF}(q)$, $\Psi_m(h) \otimes_2 \Phi_B(x)^\top = \Phi_B(hx)^\top$.

The mappings Φ_B and Ψ_m allow us to identify $\mathcal{C}_B$ as a linear block code with parity-check matrix $\mathbf{H}_B$ that is obtained by replacing each element of $\mathbf{H}_q$ by its image given by Ψ_m . We could perform BP on the Tanner graph of $\mathbf{H}_B$, denoted by $\text{BP}(\mathbf{H}_B)$, instead of the more complex $\text{BP}(\mathbf{H}_q)$. However this has worse performance in general due to the large number of short cycles introduced by $\Psi_m(h)$'s in the parity-check matrix $\mathbf{H}_B$. In particular, for the BEC, we shall show later that this bad performance can be attributed to a larger number of stopping sets of $\mathbf{H}_B$.

6.3.2 Extended Binary Image

The observation made at the end of the previous subsection leads us to the question whether it is possible to design a binary image of a non-binary code that matches the performance of the non-binary code. This question was answered in the affirmative by Savin in [3] for the BEC. We briefly describe the construction of this code, called the *extended binary image*.

We define a mapping $\Phi_E : \text{GF}(q) \mapsto \text{GF}(2)^{2^b-1}$ such that each symbol of $\text{GF}(q)$ is mapped to a codeword of the *simplex code* $\mathcal{C}_S(b)$ of length $2^b - 1$, i.e., the dual of the Hamming code of blocklength $2^b - 1$. For $X \in \text{GF}(q)$,

$$\Phi_E(X) \triangleq \Phi_B(X) \otimes_2 \mathbf{G}_S(b)$$

where $\mathbf{G}_S(b)$ is the generator matrix of the simplex code of size $b \times (2^b - 1)$. By linearity of Φ_B , $\Phi_E(X_1 + X_2) = \Phi_E(X_1) \oplus_2 \Phi_E(X_2)$. Note that the columns of $\mathbf{G}_S(b)$ are all vectors of weight $1 \leq w \leq b$. We will let the i^{th} column of $\mathbf{G}_S(b)$ be $(g_{1,i}, g_{2,i}, \dots, g_{b,i})^\top$ where $i = \sum_{j=1}^b g_{j,i} 2^{j-1}$ for $i \in [2^b - 1]$. We use the mapping Φ_E to map row-vector of symbols $\underline{X} = (X_1, \dots, X_{n_q}) \in \text{GF}(q)^{n_q}$ to a binary row-vector $\Phi_E(\underline{X}) \triangleq (\Phi_E(X_1), \dots, \Phi_E(X_{n_q}))$ of length $(2^b - 1)n_q$. The ordering of the columns of $\mathbf{G}_S(b)$ implies that for $i \in [n_q]$, $j \in [b]$

$$(\Phi_B(X_i))_j = (\Phi_E(X_i))_{2^{j-1}}, \quad (6.1)$$

where we write $(\Phi_B(X))_e$ to denote the e^{th} element of the vector representing $\Phi_B(X)$ for $X \in \text{GF}(q)$. The *extended binary image* of $\mathcal{C}_q$ is defined as the set $\mathcal{C}_E \subset \text{GF}(2)^{(2^b-1)n_q}$ where $\mathcal{C}_E = \{\Phi_E(\underline{X}) : \underline{X} \in \mathcal{C}_q\}$. For a fixed Φ_E , one can define a mapping $\Psi_M : \text{GF}(q) \mapsto \mathcal{M}_{2^b-1}$ such that

$$\Psi_M(h) \otimes_2 \Phi_E(X)^\top = \Phi_E(hX)^\top \quad \forall h, X \in \text{GF}(q).$$

Then the following can be shown.

Lemma 6.3.1. $\Psi_M(h)$ is a permutation matrix for all $h \in \text{GF}(q)$.

The matrix $\mathbf{H}_{PM}$ of dimension $m_q(2^b - 1) \times n_q(2^b - 1)$ is defined as one obtained by replacing each element of $\mathbf{H}_q$ by its image under Ψ_M . Then, the Tanner graph of $\mathbf{H}_{PM}$ is a *graph cover* [5] of $\mathcal{G}_q$ with all edge labels equal to 1. It is easy to show that the extended binary image is

$$\mathcal{C}_E = \{\underline{x} \in \text{GF}(2)^{(2^b-1)n_q} : \mathbf{H}_{PM} \otimes_2 \underline{x}^\top = \underline{0}^\top\} \cap \mathcal{C}_S(b)^{n_q}$$

where

$$\mathcal{C}_S(b)^{n_q} = \underbrace{\mathcal{C}_S(b) \times \dots \times \mathcal{C}_S(b)}_{n_q} \subset \text{GF}(2)^{(2^b-1)n_q}.$$

The overall parity-check matrix for the extended binary code is therefore of the form

$$\mathbf{H}_E = \begin{pmatrix} \mathbf{H}_{PM} \\ \mathbf{I}_{n_q} \odot \mathbf{H}_S(b) \end{pmatrix}$$

where $\mathbf{I}_{n_q}$ is the $(n_q \times n_q)$ identity matrix, $\mathbf{H}_S(b)$ is the parity-check matrix of the simplex code and $\odot$ represents the *Kronecker product*.

As a consequence of Equation (6.1), transmitting the codewords of $\mathcal{C}_B$ amounts to transmitting bits indexed by the set $\mathcal{I}_t = \{i \in [(2^b - 1)n_q] : i = 2^j \bmod (2^b - 1), j \in \{0, 1, \dots, b - 1\}\}$ of the corresponding codewords in $\mathcal{C}_E$ and puncturing the rest. We will let $\mathcal{I}_p = [(2^b - 1)n_q] \setminus \mathcal{I}_t$ denote the index set of punctured bits in $\mathcal{C}_E$. For ease of notation, we will define $x_{i,j}^B \triangleq (\Phi_B(X_i))_j$ and $x_{i,j}^E \triangleq (\Phi_E(X_i))_j$.

Example 6.3.1. *Let us consider a code over $\text{GF}(8)$ given by the following parity-check matrix $\mathbf{H}_q = (\alpha \quad \alpha^2 \quad 1)$, where α is a primitive element of $\text{GF}(8)$ with the primitive polynomial being $x^3 + x + 1 = 0$. The code may be represented as solutions to the following linear constraint $\alpha X_1 + \alpha^2 X_2 + X_3 = 0$. Table 6.1 gives the chosen mapping Φ_B . Then*

Table 6.1: Mapping $\Phi_B : \text{GF}(8) \mapsto \text{GF}(2)^3$

$X \in \text{GF}(8)$	$\Phi_B(X)$
0	(0, 0, 0)
α	(0, 0, 1)
α^3	(0, 1, 0)
α^6	(1, 0, 0)
α^2	(1, 1, 1)
α^4	(1, 1, 0)
α^5	(1, 0, 1)
α^7	(0, 1, 1)

the parity equations for the basic binary code can be written as

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{1,1}^B \\ x_{1,2}^B \\ x_{1,3}^B \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_{2,1}^B \\ x_{2,2}^B \\ x_{2,3}^B \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{3,1}^B \\ x_{3,2}^B \\ x_{3,3}^B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \text{ i.e.,}$$

$$(x_{1,2}^B + x_{1,3}^B) + x_{2,2}^B + x_{3,1}^B = 0, \quad (6.2)$$

$$(x_{1,1}^B + x_{1,2}^B + x_{1,3}^B) + x_{2,3}^B + x_{3,2}^B = 0, \quad (6.3)$$

$$(x_{1,1}^B + x_{1,3}^B) + (x_{2,1}^B + x_{2,2}^B) + x_{3,3}^B = 0. \quad (6.4)$$

The parity-check equation for the extended binary image can be written as

$$\Psi_M(\alpha)\Phi_E(X_1) + \Psi_M(\alpha^2)\Phi_E(X_2) + \Psi_M(1)\Phi_E(X_3) = 0.$$

Thus, the seven binary parity-check equations represented by the equation above are represented by $\mathbf{H}_{PM}$ given by

	$(\Phi_E(X_1))_j$							$(\Phi_E(X_2))_j$							$(\Phi_E(X_3))_j$						
j	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
P_1						1		1							1						
P_2							1			1						1					
P_3	1												1				1				
P_4					1					1								1			
P_5			1					1											1		
P_6		1											1							1	
P_7					1							1									1

Here, equation P_1 is $x_{1,6}^E + x_{2,2}^E + x_{3,1}^E = 0$ which together with the simplex constraints $x_{1,6}^E = x_{1,2}^B + x_{1,3}^B$, $x_{2,2}^E = x_{2,2}^B$ and $x_{3,1}^E = x_{3,1}^B$ can be written as Equation (6.2). Similarly, equations P_2 and P_4 correspond to Equations (6.3) and (6.4) respectively. Equations P_3, P_5, P_6, P_7 are all linear combinations of the above equations.

In [3], it was shown that the extended binary image was equivalent to the non-binary code from which it was constructed. But this equivalence was established for the BEC with BP updates at the parity-check nodes and ML updates at the simplex nodes. In order to achieve this equivalence with BP, we will assume that the parity-check matrices representing the simplex codes have redundant parity-checks assuring ML performance with BP. For the simplex codes, this is achieved by forming the parity-check matrix with rows of weight 3 [6]. Hence, $\mathbf{H}_S(b)$ is the parity-check matrix of size $(2^b - 1)(2^{b-1} - 1) \times (2^b - 1)$ for the simplex code. We will denote BP over the Tanner graph of $\mathbf{H}_E$ with $\mathbf{H}_S(b)$ having redundant checks and when only bits indexed by $\mathcal{I}_t$ are transmitted as $\text{BP}(\mathbf{H}_E)$.

6.4 Superiority of $\text{BP}(\mathbf{H}_q)$ over $\text{BP}(\mathbf{H}_B)$

From the construction of the basic binary image and the extended binary image, it is clear that for each parity-check equation in $\mathbf{H}_q$, there are b parity-check equations in $\mathbf{H}_B$ and $(2^b - 1)$ parity-check equations in the $\mathbf{H}_{PM}$ portion of $\mathbf{H}_E$.

Lemma 6.4.1. *For each row $\underline{h}_q$ in the parity-check matrix $\mathbf{H}_q$, the parity-check matrix $\mathbf{H}_E$ contains all non-trivial linear combinations of the b rows corresponding to $\underline{h}_q$ in the parity-check matrix $\mathbf{H}_B$.*

Discussion: First, note that the extra variables in the extended binary image were defined such that all non-trivial linear combinations of the transmitted bits were represented by a variable and these linear relations were maintained through the use of the simplex code for each non-binary symbol. Second, observe that by using the permutation matrices, we obtained a set of $(2^b - 1)$ parity-check equations each involving exactly one of the variables from each simplex codeword. Putting these two together, we can show the above result.

Since it is known that the performance of $\text{BP}(\mathbf{H}_q)$ is the same as that of $\text{BP}(\mathbf{H}_E)$, it suffices to show that $\text{BP}(\mathbf{H}_E)$ has superior performance in comparison with $\text{BP}(\mathbf{H}_B)$. The performance of codes over the BEC with BP decoding is determined by the *stopping sets* of the Tanner graph of their parity-check matrix [7]. By an *erasure pattern* $\mathcal{E}$, we mean the index set of the erasures in a vector. For a binary parity-check matrix $\mathbf{H}$, let $\beta(\mathbf{H}, \mathcal{E})$ denote the erasure pattern obtained at the end of decoding the received word with erasure pattern $\mathcal{E}$ using BP on the Tanner graph of $\mathbf{H}$. Then a stopping set is an erasure pattern $\mathcal{E}$ such that $\beta(\mathbf{H}, \mathcal{E}) = \mathcal{E}$. Let $\mathbb{S}(\mathbf{H}) = \{\mathcal{E} : \beta(\mathbf{H}, \mathcal{E}) = \mathcal{E}\}$ be the set of stopping sets of $\mathbf{H}$. Notice that this condition is the same as the graph-theoretic requirement that every check node neighbor of variables in the Tanner graph of $\mathbf{H}$ indexed by $\mathcal{E}$ be connected to these variables at least twice. However, with the extended binary image, we are interested in erasure patterns only among the transmitted variables. Hence we do not have a corresponding graph-theoretic definition of stopping sets of $\text{BP}(\mathbf{H}_E)$.

Definition 6.4.2 (Stopping Sets of $\text{BP}(\mathbf{H}_E)$). *Consider BP decoding of the extended binary code over the BEC with the parity-check matrix $\mathbf{H}_E$. A stopping set, $\mathcal{E}$, is a subset*

of the index set of transmitted bits such that BP decoding can recover no transmitted bit in $\mathcal{E}$, i.e.,

$$\mathcal{E} \subset \mathcal{I}_t \text{ such that } \beta(\mathbf{H}_E, \mathcal{E} \cup \mathcal{I}_p) \cap \mathcal{I}_t = \mathcal{E}.$$

Discussion: Let us denote by $\mathbb{S}_E$ the set of all stopping sets of the extended binary image as defined above. Let $\mathbb{S}(\mathbf{H}_E)$ denote the set of all stopping sets of the code with parity-check matrix $\mathbf{H}_E$ assuming all bits are transmitted. Define

$$\mathbb{S}^t(\mathbf{H}_E) = \{\mathcal{S} \cap \mathcal{I}_t : \mathcal{S} \in \mathbb{S}(\mathbf{H}_E)\}.$$

Then, $\mathbb{S}_E = \mathbb{S}^t(\mathbf{H}_E)$. This is because for each $\mathcal{S} \in \mathbb{S}^t(\mathbf{H}_E)$, $\beta(\mathbf{H}_E, \mathcal{S})$ is the maximal stopping set $\mathcal{S}' \in \mathbb{S}(\mathbf{H}_E)$ such that $\mathcal{S}' \cap \mathcal{I}_t = \mathcal{S}$. Note that for the basic binary image, the set of all stopping sets is just the normal definition $\mathbb{S}_B = \mathbb{S}(\mathbf{H}_B)$ since every bit of the basic binary image is transmitted.

Proposition 6.4.3 (BP($\mathbf{H}_E$) better than BP($\mathbf{H}_B$)). $\mathbb{S}_E \subset \mathbb{S}_B$.

Proof. Suppose to the contrary that $\exists \mathcal{S} \in \mathbb{S}_E \setminus \mathbb{S}_B$. Then, there is a parity-check equation in $\mathbf{H}_B$ that involves only one erasure, say $x_{i,e}^B = (\Phi_B(X_i))_e$. Let this equation be

$$\begin{aligned} \sum_{j \in \mathbb{I}_i} (\Phi_B(X_i))_j + \sum_{k \in \mathbb{I}} \sum_{j \in \mathbb{I}_k} (\Phi_B(X_k))_j &= 0 \\ \Rightarrow \sum_{j \in \mathbb{I}_i} x_{i,j}^B + \sum_{k \in \mathbb{I}} \sum_{j \in \mathbb{I}_k} x_{k,j}^B &= 0 \end{aligned} \tag{6.5}$$

for some index sets $\mathbb{I}_i, \mathbb{I}_k \subset [b]$, $\forall k \in \mathbb{I}, \mathbb{I} \subset [n_q] \setminus \{i\}$ and $e \in \mathbb{I}_i$. We know from the construction of the extended binary image that there exist variables

$$\begin{aligned} x_{k,\Sigma_k}^E &\triangleq (\Phi_E(X_k))_{\sum_{j \in \mathbb{I}_k} 2^{j-1}} = \sum_{j \in \mathbb{I}_k} (\Phi_B(X_k))_j \\ &= \sum_{j \in \mathbb{I}_k} (\Phi_E(X_k))_{2^{j-1}}, k \in \mathbb{I} \cup \{i\}. \end{aligned}$$

From Lemma 6.4.1, we also know that the equation

$$x_{i,\Sigma_i}^E + \sum_{k \in \mathbb{I}} x_{k,\Sigma_k}^E = 0 \quad (6.6)$$

is contained in $\mathbf{H}_{PM}$. Since every bit except $x_{i,e}^B$ was known in Equation (6.5), every bit except x_{i,Σ_i}^E is known in Equation (6.6). Hence the extended binary image can solve for x_{i,Σ_i}^E . But the simplex portion of $\mathbf{H}_E$ contains the equation

$$x_{i,\Sigma_i}^E = \sum_{j \in \mathbb{I}_i} x_{i,2^{j-1}}^E = \sum_{j \in \mathbb{I}_i} (\Phi_B(X_i))_j$$

which has only one unknown $x_{i,2^{e-1}}^E = x_{i,e}^B$, which can also be solved for. This is a contradiction to the assumption that $\mathcal{S} \in \mathbb{S}_E$. Therefore, $\mathbb{S}_E \setminus \mathbb{S}_B = \emptyset$. $\square$

Example 6.4.1. Consider the code in Example 6.3.1 and let the received word be (?0? 0?0 000), where ? denotes an erasure. It is easy to see that this received word corresponds to a stopping set of the basic binary image. However, the extended binary image can recover all erasures. Using P_6 we can recover $x_{2,7}^E$, then using the simplex equation $x_{2,7}^E = x_{2,1}^E + x_{2,2}^E + x_{2,4}^E$ we obtain $x_{2,2}^E$. In turn, $x_{2,i}^E \forall i \in \{1, \dots, 7\}$ can be obtained. Finally we can use P_3 and P_7 to recover $x_{1,1}^E$ and $x_{1,4}^E$.

Note that although we have only shown improper containment of $\mathbb{S}_E$ in the set of stopping sets of $\mathbf{H}_B$, in most cases this containment is proper, i.e., $\mathbb{S}_E \subsetneq \mathbb{S}_B$. One case where $\mathbb{S}_E = \mathbb{S}_B$ is when the Tanner graph corresponding to $\mathbf{H}_B$ is cycle-free.

Proposition 6.4.3 gives us an analytical insight into why non-binary codes perform well. However, since the superiority is established only in comparison with the basic binary image corresponding to the non-binary code, the result constitutes only a partial answer to why non-binary codes are better than their binary counterparts in general.

6.5 Enhancing $\mathcal{C}_B$ using Redundant Parities

It is known that the performance of a linear code over the BEC with BP can be improved by using a parity-check matrix with redundant parity-checks (RPCs). In fact, by adding enough parity-checks, we can guarantee ML performance with BP (See

[6] and references therein). A similar notion is that of *stopping redundancy* [8], where RPCs are added to remove stopping sets of size smaller than the minimum distance of the code. In the same spirit, we pose the question whether it is possible to achieve the performance of a non-binary code by its basic binary image with some RPCs.

Proposition 6.5.1. *For every $\mathcal{S} \in \mathbb{S}_B \setminus \mathbb{S}_E$, there exists an RPC for the basic binary image, $\underline{h}_{\mathcal{S}} \otimes_2 \Phi_B(\underline{X})^\top = \underline{0}$, such that $\mathcal{S} \notin \mathbb{S}(\hat{\mathbf{H}}_B)$ where $\hat{\mathbf{H}}_B^\top = (\mathbf{H}_B^\top | \underline{h}_{\mathcal{S}}^\top)$.*

Proof. Since $\mathcal{S} \notin \mathbb{S}_E$, the BP decoder working over $\mathbf{H}_E$ can solve for some erased transmitted bit $x_{i,2^e-1}^E = x_{i,e}^B$. This implies that the ML decoder working with $\mathbf{H}_B$ can also solve for $x_{i,e}^B$. Since ML decoding over the BEC is the same as Gaussian elimination, the above means that there exists a linear combination of the parity-check equations in $\mathbf{H}_B$ which has only $x_{i,e}^B$ as the unknown, which can be set as $\underline{h}_{\mathcal{S}}$. $\square$

Note that since there might be multiple linear combinations of parity-check equations of $\mathbf{H}_B$ that can solve for $x_{i,e}^B$, the RPC $\underline{h}_{\mathcal{S}}$ is not always unique.

We now describe an algorithm that finds the RPC $\underline{h}_{\mathcal{S}}$, pseudo-code for which is given as Algorithm 6.1. Given $\mathcal{S}$, the algorithm starts the peeling decoding over $\mathbf{H}_E$ with the erasure pattern $\mathcal{S} \cup \mathcal{I}_p$. The decoder maintains a list of all the punctured bits that were solved and the unerased transmitted bits that were used to solve them. When a transmitted but erased bit is solved, the decoder is terminated and the algorithm finds an equation involving only this recovered transmitted bit and some unerased transmitted bits chosen based on the punctured bits that appear in the computation tree for this recovered transmitted bit. Thus, for a given stopping set, the algorithm uses a partial peeling decoding attempt over the extended binary image and a traversal through the computation tree of a recoverable transmitted bit to obtain the corresponding RPC of $\mathbf{H}_B$. For a given collection of stopping sets, this algorithm is first run for low-weight stopping sets since RPCs that eliminate low-weight stopping sets may also eliminate higher weight stopping sets containing those low-weight stopping sets as a subset. It is possible to optimize the choice of RPCs to minimize the degrees of the additional check nodes introduced by them. However, this optimization was not considered in the implementation used for this paper. Note that the idea here is different from the one in [9] where the authors try to optimize the set of transmitted bits under the assumption

Algorithm 6.1 Algorithm to find $\underline{h}_S$ given $S \in \mathbb{S}_B \setminus \mathbb{S}_E$

```

1:  $\mathcal{R} \leftarrow \emptyset, \mathcal{L} \leftarrow \emptyset, \underline{h}_S \leftarrow \underline{0}$ 
2: while  $\exists$  a row  $\underline{h}_j^E \in \mathbf{H}_E$  that solves  $(\Phi_E(\underline{X}))_i$  do
3:    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(i, j)\}$ 
4:   if  $i \in \mathcal{I}_t$  then
5:      $\mathcal{L} \leftarrow \{(i, j)\}$ 
6:     while  $\mathcal{L} \neq \emptyset$  do
7:        $(i, j) \leftarrow \text{pop}(\mathcal{L})$ 
8:       Let  $\mathbb{I}_j \subset [(2^b - 1)n_q] : \sum_{k \in \mathbb{I}_j} \underline{u}_{k, (2^b - 1)n_q} = \underline{h}_j^E$ 
9:       for all  $i' \in \mathbb{I}_j \cap \mathcal{I}_p \setminus \{i\}$  do
10:        Locate  $(i', j') \in \mathcal{R}$ 
11:        push( $\mathcal{L}, (i', j')$ )
12:       end for
13:       for all  $k \in \mathbb{I}_j \cap \mathcal{I}_t$  do
14:         $\sigma_k \leftarrow \lfloor \frac{k}{2^b - 1} \rfloor b + (k \bmod (2^b - 1))$ 
15:         $\underline{h}_S \leftarrow \underline{h}_S \oplus_2 \underline{u}_{\sigma_k, bn_q}$ 
16:       end for
17:     end while
18:   return  $\underline{h}_S$ 
19: end if
20: end while

```

that bits of the extended binary image indexed by a subset of $\mathcal{I}_p$ are also transmitted to obtain a lower rate code that performs better.

Tables 6.2 and 6.3 list the number of codewords and number of stopping sets for two non-binary LDPC codes, $\mathcal{C}_4$ and $\mathcal{C}_8$, and their binary images. Tanner graph for $\mathcal{C}_4$ is constructed randomly with parameters $q = 2^2, n_q = 96, k_q = 32$, and $d_l = 2, d_r = 3$. Tanner graph for $\mathcal{C}_8$ is constructed using Progressive Edge-Growth (PEG) [10] with parameters $q = 2^3, n_q = 100, k_q = 50$, and $d_l = 2, d_r \simeq 4$. Let A_B^w denote the number of codewords of weight w in the basic binary image $\mathcal{C}_B$. Let $\mathbb{S}^w(\mathbf{H})$ denote the set of stopping sets of $\mathbf{H}$ of weight w , $\mathbb{S}_E^w$ the set of stopping sets of the extended binary image of weight w . For $\mathcal{C}_4$, RPCs were added to its basic binary image to remove extra stopping sets of weights up to 12 in $\mathbf{H}_B^{12}$ and weights up to 15 in $\mathbf{H}_B^{15}$ while, for $\mathcal{C}_8$, RPCs were added to remove extra stopping sets of weights up to 8 in $\mathbf{H}_B^8$ and weights up to 10 in $\mathbf{H}_B^{10}$. The stopping sets $\mathbb{S}_B = \mathbb{S}(\mathbf{H}_B)$ were found using the algorithm in [11] and whether these were in $\mathbb{S}_E$ was verified by running the BP decoder over $\mathbf{H}_E$. For those stopping sets in $\mathbb{S}_B \setminus \mathbb{S}_E$, the RPCs were found using the algorithm described earlier in

this section. For $\mathcal{C}_4$, the basic binary image has 192 variable nodes and 128 check nodes, while the number of RPCs in $\mathbf{H}_B^{12}$ and $\mathbf{H}_B^{15}$ are 93 and 154, respectively. For comparison, the extended binary image has 288 variable nodes and 288 check nodes. Similarly, the basic binary image for $\mathcal{C}_8$ has 300 variable nodes and 150 check nodes, while the number of RPCs in $\mathbf{H}_B^8$ and $\mathbf{H}_B^{10}$ is 60 and 185, respectively. The number of variable and check nodes in the extended binary image for $\mathcal{C}_8$ is 700 and 1050, respectively. In general, for a NB-LDPC code over $\text{GF}(q)$, the number of variable and check nodes in the basic binary image is $O(\log q)$ while it is $O(q)$ in the extended binary image. The number of variables nodes in the enhanced basic binary image is the same as basic binary image, while the number of check nodes is the sum of the number of check nodes in the basic binary image and the additional RPCs added.

Table 6.2: Stopping set weights for a $\text{GF}(4)$ code, $\mathcal{C}_4$, and its binary images

w	A_B^w	$ \mathbb{S}_E^w $	$ \mathbb{S}^w(\mathbf{H}_B) $	$ \mathbb{S}^w(\mathbf{H}_B^{12}) $	$ \mathbb{S}^w(\mathbf{H}_B^{15}) $
≤ 4	0	0	0	0	0
5	4	4	6	4	4
6	2	2	11	2	2
7	7	7	31	7	7
8	7	9	50	9	9
9	3	4	86	4	4
10	12	23	171	23	23
11	22	30	343	30	30
12	63	80	873	80	80
13	87	120	2199	151	120
14	122	204	5463	287	204
15	205	418	13891	650	418
16	369	793	35209	1322	821

For the codes under consideration, Figures 6.1 and 6.2 plot the BP performance of the non-binary codes, their basic binary images, and the enhanced binary images. The improvement in performance with RPCs is evident in the figures. The FERs of the non-binary code and the enhanced binary image are close to each other, and the corresponding BERs are also comparable.

Table 6.3: Stopping set weights for a GF(8) code, $\mathcal{C}_8$, and its binary images

w	A_B^w	$ \mathbb{S}_E^w $	$ \mathbb{S}^w(\mathbf{H}_B) $	$ \mathbb{S}^w(\mathbf{H}_B^8) $	$ \mathbb{S}^w(\mathbf{H}_B^{10}) $
≤ 2	0	0	0	0	0
3	0	0	9	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	1	1	59	1	1
7	5	5	82	5	5
8	13	13	509	13	13
9	38	38	2781	156	38
10	64	64	11763	612	64
11	143	147	45310	2102	310
12	309	358	169120	6363	871
13	799	970	663617	19346	2575
14	1906	2525	2727519	62032	7893

6.6 Concluding Remarks

We showed that when the transmission of a non-binary LDPC codeword occurs over the BEC, BP decoding over the non-binary graph has a better performance than BP decoding over the Tanner graph of the basic binary image of the code. We proposed an algorithm to efficiently find effective redundant parity-checks for the basic binary image by observation of the BP decoding iterations of the extended binary image. Through numerical results and simulations, the effectiveness of the proposed algorithm was established. Obtaining bounds on the number of RPCs of the basic binary image needed to achieve the same performance as BP over non-binary code would be of interest. Similar characterization of the reasons for the superiority of non-binary codes over other channels involving errors as well as erasures will give further insight on designing strong codes and efficient decoding algorithms for such channels.

Acknowledgements

This chapter is a reprint of the paper “Enhancing Binary Images of Non-Binary LDPC Codes,” Aman Bhatia, Aravind R. Iyengar, Paul H. Siegel; published in Proc. IEEE Globecom, Houston, Texas, December 5-9, 2011, pp. 1–6. The dissertation author

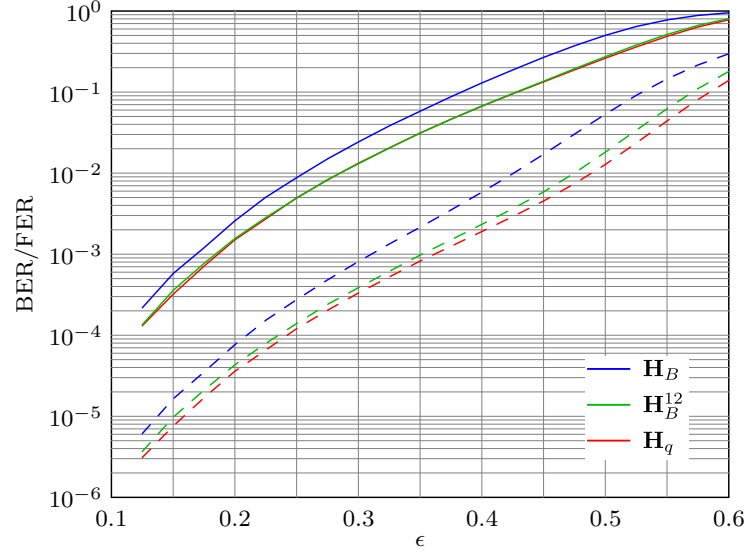


Figure 6.1: Bit (dashed curves) and frame (solid curves) error rates for the random non-binary code $\mathcal{C}_4$ with parameters $q = 2^2, n_q = 96, k_q = 32, d_l = 2, d_r = 3$ and its binary images considered in Table 6.2.

was the primary investigator and author of this paper, and co-authors have approved the use of the material for this dissertation.

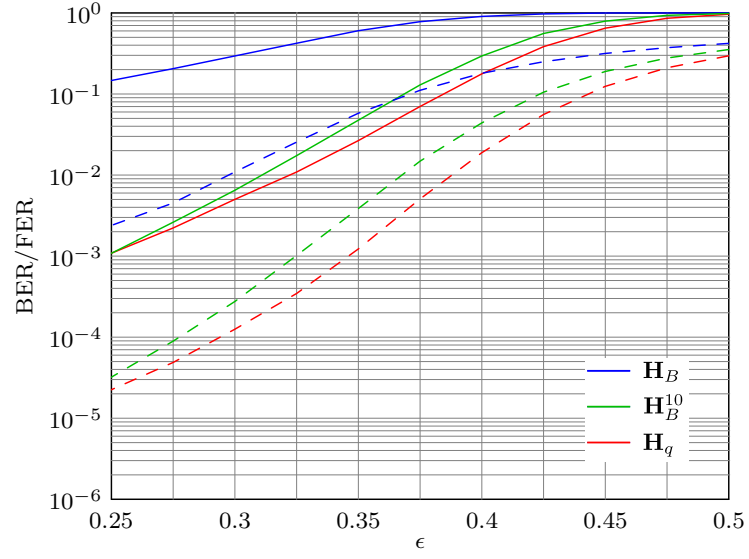


Figure 6.2: Bit (dashed curves) and frame (solid curves) error rates for the PEG constructed non-binary code $\mathcal{C}_8$ with parameters $q = 2^3, n_q = 100, k_q = 50, d_l = 2, d_r \simeq 4$ and its binary images considered in Table 6.3.

Bibliography

- [1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, Massachusetts: MIT Press, 1963.
- [2] M. Davey and D. MacKay, “Low density parity check codes over $GF(q)$,” *Proc. IEEE Information Theory Workshop*, Kilarney, Ireland, Jun. 22 - 26, 1998, pp. 70–71.
- [3] V. Savin, “Binary linear-time erasure decoding for non-binary LDPC codes,” in *2009 IEEE Information Theory Workshop*, Taormina, Italy, Oct. 11 - 16, 2009, pp. 258–262.
- [4] V. Rathi and R. Urbanke, “Density evolution, thresholds and the stability condition for non-binary LDPC codes,” *IEE Proceedings on Communications*, Dec. 9, 2005, vol. 152, no. 6, pp. 1069–1074.
- [5] T. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.
- [6] J. Han and P. Siegel, “On ML redundancy of codes,” *Proc. IEEE International Symposium on Information Theory*, Toronto, ON, Canada, Jul. 6 - 11, 2008, pp. 280–284.
- [7] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, “Finite-length analysis of low-density parity-check codes on the binary erasure channel,” *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.
- [8] M. Schwartz and A. Vardy, “On the stopping distance and the stopping redundancy of codes,” *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 922–932, Mar. 2006.
- [9] L. P. Sy, V. Savin, and D. Declercq, “Extended non-binary low-density parity-check codes over erasure channels,” *CoRR*, vol. abs/1103.2691, 2011.
- [10] X. -Y. Hu, E. Eleftheriou, and D. M. Arnold, “Regular and irregular progressive edge-growth tanner graphs,” *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
- [11] E. Rosnes and O. Ytrehus, “An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices,” *IEEE Transaction on Information Theory*, vol. 55, no. 9, pp. 4167–4178, Sep. 2009.