UNIVERSITY OF CALIFORNIA
Los Angeles

Coding Schemes to Approach Capacity in Short Blocklength with Feedback and LDPC Coding for Flash Memory

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

by

Kasra Vakilinia

2016

ABSTRACT OF THE DISSERTATION

Coding Schemes to Approach Capacity in Short Blocklength with Feedback and LDPC Coding for Flash Memory

by

Kasra Vakilinia

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2016

Professor Richard D. Wesel, Chair

This dissertation mainly focuses on two different branches of coding theory and its applications:1) coding to approach capacity in short blocklengths using feedback and 2) LDPC coding for Flash memory systems. In the first area, we study the benefits that feedback with incremental redundancy can provide to increase the maximum achievable rate in communication systems, using carefully designed adaptive non-binary LDPC codes. We show how to achieve over 90% of the idealized throughput of rate-compatible sphere-packing with maximum-likelihood decoding (RCSP-ML) for average blocklengths of 150-450 bits. This is important because it illustrates that feedback greatly reduces the number of transmitted symbols required to achieve near-capacity performance.

We then extend these ideas to feedback systems where the number of incremental transmissions is limited. In order to optimize the blocklengths for each incremental transmission we formulate an integer optimization problem involving an approximation based on the inverse-Gaussian p.d.f., the distribution of the blocklength required for successful decoding. The brute-force approach to solve this computationally complex optimization problem quickly becomes infeasible. In order to solve this problem efficiently, we introduce sequential differential optimization (SDO) algorithm that has only linear complexity to identify optimal incremental transmission lengths. The results obtained from SDO are negligibly different from the exponentially complex exhaustive-search solution. By using the optimized incremental transmission lengths (with an average blocklength of less than 500 bits), non-binary LDPC codes achieve a throughput greater than 90% of the capacity with a

two-phase scheme. Furthermore, we extend these ideas to the case of using cyclic redundancy checks (CRC). With CRC, even better performance in the blocklength range of about 500 bits is obtainable. The overhead associated with a CRC prevents great performance in short blocklength regime (fewer than 400 bits). We also extend these ideas to systems with larger constellations operating at a higher signal-to-noise ratio (SNR).

Another incremental transmission coding scheme studied in this dissertation focuses on design and use of rate-compatible protograph-based raptor-like (PBRL) LDPC codes with various blocklengths and rates that can be used in feedback systems over additive-white Gaussian noise (AWGN) channels. The codes proposed in this work use X-OR operations and density evolution to produce additional degree-one parity bits providing extensive rate compatibility. The protographs are also carefully lifted to avoid undesirable graphical structures such as problematic stopping sets. For a target frame error rate of 10^{-5}, at each rate the k = 1032 and k = 16384 code families perform within 1 dB and 0.4 dB, respectively, of both the Gallager bound and the normal approximation. The k = 16384 code family outperforms the best known standardized code family, the AR4JA and longer DVB-S2 codes.

We extend the ideas in design of PBRL codes from AWGN channels to binary symmetric channels (BSC) and binary erasure channels (BEC). We introduce two fast and efficient algorithms to calculate the threshold of LDPC codes used over BSC. These algorithms serve as alternatives to the quite complex density evolution algorithm. Since these new algorithms are quite fast, we use them to design PBRL LDPC codes for BSC.

To explore the advantage of feedback in conjunction with other modern coding schemes, in this work we use an extension of reciprocal channel approximation (RCA) to accurately and efficiently predict the frame error rate (FER) performance of polar codes by analyzing the probability density function (p.d.f.) of the log-likelihood ratio (LLR) values associated with information bits. The preliminary results show that a feedback scheme in conjunction with a repetition coding system significantly reduces the blocklength required to achieve a target FER. For example, using a rate-0.5 128-bit polar code as the initially transmitted code, the theoretical analysis verified by simulation shows a 16-fold reduction in blocklength with only about 7.4% of overhead in for-

ward channel transmissions. We also make an improvement to this feedback coding scheme which reduces the overhead to almost 3% for a similar FER performance gain.

The second part of this dissertation focuses on design of binary and non-binary LDPC codes for Flash memory systems. Usually the FER requirements in Flash memory systems is more strict than wireless communication systems. In order to improve the error correction capability of the codes used in Flash memory systems, sometimes the same memory cell is read multiple times. In this dissertation, we study the coding gain from multiple reads of the same Flash memory cell with distinct word-line voltages. The subsequent additional reads provide enhanced precision for LDPC decoding. We identify a trade-off in LDPC code design when decoding is performed with multiple precision levels and conclude that the best code at one level of precision is typically not be the best code at a different level of precision.

By studying the trade-off in LDPC code design by using extrinsic-information-transfer (EXIT)-function analysis employing the reciprocal channel approximation (RCA), we obtain the optimal LDPC code degree distributions for initial hard decoding (one-bit quantization of the channel output) and for decoding with the soft information provided by subsequent additional reads in both SLC (two-level cell) and MLC (four-level-cell) Flash memory. The results indicate that design for hard decoding can provide irregular degree distributions that have good thresholds across a range of possible decoding precisions. Finally, we illustrate that the MMI optimization of word-line voltages for five reads is a quasi-convex problem for the Gaussian model of SLC Flash.

The dissertation of Kasra Vakilinia is approved.

Dariush Divsalar

Adnan Darwiche

Lieven Vandenberghe

Lara Dolecek

Richard D. Wesel, Committee Chair

University of California, Los Angeles

2016

To my parents, Susan and Alireza and my siblings, Abtin and Sanam.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I owe my gratitude to the faculty members of the electrical engineering department at UCLA, my family members, friends, and many other great people who have helped me complete this dissertation. I would like to take this opportunity to express my gratitude to these people for their support and assistance. Among these people, several individuals, especially, the members of my Ph.D. committee, Professor Richard Wesel, Professor Dariush Divsalar, Professor Lara Dolecek, Professor Lieven Vandenberghe, and Professor Adnan Darwiche deserve special mention for their contributions to this dissertation.

Foremost, I would like to express my sincere gratitude to my advisor, Professor Richard Wesel, for his help, support, patience, motivation, enthusiasm, and knowledge. I am also very much grateful to him for teaching me how to conduct research and giving me the opportunity to work on different exciting projects at UCLA Communications Systems Laboratory (CSL). During my Ph.D. studies, Professor Wesel taught me how to think about and approach research problems. I really appreciate his comments on my work for his detailed attention to the arguments and also because he was able to find ways to improve it. Professor Wesel also helped me immeasurably in my professional socialization. He has always taken time to introduce me to people within the discipline and has always been a strong advocate for me. I could not have imagined having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I have been especially fortunate to be supervised by Professor Dariush Divsalar. His concern for graduate training and the future of the discipline and his strong opinions and his willingness to share them have been great assets to me and to many others graduate students at UCLA. Professor Divsalar has always been there to have technical discussions with me and give me advice. He was always willing to engage in ideas and invest his time (which is extremely valuable and scarce) and his energy in improving our work. His guidance helped me in all time of research and writing of this dissertation. I am deeply grateful to him for the long discussions that helped me sort out the technical details of my work and for holding me to a high research standard and enforcing strict validations for research results.

I am greatly thankful to Professor Dolecek for her insightful comments and constructive criticisms at different stages of my research. Her suggestions were always thought-provoking and helped me think about new ideas. I am also grateful to Professor Dolecek for her encouragement, advice and for listening to my presentations. Her comments helped me understand and enrich the practical aspects of my research as well as teaching me how to improve my presentation when the audience is from industry.

I would like to thank Professor Vandenberghe a lot for his great lectures and insight on many topics related to optimization algorithms. I encountered quite a few optimization problems while conducting my research that required the techniques I learned in Professor Vandenberghe's course. I would also like to thank Professor Adnan Darwiche for his helps, time, and support as a member of my committee.

Most importantly, none of this would have been possible without the love and patience of my family. My family to whom this dissertation is dedicated to has been a constant source of love, concern, support and strength all these years.

Finally, I also gratefully acknowledge the institutional support that I have received while working on this dissertation. In particular, I thank the National Science Foundation and the University of California, Los Angeles for their financial support.

VITA

2015 Ph.D. Candidate in Electrical Engineering
UCLA, Los Angeles, California.

PUBLICATIONS

[1] K. Vakilinia, D. Divsalar, and R. D. Wesel, "Optimizing Transmission Lengths for Limited Feedback With Nonbinary LDPC Examples", Accepted to appear in IEEE Transactions on Communications.

[2] K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-Based Raptor-Like LDPC Codes for the Binary Erasure, Symmetric, and Mixed Channels", Submitted to IEEE Transactions on Communications.

[3] T. Y. Chen, K. Vakilinia, D. Divsalar and R. D. Wesel, "Protograph-Based Raptor-Like LDPC Codes," in IEEE Transactions on Communications, vol. 63, no. 5, pp. 1522-1532, May 2015.

[4] J. Wang, K. Vakilinia, T.-Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. D. Wesel, "Enhanced Precision Through Multiple Reads for LDPC Decoding in Flash Memories", IEEE Journal on Selected Areas in Communication, May 2014, vol. 32, no. 5, pp 880-891.

[5] K. Vakilinia, A. R. Williamson, S. V. S. Ranganathan, D. Divsalar, R. D. Proceedings: Wesel, "Feedback Systems using Non-Binary LDPC Codes with a Limited Number of Transmissions", IEEE Inf. Theory Workshop (ITW), Hobart, Tasmania, Australia. November 2 - November 5, 2014.

[6]K. Vakilinia, T.-Y. Chen, A. R. Williamson, D. Divsalar, and R. D. Wesel, “Short-Blocklength Non-Binary LDPC Codes with Feedback-Dependent Incremental Transmissions”, IEEE Int. Symp. Inf. Theory (ISIT), Honolulu, Hawaii, USA, June 29 - July 4, 2014.

[7]S. Ranganathan, K. Vakilinia, D. Divsalar,and R. D. Wesel, “Design of High-Rate Irregular Non-binary LDPC Codes Using Algorithmic Stopping-Set Cancellation”, IEEE Int. Symp. Inf. Theory (ISIT), Honolulu, Hawaii, USA, June 29 - July 4, 2014.

[8]K. Vakilinia, D. Divsalar, and R. D. Wesel, “RCA Analysis of the Polar Code LLRs and the Rate of Polarization with and without Feedback”, Submitted to IEEE Int. Symp. Inf. Theory and Applicat. (ISIT), 2015.

[9]K. Vakilinia, D. Divsalar, and R. D. Wesel, “Optimized Degree Distributions for Binary and Non-Binary LDPC Codes in Flash Memory”, IEEE Int. Symp. Inf. Theory and Applicat. (ISITA), Melbourne, Australia, October 26 - 29, 2014.

[10]K. Vakilinia, D. Divsalar, and R. D. Wesel, “Protograph-Based Raptor-Like LDPC Codes for the Binary Erasure Channel”, Inf. Theory and Applicat. (ITA), San Diego, USA, Feb 1 - 5, 2015.

[11]S. Ranganathan, K. Vakilinia, D. Divsalar, and R. D. Wesel, “Some Results on Spatially Coupled LDPC Codes”, Inf. Theory and Applicat. (ITA), San Diego, USA, Feb 1 - 5, 2016.

[12]Richard D. Wesel, K. Vakilinia, and D. Divsalar, “Resource-Aware Incremental Redundancy in Feedback and Broadcast”, Int Zurich Seminar on Comm. (IZS), Zurich, Switzerland, March 1 - 6, 2016.

CHAPTER 1

Non-Binary LDPC Codes with Feedback

This chapter presents a general approach for optimizing the number of symbols in incremental transmissions in a feedback communication system with a limited number of increments. This approach is based on a tight normal approximation on the rate for successful decoding. Applying this approach to a variety of feedback systems using non-binary (NB) low-density parity-check (LDPC) codes shows that greater than 90% of capacity can be achieved with average blocklengths fewer than 500 transmitted bits. One result is that the performance with a maximum of ten transmission and decoding attempt closely approaches the performance with an infinite number of incremental transmissions. Most of the analysis in this section focuses on binary-input additive-white Gaussian noise (BI-AWGN) channels but also demonstrates that the normal approximation works well on examples of fading channels as well as high-SNR AWGN channels that require larger QAM constellations. The analysis covers both variable-length feedback codes with termination (VLFT) and the more practical variable length feedback (VLF) codes without termination that require no assumption of noiseless transmitter confirmation. For VLF we consider both a two-phase scheme and CRC-based scheme.

1.1 Introduction

The classical results from [1] show that feedback does not increase the capacity of discrete memoryless channels. However, Polyanskiy et al. [2] and Chen et al. [3] show that capacity can be approached in a smaller number of channel uses using feedback. Polyanskiy et al. [2] introduce random-coding lower bounds for variable-length feedback coding with termination (VLFT) and

without termination (VLF), which approach capacity with average blocklengths of hundreds of bits. A communication system without feedback, on the other hand, requires thousands of bits to closely approach capacity [4]. The goal of this section is to design practical systems using non-binary low-density parity-check (NB-LDPC) codes that match or exceed the lower bounds of [2]. Most of the analysis in this section is not exclusive for NB-LDPC codes, but NB-LDPC codes are used for demonstration because they perform well in the short-blocklength regime (150 to 600 bits) that is of interest. Polyanskiy et al. [4] illustrated that in a non-feedback communication system, the maximum achievable throughput is significantly lower for short blocklengths of up to several hundred bits.

In VLFT analysis of [2], the receiver provides full noiseless feedback to the transmitter. The transmitter sends additional incremental bits until it knows the receiver has decoded the message correctly, resulting in zero probability of error. The "T" in VLFT stands for termination and corresponds to a noiseless transmitter confirmation (NTC) bit that the transmitter uses to terminate the transmission. As shown in Fig. 1.1, the NTC is transmitted through a channel different from the main communication channel experiencing an additive Gaussian noise. In contrast, VLF (without the "T") does not have the advantage of an NTC. All VLF forward transmissions go over the same noisy channel. Thus, there is a nonzero probability of undetected error in VLF.

Fig. 1.1 illustrates the system model of VLFT with the NTC transmitted after each transmission. NTC contains 1 bit of information and is used to terminate the transmission in case of successful decoding. For most of analysis in this chapter, the communication takes place over an additive-white Gaussian noise (AWGN) channel.

VLF and VLFT are examples of hybrid automatic repeat request (HARQ) schemes. Prior to Polyanskiy et al. [2] and Chen et al. [3], HARQ feedback schemes had been studied in great detail in many papers including for example [5–10]. These papers provide an overview of HARQ, discuss how error correcting codes can be combined with ARQ and demonstrate applications of HARQ. In particular, [10] shows that hybrid ARQ is especially useful in point-to-point scenarios. The coding schemes that are most commonly explored in HARQ systems [11–13] are based on convolutional codes (CCs) or a concatenation of turbo and block parity-check codes, where the

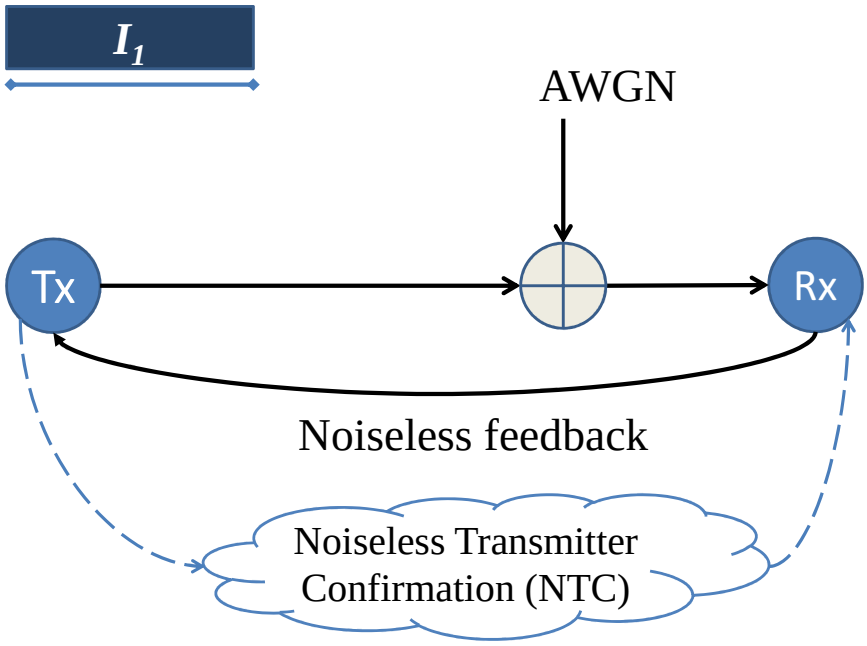


Figure 1.1: The VLFT system model with NTC transmitted after each transmission to let the receiver know if it has decoded the message correctly and if the transmission can be terminated. The transmission is over an AWGN channel and we assume noiseless feedback throughout the analysis.

Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm is used to determine which bit is unreliable and needs to be transmitted in the subsequent transmissions. These works use a genie (equivalent to NTC in VLFT) to terminate transmissions.

In order to remove the genie and realize a more practical system (equivalent to VLF) [7, 9, 14–17] consider reliability-based HARQ using convolutional codes where the transmission terminates when the probability of having a correctly decoded message is high enough. For example, in [9] the reliability metric is based on the average magnitude of the log-likelihood ratios of the source symbols.

In [18,19], Soljanin et al. study VLFT HARQ using rate-compatible binary LDPC codes. They use maximum likelihood (ML) decoding analysis to determine the size of incremental transmissions in case of decoding failure. In [20, 21] Soljanin et al. extend their analysis to time-varying binary erasure channels.

Some other high-throughput ARQ schemes use rateless spinal codes as in [22, 23], where hash functions are used for the subsequent coded symbols. In [24], Romero uses cyclic redundancy check (CRC) codes to study the performance of spinal codes in VLF setting. Use of polar codes with HARQ is also studied in [25, 26]. These works present polar-code-based HARQ schemes over binary-input additive white Gaussian noise (BI-AWGN) and Rayleigh fading channels using Chase combining.

The closest work to the analysis presented here is by Pfletschinger et al. in [27] which uses rate-adaptive, non-binary LDPC codes in a HARQ scheme over Rayleigh fading channel in the VLFT setting. They present two algorithms that use channel statistics and mutual information to optimize the blocklengths for each transmission to maximize the throughput. Based on channel state information at transmitter, the code rates, modulations, and maximum number of retransmissions are all optimized prior to initial transmission.

Chen et al. [3] and Williamson et al. [28] analyzed a VLFT scheme based on rate-compatible sphere-packing with an ML decoder (RCSP-ML). RCSP is an approximation of the performance of communication with repeated incremental redundancy and noiseless transmitter confirmation (IR-NTC). Fig. 1.2 shows how RCSP extends sphere-packing analysis from a single fixed-length code to a family of rate-compatible codes. For the ideal family of rate-compatible codes, each code in the family achieves the perfect packing and is decoded by a maximum-likelihood (ML) decoder. If the decoding is not after the incremental transmission I_i, the transmitter moves forward with transmitting the next package I_{i+1}. In each stage and after each transmission, an NTC signal in form of genie is used to either terminate the transmission or let the receiver know that the receiver has not been able to decode the original message correctly. In case of the latter scenario, the receiver expects the incremental package in the subsequent transmission.

Chen et al. [3] also simulated a VLFT scheme using tail-biting convolutional codes. The sim-

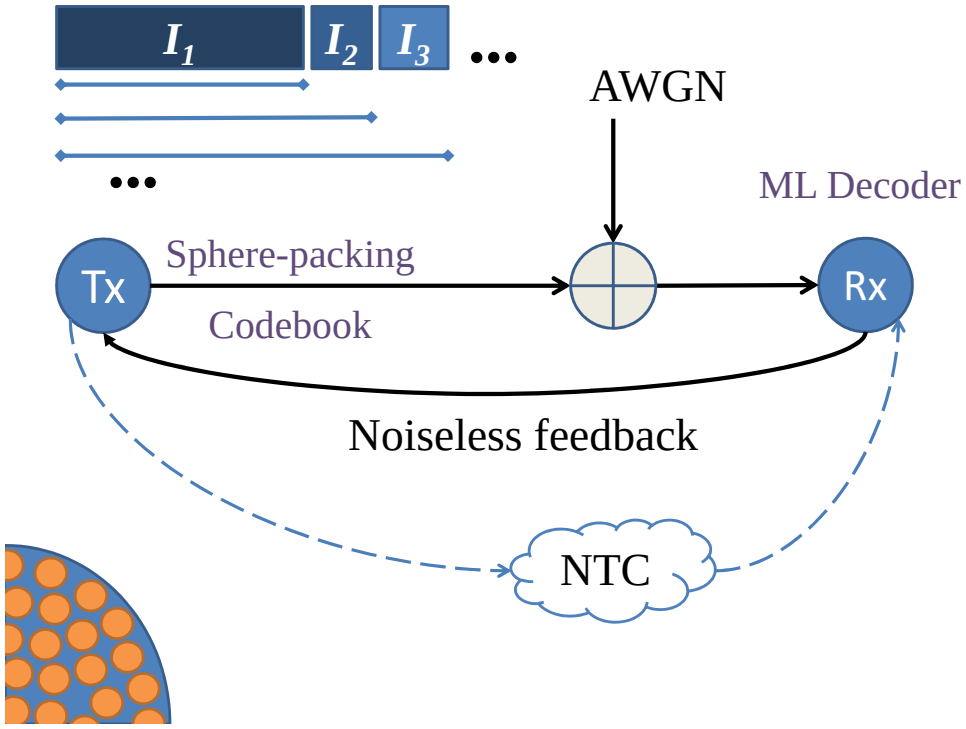


Figure 1.2: VLFT scheme based on RCSP-ML. For the ideal family of rate-compatible codes, each code in the family achieves the perfect packing and is decoded by an ML decoder.

ulation results are for very short, punctured rate-compatible tail-biting convolutional codes. These codes can achieve very high throughput in the VLFT setting. The initial transmission is accomplished by puncturing the mother code. In subsequent incremental transmissions, one bit is transmitted at a time until the receiver decodes correctly using all received information. Note that the specific bits transmitted are pre-determined and do not depend on the feedback. The feedback is used only to decide when to terminate the transmission. The use of tail biting convolutional codes avoids the penalty of additional bits required to get the decoder back to the all-zero state. Therefore, The punctured rate compatible code is derived from a mother code of which a few bits are originally not transmitted. Those bits are later on sent as the decoder fails to decode correctly.

For the 2-dB binary-input additive-white-Gaussian-noise (BI-AWGN) channel, rate-compatible

tail-biting convolutional codes with feedback achieve about 95% of the idealized RCSP-ML throughput (R_{RCSP}) for average blocklengths up to 50 bits. For the 2-dB BI-AWGN channel with feedback, the convolutional codes achieve about 95% of the idealized RCSP-ML throughput (R_{RCSP}) for average blocklengths up to 50 bits. In [29], Williamson et al. also analyzed VLF systems for similar blocklengths of up to 100 bits. However, for average blocklengths of 100 bits and larger, the throughput of the convolutional code decreases. This performance degradation worsens as the average blocklength increases because the frame error rate of the convolutional code increases as the length of code increases. This phenomenon happens because the performance of the convolutional code does not improve for blocklengths larger that what the trace back depth supports.

As Chen et al. mention in [3], rate-compatible codes for IR-NTC systems that approach the performance predicted by RCSP-ML in the VLFT setting still remained to be identified for expected latencies (average blocklengths) of 200 to 600 bits. One of the primary focuses of this chapter is to demonstrate that non-binary LDPC (NB-LDPC) codes with incremental transmissions can attain over 90% of the predicted RCSP-ML throughput in the VLFT setting for average blocklengths of 150 to 450 bits. This latency region is important because it is still short enough that feedback provides a real advantage but also long enough that good VLFT performance can translate to good VLF performance when ideal termination is replaced by a practical termination scheme within the primary channel. In this chapter, similar to [3], an incremental redundancy (IR) scheme is used to give information to the receiver one bit at a time. A genie in form of NTC informs the receiver whether the decoded codeword is correct determining if the next increment needs to be sent. Since the transmissions continue until the correct codeword is decoded, the probability of error is zero. Furthermore, we show how to optimize the lengths of the incremental transmissions and to demonstrate that NB-LDPC codes with optimized incremental transmissions can achieve throughputs close to theoretical limits for expected latencies of 150 to 500 bits in the VLF setting . Most of the following analysis is applicable to any coding scheme, but we use NB-LDPC codes to demonstrate the possible performance motivated by [30], which shows that NB-LDPC codes without feedback, perform well in this short-blocklength regime.

In this section we initially analyze the performance of NB-LDPC codes in VLFT for a BI-

AWGN channel with an SNR of 2 dB with an unlimited number of transmissions and with the number of transmissions m fixed to be five and ten. We also consider two-phase VLF system with m = 5. In VLFT, the non-binary LDPC codes attain over 90% of the predicted RCSP-ML throughput for average blocklengths of 150 to 450 bits. Also, in a VLF scheme incorporating a confirmation phase after each communication phase (hence called "two-phase"), about 92% of capacity is achieved in less than 500 bits with a maximum of five transmissions. We also consider another VLF system that uses a stopping criterion that incorporates a cyclic redundancy check (CRC). Hence we consider the system with an unlimited number of transmissions as well as a more practical scenario in which the number of transmissions is limited in various VLFT and VLFT systems.

The size of the incremental messages in each transmission significantly affects the overall throughput of the system. We describe methods to select the size of each incremental message to maximize the throughput in VLFT and VLF schemes. We describe reciprocal-Gaussian and Gaussian approximations for probability density function (p.d.f.) of the cumulative blocklengths and their corresponding instantaneous rates supported by the channel. These approximations enable us to formulate an analytical throughput maximization problem that facilitates optimization of the sizes of the incremental transmissions.

In this work, we consider a broad range of m, the number of possible transmissions. For a large value of m and by using the optimized blocklengths, , the VLF with CRC system achieves better throughput performance than the two-phase VLF schemes for the example BI-AWGN channel with an SNR of 2 dB in the blocklength regime of 150 to 600 bits. For this channel, the CRC-based VLF scheme achieves about 94% of the capacity with an unlimited number m of transmissions and about 92% of the capacity with m = 10.

Similar results are extended to a higher-SNR (8 dB) channel and use a larger 16 quadrature amplitude modulation (QAM) constellation. The capacity of the 8 dB 16-QAM AWGN channel is 2.68 bits per symbol. The VLF-with-CRC system with an unlimited number of transmissions achieves a throughput of 2.37 bits per symbol with a frame error probability of less than 10^{-3}. This throughput corresponds to 88% of the capacity in the blocklength regime of about 40 16-QAM

symbols. Furthermore, the results show good performance for SNR-5dB BI-AWGN fading channel with the channel state information (CSI) available at the receiver. The capacity of this channel is 0.67 bits. The VLF-with-CRC system with an unlimited number of transmissions achieves a throughput of 0.60 with a frame error probability of less than 10^{-3}. This throughput corresponds to 90% of the capacity in the blocklength regime of about 140 bits.

We compare the throughput achieved by our optimization applied to NB-LDPC codes with convolutional codes whose blocklengths were optimized by a coordinate-descent algorithm in [31] for information blocks of $k = 12, 24,$ and 36 $GF(256)$ symbols ($k = 96, 192,$ and 288 bits) with $m = 5$ transmissions in VLFT and two-phase VLF settings. In order to use the same NB-LDPC codes designed in analysis of VLFT and two-phase VLF, the information blocks are reduced by 7 bits of CRC to $k = 89, 185,$ and 281 bits.

The rest of the chapter proceeds as follows: Section 1.2 provides an overview of the VLFT system, presents the NB-LDPC code design, and discusses the desirability of having degree-1 variable nodes in design of the NB-LDPC codes and provides the discussion on the post-processing modifications made to the NB-LDPC decoder to further improve performance. Section 1.3 discusses the reciprocal-Gaussian approximation for the probability mass function of the cumulative blocklengths required for successful decoding and applies the Gaussian approximation ideas to other communication channels such as high SNR and Fading channels. Section 1.4 presents the sequential differential optimization algorithm (SDO) for optimizing the size of each incremental transmission in VLFT. Section 1.5 presents a VLF system with CRC and analyzes this system with an unlimited number of transmissions and extends the results to the system with a limited number of transmissions. Section 1.6 gives an overview of the two-phase VLF scheme and uses SDO to optimize the cumulative blocklength at each decoding attempt. Section 1.7 compares the throughput and the expected latency of NB-LDPC and convolutional codes in VLFT and VLF settings. Section 1.8 concludes this chapter.

1.2 VLFT with Non-Binary LDPC Codes

Feedback cannot increase the capacity of point-to-point channels, but it can facilitate higher throughput at significantly lower latency than systems without feedback. The latency improvement is made possible by capitalizing on favorable noise realizations and attempting to decode early, instead of needlessly sending additional symbols [28]. In case of an unfavorable noise realization, additional information is required which effectively lowers the communication rate to match the operational capacity of the channel. Therefore, incremental transmissions are necessary for coding systems with feedback. A rate compatible code adapts the coding rate according to the channel noise. As the channel condition deteriorates, the transmission rate is lowered up to a point that the coding system supports the channel condition. As the channel conditions improve, the code rate is again increased to obtain better throughput.

1.2.1 System Overview

Traditionally, rate-compatible codes are designed by starting from a low-rate mother code and increasing the rate by puncturing the code. In general, finding an optimized puncturing pattern for finite length codes is not easy. To lower the rate of a punctured rate-compatible code, some initially encoded but punctured (not transmitted) bits from the mother code are transmitted as subsequent incremental transmissions. A good property of punctured rate compatible codes in terms of complexity is that they can be encoded and decoded with a single encoder and decoder respectively.

The proposed NB-LDPC coding scheme has this advantage of requiring only a single encoder and a single decoder while it does not explicitly involve puncturing. Rather, we design a short, high-rate NB-LDPC code for which all symbols are transmitted in the initial transmission. Each subsequent transmission is a single bit carefully selected to help the decoder as much as possible given its current decoding state. The rate is gradually lowered by sending these additional bits, each of which is a function of selected bits in the binary representation of the non-binary symbols.

We choose high-rate protograph-based NB-LDPC codes in this chapter. See [30] for a dis-

(a) Only degree-2 variable nodes (regular)

(b) degree-1 and 2 variable nodes (irregular)

Figure 1.3: Two protographs: (a) has only degree-2 variable nodes and (b) has a both degree-2 and degree-1 variable nodes.

cussion of protograph-based LDPC design. The codes are irregular, having mostly degree-2 and a few degree-1 variable nodes. Short-blocklength NB-LDPC codes with only degree-2 variable nodes (regular) designed over large Galois field sizes are known to perform well [32]. However, we observed that for short blocklengths and large Galois field sizes, the addition of a few degree-1 variable nodes improves the convergence of the decoder in the low SNR region (see Fig. 1.4 and Table 1.1). The operating SNR in this chapter is 2 dB.

It is crucial that the code used for the initial transmission has a very high coding rate, even higher than the capacity. The coding rate is lowered in case of decoding failure. For example for the BI-AWGN channel with SNR 2 dB, the initial code can have a rate of 0.75 to 0.8 while the capacity of the channel is 0.685. By doing this we can take advantage of favorable noise realizations and decode correctly at a shorter block length.

1.2.2 Desirability of Degree-1 Variable Nodes

Simulation results of short-blocklength NB-LDPC codes over large Galois fields show an improved convergence at low SNR values for codes *with* degree-1 variable nodes. Fig. 1.4 shows the frame error rate (FER) comparison of the lifted (copied and permuted) versions of the two rate-1/2 protographs in Fig. 1.3 for low E_b/N_0 values. These protographs are lifted 4 times and the NB-LDPC codes are over $GF(2^8)$.

Fig. 1.4 shows that at 2 dB the FER for the protograph in Fig. 1.3b is lower than Fig. 1.3a. The FERs in Fig. 1.4 may seem high, but for feedback with incremental redundancy, an initial FER of

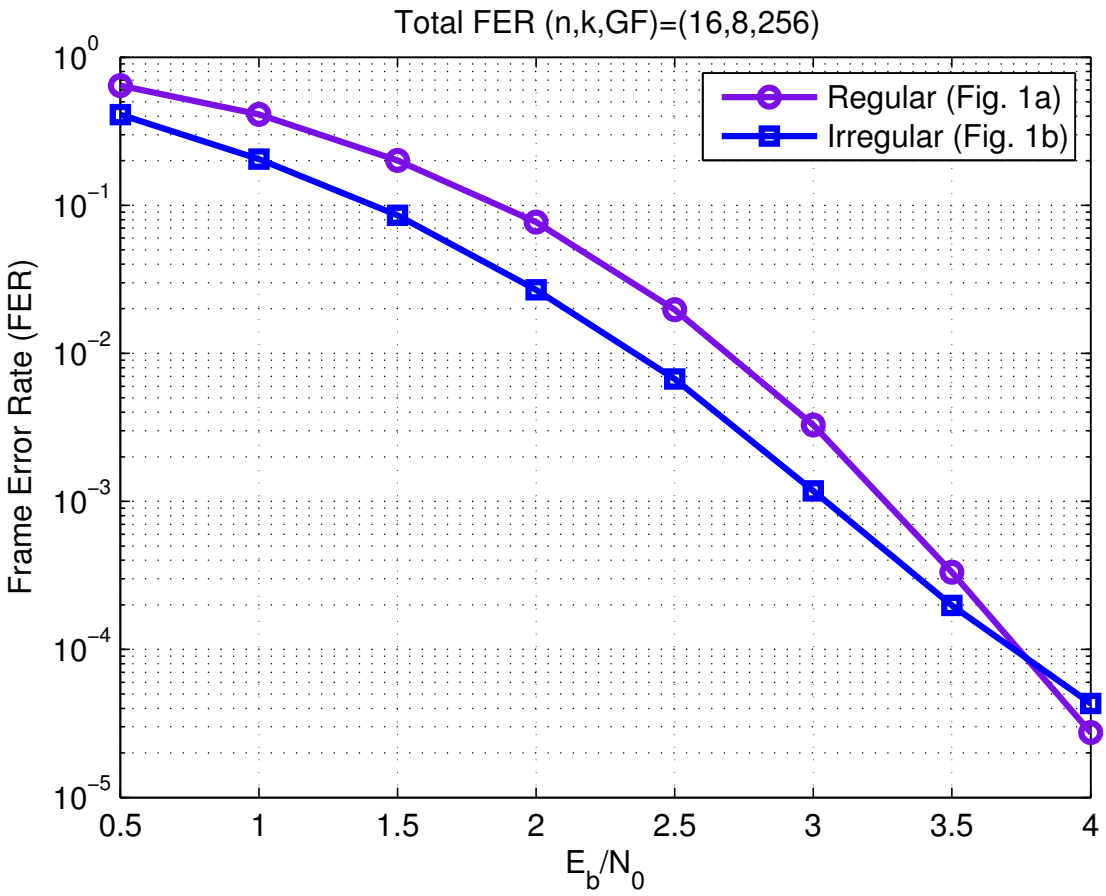


Figure 1.4: FER comparison of the lifted versions of the regular (Fig. 1.3a) and irregular (Fig. 1.3b) protographs in Fig. 1.3. Both codes are designed over $GF(256)$ with rate 1/2 (n = 16 $GF(256)$ symbols and k = 8 $GF(256)$ symbols).

0.02 is not unreasonable as it indicates 98% of codewords need no further redundancy.

The better FER for the protograph in Fig. 1.3b than the one in Fig. 1.3a is explained by the smaller number of short cycles present in the lifted version of the protograph in Fig. 1.3b. Table 1.1 lists the number of $l-cycles$ (cycles of length l) for the two protographs in Fig. 1.3.

There is an issue with having degree-1 variable nodes in NB-LDPC codes. The connection of these variable nodes to cycles results in rank-deficient sub-matrices within the non-binary parity-check matrix [33] leading to a low minimum symbol distance. However, this issue is less problematic in feedback systems where the incremental transmissions gradually increase the distance

Table 1.1: Small cycle count for the protographs in Fig. 1.3

Protograph	4-cycles	6-cycles	8-cycles	10-cycles	12-cycles
Fig. 1.a	0	0	36	0	96
Fig. 1.b	0	0	6	0	16

among candidate codewords. This form of rate compatible coding can be considered as a specific form of concatenation. I Furthermore, we present a decoding algorithm for the proposed codes with computational decoding complexity the same as that of the mother code. The codes designed in this manner perform well and retain expected throughput of higher than 90% of ML decoded RCSP analysis at average latencies 150 to 500 bits. In order to achieve even better throughput at low latency, a random initial state (RIS) dithering post processing technique is added to the non binary LDPC decoder to further improve its performance.

For most of the analysis, the operating SNR in this chapter is 2 dB, similar to the work of [34–36]. However, to emphasize the generality of the approach in this work, Section 1.3.1 shows results for higher-SNR AWGN and fading channels.

It is necessary that the initial transmission has a rate higher than the capacity to take advantage of good channel realizations. The coding rate is lowered until decoding is successful. For example for SNR-2dB BI-AWGN channel, the initial code can have a rate of 0.75 to 0.8 while the capacity of the channel is 0.685.

We will consider feedback systems that transmit incremental redundancy one bit at a time and also systems that transmit incremental redundancy in multiple-bit increments. For systems that use multiple-bit increments, a practical system may limit the maximum number m of increments. In the context of a specified m, this work also optimizes the lengths of the m possible increments to maximize throughput.

Section 1.2.3 provides a detailed description of how we generate each bit of incremental redundancy for the NB-LDPC codes that we use. Then, Section 1.3 shows that in the context of this incremental redundancy, the coding rate that first produces successful decoding is closely approx-

imated by a normal distribution. Knowing a distribution that describes the coding rate of the first successful decoding facilitates optimization of the lengths of multiple-bit increments, as described in Section 1.4.

1.2.3 Creating a Bit for Incremental Transmission

In [35], Vakilinia et al. use NB-LDPC codes in a VLFT system with 1-bit increments. In case of failure to decode to the correct codeword, after the initial transmission, the transmitter sends one bit at a time until the decoder decodes correctly.

Traditionally, rate-compatible codes are designed by starting with a low-rate mother code and increasing the rate by puncturing the code. The proposed NB-LDPC coding scheme in [35] does not explicitly involve puncturing. Rather, the design starts with a short, high-rate NB-LDPC code for which all symbols are transmitted in the initial transmission. Each subsequent transmission is a single bit carefully selected to help the decoder as much as possible given its current decoding state. The rate is gradually lowered by sending these additional bits, each of which is a function of selected bits in the binary representation of the non-binary symbols.

A rate-$\frac{K}{N}$ NB-LDPC code over $GF(2^m)$ used in a binary communication link encodes an information sequence of size Km bits into a sequence of size Nm bits. The LDPC code can be represented by a full-rank parity-check matrix H of size $(N-K, N)$. This parity-check matrix is the null space of a $K \times N$ generator matrix G with the elements of the matrix chosen from $GF(2^m)$. A valid codeword result in a zero vector when it is multiplied by this parity check matrix.

$$y = \{x \in GF(2^m)^N | Hx = 0\} \tag{1.1}$$

In order to use an NB-LDPC code with the primitive element α over binary-input channels, each $GF(2^m) = \{0, \alpha^0, \alpha^1, ..., \alpha^{(2^m-2)}\}$ symbol is converted to m bits. For example, consider $GF(2^3)$ with the primitive element of α. Table 1.2 shows how each element of $GF(2^3)$ can be uniquely represented in 3 bits (g_3, g_2, g_1).

Table 1.2: Binary representation of $GF(8)$ elements

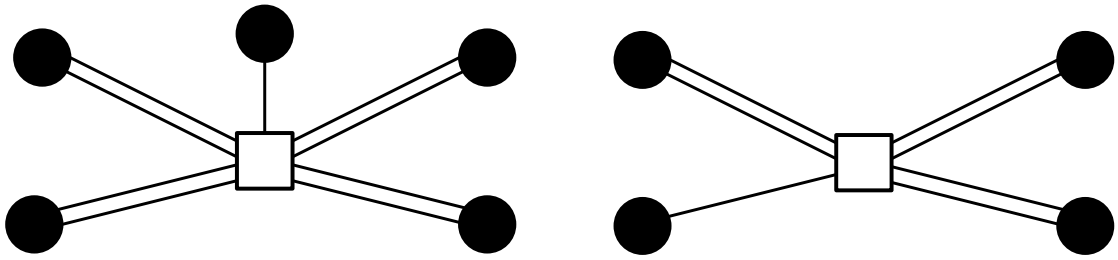
α^i	0	1	α	α^2	α^3	α^4	α^5	α^6
Poly.	0	1	α	α^2	α+1	α^2+α	α^2+α+1	α^2+1
$g_3g_2g_1$	000	001	010	100	011	110	111	101

The rate-$\frac{K}{N}$ non-binary LDPC codes proposed here initially encode a sequence of Km bits (K $GF(2^m)$ symbols) into a codeword of length Nm. The rate is lowered to $\frac{Km}{Nm+b}$ where b is number of additional incremental bits. Each additional bit is created by an xor ($\oplus$) combination of bits in the binary representation of the $GF(2^m)$ symbols. For each variable node, the reliability of each of the 2^m–1 possible combinations of the bits in the binary representation is computed. For example, in $GF(2^3)$ the reliabilities of the seven possible combinations $g_1, g_2, g_3, g_1 \oplus g_2, g_2 \oplus g_3, g_1 \oplus g_3, g_1 \oplus g_2 \oplus g_3$ are computed for each variable node. The single bit that is the combination with the lowest reliability (looking over all combinations for all variable nodes) is transmitted.

This is a form of active feedback in the sense that the feedback is telling the transmitter *what* to transmit rather than only *whether* to transmit additional bits from a pre-determined rate-compatible code family. This is a generalization of the ideas of active hypothesis testing [37]. For comparison, we also performed simulations using non-active feedback, in which the additional bits are selected at random.

The input frame consisting of K $GF(2^m)$ information symbols is initially encoded by the rate-$\frac{K}{N}$ NB-LDPC encoder into a sequence of length N $GF(2^m)$. These $GF(2^m)$ symbols are converted by their binary representations to bits. The Nm bits are modulated using BPSK and transmitted over an AWGN channel. The additive noise is modeled as an independent, zero-mean Gaussian random sequence with variance σ^2. The received Nm bits are grouped into N blocks of length m. These binary blocks of size m are used to compute the reliabilities (log-likelihood ratios) for each non-binary symbol. As in [35], SNR is calculated as $\frac{1}{\sigma^2}$, the ratio of the transmission power over the noise variance.

This coding scheme makes the unreliable variable nodes more reliable and let the decoder



(a) Rate-0.8 irregular protograph with degree-1 and 2 variable nodes

(b) Rate-0.75 irregular protograph with degree-1 and 2 variable nodes

Figure 1.5: Rate-0.75 and rate-0.8 non-binary LDPC protographs

escape from the local minima (trapping and absorbing sets) in cases of non-convergence. Additionally, this form of concatenation makes valid but wrong codewords less probable than the correct codeword.

1.2.4 Non-binary Code Designs

In this chapter we design NB-LDPC codes with symbols from $GF(2^8)$ for three different information blocks (k) of 96, 192, and 288 bits. The code for information block k = 96 (K = 12 $GF(2^8)$ symbols) is the 3-times lifted version of the rate-0.8 protograph in Fig. 1.5a. For information blocks k = 192 and k = 288 bits (K = 24, and K = 36 $GF(2^8)$ symbols), the protograph in Fig. 1.5b is lifted 8 and 12 times respectively to obtain rate-0.75 codes. The reason for selecting a higher-rate code for k = 96 is given in Sec. 1.7 and a complete description of the non-binary codes used in this work is available online.

Incremental transmissions are used progressively to lower the rate of the codes designed in the previous section until the transmission is decoded correctly. This section presents how the incremental transmissions are created bit by bit in the encoder and processed by the decoder.

It is easier to explain the decoding of non binary LDPC codes by their corresponding bipartite graph. The graph consists of N variable nodes corresponding to the transmitted codeword and N-K check nodes corresponding to the parity checks. A connection exists between variable node j and check node i if the corresponding element H_{ij} in the parity check matrix H is non-zero. Messages

are only exchanged between connected nodes. It is easier to explain the decoding of non binary LDPC codes by their corresponding bipartite graph with variable nodes and check nodes.

The iterative decoder sends vectors of log-likelihood ratios (LLRs) between variable and check nodes. In the initial iteration the LLR associated with the belief that $V_j = a$, the variable node j taking on the value $a \in GF(2^m)$, with respect to the reference belief that $V_j = 0$ is defined by $LLR_j^a = \log\left(\frac{P(V_j=a|Y)}{P(V_j=0|Y)}\right)$, where Y is the information received from the channel. For independent Gaussian noise, the prior LLR values obtained from the channel can be calculated from the sum of the individual LLRs for each received bit in the group of m bits. For example, for a variable node j over $GF(2^3)$, the channel prior LLR for $\alpha^4 = (1, 1, 0)$ is calculated as follows:

$$\begin{aligned} LLR_j^{\alpha^4} &= \log\left(\frac{P(V_j = a|Y)}{P(V_j = 0|Y)}\right) \\ &= \log\frac{P(g_3(V_j) = 1|x_3)P(g_2(V_j) = 1|x_2)P(g_1(V_j) = 0|x_1)}{P(g_3(V_j) = 0|x_3)P(g_2(V_j) = 0|x_2)P(g_1(V_j) = 0|x_1)} \\ &= \log\frac{P(g_3(V_j) = 1|x_3)P(g_2(V_j) = 1|x_2)}{P(g_3(V_j) = 0|x_3)P(g_2(V_j) = 0|x_2)} \\ &= -\frac{2}{\sigma^2}(x_3 + x_2). \end{aligned} \tag{1.2}$$

The decoding algorithm is initialized by sending these messages from variable nodes to check nodes. After this initialization step, the decoding process continues by sending LLR vectors iteratively from the variable nodes to their neighboring check nodes and vice versa. Iterative message passing only exchanges extrinsic information between connected nodes.

A hard decision about the original codeword is made when the messages go back to the variable nodes. If this decision satisfies the check-node constraints, the decoding is terminated. Otherwise, the messages between variable and check nodes are iteratively exchanged until a valid codeword is detected or the maximum number of iterations is reached.

Once the decoder terminates iterations, it provides feedback to the transmitter of the codeword it has identified (or that it has failed to identify a codeword) and the least reliable combination of bits identified. If the codeword is correct, the transmitter sends the NTC which terminates the transmission of that codeword. Otherwise, the transmitter transmits the incremental redundancy bit. Each additional bit is created by an XOR ($\oplus$) combination (summation in $GF(2)$) of bits in

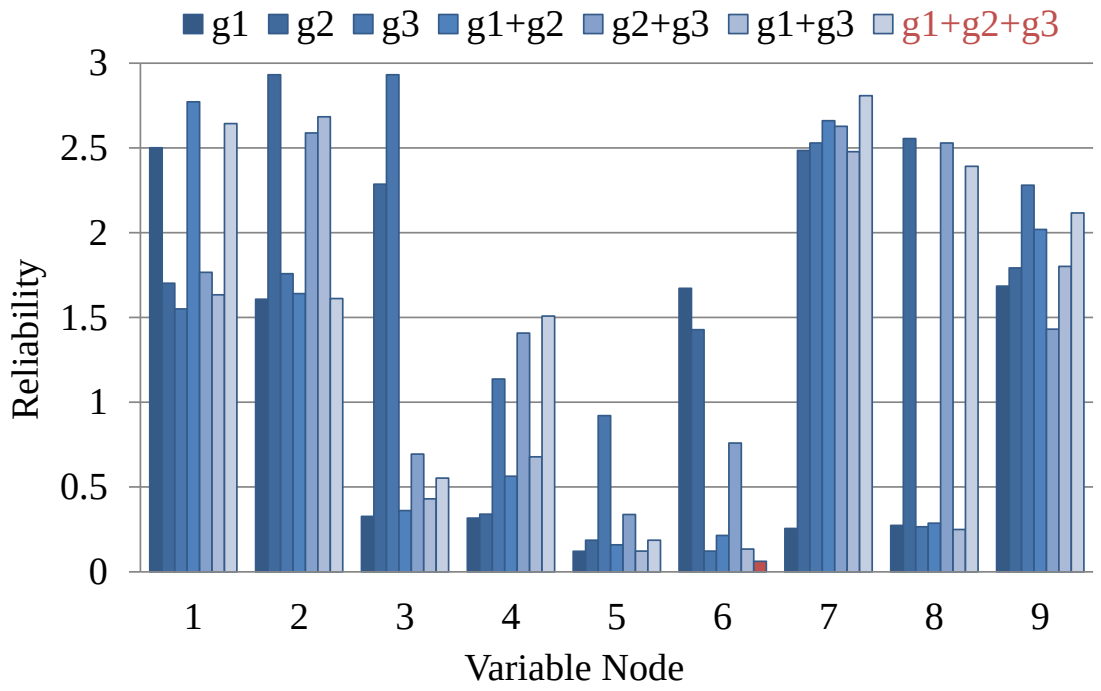


Figure 1.6: Example of the reliabilities for all seven possible combinations $g_1, g_2, g_3, g_1 \oplus g_2, g_2 \oplus g_3, g_1 \oplus g_3, g_1 \oplus g_2 \oplus g_3$ for all nine variable nodes. The combination with the lowest reliability is transmitted as incremental redundancy.

the binary representation of one $GF(2^m)$ symbol. For each variable node, the receiver computes the reliability of each of the 2^m-1 possible combinations of the bits in the binary representation is computed. For example, in $GF(2^3)$ the reliabilities of the seven possible combinations $g_1, g_2, g_3, g_1 \oplus g_2, g_2 \oplus g_3, g_1 \oplus g_3$, and $g_1 \oplus g_2 \oplus g_3$ are computed for each variable node. Finally, the single combination bit that has the least reliability (e.g. considering all seven combinations for all variable nodes and choosing the least-reliable combination for a single variable node) is requested from the transmitter.

The decoder re-computes the initialization only for the variable node that has been updated with an additional bit. Continuing the previous example, if bits 1 and 3 in variable node j are

combined for the transmitted incremental bit, the new initialization LLR is computed as:

$$LLR'^{\alpha^4}_j = LLR^{\alpha^4}_j + \log \frac{P(g_4(V_j) = (0 \oplus 1)|x_{\text{new}})}{P(g_4(V_j) = 0|x_{\text{new}})} \tag{1.3}$$

$$= LLR^{\alpha^4}_j - \frac{2}{\sigma^2}(x_{\text{new}}). \tag{1.4}$$

The rest of the iterative process is not changed. The transmission of addition bits based on lowest reliability combination continues until the decoder decodes to the correct codeword.

Finally, the $GF(2^m)$ symbols are converted back to binary and the additional transmitted bits are also computed based on the desired binary combinations. For instance, in the previous example, α^4 as a symbol in $GF(2^3)$ will be converted to binary $(1, 1, 0, 1)$ where the rightmost 1 is an xor combination of the 1^{st} and 3^{rd} bits from the left.

This is a form of active feedback in which relatively extensive feedback tells the transmitter *what* to transmit in contrast to non-active feedback in which a single bit of feedback indicates *whether* to transmit. This is a generalization of the ideas of active hypothesis testing [37]. In [35] Vakilinia et al. compared the performance of a non-active feedback system and the active feedback system discussed earlier for NB-LDPC codes and showed significantly better performance with the active feedback system. The active feedback used in [35] tells the transmitter which bit combination to be transmitted next. This active feedback scheme does not require the receiver to transmit back the entire message, contrary to the analysis of [2]. In the non-active feedback scheme of [35] the additional bits are selected at random.

In this dissertation, we consider both active and non-active feedback. The non-active feedback in this work corresponds to sending the XOR of all bits representing one of the variable nodes of the original rate-k/N_0 NB-LDPC code. This predetermined non-active feedback system performs close to the system with active feedback since the active feedback of [35] usually asks for the XOR of all bits for the subsequent transmissions. The figures and results in this chapter indicate whether active or non-active feedback scheme was used to generate them.

This coding scheme makes the unreliable variable nodes more reliable and let the decoder escape from the local minima (trapping and absorbing sets) in cases of non-convergence. Additionally, this form of concatenation makes valid but wrong codewords less probable than the

correct codeword. In order to improve the performance of NB-LDPC codes, we can also use the post processing algorithm discussed next.

1.2.5 Post Processing

The performance of a belief-propagation (BP) decoder is suboptimal compared to the optimal ML decoding. This suboptimality is usually due to the failure of the BP decoder to converge to a valid codeword. Post-processing techniques improve the performance of BP decoders by helping the decoder converge to valid codewords more frequently.

The post-processing technique used here is a modified version of Random Initial State (RIS) algorithm which originally [38] resulted in improvements in the floor and waterfall regions for a binary LDPC decoder. The RIS algorithm explores the neighborhood of the received vector to find a convergence region by adding a dithering random vector to the received vector. The dithering slightly perturbs the initial vector so that the decoder decodes to a nearby codeword.

In case of non convergence to a valid codeword, dithering begins by choosing a vector with independent and identically Gaussian distributed zero-mean components with variance of σ'^2 smaller than the variance of the channel noise. This process is performed for a certain number of times t.

If the decoder fails to converge to a valid codeword after t decoding attempts, the incremental bit is transmitted. If only a unique valid codeword is detected, this codeword is fed back to the transmitter. If the decoded codeword is correct the next block of information is sent, otherwise, an additional incremental bit is transmitted to decode the current block. The RIS algorithm may result in the decoder converging to multiple valid codewords in the t decoding attempts. In this case, the receiver computes the squared Euclidean distance between the detected codewords and the received sequence and selects the codeword with the smallest squared Euclidean distance.

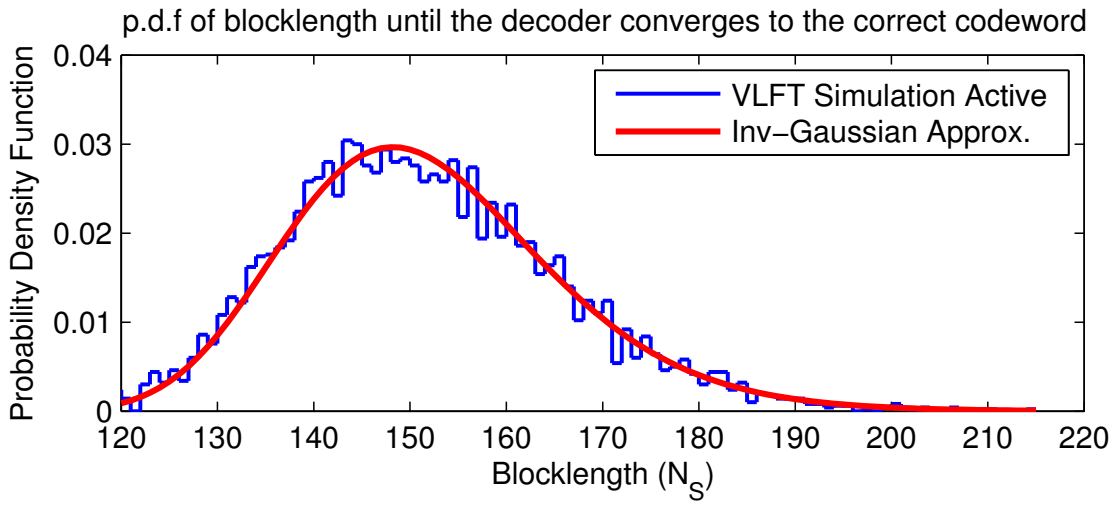


Figure 1.7: Empirical probability mass function (p.m.f.) corresponding to the blocklength required for successful decoding for the first time in VLFT using $GF(256)$ NB-LDPC code over SNR-2dB AWGN channel. Also shown is the reciprocal-Gaussian approximation of (1.7) with $\mu_S = 0.6374$ and $\sigma_S = 0.0579$. Smallest blocklength is $N_0 = 120$ bits with $k = 96$ information bits so that the initial rate is $R_0 = \frac{k}{N_0} = 0.8$.

1.3 Gaussian and Reciprocal-Gaussian Approximations

The input frame consisting of K $GF(2^m)$ information symbols is initially encoded by the rate-$\frac{K}{N}$ NB-LDPC encoder into a sequence of length N $GF(2^m)$ symbols. These $GF(2^m)$ symbols are converted using their binary representations to bits. The Nm bits are modulated using binary phase shift keying (BPSK) and transmitted over an AWGN channel. The additive noise is modeled as an independent, zero-mean Gaussian random sequence with variance σ^2. As in [34], SNR is calculated as $\frac{1}{\sigma^2}$, the ratio of the transmission power to the noise variance.

Consider a stream of incremental redundancy as described in Section 1.2.3 arriving one bit at a time at the receiver (after an initial transmission of a high-rate NB-LDPC code). We are interested in the statistical behavior of the random variable describing the blocklength of the first successful decoding and the corresponding random variable describing the coding rate of that first successful decoding.

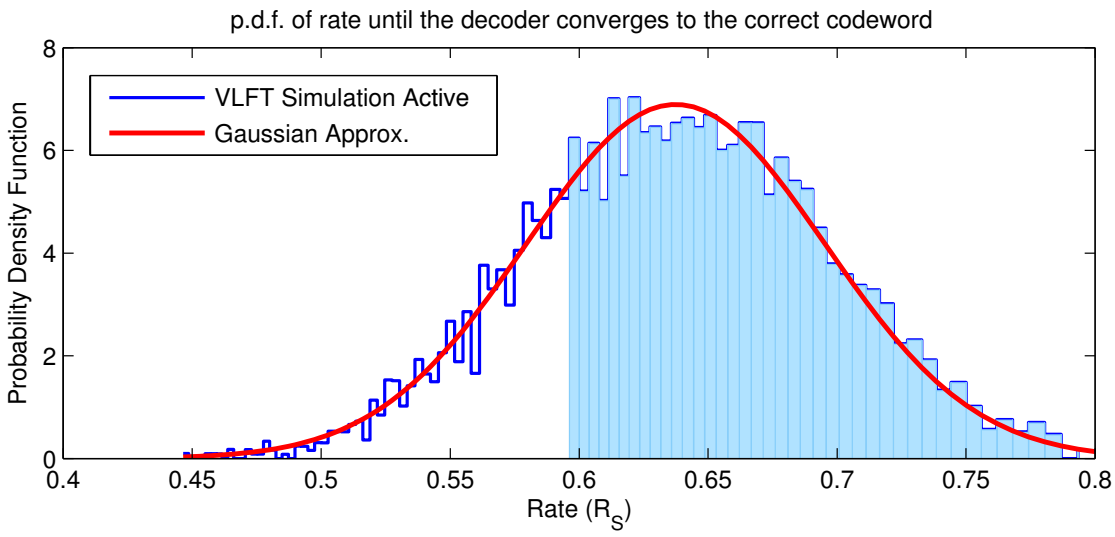


Figure 1.8: Empirical p.m.f. corresponding to $R_S = \frac{k}{N_S}$ computed from Fig. 1.7 and Gaussian approximation of (1.5) with $\mu_S = 0.6374$ and $\sigma_S = 0.0579$.

For the system of [35], the "VLFT simulation active" plot in Fig. 1.7 shows the empirical p.m.f. of the blocklength of first successful decoding. The total blocklength N_S includes the initial block and all incremental transmissions, (with active feedback) required for receiver to decode the NB-LDPC codeword correctly for the first time. The "VLFT simulation active" plot in Fig. 1.8 shows the empirical p.m.f. of the instantaneous rate $\left(R_S = \frac{k}{N_S}\right)$ at which decoding is successful for the first time. Fig. 1.8 shows that R_S is well-approximated by a normal distribution

$$f_{R_S}(r) = \frac{1}{\sqrt{2\pi\sigma_S^2}} e^{-\frac{(r-\mu_S)^2}{2\sigma_S^2}} \tag{1.5}$$

with mean $\mu_S = \mathrm{E}(R_S)$ and variance $\sigma_S^2 = \mathrm{Var}(R_S)$. The intuition behind these approximations is consistent with the "normal approximation" of the accumulated information density due to the law of large numbers (LLN) in [4].

To maximize throughput, the initial code-rate of the NB-LDPC code is chosen so that almost no codeword is successfully decoded in the initial transmission. Thus, the empirical probability mass function (p.m.f.) of the number of additional increments required to decode correctly does not have a spike at zero.

Fig. 1.9 shows the complementary cumulative distribution function (c.c.d.f.) for the distribution

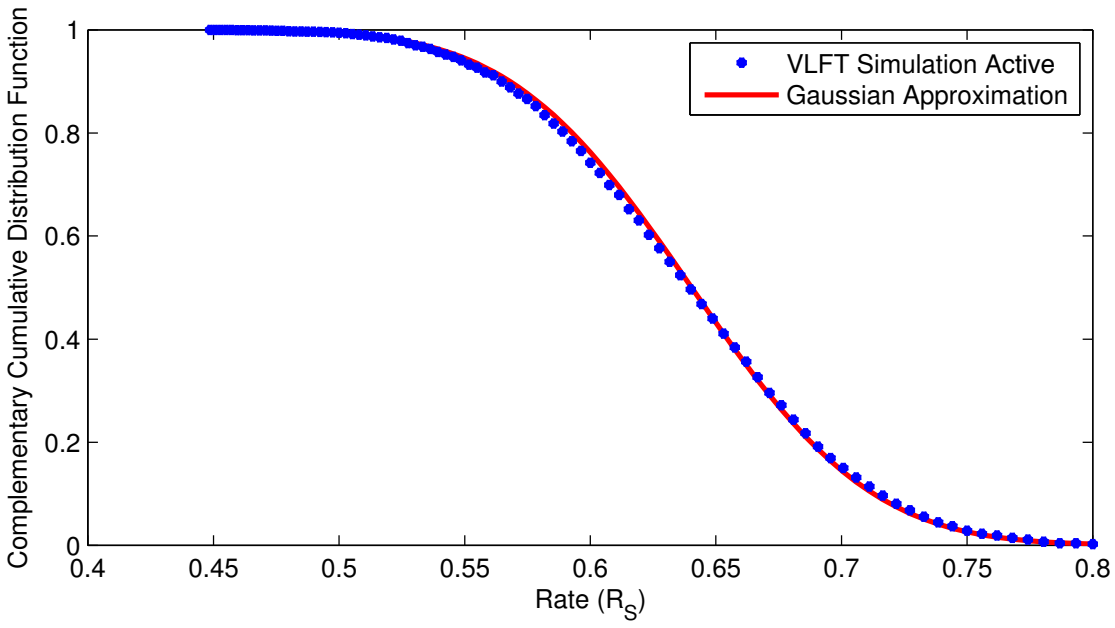


Figure 1.9: Empirical c.c.d.f. and the approximation on the tail of a normal distribution (Q-function) corresponding to the shaded area of Fig. 1.8.

of R_S and the Gaussian approximation of Fig. 1.8. Fig. 1.9 confirms that the distribution of R_S is well approximated by a Gaussian distribution. As discussed later, the empirical c.c.d.f is used to show that the Gaussian approximation is valid for a variety of AWGN channels including the high SNR ones using larger constellations and also for fading channels. The "VLFT simulation active" plot in Fig. 1.9 shows the empirical c.c.d.f. of the instantaneous rate $\left(R_S = \frac{k}{N_S}\right)$ at which decoding is successful for SNR-2dB BI-AWGN of [36]. This c.c.d.f. plot shows the cumulative probability that the channel supports a rate higher than the rate on the x axis. This higher rate means that the decoding has been successful with a lower number of transmitted bits. The c.c.d.f. plot corresponds to the shaded area of Fig. 1.8. The "Gaussian Approximation" plot of Fig. 1.9 corresponds to the tail probability of the standard normal distribution of Fig. 1.8.

The parameters μ_S and σ_S^2 in (1.5) for a particular code need to be determined through simulation and curve fitting. Having the p.m.f. of the N_S, the curve fitting process involves calculating the p.m.f. and c.c.d.f. of R_S and solving a linear regression problem to obtain μ_S and σ_S. Note that μ_S is *not* the expected throughput but rather the average of the instantaneous rates supported

by the channel.

The cumulative distribution function (c.d.f.) of N_S is $F_{N_S}(n) = P(N_S \leq n)$, and we have

$$F_{N_S}(n) = P\left(\frac{k}{R_S} \leq n\right) = P\left(R_S \geq \frac{k}{n}\right) = 1 - F_{R_S}(\frac{k}{n}). \tag{1.6}$$

Taking the derivative of F_{N_S} using the Gaussian approximation of F_{R_S} produces the following "reciprocal-Gaussian" approximation for p.d.f. of N_S:

$$f_{N_S}(n) = \frac{k}{n^2\sqrt{2\pi\sigma_S^2}} e^{\frac{-\left(\frac{k}{n}-\mu_S\right)^2}{2\sigma_S^2}}. \tag{1.7}$$

This p.d.f as shown in Fig. 1.7 closely approximates the empirical distribution of N_S. For $N_1 < N_2$, the probability of the decoding attempt being successful at blocklength N_2 but not at N_1 using this approximation is

$$\int_{N_1}^{N_2} f_{N_S}(n)dn = \int_{N_1}^{N_2} \frac{k}{n^2\sqrt{2\pi\sigma_S^2}} e^{\frac{-\left(\frac{k}{n}-\mu_S\right)^2}{2\sigma_S^2}} dn \tag{1.8}$$

$$= Q\left(\frac{\frac{k}{N_2} - \mu_S}{\sigma_S}\right) - Q\left(\frac{\frac{k}{N_1} - \mu_S}{\sigma_S}\right). \tag{1.9}$$

The increase in blocklength from N_1 to N_2 reduces the rate from $\frac{k}{N_1}$ to $\frac{k}{N_2}$. Note that (1.9) gives the probability that the channel supports rate $\frac{k}{N_2}$ while not supporting the higher rate $\frac{k}{N_1}$. The Q functions in (1.9) are due to the normally-distributed highest-rate-of-successful-decoding (R_S) at $\frac{k}{N_1}$ and $\frac{k}{N_2}$.

1.3.1 General Applicability of the Normal Approximation

A similar Gaussian analysis is obtainable for other channels and different SNR values. Fig. 1.10 shows a similar complementary cumulative Gaussian approximation for the same $GF(256)$ NB-LDPC code of Fig. 1.8 with an initial binary rate of 0.8 on SNR-8dB 16-QAM AWGN channel. Each non-binary element of the NB-LDPC code is mapped onto two 16-QAM symbols. Once again, the distribution of the instantaneous rate that the channel supports is well approximated by a normal distribution.

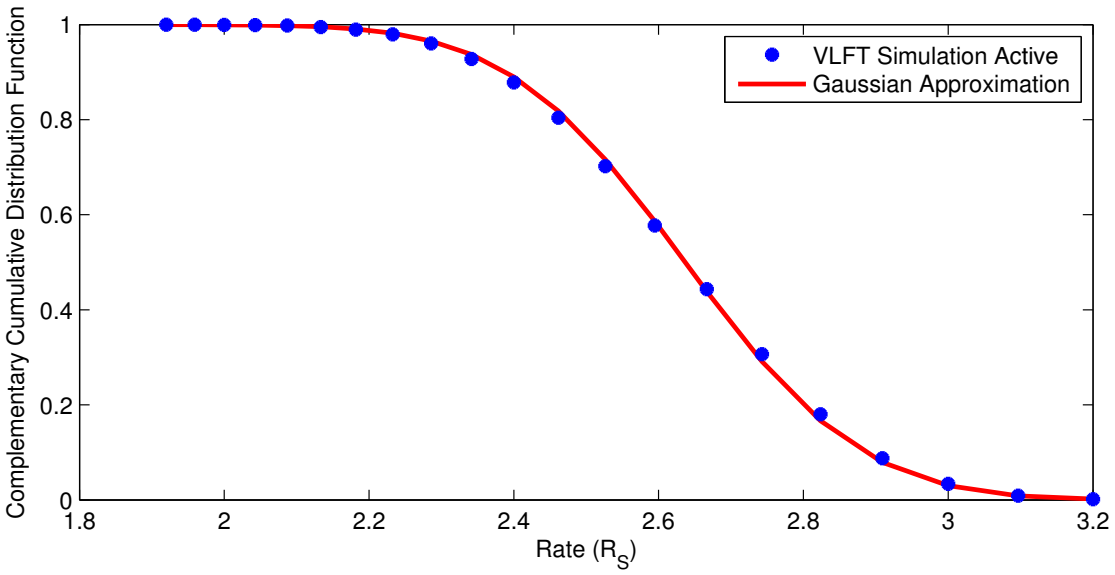


Figure 1.10: Empirical c.c.d.f. and the approximation on the tail of a normal distribution with $\mu_S = 2.63$ and $\sigma_S = 0.19$ of the instantaneous rate $\left(R_S = \frac{k}{N_S}\right)$ at which decoding is successful for SNR-8dB 16-QAM AWGN channel.

Furthermore, Fig. 1.11 shows the complementary cumulative Gaussian approximation for the same $GF(256)$ NB-LDPC code of Fig. 1.8 with an initial binary rate of 0.8 on SNR-5dB BI-AWGN fading channel with CSI knowledge at the receiver. The output of the channel, $Y = \beta X + N$ where the input X is a binary phase-shift keying (BPSK) modulated signal and N is the Gaussian noise with $\text{Var}(N) = \sigma^2$. The average SNR of this channel is $\frac{1}{\sigma^2}$. The coefficient β is a Rayleigh distributed random variable satisfying $E[\beta^2] = 1$. The value of β is known at the receiver. The distribution of the instantaneous rate that the channel supports is again well approximated by a normal distribution. Since the normal distribution approximation is valid for various channels, most of the analyses in the subsequent sections of this chapter are also valid for various channels with different SNR values.

To further discuss the generality of the Gaussian approximation on the rate that the channel supports in our feedback system, consider the accumulated information density $i(X, Y)$ at the receiver at the time of successful decoding. The expected value of $i(X, Y)$ is the capacity of the channel. For BI-AWGN channel, the $i(X, Y)$ is derived as follows:

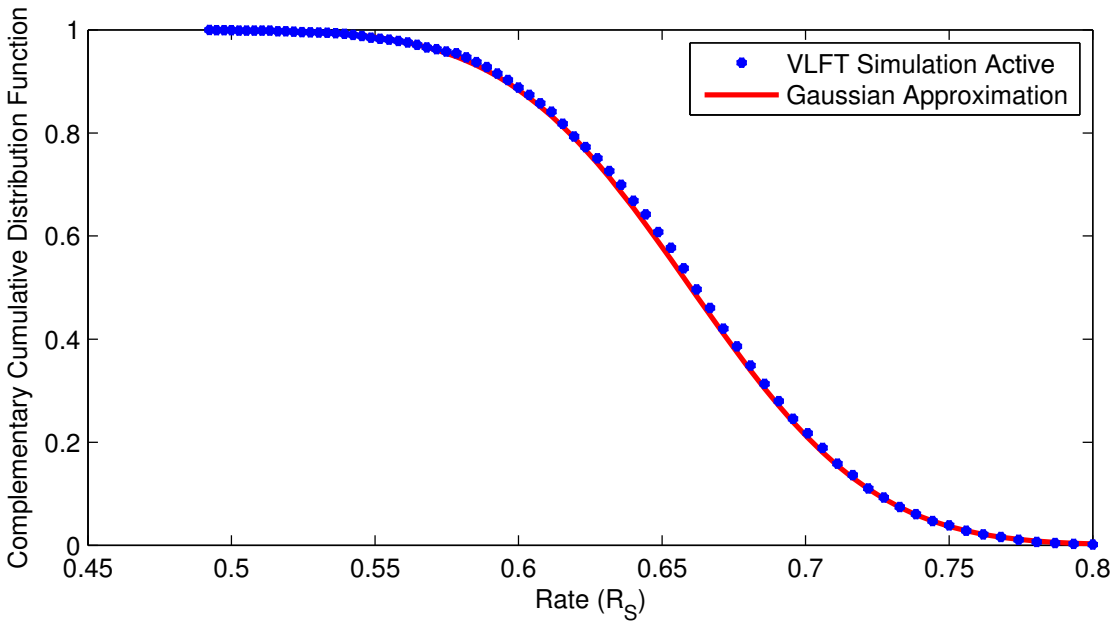


Figure 1.11: Empirical c.c.d.f. and the approximation on the tail of a normal distribution with $\mu_S = 0.66$ and $\sigma_S = 0.05$ of the instantaneous rate $\left(R_S = \frac{k}{N_S}\right)$ at which decoding is successful for SNR-5dB AWGN fading channel.

$$
\begin{aligned}
i(X,Y) &= \log_2 \frac{f_{Y|X}(y|x)}{f_Y(y)} && (1.10)\\
&= \log_2 \frac{e^{-(y-x)^2/(2\sigma^2)}}{\frac{1}{2}\left(e^{-(y-1)^2/(2\sigma^2)} + e^{-(y+1)^2/(2\sigma^2)}\right)} && (1.11)\\
&= \log_2 \frac{e^{-z^2/(2\sigma^2)}}{\frac{1}{2}\left(e^{-z^2/(2\sigma^2)} + e^{-(z+2)^2/(2\sigma^2)}\right)} && (1.12)\\
&= 1 - \log_2\left(1 + e^{-2(z+1)/\sigma^2}\right). && (1.13)
\end{aligned}
$$

For BI-AWGN channel, $i(X,Y)$ is a function only of the noise realization $z = y - x$ for $x = \pm 1$, and hence $i(X,Y) = i(z)$. For each transmitted bit from the NB-LDPC code over the channel, there is some amount of information density accumulated. The total amount of information density accumulation (I) at the receiver until the receiver decodes the message correctly

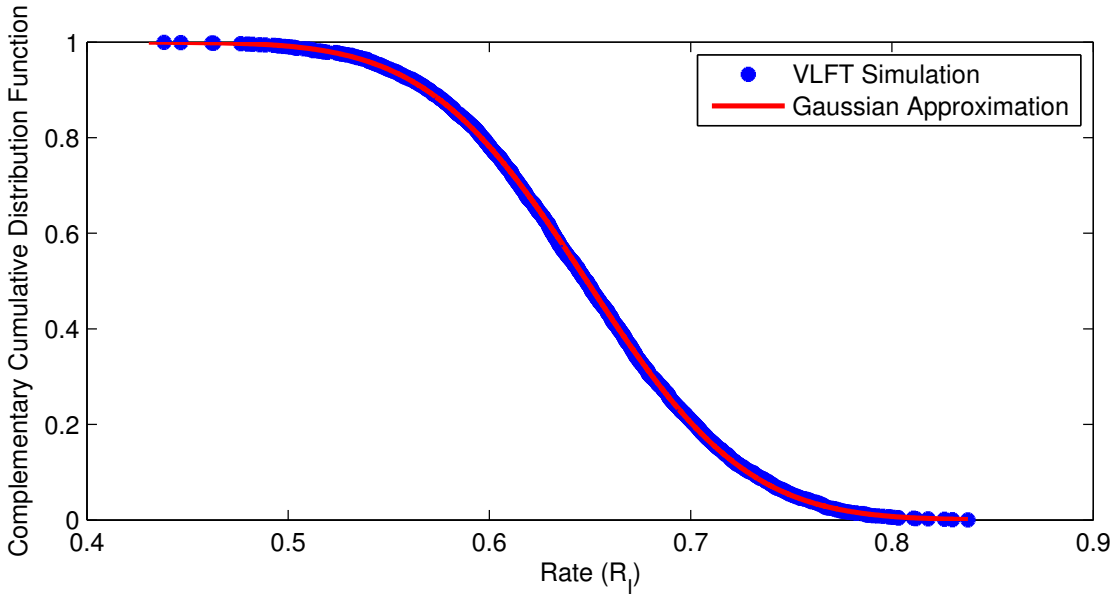


Figure 1.12: Empirical c.c.d.f. and the approximation on the tail of a normal distribution with $\mu_S = 0.64$ and $\sigma_S = 0.06$ of the average accumulated information density $\left(R_I = \frac{I}{N_S}\right)$ at which decoding is successful for SNR-2dB AWGN channel.

$$I = \sum_{k=1}^{N_s} i(z_k). \tag{1.14}$$

The corresponding rate associated with the accumulated information density is $R_I = \frac{I}{N_s}$. As pointed out by [4], (1.14) is a sum of independent random variables for which the central limit theorem will converge quickly to a normal distribution. An important consideration for our approach is whether the rate at which a practical decoder succeeds also follows a normal distribution. This hinges on the ability of a rate-compatible code family as in [39] to operate with a small gap from capacity over the rate range of interest.

For the previously discussed SNR-2dB BI-AWGN channel, Fig. 1.12 shows the c.c.d.f. of R_I and the corresponding Gaussian approximation. The rate corresponding to the accumulated information density at the receiver until the decoding is successful also follows the Gaussian approximation.

Fig. 1.13 shows the average accumulated information density for decoding correctly at a partic-

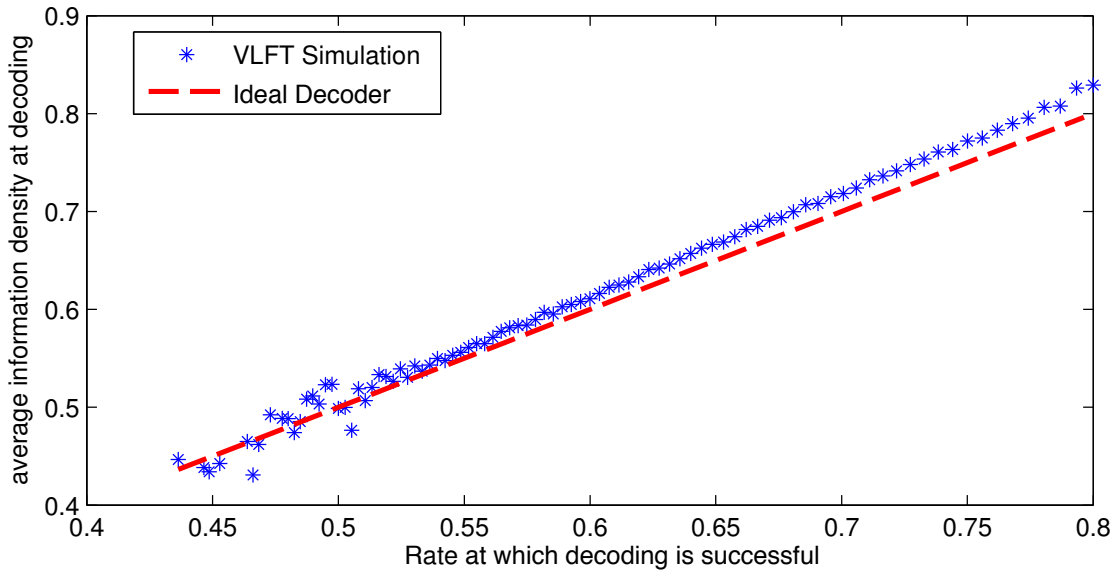


Figure 1.13: Average amount of the accumulated information density for decoding correctly at a particular code rate for the GF(256) NB-LDPC of Fig. 1.7 code over SNR-2dB AWGN channel.

ular code rate for the NB-LDPC code. This figure shows on average, how much more information in number of bits the NB-LDPC code requires to decode the message correctly compared to the operating rate. The "ideal decoder" plot in Fig. 1.13 corresponds to the average accumulated information density being equal to the rate (the line of equality).

1.4 Optimizing Transmission Lengths

Consider the scenario in which the number of increments (packets of incremental redundancy) associated with a codeword that can be accumulated at the receiver is limited to m. Using the p.d.f. of N_S from (1.7) we find the optimal blocklengths $\{N_1, N_2, \cdots, N_m\}$ to maximize the throughput. The initial blocklength N_1 satisfies $N_1 \geq N_0$ where N_0 is the smallest possible blocklength (of the original NB-LDPC code). Each of the additional bits beyond N_0 transmitted in the first transmission is the exclusive-or of all eight bits representing one of the variable nodes of the original rate-k/N_0 GF(256) NB-LDPC code. The other transmissions use the scheme in Section 1.2.3 to generate the subsequent bits.

1.4.1 Throughput Optimization Through Exhaustive Search

An accumulation cycle (AC) is a set of m or fewer transmissions and decoding attempts ending when decoding is successful or when the m^{th} decoding attempt fails. If decoding is not successful after the m^{th} decoding attempt, the accumulated transmissions are forgotten and the process starts over with a new transmission of the first block of N_1 symbols. From a strict optimality perspective, neglecting the symbols from the previous failed AC is sub-optimal. However, the probability of an AC failure is sufficiently small that the performance degradation is negligible. Neglecting these symbols greatly simplifies analysis.

Define the throughput as $R_T = \frac{E[K]}{E[N]}$, where $E[N]$ represents the expected number of channel uses in one AC and $E[K]$ is the effective number of *information* bits transferred correctly over the channel in one AC.

The expression for $E[N]$ is

$$E[N] = N_1 Q\left(\frac{\frac{k}{N_1} - \mu_S}{\sigma_S}\right) \tag{1.15}$$

$$+\sum_{i=2}^{m} N_i \left[Q\left(\frac{\frac{k}{N_i} - \mu_S}{\sigma_S}\right) - Q\left(\frac{\frac{k}{N_{i-1}} - \mu_S}{\sigma_S}\right)\right] \tag{1.16}$$

$$+N_m \left[1 - Q\left(\frac{\frac{k}{N_m} - \mu_S}{\sigma_S}\right)\right]. \tag{1.17}$$

The right hand side of (1.15) shows the contribution to expected blocklength from successful decoding on the first attempt in the AC. $Q\left(\frac{\frac{k}{N_1}-\mu_S}{\sigma_S}\right)$ is the probability of decoding successfully with the initial block of N_1. Similarly, the terms in (1.16) are the contributions to expected blocklength from decoding that is first successful at total blocklength N_i (at the i^{th} decoding attempt). Finally, the contribution to expected blocklength from not being able to decode even at N_m is $1-Q\left(\frac{\frac{k}{N_m}-\mu_S}{\sigma_S}\right)$ which is shown in (1.17). Even when the decoding has not been successful at N_m, the channel has been used for N_m channel symbols.

The expected number of successfully transferred information bits $E[K]$ is

$$E[K] = kQ\left(\frac{\frac{k}{N_m} - \mu_S}{\sigma_S}\right), \tag{1.18}$$

where $Q\left(\frac{\frac{k}{N_m}-\mu_S}{\sigma_S}\right)$ is the probability of successful decoding at some point in the AC. Note that $E[K]$ depends only upon N_m. In fact, for large values of N_m, $E[K] \approx k$ and thus not sensitive to the choice of N_m

Exhaustive search (ES) can be used to optimize $\{N_1, N_2, \cdots, N_m\}$ to maximize $R_T = \frac{E[K]}{E[N]}$. The order of complexity for ES is $O\left(\binom{N_{max}-N_0+1}{m}\right)$, where N_{max} is the maximum allowable overall blocklength for an AC. Since $E[K] \approx k$, maximization of R_T is equivalent to minimization of $E[N]$.

1.4.2 Sequential Differential Optimization

Sequential differential optimization (SDO) is an extremely effective alternative to ES. Over a range of possible N_1 values, SDO optimizes $\{N_2, \cdots, N_m\}$ to minimize $E[N]$ for each fixed value of N_1 by setting derivatives to zero as follows:

$$\left\{N_2, \cdots, N_m : \frac{\partial E[N]}{\partial N_i} = 0, \ \forall i = 1, \cdots, m-1\right\}. \tag{1.19}$$

For each $i \in \{2, \ldots, m\}$, the optimal value of N_i is found by setting $\frac{\partial E[N]}{\partial N_{i-1}} = 0$, yielding a sequence of relatively simple computations. In other words, we select the N_i that makes our previous choice of N_{i-1} optimal in retrospect. For example to find N_2 we compute the derivative

$$\frac{\partial E[N]}{\partial N_1} = Q\left(\frac{\frac{k}{N_1} - \mu_S}{\sigma_S}\right) + (N_1 - N_2)Q'\left(\frac{\frac{k}{N_1} - \mu_S}{\sigma_S}\right) = 0 \tag{1.20}$$

and solve for N_2 as

$$N_2 = \frac{Q\left(\frac{\frac{k}{N_1}-\mu_S}{\sigma_S}\right) + N_1 Q'\left(\frac{\frac{k}{N_1}-\mu_S}{\sigma_S}\right)}{Q'\left(\frac{\frac{k}{N_1}-\mu_S}{\sigma_S}\right)}, \tag{1.21}$$

where

$$Q'\left(\frac{\frac{k}{N_i} - \mu_S}{\sigma_S}\right) = \frac{k}{N_i^2 \sigma_S}\frac{1}{\sqrt{2\pi}} e^{\frac{-\left(\frac{k}{N_i}-\mu_S\right)^2}{2\sigma_S^2}}. \tag{1.22}$$

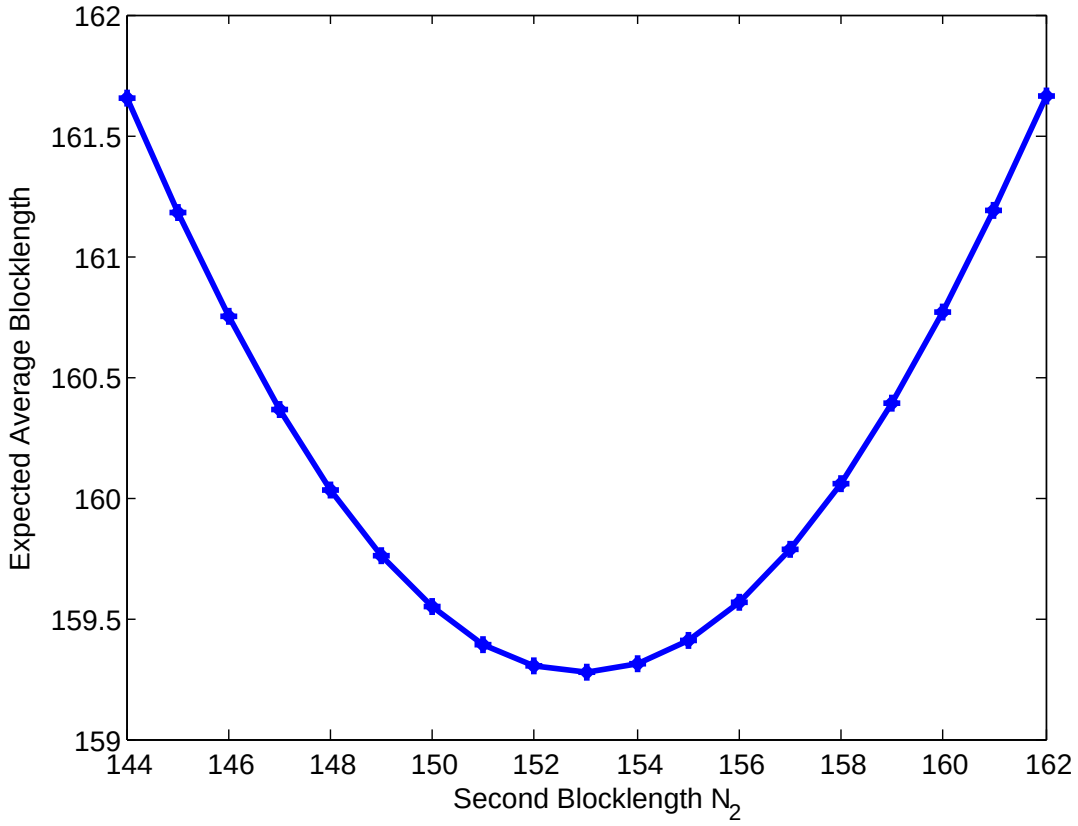


Figure 1.14: The expected average blocklength as a function of the size of the second increment is convex. Therefore, there is a minimum average blocklength for the optimum choice of the second blocklength size N_2.

For $i > 2$, $\frac{\partial E[N]}{\partial N_{i-1}} = 0$ depends only on $\{N_{i-2}, N_{i-1}, N_i\}$ as follows:

$$\frac{\partial E[N]}{\partial N_{i-1}} = Q\left(\frac{\frac{k}{N_{i-1}} - \mu}{\sigma}\right) + (N_{i-1} - N_i)Q'\left(\frac{\frac{k}{N_{i-1}} - \mu}{\sigma}\right) - Q\left(\frac{\frac{k}{N_{i-2}} - \mu}{\sigma}\right).$$

Thus we can solve for N_i as

$$N_i = \frac{Q\left(\frac{\frac{k}{N_{i-1}} - \mu}{\sigma}\right) + N_{i-1}Q'\left(\frac{\frac{k}{N_{i-1}} - \mu}{\sigma}\right) - Q\left(\frac{\frac{k}{N_{i-2}} - \mu}{\sigma}\right)}{Q'\left(\frac{\frac{k}{N_{i-1}} - \mu}{\sigma}\right)}. \tag{1.23}$$

Actually, for each possible value of N_1, SDO can be used to produce an infinite sequence of N_i values that solve (1.19). Each such sequence is an optimal sequence of increments for a given

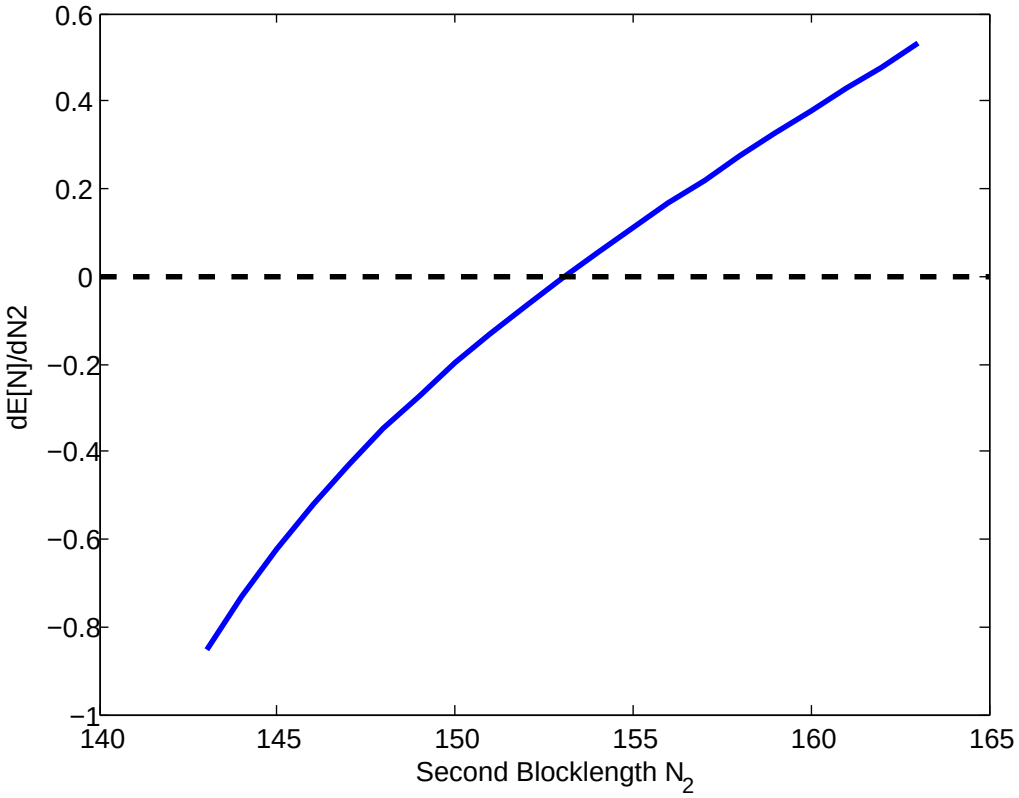


Figure 1.15: The derivative of the expected average blocklength with respect to the size of the second increment is zoo at the optimal point where the average blocklength is minimized for the choice of the second blocklength size N_2.

density of retransmission points on the transmission axis. As N_1 increases, the density decreases. Using SDO to compute the optimal m points is equivalent to selecting the most dense SDO-optimal sequence that when truncated to m points results in the highest throughput. Fig. 1.14 shows that the expected average blocklength as a function of the size of the second increment is convex. Therefore, there is a minimum average blocklength for the optimal choice of N_2. As shown in Fig. 1.15, the optimal choice of N_2 makes the derivative of the average blocklength with respect to the size of the second increment equal zero.

Table 1.3: Optimized $\{N_1, N_2, \cdots, N_m\}$, R_T, and λ from ES and SDO for k = 96 bits for VLFT on a 2 dB SNR binary-input AWGN channel using μ_S = 0.6374 and σ_S = 0.0579.

Alg.	m	$\{N_1, N_2, \cdots, N_m\}$	R_T	λ
ES, SDO	2	158 , 188	0.566	169.6
ES	3	150, 167, 194	0.58638	163.71
SDO	3	150, 167, 195	0.58635	163.72
ES	4	146, 158, 172, 198	0.59709	160.77
SDO	4	146, 158, 172, 197	0.59707	160.78
ES, SDO	5	143, 153, 163, 176, 201	0.603	159.2
ES, SDO	6	140, 149, 157, 166, 179, 204	0.608	157.9
ES, SDO	7	139, 147, 154, 161, 170, 182, 206	0.611	157.1

1.4.3 Application to VLFT with m Transmissions

Table 1.3 shows the optimized $\{N_1, N_2, \cdots, N_m\}$, resulting throughput R_T, and expected block-length $\lambda = k/R_T$ for various m. The values obtained by SDO are very close to the values obtained by ES.

For $m = 2, 5, 6,$ and 7, the optimized blocklengths for both approaches are the same. For $m = 3$ and 4 the blocklengths differ only in the value of N_m (shown in bold) and only by one bit. This small difference in N_m causes a negligible difference in the maximum throughput R_T and minimum expected blocklength $\lambda = \frac{k}{R_T}$. Since the complexity of ES is exponential in m, it is infeasible to obtain a globally optimal solution for $m > 7$; whereas SDO, with complexity $O(N_{max} - N_0)$, can find a solution within seconds even for large m.

Fig. 1.17 and Fig. 1.16 show the optimum R_T and λ for various m using SDO. The dashed lines show the maximum achievable R_T and the corresponding minimum achievable λ with an

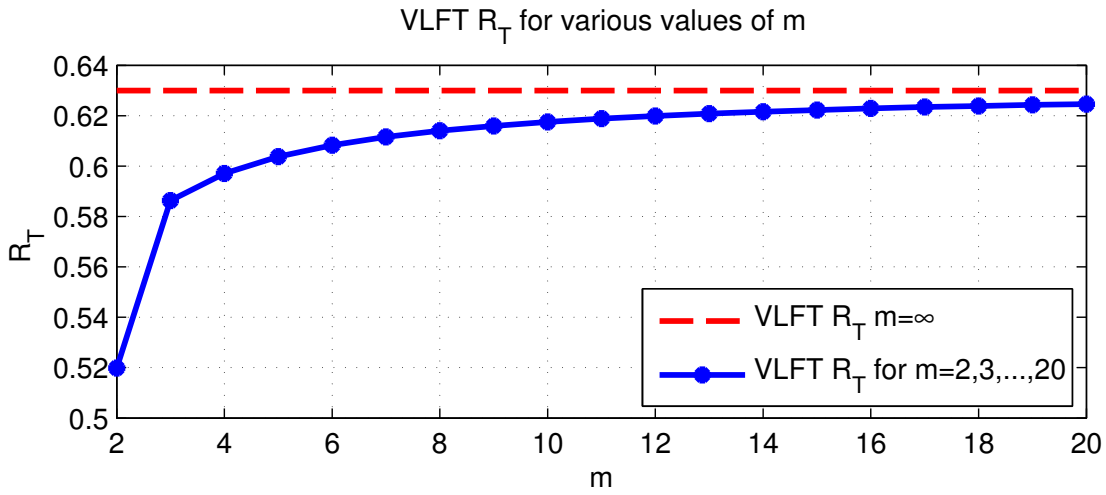


Figure 1.16: Throughput (R_T) and the expected blocklength (λ) as a function of the number of transmissions m achieved by non-binary LDPC codes in the VLFT setting for $k = 96$.

Table 1.4: Optimized R_T, and λ using SDO for various values of m with $k = 96$ bits for VLFT on a 2 dB SNR binary-input AWGN channel using $\mu_S = 0.6374$ and $\sigma_S = 0.0579$.

m	2	3	5	10	15	20	∞
λ	167.1	163.7	159	155.5	154.3	153.7	151.9
R_T	0.566	0.586	0.603	0.618	0.622	0.624	0.63742

unlimited m as in [35]. As a function of m, R_T quickly converges to the $m = \infty$ asymptote and even for $m \approx 10$ the throughput is close to the value achievable with an unlimited number of increments. Correspondingly, the expected latency also converges quickly and for $m \approx 10$ the expected blocklength is close to the minimum λ achievable by unlimited transmissions of one bit at a time.

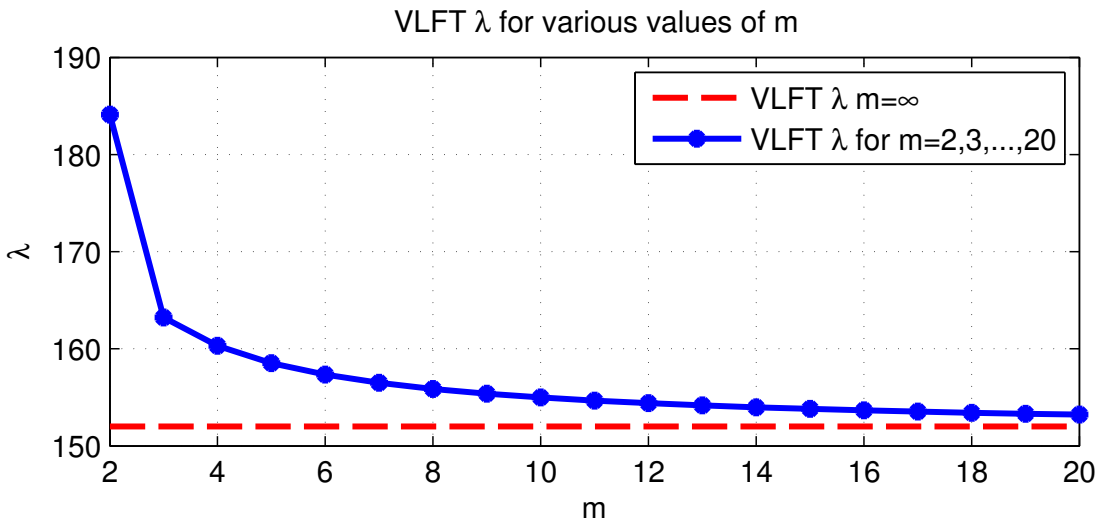


Figure 1.17: Throughput (R_T) and the expected blocklength (λ) as a function of the number of transmissions m achieved by non-binary LDPC codes in the VLFT setting for $k = 96$.

1.5 VLF with CRC

In this section, similar to the case of $m = \infty$ VLFT, the transmitter sends one bit of incremental redundancy at a time until the decoder converges to the correct codeword or converges to an incorrect codeword that passes the CRC check. The difference here compared with the VLFT scheme is that instead of using NTC as a genie, cyclic redundancy check (CRC) codes are used as error-detecting codes to detect whether there is an error in the decoded message. In systems incorporating CRCs, a certain number of check bits, L_{crc}, are computed and added to the information message of length k_{inf}. Fig 1.18 shows how the checksum bits are computed by initially appending the information bits with an all-zero block of length L_{crc} and then computing the remainder of the division by a divisor polynomial of degree $L_{crc} + 1$. The total block containing the original information block and the additional CRC check bits are finally encoded by the NB-LDPC encoder and the encoded message is transmitted over the communication channel.

At the receiver, the NB-LDPC decoder initially attempts to decode the received block. If decoding results in a codeword, the CRC check determines whether the check bits agree with the data by computing the checksum from the first k_{inf} bits of the received data and comparing this

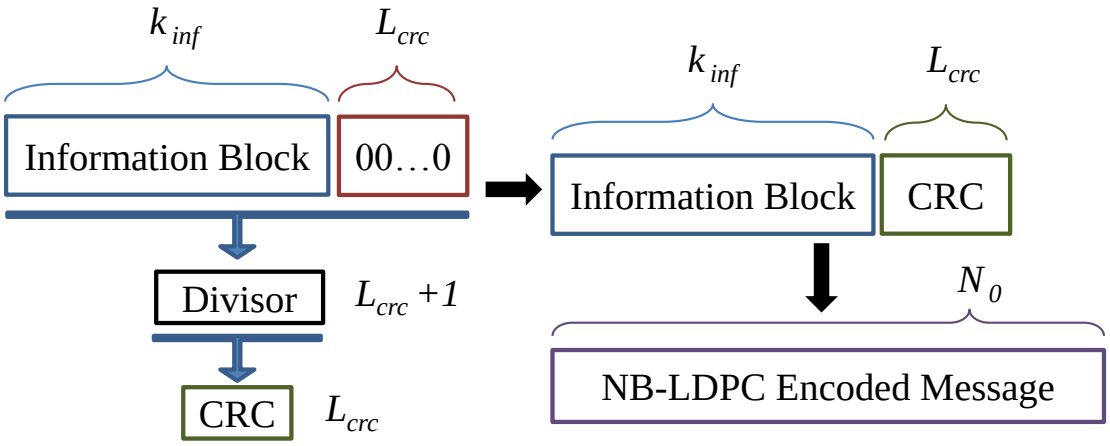


Figure 1.18: The block diagram corresponding to coding with CRC.

checksum with the last L_{crc} received bits. Alternatively, the receiver can divide all received bits by the generator polynomial and check if the L_{crc}-bit remainder is all-zero. In order to achieve an undetected error probability of ϵ, of the CRC code length L_{crc} is chosen so that the overall probability of error resulting from the NB-LDPC and CRC codes combined is smaller than ϵ.

The transmitter terminates transmission when the receiver sends feedback indicating that the decoded message passes the CRC check correctly. If the message is correctly decoded, it passes the CRC detection of CRC and the transmitter moves on to the next message. If the message is decoded incorrectly and the decoded message fails to pass CRC, the transmitter sends more bits to increase reliability of the bits already transmitted. if the receiver decodes the message incorrectly and the erroneously decoded message passes the CRC check, the transmitter moves on to the next message and the packet is assumed in error. This error is undetected by the receiver. In the case of unlimited transmissions ($m = \infty$), the transmitter transmits one bit at a time until the decoder either decodes the message correctly or until it decodes to a message that passes the CRC check.

With a limited number of transmissions, the blocklength corresponding to each transmission and the length of CRC are chosen to guarantee a probability of undetected error of at most ϵ. If the message is not decoded correctly even after m transmissions (and the NACKs are correctly received), the receiver deletes all received symbols and a new transmission cycle begins with the transmitter sending the original block of N_1 symbols.

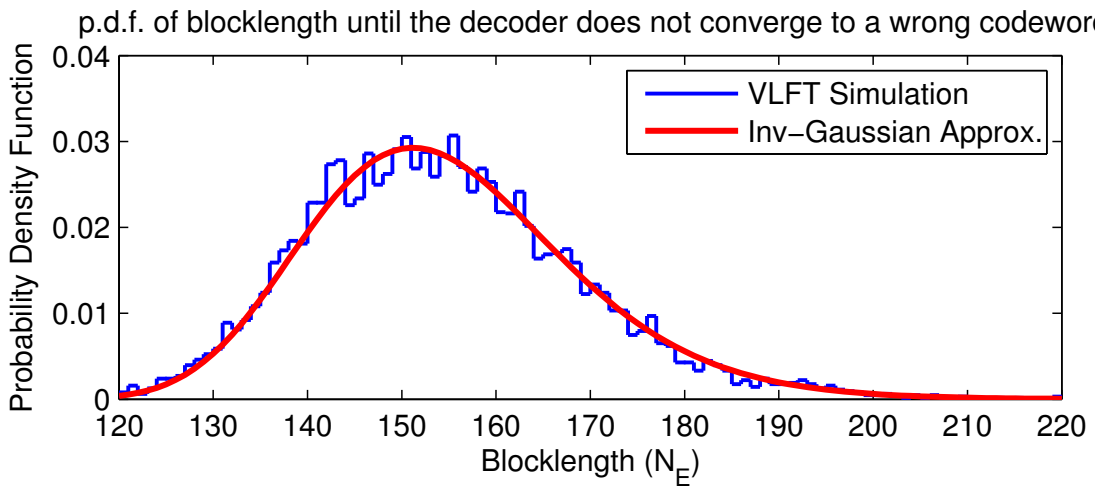


Figure 1.19: Empirical p.m.f. and reciprocal-Gaussian fit for the shortest cumulative blocklength (N_E) after which decoding never again converges to an incorrect codeword. The smallest blocklength for the GF(256) NB LDPC code is $N_0 = 120$ bits with $k = 96$ information bits. Thus, the initial rate is $R_0 = \frac{k}{N_0} = 0.8$.

Since the CRC as an error detection tool is used only when the decoder converges to a codeword, it is crucial to differentiate between *erroneous* decoding and failure to converge to a codeword. Fig. 1.19 shows the empirical p.m.f. of the required cumulative number of symbols (N_E) until the receiver will never again converge to an incorrect codeword. Note that Fig. 1.19 is conditioned on the decoder initially decoding to a wrong codeword at $N_0 = 120$. The probability that the decoder decodes incorrectly at N_0 is γ. (For the experiment that produced the p.m.f. in Fig. 1.19 $\gamma = 0.165$.)

For blocklengths larger than N_E, the decoder either decodes correctly or fails to converge to any codeword. This is a different condition than correct decoding, which was modeled in Figs. 1.7 and 1.8. Fig. 1.20 shows the empirical p.m.f. of $R_E = \frac{k}{N_E}$, the instantaneous rate at which the decoder stops decoding to the wrong codeword, and the corresponding Gaussian approximation.

Fig. 1.21 shows the state diagram representing all the scenarios that can happen based on our simulations. According to our simulations, if the decoder converges to a wrong codeword, it continues to decode to the same wrong codeword even with additional incremental transmissions.

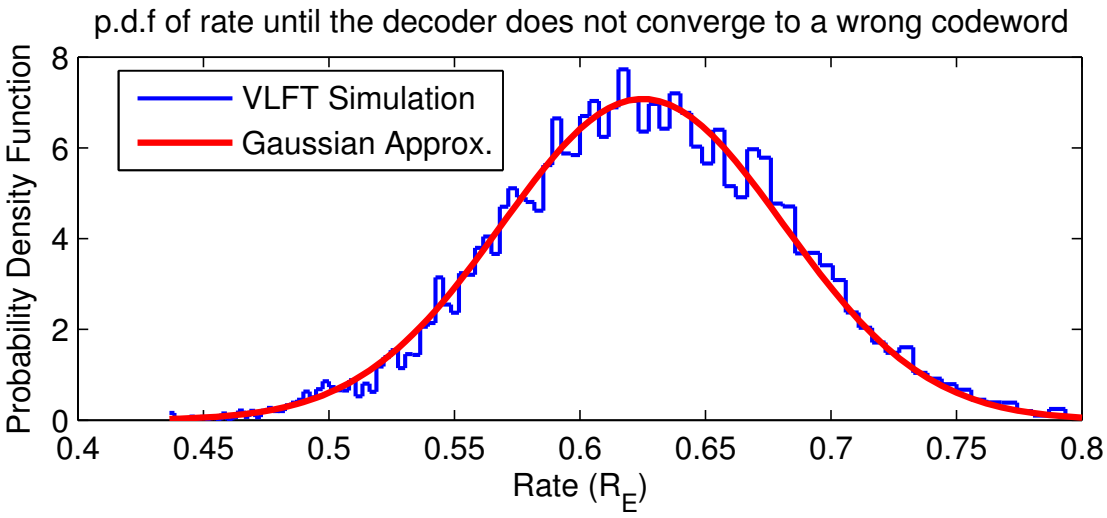


Figure 1.20: Empirical p.m.f. and Gaussian approximation with $\mu_E = 0.626$ and $\sigma_E^2 = 0.056$ of R_E in VLFT setting.

The increased reliability from incremental transmissions never moves the decoder from one wrong codeword to another wrong codeword. It only helps the decoder either to converge to the correct codeword or not to converge to any codeword at all. Figs. 1.7, 1.8 correspond to the blocklength and rate of entry to state 3. Figs. 1.19, 1.20 correspond to the blocklength and rate of leaving state 1 where the decoder converges to a wrong codeword.

We require an undetected error probability of smaller than ϵ. If the transmission starts with a blocklength of length N_0, the total probability of error is $\gamma \times 2^{-L_{\text{crc}}}$, where $2^{-L_{\text{crc}}}$ is approximately the probability of error that the CRC checks for a wrong codeword. We use standard CRC codes in this chapter. However, for the best error detection, the CRC codes can be designed specifically for a particular code as shown in [40].

For the error probability constraint of ϵ, we choose the length of the CRC code so that $\gamma \times 2^{-L_{\text{crc}}} < \epsilon$. For example, if ϵ is set to be 10^{-3} and $\gamma = 0.165$, the length of the CRC code $L_{\text{crc}} = 8$ is required to guarantee the overall probability of error, $\gamma \times 2^{-L_{\text{crc}}} = 6.25 \times 10^{-4} < \epsilon = 10^{-3}$.

As will be illustrated in the results section (Section 1.7), the throughput of this scheme can be well predicted by the results obtained from VLFT with unlimited transmissions (Section 1.4)

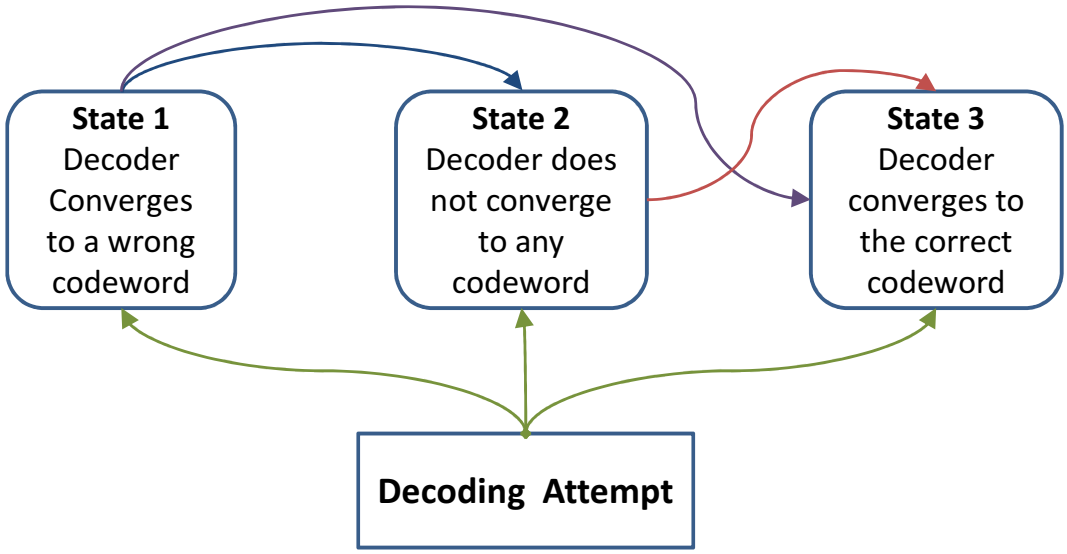


Figure 1.21: The state diagram corresponding to LDPC coding with incremental transmissions.

modified by a factor of $\frac{k-L_{\text{crc}}}{k}$ that captures the back-off in rate due to the CRC overhead. For example, in our previous analysis from Table 1.3 for $m = \infty$, the rate is 0.632 while with a CRC of length 8, for $k_{\text{inf}} = 96 - 8 = 88$ the rate is predicted to be $\frac{96-8}{96} \times 0.632 = 0.579$. As the simulation results of Section 1.7 show, the actual achieved rate is 0.575 with an undetected error probability of 8.04×10^{-4}. We will discuss these results in more detail in Section 1.7.

1.5.1 VLF with CRC and Limited Transmissions

In VLF with a limited number of transmissions, the length of each incremental transmission should be selected to maximize $R_T = \frac{E[K|L_{\text{crc}}]}{E[N]}$, where $E[N]$ is given by (1.15) and $E[K|L_{\text{crc}}]$ is the effective number of transmitted information bits, computed as

$$E[K|L_{\text{crc}}] = (K - L_{\text{crc}}) \left[Q\left(\frac{\frac{K}{N_m} - \mu_S}{\sigma_S} \right) - P_{N_1} 2^{-L_{\text{crc}}} \right], \tag{1.24}$$

under the constraint that the probability of undetected error $P_{N_1}\, 2^{-L_{\text{crc}}} < \epsilon$. P_{N_1} is the probability of converging to an incorrect codeword at blocklength N_1.

An approximation technique similar to the one used in optimizing the length of each incre-

Table 1.5: Optimized $\{N_1, \cdots, N_m\}$ for m=5 in VLF-with-CRC using SDO for different values of L_{crc}. The exact same values were obtained by ES.

L_{crc}	$\{N_1, N_2, \cdots, N_5\}$	λ	R_T	ϵ
1	193, 198, 205, 216, 241	193.27	0.49	8.95×10^{-4}
2	187, 192, 199, 210, 235	187.38	0.50	9.02×10^{-4}
3	180, 185, 192, 203, 228	180.67	0.51	9.82×10^{-4}
4	174, 180, 187, 198, 222	175.14	0.52	9.14×10^{-4}
5	166, 172, 180, 192, 216	168.48	0.54	9.62×10^{-4}
6	157, 164, 172, 184, 209	162.68	0.55	9.58×10^{-4}
7	143, 153, 163, 176, 201	159.14	0.56	9.44×10^{-4}
8	143, 153, 163, 176, 201	159.07	0.55	4.72×10^{-4}
9	143, 153, 163, 176, 201	159.04	0.54	2.36×10^{-4}
10	143, 153, 163, 176, 201	159.02	0.54	1.18×10^{-4}

mental redundancy block in VLFT is used here: $\left[Q\left(\frac{\frac{k}{N_m}-\mu_S}{\sigma_S}\right) - P_{N_1}\ 2^{-L_{\text{crc}}}\right] \approx 1$. The optimization problem of maximizing $R_T = \frac{E[K|L_{\text{crc}}]}{E[N]}$ reduces to minimizing $E[N]$ for each L_{crc}. The SDO technique used in Section 1.4 can be used here under the additional constraint that $P_{N_1}\ 2^{-L_{\text{crc}}} < \epsilon$.

For each L_{crc}, the optimized $\{N_1, \cdots, N_m\}$ values for this case are identical for SDO and ES and the values are given in Table 1.5. For small values of L_{crc} we need to use a large value of N_1 to make sure $P_{N_1}\ 2^{-L_{\text{crc}}} < \epsilon$. As a larger value of L_{crc} is selected, N_1 and consequently $\{N_2, \cdots, N_5\}$ decrease while the error probability constraint is still satisfied. For $L_{\text{crc}} = 7$ the set of $\{N_1, \cdots, N_5\} = \{143, 153, 163, 176, 201\}$ minimizes the expected latency λ and maximizes R_T. For larger values of $L_{\text{crc}} > 7$, the set of optimum blocklengths does not change and only the overall probability of error decreases as the CRC length is increased.

The optimal set of blocklengths for $L_{\text{crc}} \geq 7$ and $m = 5$ is the same as the set for VLFT and $m = 5$ from Table 1.3. The intuition for this is that once L_{crc} is large enough that decoding decisions

are extremely reliable, the optimal blocklengths for VLF-with-CRC should match those of VLFT. Because the blocklengths are identical, the throughput R_T for m = 5 with L_{crc} = 7 can be computed by reducing the R_T in Table 1.3 to account for the overhead of the CRC. The reduction from the m = 5 VLFT rate R_T = 0.603 is $\frac{96-7}{96}$ where $\frac{96-7}{96} \times 0.603 = 0.559$ which corresponds to the R_T from Table 1.5 for L_{crc} = 7. While both SDO and ES give the same values for different L_{crc} values, the order of complexity for SDO is $O(L_{\text{crc}}(N_{max} - N_0))$ while with ES algorithm the complexity has the much larger order of $O\Big(L_{\text{crc}}\binom{N_{max}-N_0}{m}\Big)$. As the simulation results of Section 1.7 show, the actual achieved rate is 0.541 with an undetected error probability of 5.75×10^{-4}.

For the simulations in Section 1.7 the CRC code used for L_{crc} = 7 has a polynomial representation of 0x09 ($x^7 + x^3 + 1$). This CRC code has been used by Telecommunication Standardization Sector of the International Telecommunications (CCITT) which sets international communications standards. The CRC code used for L_{crc} = 8 has a polynomial representation of 0x07 ($x^8 + x^2 + x + 1$) and is used in MultiMedia Cards (MMC) and Secure Digital (SD) cards.

1.6 Two-Phase VLF

Now we consider the two-phase VLF model in which the transmitter (source) uses the primary communication channel to confirm whether the receiver (destination) has decoded to the correct codeword. As in [41], the two-phase incremental redundancy scheme has a *communication* phase followed by a *confirmation* phase.

Fig. 1.22 shows a block diagram for the two-phase communication scheme. Starting at the left, a message block of size N_1 is transmitted (communication phase). If the destination decodes correctly, the source sends a coded forward "ACK" on the same forward noisy channel to confirm the successful decoding (confirmation phase). If the destination decodes incorrectly, the source sends a coded forward NACK. The ACKs and NACKs are repetition codes of length A_1 symbols and are transmitted over the same forward noisy channel from the transmitter (source) to the receiver (destination). If the decoder does not converge to any codeword with N_1 symbols, the transmitter skips the unnecessary confirmation phase and immediately transmits the second increment of $N_2 - N_1$

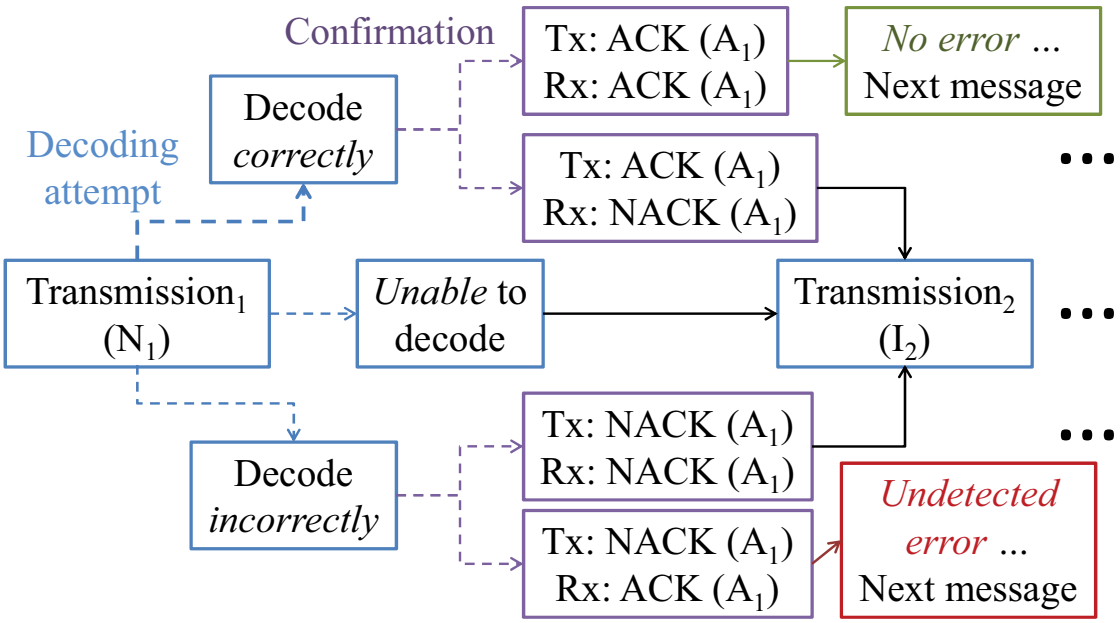


Figure 1.22: Two-phase VLF block diagram and the forward transmission stages in two-phase VLF systems.

bits.

Fig. 1.23 shows the first and second stages in decoding the message in two-phase VLF system. Initially, the receiver tries to decode the message by only receiving the blocklength of I_1. If it is able to decode the message, the transmitter encodes and sends the one bit information that whether the receiver has decoded the message the message correctly. The transmission continues afterward based on whether the receiver decodes the confirmation message correctly.

In the two-phase VLF setting, we use the probability distributions of N_S, R_S, N_E and R_E from Figs. 1.7, 1.8, 1.19, and 1.20. The optimization problem is to maximize $R_T = \frac{E[K]}{E[N]}$ where

$$E[K] = k\left(\sum_{i=1}^{m} P_i^{SS}\right) \approx k\left(Q\left(\frac{\frac{k}{N_m}-\mu_S}{\sigma_S}\right) - \sum_{i=1}^{m-1} P_i^{EE}\right), \tag{1.25}$$

with P_i^{EE} representing the probability the receiver decodes both the message and the NACK erroneously and P_i^{SS} is the probability the receiver decodes both message and ACK successfully. Note that (1.25) assumes (consistent with our simulation results) that once the decoder is in state 3 of Fig. 1.21, it does not return to state 1 even if a forward ACK is incorrectly received as a forward

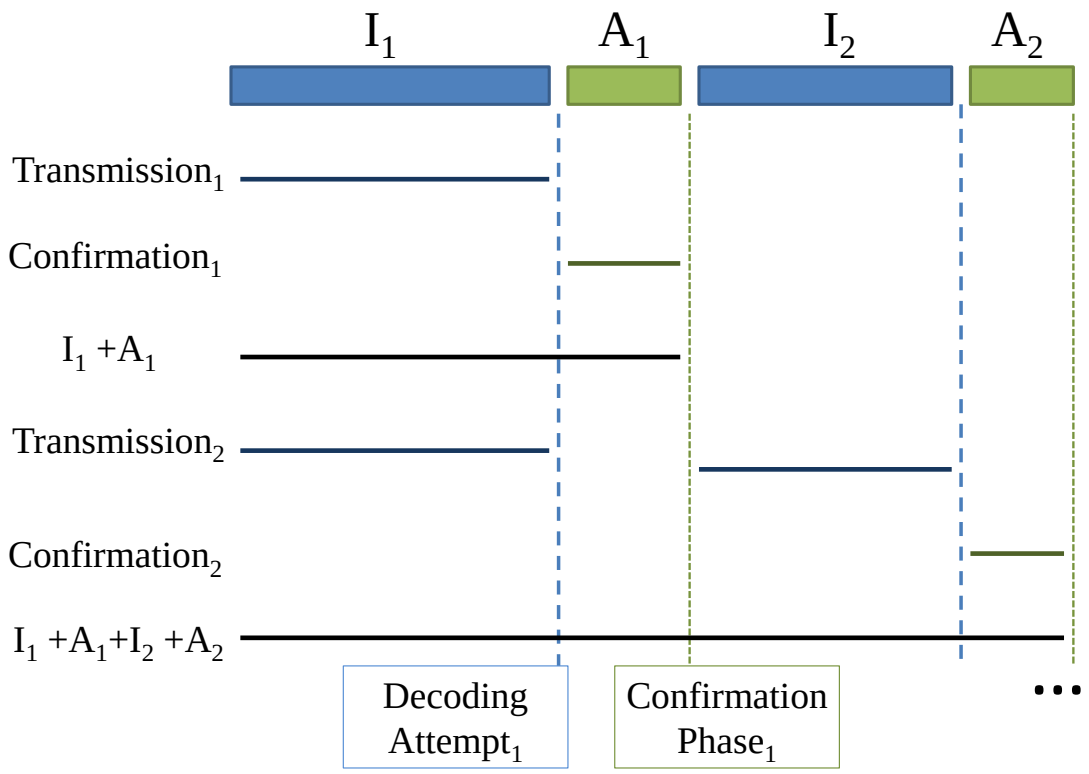


Figure 1.23: First and second stages of the transmission and confirmation phases in two-phase VLF systems.

NACK. In any case, as in Section 1.2 we assume $E[K] \approx k$.

The expected number of symbols transmitted in an AC is

$$E[N] = \sum_{i=1}^{m} (N_i + A_i)\left[P_i^{SS} + P_i^{EE}\right] + A_i\left[P_i^{SE} + P_i^{ES}\right] \tag{1.26}$$

$$+ N_m\left(1 - \left(\sum_{i=1}^{m} P_i^{SS} + \sum_{i=1}^{m} P_i^{EE}\right)\right), \tag{1.27}$$

where P_i^{SE} is the probability of decoding the message successfully but decoding the ACK as a NACK. Conversely, P_i^{ES} is the probability of decoding the message erroneously but decoding the NACK successfully. The term multiplying N_m in (1.27) is the probability that an AC ends without satisfying either of the stopping conditions. (1.27) is also approximated to $N_m\left(1 - Q\left(\frac{\frac{k}{N_m} - \mu_S}{\sigma_S}\right) + P_m^{SE}\right)$. The probabilities P_i^{SS}, P_i^{EE}, P_i^{SE}, and P_i^{ES} are computed as follows:

$$P_i^{SS} = \left[Q\left(\frac{\frac{k}{N_i}-\mu_S}{\sigma_S}\right)-Q\left(\frac{\frac{k}{N_{i-1}}-\mu_S}{\sigma_S}\right)\right]\left[1-Q\left(\frac{\sqrt{A_i}}{\sigma_c}\right)\right] \tag{1.28}$$

$$P_i^{EE} = \left[\gamma\left(1-Q\left(\frac{\frac{k}{N_i}-\mu_E}{\sigma_E}\right)\right)\right]\left[Q\left(\frac{\sqrt{A_i}}{\sigma_c}\right)\right] \tag{1.29}$$

$$P_i^{SE} = \left[Q\left(\frac{\frac{k}{N_i}-\mu_S}{\sigma_S}\right)-Q\left(\frac{\frac{k}{N_{i-1}}-\mu_S}{\sigma_S}\right)\right]\left[Q\left(\frac{\sqrt{A_i}}{\sigma_c}\right)\right] \tag{1.30}$$

$$P^{ES} = \left[\gamma\left(1-Q\left(\frac{\frac{k}{N_i}-\mu_E}{\sigma_E}\right)\right)\right]\left[\left(1-Q\left(\frac{\sqrt{A_i}}{\sigma_c}\right)\right)\right]. \tag{1.31}$$

In (1.28) the probability of decoding correctly at N_i and not at blocklengths smaller than or equal to N_{i-1} is $Q\left(\frac{\frac{k}{N_i}-\mu_S}{\sigma_S}\right) - Q\left(\frac{\frac{k}{N_{i-1}}-\mu_S}{\sigma_S}\right)$ and $Q\left(\frac{\sqrt{A_1}}{\sigma_c}\right)$ is the probability that the ACK is decoded as a NACK, where σ_c is the standard deviation of the channel noise. In (1.29), $\gamma\left[1 - Q\left(\frac{\frac{k}{N_i}-\mu_E}{\sigma_E}\right)\right]$ is the probability of decoding erroneously at N_i.

We optimize the blocklengths for two-phase VLF to maximize R_T under the constraint that $\sum_{i=1}^{m} P_i^{EE} < \epsilon$, using both ES and SDO approaches from Section 1.4 for fixed values of $\{A_1, \cdots, A_m\}$. For ES we considered values of $N_1 \leq N_2 \leq \cdots \leq N_m$ and constrained N_m to be no larger than the blocklength corresponding to a rate-0.1 code ($N_m \leq 10k$). For SDO we considered N_1 values ranging from the initial coding length N_0 to $3k$, which was the range that gave useful values of ϵ.

Table 1.6 shows two sets of $\{N_1, \cdots, N_m\}$ with $m = 5$ obtained for different N_1 in SDO with $\epsilon \approx 10^{-3}$. The optimized $\{N_1, \cdots, N_m\}$ with $\epsilon \leq 10^{-3}$ from ES is close to the SDO optimized blocklengths. The optimized blocklengths from SDO can also be used as optimization limits for ES algorithm and significantly reduce the ES optimization space.

Selecting a maximum size of the confirmation blocks (A_{max}), the order of complexity for SDA considering every set of values $\{A_1, \cdots, A_m\}$ is $O\left(A_{max}^m (N_{max} - N_0)\right)$. In contrast, ES requires $O\left(A_{max}^m \binom{N_{max}-N_0}{m}\right)$. ES is not feasible for all possible values of $\{A_1, \cdots, A_m\}$, but SDO can be used to explore the confirmation block sizes with ES employed to make small adjustments in the values of N_i for the best values of $\{A_1, \cdots, A_m\}$ according to SDO. As seen in Section 1.4 ES and SDO give very similar results, but SDO is much less complex.

1.7 Results

Fig. 1.24 shows R_T versus λ for NB-LDPC and convolutional codes using VLFT. In VLFT with an unlimited number of transmissions (1-bit increments), convolutional codes with ML decoders perform very well at short average blocklengths of up to 100 bits. VLFT schemes have throughputs greater than capacity at short blocklengths because of the NTC. Convolutional codes follow the marginal RCSP-ML (with unconstrained input) plot closely at short-blocklength with a small gap that is due to the binary input for convolutional codes. At longer blocklengths of about 200 bits, marginal RCSP-ML rate approaches the capacity. NB-LDPC codes outperform convolutional codes at longer blocklengths because the codeword error rate of convolutional codes increases once the blocklength exceeds twice the traceback depth [42] whereas the NB-LDPC code performance continues to improve with blocklength. The gaps between the throughputs for $m = \infty$, $m = 5$, and $m = 10$ NB-LDPC codes are similar to the gap observed in Fig. 1.17 . For $m = 10$ the performance of NB-LDPC codes in VLFT is much closer to the case of $m = \infty$. The NB-LDPC codes of Fig. 1.24 are over $GF(256)$. The shortest code for $k = 96$ bits has an initial blocklength of 15 $GF(256)$ symbols (120 bits), corresponding to an initial rate of 0.8. The NB-LDPC codes for $k = 192$ and $k = 288$ have initial blocklengths of 256 and 384 bits, respectively. Some results for the finite-m systems follow the non-active feedback scheme described in Section 1.2.3.

Fig. 1.25 shows the percentage of RCSP-ML rate for VLFT achieved by NB-LDPC and convolutional codes in VLFT. In the expected-blocklength range of 150-600 bits, NB-LDPC codes

Table 1.6: Optimized $\{N_1, \cdots, N_m\}$ for m=5 two-phase VLF using SDO and ES with $\{A_1, \cdots, A_5\}$ = $\{5, 4, 3, 3, 3\}$.

Alg.	k	$\{N_1, N_2, \cdots, N_5\}$	λ	R_T	ϵ
SDO	96	145, 156, 167, 180, 202	166.1	0.5779	1.2E-3
SDO	96	146, 158, 171, 188, 230	166.6	0.5762	9.4E-4
ES	96	146, 158, 170, 184, 211	166.4	0.5771	9.9E-4

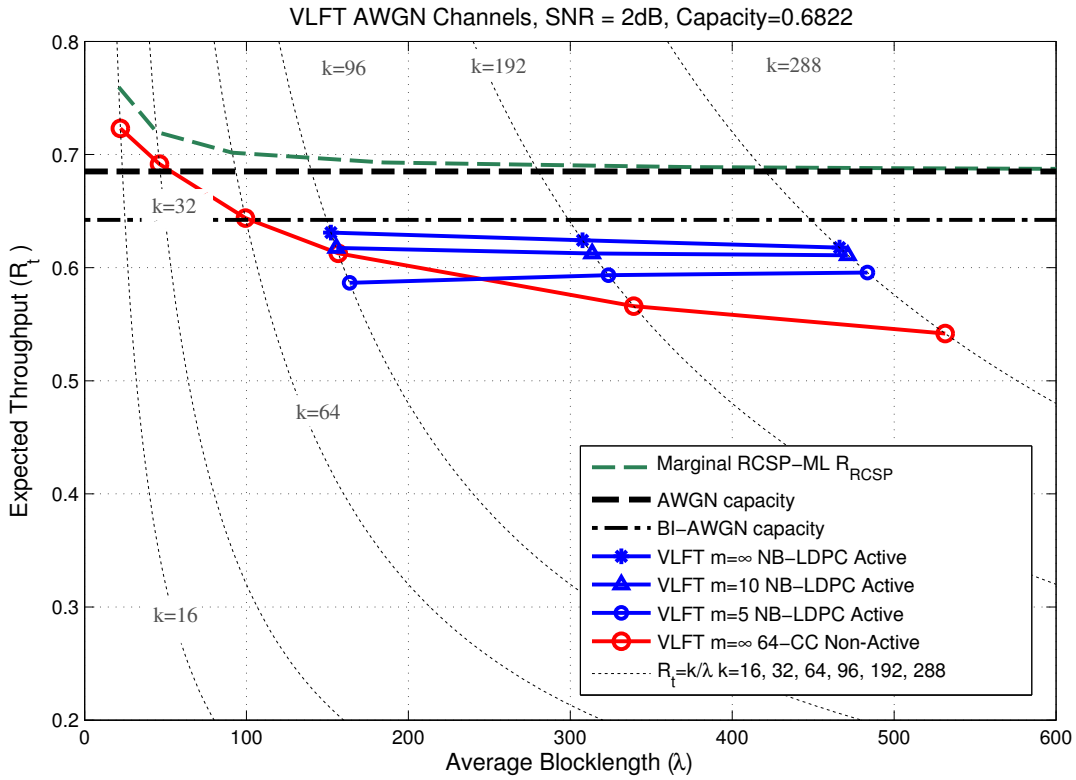


Figure 1.24: R_T vs. λ for NB-LDPC and 1024-state convolutional codes for VLFT with $m = \infty$, $m = 10$, and $m = 5$.

achieve a throughput of about 90% of RCSP-ML throughput (and about 91% and 96% of unconstrained and binary-input capacity, respectively) with an unlimited number of transmissions. When the number of the transmissions is limited to 10 and 5, the throughput percentage decreases to about 90% and 85%, respectively. RCSP-ML analysis is applied to the unconstrained-input AWGN channel at SNR 2-dB, for which the capacity is 0.684. The capacity of BI-AWGN channel at 2-dB SNR is 0.642 which is about 6% lower than the unconstrained-input AWGN capacity.

Table 1.7 summarizes the blocklengths that maximize the throughput in the two-phase VLF and VLF-with-CRC settings with ϵ=10^{-3}, for both NB-LDPC codes and (for comparison) tail-biting convolutional codes. Blocklengths for the NB-LDPC codes are obtained from (1.25-1.27) using ES on an optimization space limited by initial SDO results. Blocklengths for the convolutional codes are based on the coordinate-descent algorithm in [31] using the assumption of rate-compatible

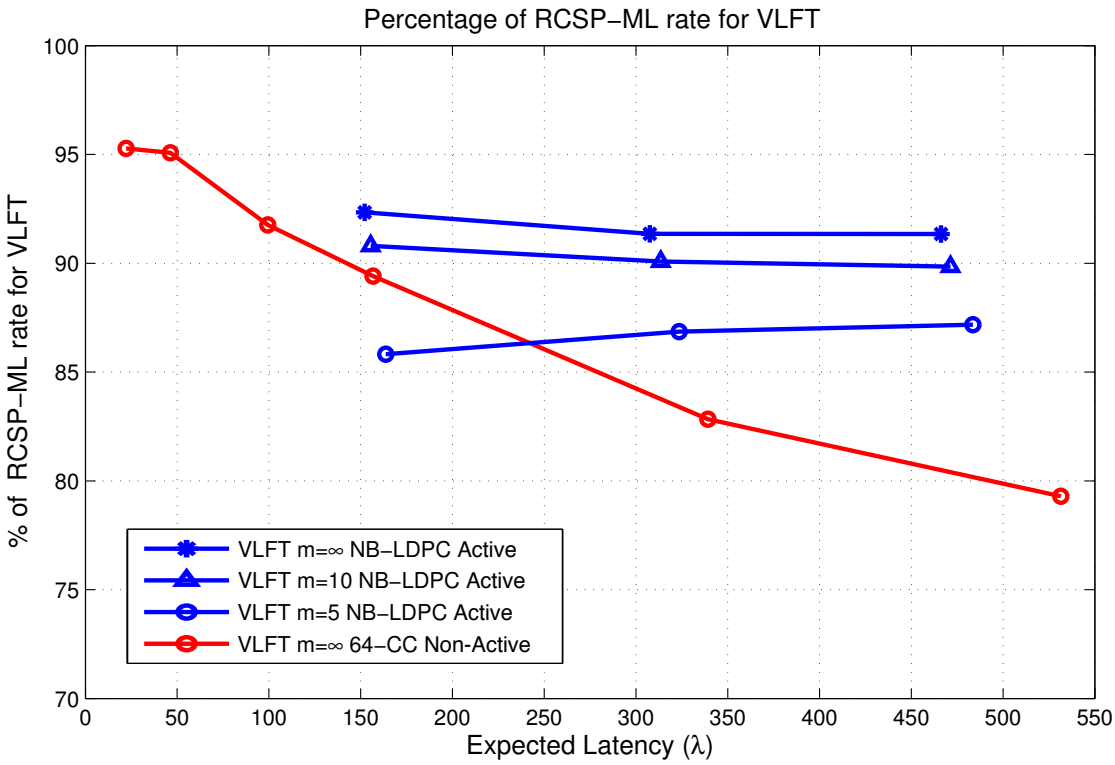


Figure 1.25: Percentage of VLFT R_T that NB-LDPC achieves with $m = \infty$, $m = 10$, and $m = 5$.

sphere-packing. Table 1.7 also shows the percentage of BI-AWGN capacity obtained in the two-phase VLF setting with $m = 5$ transmissions.

For $k = 192$ and 288, the NB-LDPC code obtains throughputs greater than 90% of BI-AWGN capacity with an average blocklengths λ of less than 500 bits in the 2-phase setting. NB-LDPC codes in the VLF-with-CRC setting with $m = 5$ achieve throughputs slightly lower than the ones in the 2-phase setting with $m = 5$. However, similar to Fig. 1.17 if m is increased to 10, VLF-with-CRC results in higher throughputs. Large values of m lead to a degradation in throughput performance for two-phase VLF due to the overhead associated with the more frequent forward ACK and NACK messages in the confirmation phase.

The rate-$1/3$ convolutional codes in Table 1.7 have octal generator polynomials $(117, 127, 155)$ for the 64-state code and $(2325, 2731, 3747)$ for the 1024-state code [35]. The NB-LDPC codes are described completely online[1].

[1]UCLA Communication Systems Laboratory (CSL) website at http://www.seas.ucla.edu/csl/resources/index.htm

Table 1.7: Optimized $\{N_1, \cdots, N_5\}$ for two-phase VLF and VLF-with-CRC with m=5 at SNR 2 dB, and corresponding R_T and λ values achieved in simulations. $\{A_1, \cdots, A_5\}$ = $\{5, 4, 3, 3, 3\}$ for two-phase VLF using NB-LDPC codes. For the convolutional codes, A_i = 6, 8, and 9 $\forall i$ for k = 96, 192, and 288 bits, respectively.

Code	k	$\{N_1, N_2, \cdots, N_5\}$	λ	R_T	%
CRC NB	89	143,153,163,176,201	164.5	0.541	84.2
2-Phase NB	96	146,158,170,184,211	170.4	0.563	87.7
2-Phase CC	96	138,153,166,180,204	168.6	0.569	88.6
CRC NB	185	293,309,325,346,386	323.4	0.572	89.1
2-Phase NB	192	301,322,344,369,408	330.5	0.581	90.5
2-Phase CC	192	287,309,331,352,384	349.4	0.549	85.4
CRC NB	281	459,487,518,550,597	491.3	0.572	89.1
2-Phase NB	288	459,487,518,550,597	495.7	0.581	90.5
2-Phase CC	288	416,441,463,488,532	599.6	0.480	74.8

Fig. 1.26 shows the throughput obtained in the VLF setting for NB-LDPC codes, 64-state and 1024-state tail-biting convolutional codes with m = 5, m = 10, m = ∞ for ϵ = 10^{-3}. As the blocklength increases, as mentioned in [34], the performance of the codes in VLF gets closer to the performance in VLFT. The plots for m = 5 are from Table 1.7. With m = ∞, the k = 89 the NB-LDPC code achieves a throughput greater than the random coding lower bound obtained from the analysis in [2].

Fig. 1.27 shows the percentage of the capacity of the BI-AWGN channel at 2-dB SNR achieved by NB-LDPC and convolutional codes using VLF. In the expected blocklength range of 300-500 bits, NB-LDPC codes with CRC achieve a throughput of about 94% of capacity with an unlimited number of transmissions. When the number of the transmissions is limited to 10, the throughput percentage decreases to about 93%. For m = 5, NB-LDPC codes perform slightly better in two-

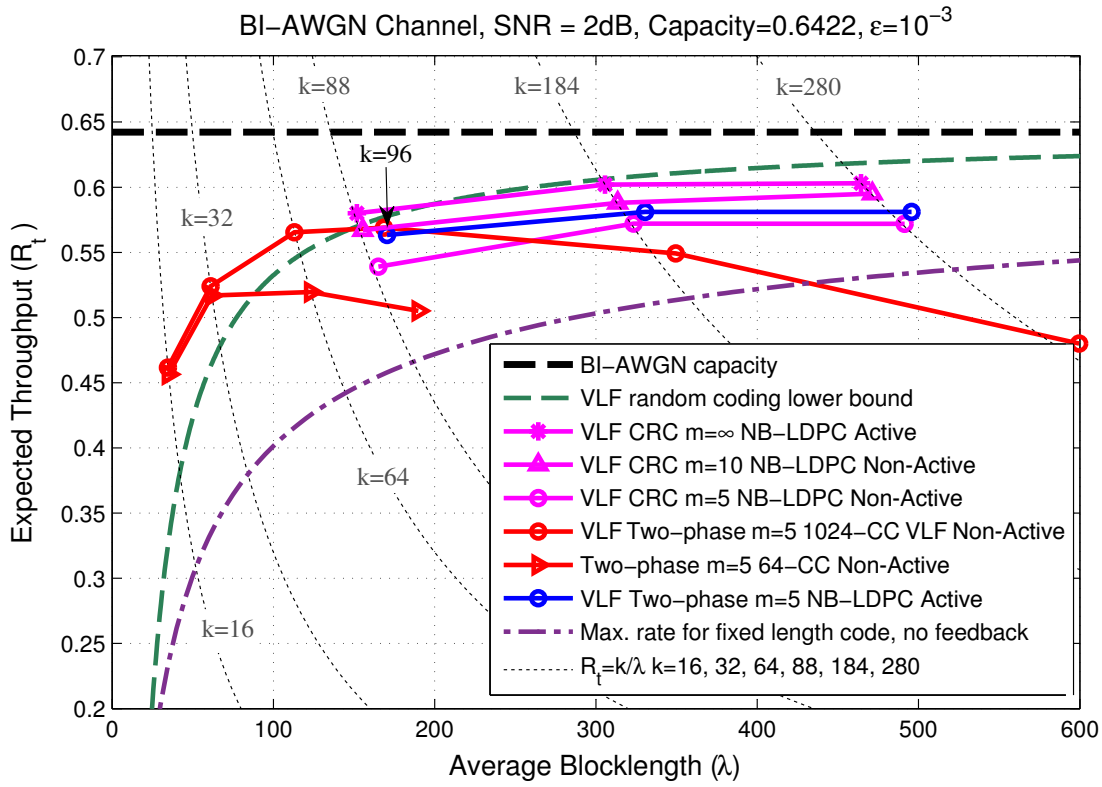


Figure 1.26: R_T vs. λ for NB-LDPC with $m = \infty$ in VLF-with-CRC and 64 and 1024-state convolutional codes and NB-LDPC codes with $m = 5$ in VLF.

phase VLF setting than in VLF-with-CRC. Note that for $m = \infty$ or even $m = 10$ two-phase VLF will not perform well because of the overhead associated with the confirmation messages.

As discussed in Section 1.3 similar Gaussian approximation analysis can be done for higher-SNR AWGN channels. for instance, for SNR-8dB AWGN channel which uses a larger 16-QAM constellation, the VLF-with-CRC system with an unlimited number of transmissions achieves a throughput of 2.37 bits per symbol with a frame error probability of less than 10^{-3}. This throughput corresponds to 88% of capacity in the blocklength regime of 40 16-QAM (quadrature amplitude modulation) symbols. Furthermore, the VLF-with-CRC system on 5-dB BI-AWGN fading channel with an unlimited number of transmissions achieves a throughput corresponding to 90% of capacity in the blocklength regime of about 140 bits.

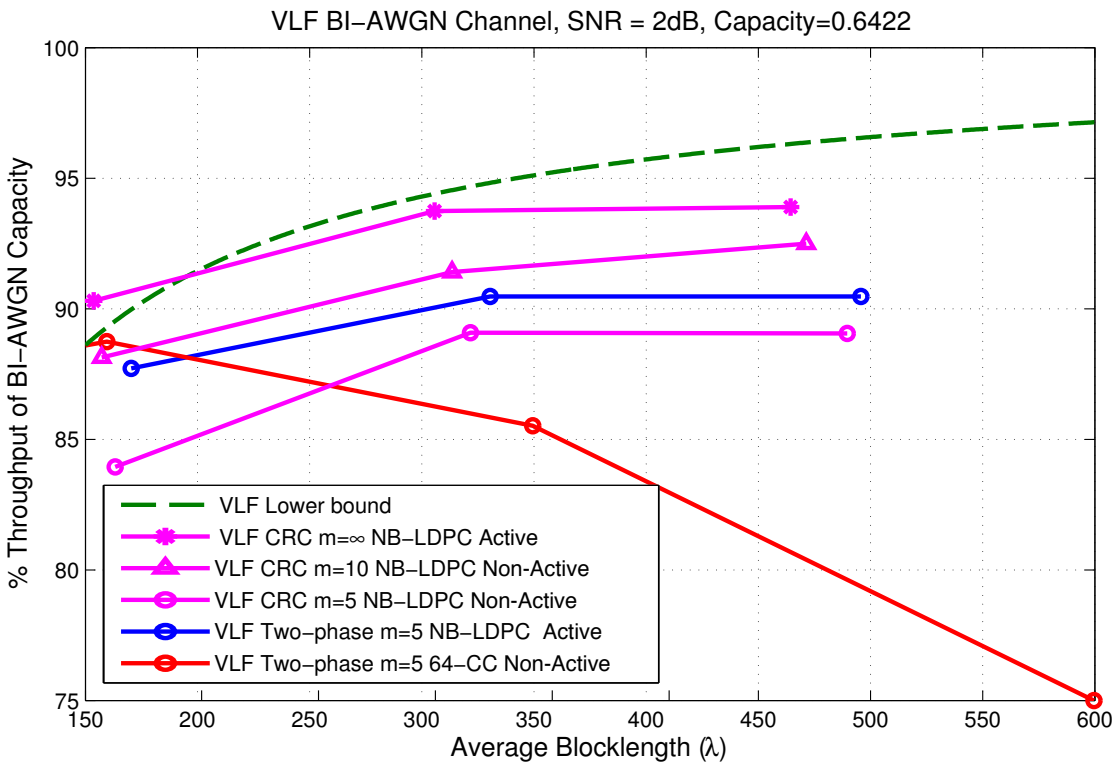


Figure 1.27: Percentage of BI-AWGN capacity that NB-LDPC and convolutional codes achieve in VLF.

1.8 Conclusions

We used the reciprocal-Gaussian approximation for the blocklength of first successful decoding to optimize the size of each incremental transmission to maximize throughput in VLFT and VLF settings. For feedback with a limitation on the number of transmissions, the sequential differential optimization (SDO) algorithm can be used quickly and accurately to find the optimal transmission lengths for a wide range of channels and codes. In this chapter, we discussed and applied SDO to non-binary LDPC codes for a variety of feedback systems. We focused on the binary-input AWGN channel but verified the effectiveness of the Gaussian approximation and SDO on the standard AWGN channel with a 16-QAM input and on a fading channel. In the 300-500 bit average blocklength regime, this work reports the best VLFT and VLF throughputs yet. VLFT throughputs are higher than VLF, but VLF is more practical because it does not assume a noiseless transmitter

confirmation symbol. For VLF-with-CRC with $m = \infty$, NB-LDPC codes with optimized blocklengths achieve about 94% of the capacity of 2-dB BI-AWGN channel for an average blocklength of 300-500 bits. In the same blocklength regime, for VLF-with-CRC with $m = 10$, NB-LDPC codes with optimized blocklengths achieve about 93% of the capacity.

The performance results can also be considered in terms of SNR gap. In Fig. 1.27, the random-coding lower bound for a system with feedback is 0.27 dB from the Shannon limit for $k = 280$ with a blocklength of less than 500 bits. Looking at the VLF-CRC NB-LDPC codes for $k = 280$ in Fig. 1.27, the $m = \infty$ NB-LDPC code is 0.53 dB from Shannon limit. The NB-LDPC non-active feedback system in Fig. 1.27 uses ten rounds of single-bit feedback to operate within 0.65 dB of the Shannon limit with an average blocklength of less than 500 bits. Similar analysis can also be done for higher-SNR AWGN and fading channels.

CHAPTER 2

Polar Codes with Feedback

This chapter uses an extension of Reciprocal Channel Approximation (RCA) to accurately and efficiently predict the frame error rate (FER) performance of polar codes by analyzing the probability density function (p.d.f.) of log likelihood ratios (LLR) associated with information bits. A feedback scheme uses the RCA to predict the p.d.f of LLRs in conjunction with a repetition coding system to decrease the blocklength required for a target FER by a factor of 16. Using a rate-0.5 128-bit polar code as the initially transmitted code, the FER of the system with feedback is obtained by theoretical analysis and verified by simulation. Including the additional incremental transmissions the average blocklength for the system with feedback is 137.55 bits and the rate is 0.4653. Without feedback, a polar code with blocklength 2048 is required to achieve a comparable FER at a comparable rate. Intuitively, feedback allows the polar code to use fewer frozen bits in the initial transmission and then uses repetition codes to provide the needed reliability to resolve unreliable unfrozen bits identified by feedback.

2.1 Introduction

Polar codes are a class of modern error correcting codes introduced by Arikan [43]. These codes can achieve the capacity of binary-input symmetric discrete memoryless channels universally. Polar codes have an explicit construction using a kernel matrix and have low encoding and decoding complexity. Originally, Arikan [43] used the 2 x 2 kernel matrix F in (2.1) to show the effect of

polarization.

$$F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.1}$$

When a large number of these polarizing matrices are concatenated, the channel polarizes in the sense that the fraction of the transmitted bits that experience a noiseless channel to the total number of channel uses is equal to the capacity of the channel. The rest of the fraction of the bits are deemed useless as they experience very noisy synthetic channels. The information (unfrozen) bits are allocated to the noiseless synthetic channels and pre-determined values (usually zeros) are allocated to the noisy channels.

Researchers [44] have studied the polarization phenomenon with larger kernel matrices than (2.1). The general requirement for a square matrix to be a valid polarizing kernel matrix is that none of the column permutations of the matrix should result in an upper triangular matrix. Korada et al. [45] show that larger polarizing matrices can increase the speed of polarization. Asymptotically, the rate of polarization shows how fast the probability of error decays to zero as the blocklength B of the polar code approaches infinity. Arikan and Telatar [46] derived an upper bound suggesting that this rate scales exponentially in $-B^{\beta}$ where β is the rate of polarization. For kernel matrices smaller than 16×16, $\beta < 1/2$.

In this chapter, we use the original polarization kernel matrix (2.1) to analyze how feedback can help in achieving a target FER at a much smaller blocklength with feedback than without feedback. The figure of merit used in this work is the expected total number of forward channel uses required to achieve a particular FER. This total number of forward channel transmissions includes the blocklength of the polar code initially transmitted as well as the subsequent forward transmissions based on the feedback. The FER performance gain is compared with the FER of the no-feedback system with an average rate similar to the scheme where feedback is used. In other words, we take into account the overhead due to the additional transmissions and the consequent rate reduction in our comparison.

Reciprocal channel approximation (RCA) was used in [47, 48] to design LDPC codes. We use RCA to accurately estimate the FER of short blocklength polar codes (without feedback).

Moreover, RCA gives an approximation on the distribution of the LLR values of the information bits (the unfrozen bits).

The approximate distribution of the LLR values obtained by RCA informs the use of feedback during decoding. Feedback requests additional bits only when the reliability of the successive-cancellation-decoded information bits falls in a specified low-reliability region of the RCA-obtained distribution. A repetition code of a specified length is used to increase the reliability of those particular bits. Significant coding gain or equivalently polarization-rate improvement is obtained by this scheme. Furthermore, an RCA-based analysis accurately predicts the simulation performance of this feedback scheme.

The rest of the chapter is organized as follows: Sec. 2.2 gives an introduction to successive cancelation (SC) decoding of polar codes and the LLR calculation of the information bits. Sec. 2.3 provides the system model and discusses the application of RCA to polar codes. This section also compares the FER performance predicted by RCA with simulation results. Sec. 2.4 presents the feedback scheme, demonstrates its significant improvement of FER, and compares the RCA analysis of that scheme with simulation results. Furthermore, it defines an optimization problem to achieve a particular FER in short blocklength and shows the mathematical derivations for the results obtained in the this section. Sec. 2.5 concludes this chapter.

2.2 Polar Codes

Fig. 2.1 shows the graphical representation of the polarization matrix used in this section. The bits V_1 and V_2 are combined according to the kernel matrix (2.1) to give W_1 and W_2

$$W_1 = V_1 \oplus V_2 \tag{2.2}$$

$$W_2 = V_2 \tag{2.3}$$

From (2.2) and (2.3), V_1 can be calculated by $V_1 = W_1 \oplus W_2$. The probability of error for V_1 is high because if either W_1 or W_2 is decoded incorrectly, V_1 will be decoded incorrectly.

The LLR value for V_1 can be calculated as if V_1 is connected to a check node in an LDPC code

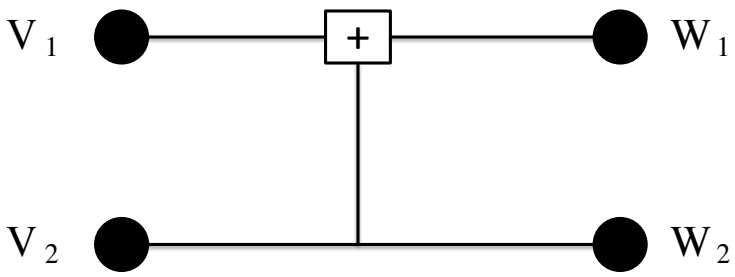


Figure 2.1: The graph of the kernel matrix.

that also connects to the variable nodes W_1 and W_2. The LLR value of V_1 can be calculated as:

$$LLR_{V_1} = 2\tanh^{-1}(\tanh(\frac{LLR_{W_1}}{2})\tanh(\frac{LLR_{W_2}}{2})). \tag{2.4}$$

If V_1 is in fact decoded correctly (or it has a value already determined) the probability of error for V_2 is significantly lower since the decision about V_2 is from two independent channel observations:

$$V_2 = W_2 \tag{2.5}$$

$$V_2 = W_1 \oplus V_1 . \tag{2.6}$$

Assuming V_1 is previously determined, the LLR value of LLR_{V_2} is calculated as follows:

$$LLR_{V_2} = \begin{cases} LLR_{W_1} + LLR_{W_2} & \text{if } V_1 = 0 \\ -LLR_{W_1} + LLR_{W_2} & \text{if } V_1 = 1 \end{cases} \tag{2.7}$$

Computation of the LLRs of all the bits in a polar code follow from repeated applications of (2.4) and (2.7).

In this chapter we consider a channel with binary input and additive white Gaussian noise (BI-AWGN) having signal-to-noise ratio (SNR) s_{ch} where equiprobable information bits are coded according to the polar code designed with the polarization matrix (2.1).

2.3 RCA for Polar Codes

The RCA for BI-AWGN channel in the study and design of LDPC codes uses a single real-valued parameter s, the SNR, to approximate the distribution of LLR messages exchanged between variable and check nodes. RCA for LDPC codes is a low complexity alternative to the density evolution algorithm. The RCA approach can be applied to polar codes with SC decoders.

Assume a BI-AWGN channel with input x and output y,

$$y = x + z, \tag{2.8}$$

where $z \sim N(0, \sigma_{ch}^2)$ is a white Gaussian noise sample and $x \in \{1, -1\}$ are binary-phase-shift-keying (BPSK) modulated channel inputs corresponding to 0 and 1 respectively. Note that for this scenario $s_{ch} = 1/\sigma_{ch}^2$. The bit LLR of a message received at the receiver is given by

$$L = \mathrm{Ln}\left(\frac{P(y|x=+1)}{P(y|x=-1)}\right) = \frac{2y}{\sigma_{ch}^2} = 2ys_{ch}\,. \tag{2.9}$$

Since polar codes are linear block codes, without loss of generality assume that the all-zero codeword is transmitted where all information and frozen bits are set to zero. Under the assumption that only $x = +1$ is transmitted over the channel, the channel output y will have a Gaussian distribution, $y \sim N(1, \sigma_{ch}^2)$. Consequently, the LLR message L is normally distributed with mean $E(L)$ and variance var(L) given by

$$\mathrm{E}(L) = \frac{2\mathrm{E}(y)}{\sigma_{ch}^2} = \frac{2}{\sigma_{ch}^2} \tag{2.10}$$

$$\mathrm{Var}(L) = \frac{4}{\sigma_{ch}^4}\mathrm{Var}(y) = \frac{4}{\sigma_{ch}^2}\,. \tag{2.11}$$

Define the reciprocal SNR as $r \in R$ such that

$$C(s) + C(r) = 1\,, \tag{2.12}$$

where $C(s)$ is the capacity of the BI-AWGN channel with SNR s. For computational simplicity and numerical precision we prefer to express $C(s)$ as

$$C(s) = 1 - \int_{-\infty}^{\infty} \log_2\left(1 + e^{-(2\sqrt{2s}u + 2s)}\right)\frac{e^{-u^2}}{\sqrt{\pi}}du, \tag{2.13}$$

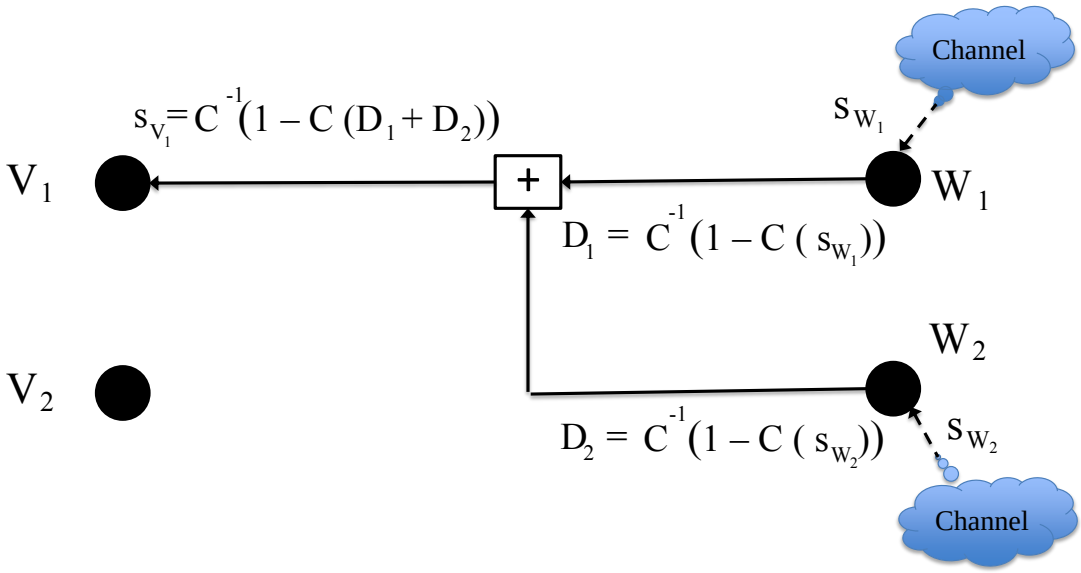


Figure 2.2: RCA to calculate the LLR p.d.f. of V_1

which is obtained from [49, (15)] through a change of variables. The self-inverting reciprocal energy function

$$R(s) = C^{-1}(1 - C(s)) \tag{2.14}$$

in [47] transforms between s and r: $r = R(s)$ and $s = R(r)$.

Let s_{ch} be the channel SNR and s_{W_i} for $i \in 1, 2$ be the SNR corresponding to the nodes W_i in Fig. 2.1. Using $C(s)$ and $C^{-1}(s)$ we can calculate the SNR observed at V_1. The function $R(s)$ makes the corresponding values D_i in Fig. 2.9 additive at the check node (+) in Fig. 2.9. Thus the SNR value at V_1 is calculated by $C^{-1}(1 - C(D_1 + D_2))$ as in Fig. 2.9. The distribution of V_1 is approximately Gaussian with a mean of $2s_{V_1}$ and variance of $4s_{V_1}$. Under the assumption that V_1 is correctly decoded ($V_1 = 0$), W_1 and W_2 provide two independent looks at V_2 and the SNR at V_2 is the summation of the SNR values observed at W_1 and W_2:

$$s_{V_2} = s_{W_1} + s_{W_2}\,. \tag{2.15}$$

The LLR distribution of each information bit (assumed by RCA to be Gaussian) is calculated at each stage of SC decoding. If any of the information bits is decoded in error the entire block will be decoded in error as the errors propagate in an SC decoder. The frame error probability is

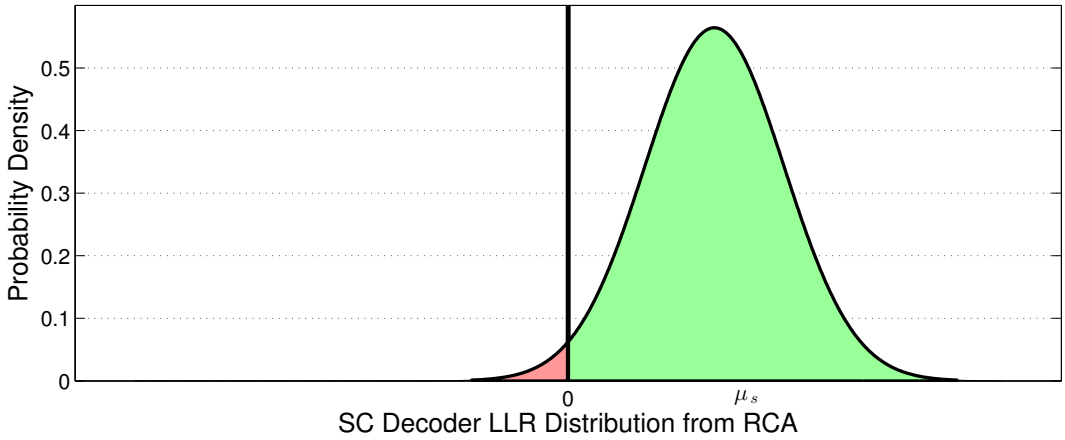


Figure 2.3: RCA to calculate the LLR p.d.f. of V_1

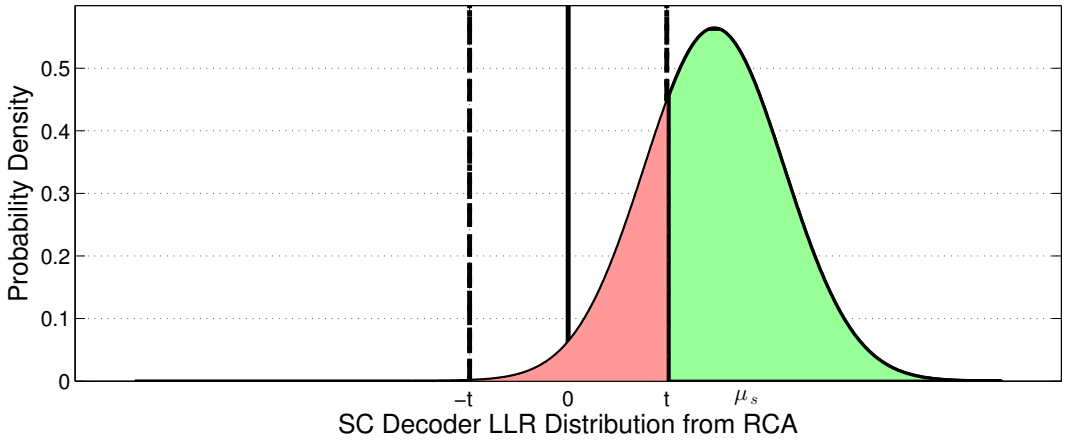


Figure 2.4: RCA to calculate the LLR p.d.f. of V_1

the probability that none of the bits is decoded incorrectly:

$$P_{FE} = 1 - \prod_{1}^{k} P(\hat{u}_i = 0 | u_1 = 0, \ldots, u_{i-1} = 0)\,. \tag{2.16}$$

Note that $\hat{u}_i$ is the decision about the information bit u_i for $i \in \{1, \ldots, k\}$, made by the SC decoder. Assuming all information bits from 1 to $i - 1 < k$ are decoded correctly, the SNR for each information (unfrozen) bit is s_i. The distribution is approximately $N(\mu_{i,\mathrm{sc}}, 2\mu_{i,\mathrm{sc}})$ where $\mu_{i,\mathrm{sc}} = 2s_{u_i}$. The probability that the i^{th} successively decoded information bit is in error under the assumption that all the previous information bits are decoded correctly is approximated by

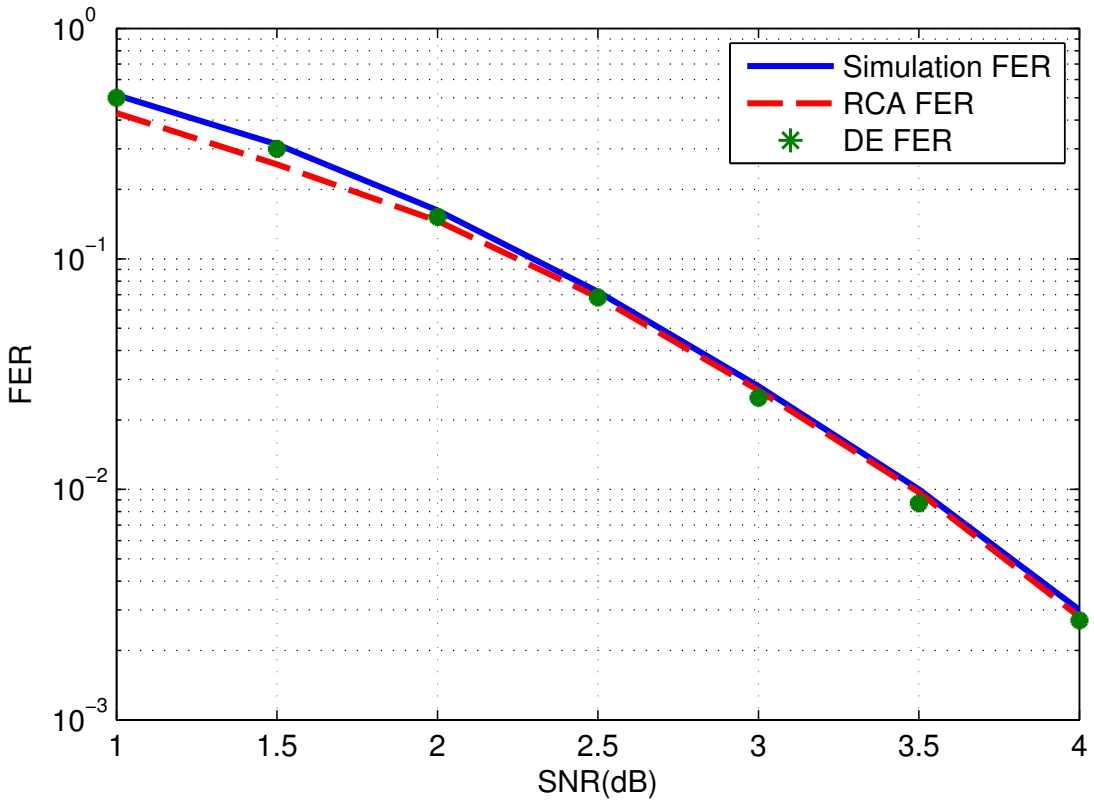


Figure 2.5: Simulation FER and FER predicted by RCA and DE for a rate-0.5 polar code of length 128 bits constructed using the original kernel matrix of (2.1).

$$P(\hat{u}_i = 0 | u_1 = 0, \ldots, u_{i-1} = 0) \approx Q\left(-\frac{\mu_{i,\text{sc}}}{\sqrt{2\mu_{i,\text{sc}}}}\right). \tag{2.17}$$

Thus the FER (P_{FE}) is approximately

$$P_{FE} \approx 1 - \prod_{i=1}^{k} Q\left(-\frac{\mu_{i,\text{sc}}}{\sqrt{\mu_{i,\text{sc}}}}\right). \tag{2.18}$$

Fig. 2.5 shows the FER predicted by RCA and illustrates how closely it follows the FER obtained from simulations and also density evolution (DE) for different SNR values.

Fig. 2.6 shows the probability of error for each information bit and its comparison to the RCA-predicted probability of error of (2.17). RCA can be effectively used to select which bits are more unreliable. Those bits can be selected as frozen bits to lower the rate and increase reliability.

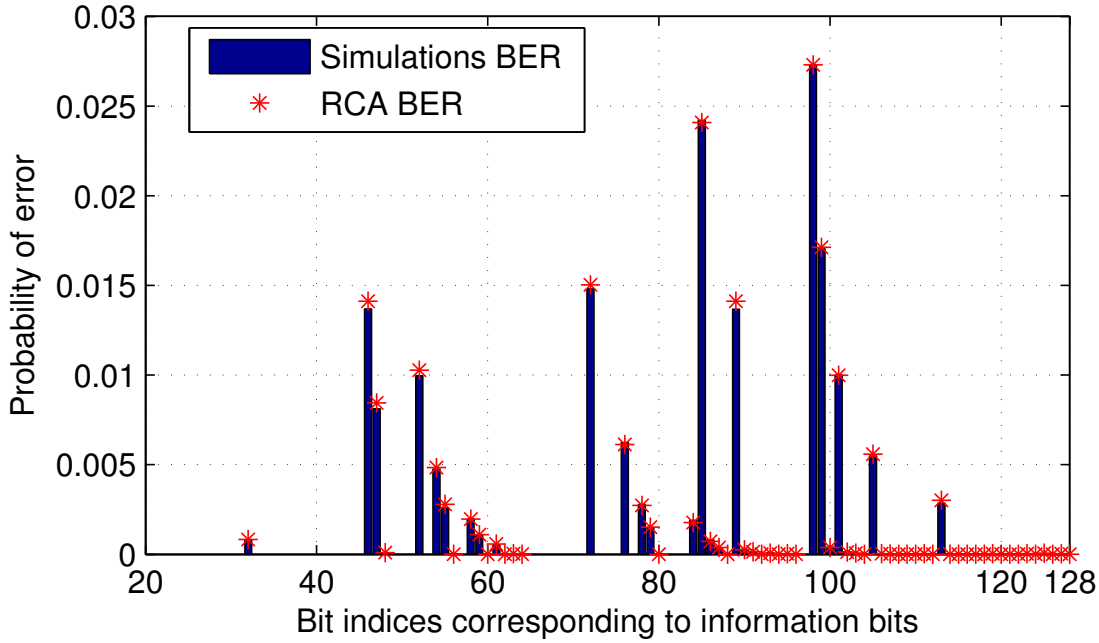


Figure 2.6: Information bit probability of error from simulation (bar plot) and RCA predicted probability of error as in (2.17).

2.4 Reduction in Blocklength to Achieve a Target FER

In this section we show how feedback can be used to reduce the blocklength required to achieve a specified FER. Essentially, feedback allows the polar code to use fewer frozen bits in the initial transmission and then uses repetition codes to provide the needed reliability to resolve unreliable unfrozen bits identified by feedback.

In order to use feedback to reduce the error probability of the SC decoded information bit u_i, a symmetric LLR threshold range around zero ($[-t_i, t_i]$) is determined. If the LLR from SC decoding is in this range, feedback instructs the transmitter to increase reliability by sending a repetition code of size n_i indicate the value of bit currently being decoded.

The probability that the LLR value of the SC-decoded bit u_i falls in the less reliable range of $[-t_i, t_i]$, where t_i is the threshold, is given by

$$P(-t_i < L_i < t_i) = Q\left(\frac{-t_i - \mu_{i,\text{sc}}}{\sigma_{i,\text{sc}}}\right) - Q\left(\frac{t_i - \mu_{i,\text{sc}}}{\sigma_{i,\text{sc}}}\right). \tag{2.19}$$

If the LLR L_i of u_i is in the range of $[-t_i, t_i]$, feedback initiates the use of a repetition code to increase the number of independent observations of the channel and consequently increase the reliability of u_i. The probability of having the increased LLR value (L_i') become positive and hence be decoded correctly is

$$P(L_i' > 0 | -t_i < L_i < t_i), \tag{2.20}$$

where L_i' includes the additional reliability $L_{i,\text{rc}}$ from the repetition code, i.e. $L_i' = L_i + L_{i,\text{rc}}$.

The total probability of correctly decoding the information bit u_i under the assumption that all the previous information bits are decoded correctly is

$$P(\hat{u}_i = u_i) = P(L_i > t_i) + P(L_i' > 0, |L_i| < t_i)\,. \tag{2.21}$$

The second term (joint probability) in (2.21) can be decomposed into the product of two terms $P(L_i' > 0 | -t_i < L_i < t_i) P(-t_i < L_i < t_i)$.

Note that $P(\hat{u}_i = u_i)$ depends on the length of the repetition code as well as the threshold t_i. The selection of these parameters will be discussed later in this in section.

Under the assumption that $u_i = 0$, for a repetition-coded signal with a blocklength of n_i, $L_{i,\text{rc}}$ is normally distributed with a mean of $\mu_{i,\text{rc}} = n_i(\frac{2}{\sigma_{ch}^2}) = 2n_i s_{ch}$ and a variance of $\sigma_{i,\text{rc}}^2 = n_i(\frac{4}{\sigma_{ch}^2})$. For simplicity of notation, the index i is omitted in the following. For the general case where $t_i, \mu_{i,\text{sc}}$ and $\sigma_{i,\text{sc}}$ of (2.19) are represented by t, μ_{sc}, and σ_{sc} (for successive cancellation) respectively, the probability $P(L' > 0 | -t < L < t)$ has the following expression:

$$\int_0^\infty c_1 e^{c_2(v)} \left[Q\left(\frac{v - G_1}{G_2}\right) - Q\left(\frac{v + G_1}{G_2}\right) \right] dv\,, \tag{2.22}$$

where

$$c_1 = \frac{\sqrt{\frac{1}{\sigma_{sc}^2 + \sigma_{rc}^2}}}{\sqrt{2\pi}\left(Q\left(\frac{-t - \mu_{\text{sc}}}{\sigma_{\text{sc}}}\right) - Q\left(\frac{t - \mu_{\text{sc}}}{\sigma_{\text{sc}}}\right)\right)} \tag{2.23}$$

$$c_2(v) = \frac{-(v - \mu_{\text{sc}} - \mu_{\text{rc}})^2}{2(\sigma_{\text{sc}}^2 + \sigma_{\text{rc}}^2)} \tag{2.24}$$

$$G_1 = \frac{t(\sigma_{\text{sc}}^2 + \sigma_{\text{rc}}^2)}{\sigma_{\text{sc}}^2} \tag{2.25}$$

$$G_2 = \sqrt{(\sigma_{\text{sc}}^2 + \sigma_{\text{rc}}^2)}\frac{\sigma_{\text{rc}}}{\sigma_{\text{sc}}}\,. \tag{2.26}$$

In this section we derive the p.d.f. corresponding to (2.22). Equations (2.27)-(2.28) show the p.d.f.s of L_{rc} and L_{sc}.

$$f_{L_{rc}}(x) = \frac{1}{\sqrt{2\pi\sigma_{rc}^2}} e^{-\frac{(x-\mu_{rc})^2}{2\sigma_{rc}^2}} \tag{2.27}$$

$$f_{L_{sc}}(y) = \frac{c_1}{\sqrt{2\pi\sigma_{sc}^2}} e^{-\frac{(y-\mu_{sc})^2}{2\sigma_{sc}^2}}, -t < y < t, \tag{2.28}$$

where

$$c_1 = \frac{1}{Q\left(\frac{-t-\mu_{sc}}{\sigma_{sc}}\right) - Q\left(\frac{t-\mu_{sc}}{\sigma_{sc}}\right)}. \tag{2.29}$$

Let $L' = L_{rc} + L_{sc}$. The p.d.f of L' is the convolution of the p.d.fs of L_{rc} and L_{sc} and is given by

$$f_{L'}(v) = \int_{-\infty}^{\infty} f_{L_{rc}}(u) f_{L_{sc}}(v-u) du. \tag{2.30}$$

The integrand of (2.30) can be expressed as a normal term multiplied by the term c_6 as follows:

$$f_{L_{rc}}(u) f_{L_{sc}}(v-u) = c_2 \frac{1}{\sqrt{2\pi c_3^2}} e^{-\frac{(u-c_4)^2}{2c_3^2}}, \tag{2.31}$$

where

$$c_2 = c_5 e^{c_6} \sqrt{2\pi c_3^2} \tag{2.32}$$

$$c_5 = \frac{c_1}{2\pi\sigma_{sc}\sigma_{rc}}, \tag{2.33}$$

$$c_6 = \frac{-(v-\mu_{sc}-\mu_{rc})^2}{2(\sigma_{sc}^2+\sigma_{rc}^2)}, \tag{2.34}$$

$$c_3^2 = \frac{\sigma_{rc}^2\sigma_{sc}^2}{\sigma_{sc}^2+\sigma_{rc}^2}, \tag{2.35}$$

$$c_4 = \frac{\sigma_{sc}^2\mu_{rc} + (v-\mu_{sc})\sigma_{rc}^2}{\sigma_{sc}^2+\sigma_{rc}^2}. \tag{2.36}$$

The variable u is only defined in the range of $[v-t, v+t]$ consistent with (2.28). Integrating (2.31) over $u \in [v-t, v+t]$, the p.d.f of $f_{L'}(v)$ is derived as

$$f_{L'}(v) = c_2 \left(Q\left(\frac{v-t-c_4}{c_3}\right) - Q\left(\frac{v+t-c_4}{c_3}\right)\right). \tag{2.37}$$

By simplifying terms, (2.37) reduces to

$$f_{L'}(v) = c_2\left(Q\left(\frac{v+G_1}{G_3}\right) - Q\left(\frac{v+G_2}{G_3}\right)\right), \tag{2.38}$$

$$\text{where } G_1 = \frac{-\sigma_{sc}^2\mu_{rc} + \mu_{sc}\sigma_{rc}^2 - t(\sigma_{sc}^2+\sigma_{rc}^2)}{\sigma_{sc}^2}, \tag{2.39}$$

$$G_2 = \frac{-\sigma_{sc}^2\mu_{rc} + \mu_{sc}\sigma_{rc}^2 + t(\sigma_{sc}^2+\sigma_{rc}^2)}{\sigma_{sc}^2}, \tag{2.40}$$

$$G_3 = \sqrt{(\sigma_{sc}^2+\sigma_{rc}^2)}\frac{\sigma_{rc}}{\sigma_{sc}}. \tag{2.41}$$

Under the assumptions that $\sigma_{sc}^2 = 2\mu_{sc}$ and $\sigma_{rc}^2 = 2\mu_{rc}$,

$$G_2 = -G_1 = \frac{t(\sigma_{sc}^2+\sigma_{rc}^2)}{\sigma_{sc}^2}. \tag{2.42}$$

Finally, the probability of (2.22) is given by

$$\int_0^\infty c_6\left(Q\left(\frac{v-G_1}{G_3}\right) - Q\left(\frac{v+G_1}{G_3}\right)\right)dv. \tag{2.43}$$

Equation (2.22), which is derived, shows the probability that a particular bit u is decoded correctly if the original LLR value from the initial SC decoding is within the thresholds $[-t, t]$ and a repetition code of length n is used. c_1 and G_1 are variables that depend on the choice of the threshold t and the channel SNR. G_2 only depends on the SNR of the channel and the LLR distribution of the SC decoded bit u.

The probability that a particular information bit u is correctly and reliably decoded by the initial SC decoding or with the increased reliability from the repetition code when its LLR is within the threshold $[-t_i, t_i]$ is

$$\begin{aligned} P_{\text{fb}(t,n)}(\hat{u}=u) &= P(L>t) + P(L'>0 \,|\, -t<L<t)P(|L|<t) \\ &= Q\left(\tfrac{t-\mu_{\text{sc}}}{\sigma_{\text{sc}}}\right) + \frac{\int_0^\infty e^{c_2(v)}\left[Q\left(\frac{v-G_1}{G_2}\right) - Q\left(\frac{v+G_1}{G_2}\right)\right]dv}{2\pi\sigma_{\text{sc}}\sigma_{\text{rc}}}. \end{aligned} \tag{2.44}$$

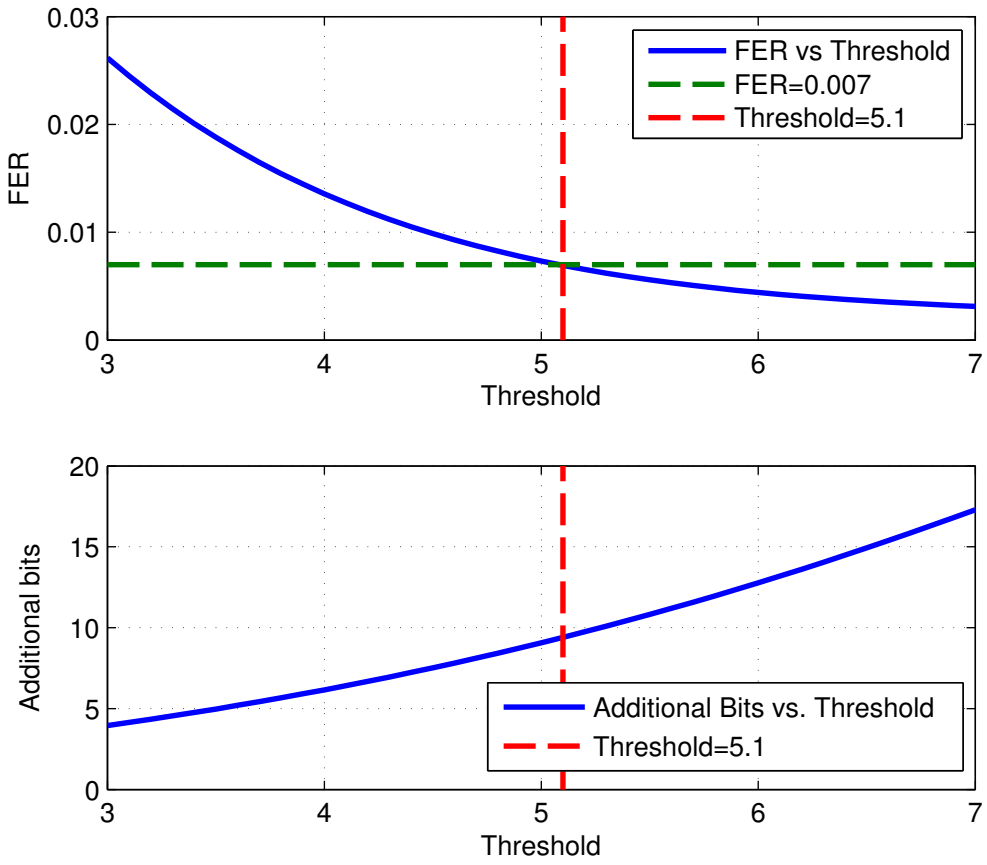


Figure 2.7: FER prediction based on RCA versus the threshold (t) for the optimization problem of (2.46) where $n = 5$ and $\epsilon = 0.007$. The total expected number of additional bits $\sum_{i=1}^{k} \Delta_{\text{fb}}(u_i, t, n) = 9.40$ in the transmitted repetition codes.

The expected number of additional bits transmitted in the forward direction to increase the reliability of the single information bit u in response to feedback is

$$\Delta_{\text{fb}}(u,t,n) = n\left(Q\left(\frac{-t-\mu_{\text{sc},u}}{\sigma_{\text{sc}}}\right) - Q\left(\frac{t-\mu_{\text{sc},u}}{\sigma_{\text{sc}}}\right)\right). \tag{2.45}$$

For a particular target FER ϵ, the optimization problem is to find the t_i and n_i values such that the total expected number of additional bits is minimized. Therefore, the optimization problem

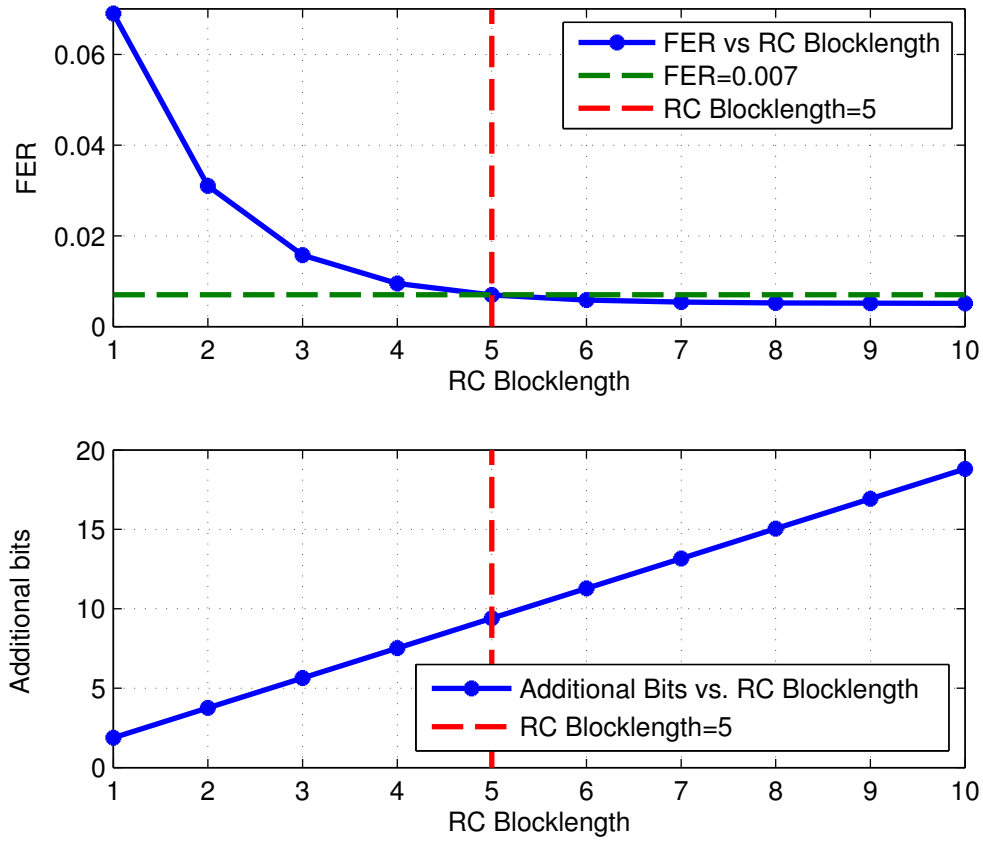


Figure 2.8: FER prediction based on RCA versus the RC blocklength (n) for the optimization problem of (2.46) where $t = 5.1$ and $\epsilon = 0.007$. The total expected number of additional bits $\sum_{i=1}^{k} \Delta_{\text{fb}}(u_i, t, n) = 9.40$ in the transmitted repetition codes.

reduces to

$$\underset{\{t_1,\ldots,t_k,n_1,\ldots,n_k\}}{\text{Minimize}} \quad \sum_{i=1}^{k} \Delta_{\text{fb}}(u_i, t_i, n_i) \tag{2.46}$$

$$\text{s.t.} \quad 1 - \prod_{i=1}^{k} P_{\text{fb}(t_i,n_i)}(\hat{u}_i = u_i) < \epsilon\,. \tag{2.47}$$

The size of the above optimization problem is very large where n_i can be any positive integer number and each t_i is any positive real number. To simplify the optimization space, in this section, we assume $t_i = t$ and $n_i = n$ for all i, even though we will continue to reflect in our calculations that $\mu_{\text{sc},i}$ and $\sigma_{\text{sc},i}$ are distinct for each i. We have found that constraining the optimization to a single t and a single n in fact does not substantially diminish performance.

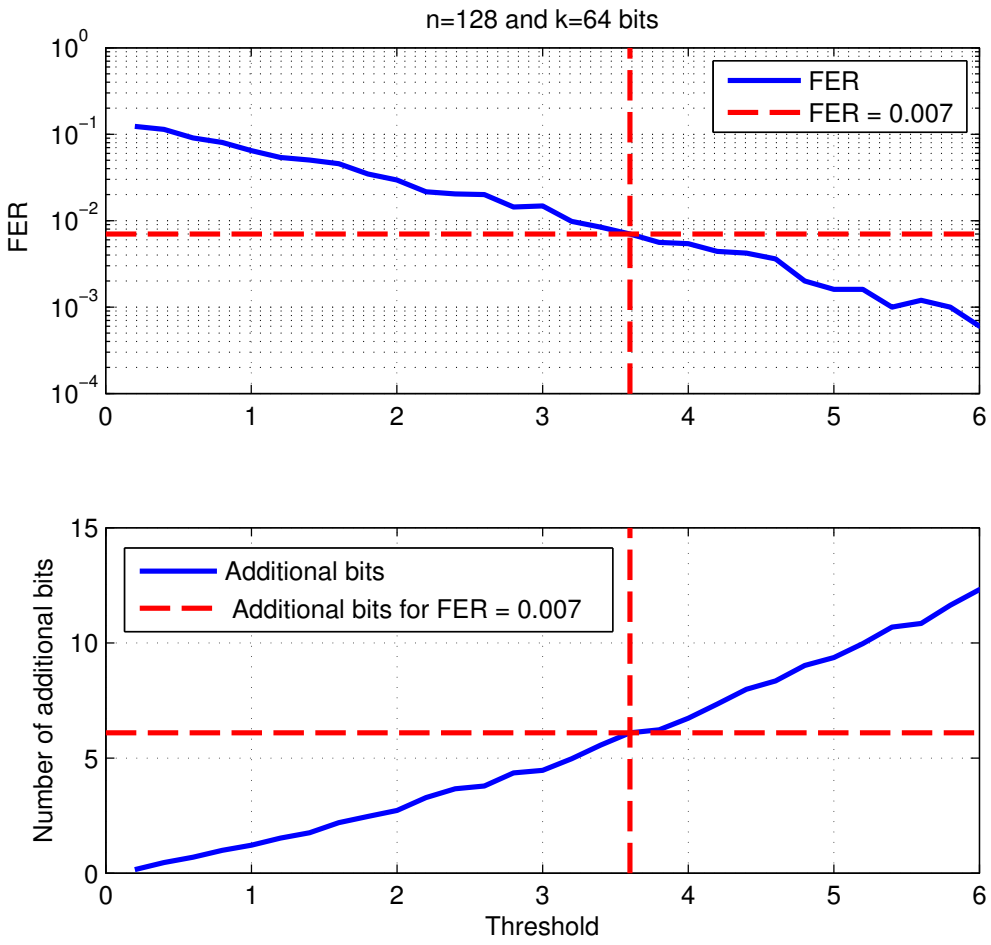


Figure 2.9: The improved results by re-transmitting the unreliable bits from the originally transmitted bits as the new additional bits.

In this section, we show the analysis for the optimal solution to the optimization problem of (2.46) under the assumption that $t_i = t$ and $n_i = n$ for all i. For a constant repetition-code length of $n = 5$, Fig. 2.7 shows the average number of the repetition-transmitted additional bits under the FER constraint of $\epsilon = 0.007$. For a fixed n, the probability of error as in (2.47) decreases, but the expected number of additional bits transmitted in the forward direction increases since the size of the unreliable region has increased. For $n = 5$, the minimum threshold t that achieves an FER of smaller than $\epsilon = 0.007$ is $t = 5.1$. This value of t corresponds to an unreliable LLR range of $[-5.1, 5.1]$.

For a constant LLR threshold of $t = 5.1$, Fig. 2.8 shows the average number of the repetition-

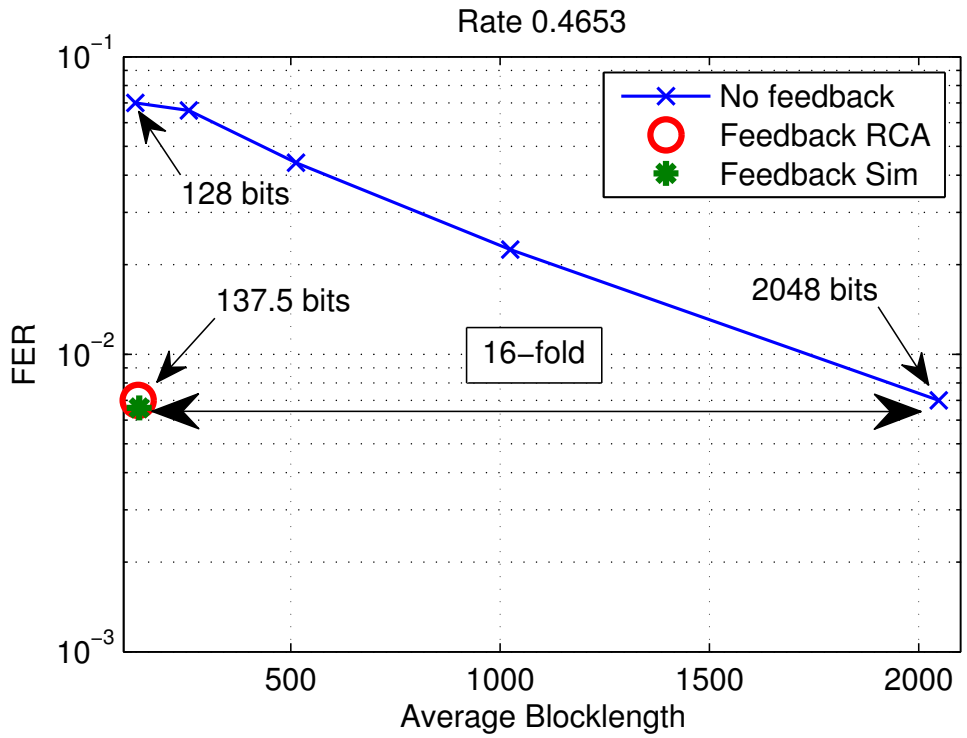


Figure 2.10: FER of the feedback system obtained by simulations and its comparison to the analysis of Sec. 2.4. The feedback system shows about 16-fold reduction in blocklength required to achieve a target FER of 7×10^{-3} compared to the case without feedback.

transmitted additional bits under the FER constraint $\epsilon = 0.007$. For a fixed t, the probably of error as in (2.47) decreases as the length of the repetition code n increases. After a certain point, the rate of decay of the probability of error decreases with an increase in n. This is due to the choice of threshold t resulting in a small enough unreliable region that with a high probability the SC decoded LLR values become less than the lower threshold $-t$. In this scenario, increasing n does not significantly lower the overall probability of error while it increases expected number of additional bits transmitted in the forward direction. For $t = 5.1$ the minimum length of the repetition code n that achieves an FER smaller than $\epsilon = 0.007$ is $n = 5$.

Fig. 2.10 shows simulation results on a 2 dB binary-input AWGN channel including a polar code with feedback using the values $t = 5.1$ and $n = 5$, which optimize $\sum_{i=1}^{k} \Delta_{\text{fb}}(u_i, t, n)$ for the target FER of $\epsilon = 7 \times 10^{-3}$ at 2 dB for the original rate 0.5 polar code shown in Fig. 2.5. This target was chosen because it is an order of magnitude below the FER of a rate-0.4653 blocklength-128 polar code at 2 dB. Based on RCA analysis, the effective rate is 0.4658 after accounting for the expected additional bits, which total $\sum_{i=1}^{k} \Delta_{\text{fb}}(u_i, t, n) = 9.40$ due to the repetition codes. The simulation results used an average of 9.55 additional bits and the effective rate of the simulation is 0.4653.

Fig 2.10 also shows the FER simulation results for polar codes without feedback for various blocklengths ranging from 128 to 2048 with rates set as closely as possible to 0.4653. The actual rates were 0.4688 for the blocklength 128, 0.4648 for blocklengths 256 and 512, and 0.4658 for blocklengths 1024 and 2048.

For rates near 0.465, the polar code with feedback achieved an FER of 7×10^{-3} with an average blocklength of $128 + 9.55 = 137.55$. In contrast, a polar code without feedback requires a blocklength of 2048 for a comparable FER. Thus the reduction in blocklength achieved by feedback is a about factor of 16.

2.5 Conclusions

In this chapter we have shown how to use an extension of RCA to closely and efficiently predict the FER performance of polar codes by analyzing the p.d.f of LLR values associated with information bits. Using this distribution and a feedback scheme incorporating a repetition code, the blocklength required to achieve a target FER is decreased by a factor of 16 as compared to a polar code without feedback. The use of feedback allows a smaller faction of the initial bits to be frozen because feedback provides the ability to effectively freeze a few more bits after the initial transmission through the use of a repetition codes to provide additional needed reliability.

CHAPTER 3

Protograph-Based Raptor-like Codes

In this chapter, we propose protograph-based Raptor-like (PBRL) codes as a class of rate-compatible (RC) LDPC codes for binary-input AWGN channels. As with the Raptor codes, exclusive-OR operations on precoded bits produce additional parity bits providing extensive rate compatibility. Unlike Raptor codes, each additional parity bit in the protograph is explicitly designed to optimize the density evolution threshold. During the lifting process, ACE and CPEG constraints are used to avoid undesirable graphical structures. Some density-evolution performance is sacrificed to obtain lower error floors, especially at short blocklengths.

Simulation results are shown for information block sizes of $k = 1032$ and 16384. For a target frame error rate of 10^{-5}, at each rate the $k = 1032$ and 16384 code families perform within 1 dB and 0.4 dB of both the Gallager bound and the normal approximation, respectively. The 16384 code family outperforms the best known standardized code family, the AR4JA codes. The PBRL codes also outperform DVB-S2 codes that have the advantages of longer blocklengths and outer BCH codes. Performance is similar to RC code families designed by Nguyen et al. that do not constrain codes to have the PBRL structure and involve simulation in the optimization process at each rate.

3.1 Introduction

This chapter provides a general technique for constructing families of rate-compatible (RC) low-density parity-check (LDPC) codes and provides numerical results showing excellent performance.

3.1.1 Rate-Compatible Channel Codes

RC punctured convolutional (RCPC) codes and RC punctured turbo (RCPT) codes are among the most popular RC channel codes used in incremental redundancy (IR) systems [50]. For both RCPC [51] and RCPT [52] codes, a collection of RC puncturing patterns are often carefully designed to ensure good error rate performance across the family of rates despite the RC constraint. Liu and Soljanin showed that RCPT code performance degrades significantly when the punctured code rate is above a threshold [53].

Low-Density Parity-Check (LDPC) codes were first introduced by Gallager in his dissertation in 1963 [54]. Tanner [55] introduced the representation of LDPC codes as bipartite graphs. MacKay [56] showed that LDPC codes provide capacity-approaching performance similar to turbo codes [57] when decoded by a message-passing algorithm with soft information.

Irregular LDPC codes have parity-check matrices that have a variety of column weights and row weights. By optimizing the variable-node and check-node degree distributions, Luby et al. [58] showed that properly constructed irregular LDPC codes can achieve rates even closer to capacity than regular codes, which have a single column weight and a single row weight. Richardson, Shokrollahi and Urbanke [59] created a systematic method called density evolution to design and analyze the optimal degree distribution of LDPC codes based on the assumption that the block-length can be infinitely long.

Inspired by the capacity-approaching performance of LDPC at individual rates, approaches including [60–74] construct RC LDPC code families.

Any RC code may be considered as a low-rate code that is punctured to produce higher rates, but design approaches can be distinguished as to whether the design begins with the lowest rate or the highest rate. In [60–65], the lowest rate "mother code" is designed first and then puncturing is designed to obtain the higher rates. However, as observed in [66] and elsewhere, finite-length LDPC RC code families obtained in this way suffer from a larger performance degradation than punctured turbo codes at high rates.

To avoid this problem at high rates, the method of *extension* designs the high-rate code first

and then the lower rate codes are designed on top of that foundation. This approach has been explored in articles including [66–74]. Our analysis uses the extension approach. Also, as in [61, 66, 68] our design approach focuses on maximizing the density-evolution/EXIT threshold as a design objective.

As in [67, 73], our RC code family is designed by extending a protograph. Thorpe [75] introduced protograph-based LDPC codes, or protograph codes. These codes were studied extensively by Divsalar et al. [76]. The design of protograph codes begins with the construction of a relatively small bipartite graph called the protograph. After using density evolution to properly design the protograph, a copy-and-permute operation, often referred to as "lifting", is applied to the protograph to obtain larger graphs of various sizes, resulting in longer-blocklength LDPC codes. As part of lifting, the variable-node connections of the edges of the same type are permuted among the protograph replicas. Even when the protograph has parallel edges, lifting can ensure that the final code does not have parallel edges. A detailed discussion on parallel edges in protographs can be found in [76], which also discusses how protograph codes facilitate efficient decoder implementation in hardware.

In contrast to [67] and [73], but similar to [71, 72], we restrict the code family to have the basic structure of Raptor codes [77]. Constraining the design in this way makes the construction and optimization manageable while still providing outstanding performance and extensive rate-compatibility. Introduced by Luby [78] and Shokrollahi [77] respectively, LT codes and Raptor codes share many similarities with LDPC codes and are shown to achieve the capacity of binary erasure channel (BEC) universally. Etesami et al. [79] explored the application of Raptor codes to binary memoryless symmetric channels and derive various results, including the fact that Raptor codes are not universal except for the BEC. Note that results on Raptor codes such as [77] and [79] rely heavily on the assumption of large information blocks.

Following the Raptor-like structure proposed in [71], Nitzold et al. applied spatial coupling [80] to improve the threshold, These spatially coupled codes can be viewed as Raptor-like LDPC convolutional codes [81]. Nitzold et al. focused on the analysis of the asymptotic decoding threshold where the rate loss due to the time-spreading number L is negligible.

Recent work by Nguyen and Nosratinia [74] considered a general structure for extending RC protograph codes but ends up proposing an example protograph that has the Raptor-like structure as first proposed in [71] and [72].

3.1.2 Organization and Main Contributions

We propose a class of RC LDPC codes called protograph-based raptor-like (PBRL) LDPC codes. The construction and optimization of PBRL codes are discussed and simulation results are presented. Comparing to existing codes in the literature (e.g. AR4JA codes in [82], DVB-S2 codes in [83] and protograph codes in [73]), PBRL codes show outstanding performance while providing extensive rate-compatibility.

Our analysis is based on the precursor conference papers [71, 72]. The PBRL approach was introduced in [71] and used to design $k = 192$ RC code families. In [71] a PBRL code family with multiple parallel edges to the punctured node produced thresholds within 0.34 dB of capacity at every rate, but the best simulated performance was achieved by a code family with higher thresholds that had only one pair of parallel edges connected to the punctured node. This is an example of the well-known inability of threshold alone to guide short-blocklength design.

In [72] the PBRL approach is applied to the design of long-blocklength code families, providing a $k = 16368$ LDPC code family as an example. Also, reciprocal channel approximation (RCA) replaces the standard density evolution of [71] to provide a fast and accurate approximation of the density evolution threshold to speed up the optimization process. In [72] the punctured node is connected to every check node in the IR part at least once to provide low thresholds. As in [71], parallel edges to the punctured node from the IR check nodes are kept to a minimum, in this case two check nodes had a pair of parallel edges to the punctured node. In both [71] and [72] the circulant progressive edge growth (CPEG) algorithm [84] was used for lifting.

This chapter unifies and extends the material in the conference papers [71] and [72]. The contributions beyond the precursor papers include the description of a modified RCA algorithm for use with single-check variable nodes (which are essential to PBRL codes), the design of new code

families for $k = 1032$ and $k = 16384$ that use lifting that incorporates both CPEG and Approximate Cycle Extrinsic Message Degree (ACE) constraints, comparison with finite-length performance bounds and approximations, and a careful discussion of how constraints on the connections can be used to sacrifice some threshold performance to avoid problematic error floors. These constraints become less stringent for longer blocklengths.

The rest of the chapter is organized as follows: Sec. 3.2 presents the PBRL code structure. Sec. 3.3 provides the design procedure to construct PBRL codes. Sec. 3.4 constructs example PBRL code families and presents analysis and simulation results. Finally, Sec. 3.5 concludes the chapter.

3.2 Protograph-Based Raptor-Like LDPC Code

This section introduces the structure, encoding, and decoding of PBRL codes.

3.2.1 The Structure of PBRL Codes

Fig. 4.1 shows the protograph structure of a PBRL code. This protograph consists of two parts: (1) a highest-rate code (HRC) protograph and (2) an incremental redundancy code (IRC) protograph. The IRC provides lower rates as more of its variable nodes are transmitted, starting from the top.

The bipartite graph of a PBRL protograph can be described by a protomatrix, which is the parity-check matrix of a protograph. Let $\mathbf{0}$ be the all-zeros matrix and $\mathbf{I}$ be the identity matrix with the appropriate dimensions. The protomatrix of the protograph shown in Fig. 4.1 is given as

$$H = \begin{bmatrix} H_{\text{HRC}} & \mathbf{0} \\ H_{\text{IRC}} & \mathbf{I} \end{bmatrix} \tag{3.1}$$

where the HRC parity-check matrix H_{HRC} and IRC parity-check matrix H_{IRC} are given as

$$H_{\text{HRC}} = \begin{bmatrix} 1 & 1 & 2 & 1 & 2 & 1 \\ 2 & 2 & 1 & 2 & 1 & 2 \end{bmatrix} \tag{3.2}$$

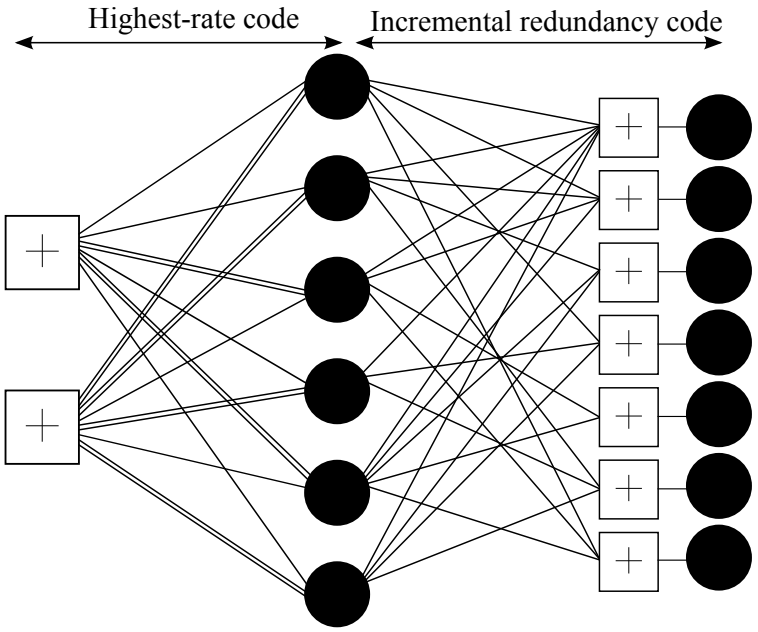


Figure 3.1: Protograph for a PBRL code with a highest-rate code (HRC) with rate $2/3$ followed by an incremental redundancy code (IRC) that uses only degree-one variable nodes. The IRC provides lower rates as more of its variable nodes are included, starting from the top.

and

$$H_{\text{IRC}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}. \tag{3.3}$$

Hence to fully express the protomatrix of a PBRL protograph it is enough to specify H_{HRC} and H_{IRC}. Note that as long as H_{HRC} is full rank, the proto-matrix at each rate (i.e. as each row of H_{IRC} is added) is also full rank.

The overall protograph determined by HRC and IRC protographs is lifted to produced the final code. We use circulant matrices in the lifting process. After lifting, the HRC code is structurally identical to the precode in a Raptor code. Similarly. the degree-one variable nodes of the second

part can be efficiently encoded as modulo-2 sums of the precode symbols in a manner similar to the LT code in a Raptor code.

The PBRL structure resembles a Raptor code, but with some important differences: 1) the variable nodes of the HRC are transmitted for a PBRL code but the precode variable nodes are not transmitted for a Raptor code 2) the connections that create a variable node in the LT part are random and potentially infinite for a Raptor code, but the connections that create a variable node in the IRC are deterministic, finite in number, and carefully designed for a PBRL code, and 3) the decoding of PBRL codes is different from that of Raptor codes as we discuss in the next subsection.

The overall PBRL protograph may be considered as the concatenation of the highest-rate LDPC code having parity-check matrix H_{HRC} and the low-density generator matrix (LDGM) code [85] having the systematic generator matrix

$$G = \begin{bmatrix} I & H_{\text{IRC}}^T \end{bmatrix}. \tag{3.4}$$

LDGM codes by themselves are known generally to have high error floors and poor asymptotic minimum distance [56, 85, 86]. However, as observed in [85, 86], concatenating an LDGM code with an outer code (the HRC in our case) mitigates the high error floor problem.

The IRC is created by the addition of variable nodes that are exclusively degree-one, which correspond to the identity matrix in (4.1). Such degree-one nodes are generally associated with poor error-floor performance. However, rate compatibility forces the sub matrix that is an identity matrix in (4.1) to be at least lower triangular. Any full-rank choice of the overall matrix in (4.1) that incorporates this lower triangular component will be range-equivalent to a matrix for which the lower triangular part is the identity matrix. Thus, the possible codes (i.e. the possible sets of valid codewords) is not affected by restricting the variable nodes in the IRC have degree one.

The remaining concern is that these degree-one variable nodes might introduce bad performance (i.e. a high error floor) under iterative decoding. As observed in [85, 86] the error floor problem can be resolved by concatenation with another code, as we have done by concatenating with the HRC. Moreover, increasing the degree of these variable nodes can introduce small cycles which must be addressed during lifting.

With the PBRL approach, each additional degree-one variable node boosts the reliability of the variable nodes with which it connects through its only neighboring check node (as described by the corresponding row of H_{IRC}). This is similar to the active feedback incremental redundancy approach used in [87,88]. A PBRL code family is thus a well-designed highest-rate code combined with additional variable nodes that provide efficient incremental redundancy to that highest-rate code such that each additional variable node in the protograph lowers as much as possible the SNR at which decoding succeeds.

3.2.2 Decoding and Encoding of PBRL Codes

A traditional Raptor code encodes the information bits to produce the precoded symbols and then encodes the precoded symbols with an LT code. In the case of an LDPC precode used with an LT code, the decoding often proceeds as follows: the decoder first performs BP decoding on the LT code and then performs BP decoding on the precode using the results of the completed LT decoding.

In [89], the authors note that because of the two-stage decoding, the complexity of Raptor codes is higher than that of RC LDPC codes. The PBRL code family always transmits the output symbols of the precode and has deterministic connections in the LT code. These two properties facilitate joint decoding of the HRC code part and the IRC code part, an idea that first appeared in [90] for Raptor codes.

For traditional Raptor codes that do not transmit the precode symbols and use randomized encoding, the initial transmission of the LT symbols may not contain enough information for BP decoding to succeed even in a noiseless setting. Always transmitting the precode symbols allows PBRL codes to have the potential for successful decoding at the initial transmission.

For high-rate PBRL codes, the decoder can deactivate those check nodes in the IRC part for which the neighboring degree-one variable node is not transmitted, offering significant complexity reduction.

Encoding of PBRL codes is as efficient as of Raptor codes: after encoding the precode, the

encoding of the IRC part only involves exclusive-or operations on the precode output symbols. For efficient encoding of the precode, see the discussion in [76] on efficient encoding of protograph codes.

The Raptor-like structure is very restrictive. One might expect the structural constraints to limit performance as compared to less-restrictive structures for extending LDPC codes. One of the main conclusions that we draw in this chapter is that despite these constraints, we obtain Raptor-like protographs with very low iterative decoding thresholds. By careful design of the protograph and the lifting process, the resulting finite-length codes can outperform existing RC LDPC codes that have been designed without the constraint of a Raptor-like structure.

3.3 Optimization of PBRL LDPC Codes

This section presents optimization procedures for finding the HRC and IRC components that comprise a good PBRL code family, considering both short and long blocklengths. Belief propagation (BP) decoding is assumed. The optimization criteria for both long and short-blocklength codes are primarily based on minimizing the iterative decoding threshold at each rate while enforcing constraints on the connections to avoid problematic error floors. These constraints are more stringent for short blocklength designs than for long blocklength designs.

To simplify the threshold computations, we use a modified version of the reciprocal channel approximation (RCA) algorithm. After presenting the modified RCA in subsection 3.3.1, subsection 3.3.2 describes the design of the HRC and subsection 3.3.3 describes the design of the IRC.

3.3.1 Density Evolution with Reciprocal Channel Approximation

The asymptotic *iterative decoding threshold* [91] characterizes the performance of the ensemble of LDPC codes that share a specified protograph. This threshold indicates the minimum SNR required to transmit reliably, meaning that the expected bit error rate goes to zero with that ensemble of codes as the blocklength grows to infinity. Note that this may not coincide with the block error rate

going to zero.

Computing the exact iterative decoding threshold for BI-AWGN requires significant computation. The RCA [47] [76] provides a fast and accurate approximation to the density evolution algorithm originally proposed by Richardson et al. [91] [59]. Experimental results [76], [47] show that the deviation of RCA from the exact density evolution threshold is less than 0.01 dB.

The RCA for BI-AWGN channel uses a single real-valued parameter s, the SNR, to approximate the density. Define the reciprocal SNR as $r \in \mathbb{R}$ such that $C(s) + C(r) = 1$ where $C(s)$ is the capacity of the BI-AWGN channel with SNR s. For computational simplicity and numerical precision we prefer to express $C(s)$ as

$$C(s) = 1 - \int_{-\infty}^{\infty} \log_2\left(1 + e^{-(2\sqrt{2s}u+2s)}\right) \frac{e^{-u^2}}{\sqrt{\pi}} du, \tag{3.5}$$

which is obtained from [49, (15)] through a change of variables. The self-inverting reciprocal energy function

$$R(s) = C^{-1}\left(1 - C(s)\right) \tag{3.6}$$

in [47] transforms between s and r: $r = R(s)$ and $s = R(r)$.

Let s_{chl} be the channel SNR $= 2\frac{E_c}{N_o} = 2R_c\frac{E_b}{N_o}$ where $\frac{E_c}{N_o}$ is the code symbol signal-to-noise ratio, $\frac{E_b}{N_o}$ is information bit signal-to-noise ratio, and R_c is the code rate. Let s_e be the message passed along an edge e from a variable node to a check node and r_e be the message passed along an edge e from a check node to a variable node. Let E_c be the set of edges that connect to a check node c and E_v be the set of edges that connect to a variable node v.

RCA first initializes the message s_e to 0 if the edge e is connected to a punctured variable node and to s_{chl} otherwise. For all edges e in the graph, RCA then computes a sequence of messages $(s_e^{(n)}, r_e^{(n)}),\ n = 0, \ldots, N$ where N is the maximum number of iterations.

The original density evolution [91] determines the threshold based on the pdfs of all outgoing messages from variable nodes. Aiming to approximate this density evolution, RCA [47] determines the decoding threshold s_{th} as the minimum s_{chl} such that $s_e^{(N)} > T$ for all edges e in the graph, where T is a stopping threshold.

Note that for the edge connecting to a degree-one variable node, $s_e = s_{\text{chl}}$ regardless of the number of iterations. For this reason, the original RCA does not work if the graph contains degree-one variable nodes. We use a slightly modified version of the RCA that focuses on the overall reliability of each variable node S_v, rather than the reliability of every edge s_e. This modification allows computation of a meaningful decoding threshold for protographs with degree-one variable nodes. Letting N be the maximum number of iterations and $T > 0$ be the stopping threshold, the modified RCA is summarized as follows:

Algorithm 1 (Modified Reciprocal Channel Approximation) *Let $f_{RCA}(s_{chl})$ be a binary-valued function that returns 1 if s_{chl} is higher than s_{th} and 0 otherwise. To determine its output, the modified RCA computes the sequence $\left(s_e^{(n)}, r_e^{(n)}\right), n = 0, \ldots, N$, for all edges e in the graph. The computation of the sequence is given as follows:*

0) For edges e connected to punctured variable nodes, set $s_e^{(0)} = 0$. For all other edges set $s_e^{(0)} = s_{chl}$.

1) For $n = 1, \ldots, N$, generate $(s_e^{(n)}, r_e^{(n)})$ as follows:

$$r_e^{(n)} = \sum_{i \in E_c \setminus e} R\left(s_i^{(n-1)}\right), \tag{3.7}$$

$$s_e^{(n)} = s_e^{(0)} + \sum_{i \in E_v \setminus e} R\left(r_i^{(n)}\right). \tag{3.8}$$

2) At each iteration compute $S_v^{(n)}$ as

$$S_v^{(n)} = S_v^{(0)} + \sum_{e \in E_v} R\left(r_e^{(n)}\right), \tag{3.9}$$

for all variable nodes v in the graph, where $S_v^{(0)}$ is

$$S_v^{(0)} = \begin{cases} 0 & \text{if } v \text{ is punctured} \\ s_{chl} & \text{otherwise} \end{cases}. \tag{3.10}$$

3) Let $S^*(n) = \min\left\{S_v^{(n)} : \forall v \text{ in the graph}\right\}$. At each iteration, compute $S^*(n)$. If $S^*(n) > T$, set $f_{RCA}(s_{chl}) = 1$ and stop; otherwise, set $f_{RCA}(s_{chl}) = 0$ and continue to the next iteration, stopping if $n = N$.

By monotonicity of the threshold [92] we can perform bisection search at the desired level of precision using the function f_{RCA}. To increase computation speed, a lookup table and linear interpolation are used to compute $R(s)$.

3.3.2 Optimizing the Highest-Rate Code of a PBRL Family

Primarily, the HRC photograph is simply the protograph of a good code at the desired rate. Our design follows the work of Divsalar et al. [76, Sec.III], with some additional optimization through RCA analysis and LDPC code simulation.

One main conclusion of [76] as also observed in [93–96] is that protograph ensembles with a minimum variable-node degree of 3 or higher are guaranteed to have linear minimum distance growth with the blocklength. This observation for irregular LDPC codes was also recognized in [97] and [98].

Divsalar et al. [76] also noted that judiciously adding a few variable nodes with degree 2 or even degree 1 improves the protograph's threshold. A small number of variable nodes with degree less than 3 can be present in the protograph while still retaining the linear minimum distance growth property as long as there is no loop comprised entirely of degree-2 nodes. The HRC protographs designed in Secs. 3.4.1 and 3.4.2 ensure the linear minimum distance growth property.

As explained in [76], puncturing a node in the HRC can improve the threshold performance. A variant of PBRL codes for which the protograph of the HRC has at least one punctured (untransmitted) node is referred to as the Punctured-Node PBRL (PN-PBRL) codes in [71]. In this work, all of our examples belong to the class of PN-PBRL codes due to their superior performance. We will refer to these codes simply as "PBRL codes", but they are also PN-PBRL codes since they have a punctured node.

We identify the HRC for a family of PBRL codes by a triplet of parameters (n_v, n_c, n_p): number of variable nodes n_v, number of check nodes n_c and number of punctured nodes n_p. For the code families designed in Secs. 3.4.1 and 3.4.2, $n_p = 1$. Assuming that H_{HRC} is full rank, the set of possible rates of this family of RC codes is given by $(n_v - n_c)/(n_v - n_p + i)$ for all integers $i > 0$.

3.3.3 Optimizing the IRC Protograph of a PBRL Family

Optimization of the IRC part includes consideration of linear minimum distance growth, the threshold values obtained at each rate, and the ability to realize the performance those threshold values predict by avoiding high error floors.

At the heart of IRC optimization is a greedy algorithm to optimize the threshold of the protograph of each rate given the protograph of the previous (higher) rate. This optimization algorithm is summarized as follows:

Algorithm 2 *Given an HRC protograph, the protograph for the IRC of a PBRL code family is obtained by the following steps:*

1. *Add a new check node and a new degree-one variable node connected to the new check node.*
2. *Use the modified RCA to find the connections between the new check node and the variable nodes of the HRC that give the lowest threshold under specified constraints on the connections. These constraints can vary with the intended blocklength but always include the constraint that each new row of H_{IRC} has at least two edges.*
3. *Stop when the lowest rate desired has been obtained. Otherwise, go to step 1.*

PBRL code families retain linear minimum distance growth by requiring that each new row in H_{IRC} has at least two edges. As long as the HRC has linear minimum distance growth, ensuring non-trivial connections for each new check node in the IRC in this way preserves linear minimum distance growth for all rates. This follows from the analysis of linear minimum distance growth in the context of LDPC codes concatenated with LDGM codes in [85, 86, 99, 100].

There are two key features that dramatically improve protograph thresholds at low rates. The first is that parallel edges improve the threshold and the second is that the punctured node of the precode should connect to all (or almost all) of check nodes in the IRC part with at least a single edge. Fig. 3.2 illustrates the importance of these features by comparing two PBRL code families that both use $(8, 2, 1)$ HRC protographs and eleven degree-one variable nodes in the IRC protograph. The protograph family from [71] connects the punctured node to seven of the eleven check

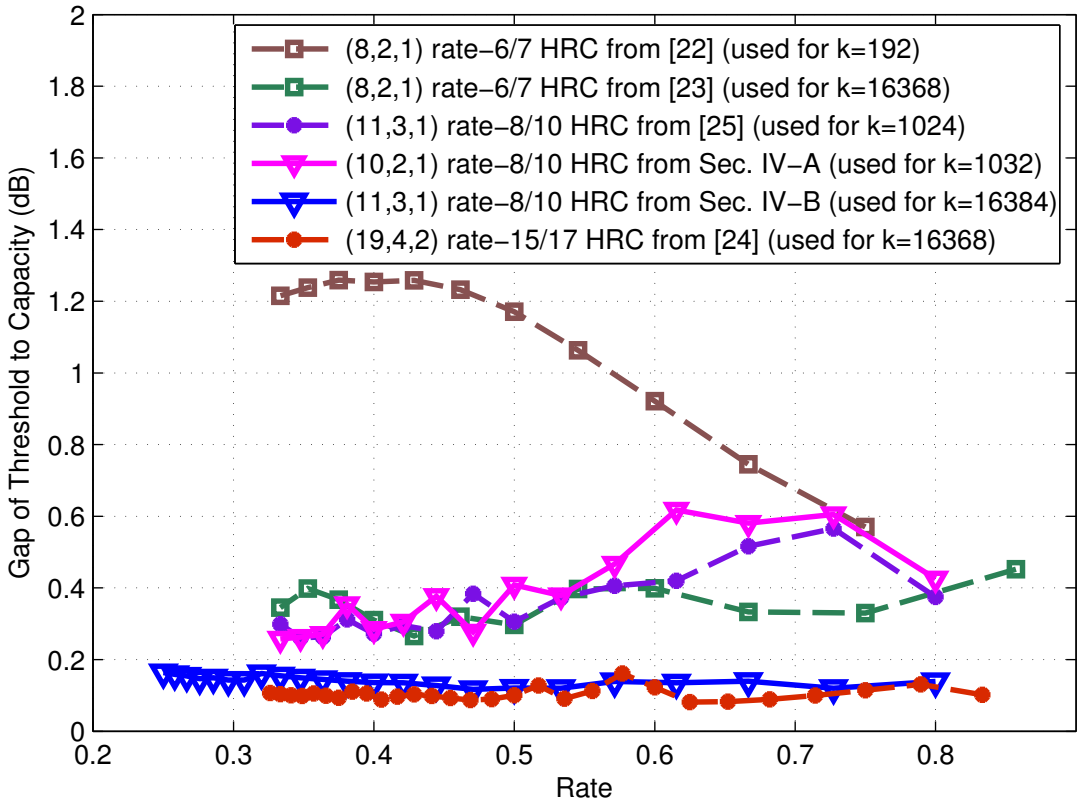


Figure 3.2: Gap of protograph threshold to binary AWGN capacity for each rate in the PBRL code family for four different PBRL protographs. These four protographs correspond to the following codes: the $k = 192$ PBRL family featured in [71], the $k = 16368$ PBRL family featured in [72], the $k = 16384$ PBRL family whose protograph is designed in Sec. 3.4.2 and the protograph for $k = 1032$ PBRL families designed in 3.4.1.

nodes connected to degree-one IRC variable nodes. Only the first such check node is connected to the punctured node with a parallel edge. In contrast, the protograph family from [72] connects the punctured node to all eleven check nodes connected to degree-one IRC variable nodes, Also, for this family the first two such check nodes are connected to the punctured node with parallel edges. Fig. 3.2 shows the dramatic threshold improvement of the protograph family of [72] over the one of [71].

Increased connectivity to the punctured node through single and parallel edges and the use of parallel edges elsewhere in the IRC lowers the thresholds significantly for the lower rates. The

threshold improvement, however, diminishes as the number of parallel edges increases. Moreover, even at long blocklengths lifting cannot overcome the resulting damage to the graphical structure of the code if the parallel edges are added too aggressively. Thus, an important aspect of IRC design is controlling the number of parallel edges to give the best possible thresholds for which lifting can successfully translate the excellent thresholds into excellent frame error rate (FER) and bit error rate (BER) performance. We will explore this trade-off in Secs. 3.4.1 and 3.4.2 to see how differently it plays out in the context of short-blocklength and long-blocklength codes.

3.4 Design Examples

This section presents design examples. Subsection 3.4.1 presents a short-block-length design example. Subsection 3.4.2 presents a long-block-length design example. Subsection 3.4.3 provides the gaps of SNR to the Shannon limit of SNR for reliable communication for FER 10^{-5} for PBRL families designed in this chapter and previously in [71, 72]. The description of the codes discussed here can be found on the UCLA CSL website.

3.4.1 A Short-Blocklength PBRL Design Example

This subsection provides an example PBRL protograph family designed for information blocklength $k = 1032$ and compares it to a comparable code from [74]. The HRC uses the $(10, 2, 1)$ protograph shown below:

$$H_{\text{HRC}}^{(1032)} = \begin{bmatrix} 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 2 & 1 \\ 1 & 3 & 1 & 3 & 1 & 3 & 1 & 2 & 1 & 2 \end{bmatrix}. \tag{3.11}$$

The IRC adds 15 degree-one check nodes at the lowest rate. The first variable node in the HRC is always punctured, and the first degree-one variable node in the IRC is always transmitted. Thus the range of rates of the form $8/(9+i)$ from $8/10$ to $8/24$. The H_{IRC} obtained from Algorithm 3

(with constraints we discuss below) is

$$H_{\text{IRC}}^{(1032)} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{3.12}$$

For this code design, we restrict our attention to potential HRC protographs with variable nodes having either degree 3 or degree 4. This restriction ensures linear minimum distance growth according to [76].

For this short-blocklength HRC, we address the trade-off between threshold and error floor by varying the number of degree-4 nodes in the HRC photograph. Using all degree-3 variable nodes provides the best threshold. Increasing the number of degree-4 variable nodes in the HRC lowers the error floor but increases the threshold as shown in Fig. 3.3. With fewer than 6 degree-4 variable nodes, the ACE-and-CPEG lifted codes for $k = 1032$ result in high error floors. As shown in (3.11), our H_{HRC} has seven degree-4 variable nodes, the smallest number of degree-4 variable nodes that yielded good error-floor performance.

For the IRC design, we require every check node in the IRC to connect to the punctured variable node of the HRC. These connections alternate between a single edge and a parallel edge (alternating 1s and 2s in the first column of (3.12)). In [72] two of the eleven check nodes in the IRC (the

two associated with highest two rates) connected to the punctured node with parallel edges. As discussed in Sec. 3.3.3, both [71,72] show that further threshold improvement can be obtained with more parallel edges. However, use of parallel edges is limited in [71, 72] because the CPEG-lifted codes displayed high error floors when more parallel edges were used.

The PBRL code families in [71] and [72] used CPEG alone for lifting. Combining CPEG with the ACE algorithm produces a lifting that better avoids damage to the graphical structure so that the alternating parallel edges to the punctured node could be used. This makes sense because extrinsic connections have an important influence on the error floor and the ACE algorithm [101] ensures that all small cycles have a minimum number (η) of these connections.

For this short-blocklength design, combined ACE-and-CPEG lifting allowed half the connections to the punctured node to be parallel edges, but we found that using parallel edges elsewhere in the IRC led to error floors for these short blocklengths.

We use two-step lifting as in [76] and employ CPEG and ACE algorithm jointly to lift the lowest rate code. The pre-lifting step has a lifting number of 3 to remove the parallel edges. The lifting number in the second stage is 43, giving information blocklength of $k = 1032$.

CPEG enforces a girth of 6. We used $\eta = 12$ and $d_{\text{ACE}} = 5$ for the ACE algorithm applied to the lowest rate. A pair (d_{ACE}, η) implies that every cycle consisting of up to d_{ACE} variable nodes, i.e. every cycle of length up to $2d_{\text{ACE}}$, is connected to at least η extrinsic check nodes. In other words, all the cycles with length 10 or less have at least 12 extrinsic connections for the lowest-rate (rate-1/3) code.

Fig. 3.4 shows that CPEG + ACE lifting for the protographs given in (3.11) and (3.12), provides larger η values than CPEG alone especially at the low end of our rate range. In Fig. 4.6, comparing the PBRL CPEG-only curves with the ACE+CPEG curves in the lower-rate range shows how the higher η values achieved by ACE+CPEG translate to an improved error floor.

Nguyen and Nosratinia constructed a family of RC protograph codes in [74] with short blocklengths ($k = 1024$). The design process in [74] featured optimization based on EXIT analysis similar to this work, but did not constrain the code to have the PBRL structure. The fact that this

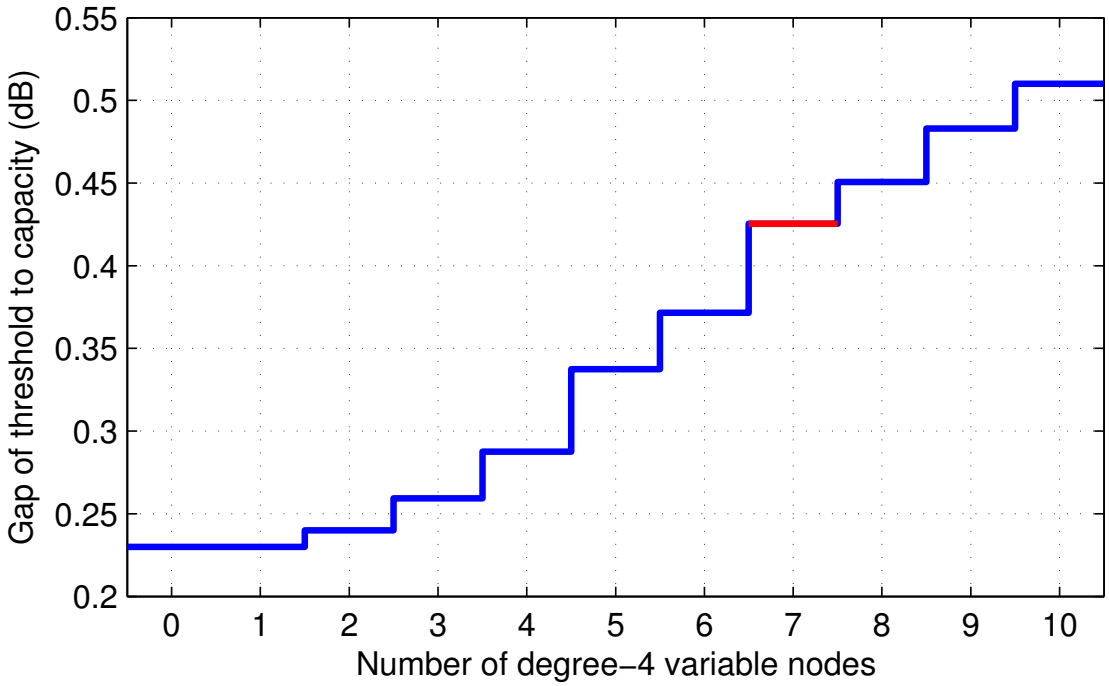


Figure 3.3: Gap of threshold from binary AWGN capacity vs. number of degree-4 variable nodes for potential highest-rate-code protographs based on a (10,2,1) protograph with all variable nodes having degrees 3 or 4. Gap is shown for the rate-0.8 protograph where the additional row has two parallel edges connected to the punctured node and the remaining connections are optimized to minimize the threshold under the constraint that each connection is either a single edge or not present. The result with seven degree-4 variable nodes corresponds to the protomatrix in (3.11).

code family has the PBRL structure even though it was not *constrained* to is a further validation of the PBRL approach.

The example in [74] uses 6 pairs of parallel edges in the IRC, all of which are connected to the punctured node in the HRC. The results agree with the observations in the current analysis and also in [71] and [72] that having parallel edges connected to the punctured node improves the threshold.

The protograph family described by (3.11)-(3.12) has the same size as the family proposed in [74]. Fig. 4.6 compares the FER performance of the RC codes proposed in [74] and the PBRL codes described by (3.11)-(3.12). The code family described by (3.11)-(3.12) has similar performance overall to that of [74] as shown in Figs. 4.6 and 3.9. The similarity in performance is not

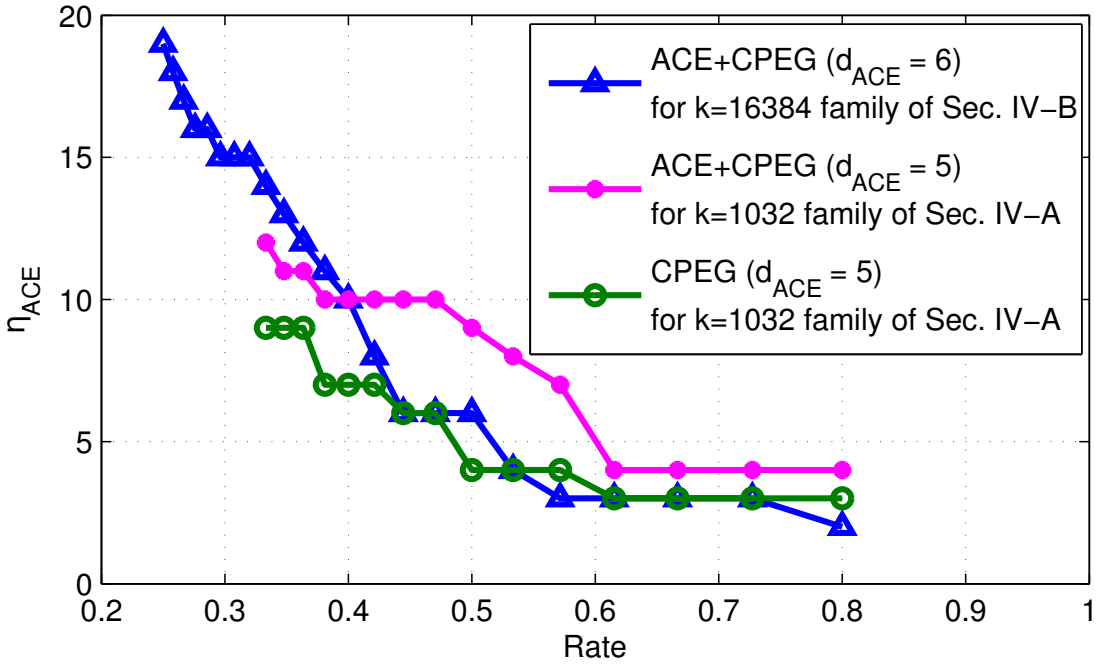


Figure 3.4: η_{ACE} for the ACE+CPEG-lifted families of codes for $k = 16384$ and $k = 1032$ and CPEG-only lifted family for $k = 1032$.

surprising since the code family in [74] was found by a similar optimization approach optimizing over a larger family of codes that include the PBRL family as a sub-class.

However, the search in [74] required experimental testing (simulation) of numerous codes with near-optimal thresholds at each rate. In contrast, after the design of the HRC protograph, our search simply optimized the threshold for each successive rate in a greedy fashion under certain constraints on edges that were imposed to control the error floor. The surprising result is that greedy threshold optimization can produce a family that is competitive if the proper constraints on edge connections are imposed.

Now we explore the constraints we imposed on the edge connections. Recall the intuition of the IRC variable nodes providing reliability back to the original HRC. As the rate decreases we desire the check nodes of the IRC eventually to connect to all of the variable nodes in the HRC. However, density evolution threshold optimization tends to favor asymmetric degree distributions so that the rich get richer in that IRC check nodes added later tend to connect to HRC variable

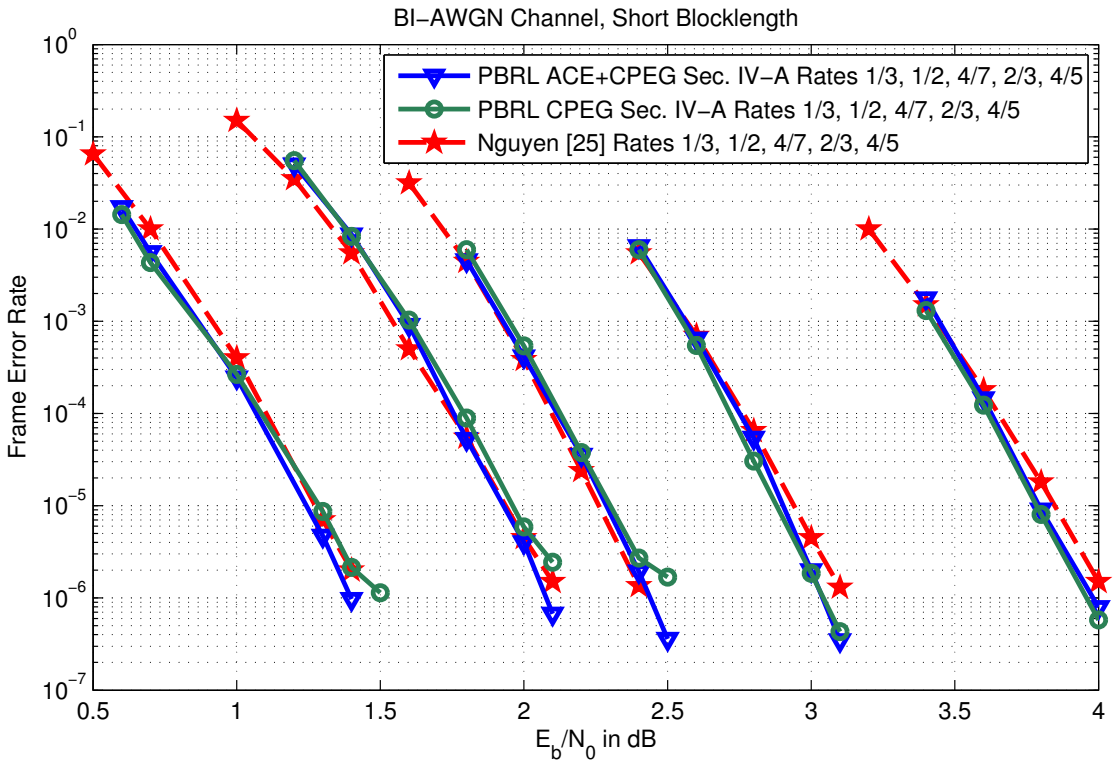


Figure 3.5: Comparison of frame error rates between PBRL codes and the codes in [74] with $k = 1032$. The protographs for the PBRL code family are from (3.11) and (3.12).

nodes that already connect to IRC check nodes.

Thus for short block-length codes we ensure good connectivity by requiring the IRC check nodes eventually to connect with all the HRC variable nodes as rate decreases. The specific constraints for the IRC in (3.11) are as follows: at rate 8/11 there can be at most two degree-3 variable nodes, at rate 8/13 there can be at most one degree-3 variable node, at rate 8/16 the minimum degree of the variable nodes is 4, at rate 8/18 there is at most one degree-4 variable node and finally at rate 8/21, the degree-4 variable nodes in HRC have at least degree five in the rate-8/21 parity-check matrix.

Fig. 3.2 allows comparison of the gaps from capacity for the thresholds of the protographs defined by (3.11)-(3.12) to the gaps of the $k = 1024$ protographs of [74]. The gaps are similar except at rate 8/13 where the protograph from [74] has a threshold about 0.2 dB lower than the

protograph from (3.11)-(3.12) whose threshold was hampered by the constraints described above. The average gap of threshold from capacity for the protographs defined by (3.11)-(3.12) over the rate range shown in Fig. 3.2 is 0.3914 dB.

Our simulations use floating-point iterative decoders with a flooding schedule for message-passing unless otherwise stated. Decoding terminates early if all parity checks are satisfied before reaching the maximum number (200) of iterations.

3.4.2 A Long-Blocklength PBRL Design Example

This subsection designs a PBRL code family with blocklength $k = 16384$ and compares it to the $k = 16368$ PBRL family designed in [72] and the $k = 16384$ family designed in [73], which does not use the PBRL structure. To improve performance, we use a PBRL protograph featuring an $(11, 3, 1)$ HRC protograph instead of the smaller $(8, 2, 1)$ HRC protograph used for the PBRL code family of [72]. The first variable node in the HRC is always punctured and H_{IRC} has 22 rows, each corresponding to an additional degree-one check node as described in Sec. 3.2.1. Thus the protograph family provides a range of rates of the form $8/(10 + i)$ from $8/10$ to $8/32$. The HRC protograph for the $k = 16384$ PBRL code is

$$H_{\text{HRC}}^{(16384)} = \begin{bmatrix} 3 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \end{bmatrix}. \tag{3.13}$$

Applying Algorithm 3 with constraints on the connections of the IRC check nodes to the variable nodes of the HRC produces the H_{IRC} given in (3.14) at the end of the chapter. As in the short-blocklength example, we require every check node in the IRC to connect to the punctured variable node of the HRC. These connections alternate between a single edge and a parallel edge (alternating 1s and 2s in the first column of (3.14)).

Unlike [72] and in contrast to the short-blocklength design in (3.11)-(3.12), the constraints applied to Algorithm 3 for this design allow parallel edges for connections to the non-punctured nodes of the HRC. Specifically, when optimizing the IRC using Algorithm 3, each connection to

a non-punctured node may have zero, one, or two edges leading to the 0, 1, and 2 values in the columns of (3.14). In [72] the only choices were zero and one, but the addition of ACE to the lifting process allows more parallel edges.

Nguyen et al. [73] constructed a family of RC protographs with a (19,4,2) protograph for the highest rate 15/17 and a (47,32,1) protograph for the lowest rate of 15/46. Fig. 3.2 shows that the protograph family defined by (3.13)-(3.14), with an (11,3,1) protograph for the highest rate of 8/10 and a (33, 25,1) protograph for the lowest rate of 8/32, gives thresholds comparable to those obtained in [73] despite the smaller protograph sizes. Fig. 3.2 also shows that the resulting thresholds have at least 0.1 dB of improvement at every rate compared to the thresholds of the (smaller) protographs used for the $k = 16368$ PBRL family in [72]. The gaps to capacity in Fig. 3.2 for the codes using (3.13)-(3.14) are all less than or equal to 0.164 dB. Furthermore, the average gap of threshold to capacity is 0.1408 dB.

We use two-step lifting as in [76] and employ CPEG and ACE algorithm jointly to lift the lowest rate code. The pre-lifting step has a lifting number of 4 to remove the parallel edges. The lifting number in the second stage is 512, giving information blocklength of $k = 16384$. CPEG enforces a girth of 8. Fig. 3.4 shows the η values of the $k = 16384$ PBRL code family obtained by lifting the protographs of (3.13)-(3.14) with $d_{\text{ACE}} = 6$. Achievable η (minimum number of extrinsic edges for cycles of length 12 or smaller) increases as rate decreases.

Fig. 3.6 compares the FER performance of the PBRL code family in [72] at rates $1/3$, $1/2$, and $2/3$ to the DVB-S2 standard (with and without an outer BCH code) [83] and AR4JA codes from the CCSDS standard [82]. In some cases for DVB-S2 codes, only results for a maximum of 50 iterations were available. The blocklengths of the DVB-S2 codes are fixed to 64800 bits, whereas the PBRL and AR4JA codes have a fixed information length of 16368 bits and blocklengths of 32736 bits and 24552 bits for rate $1/2$ and rate $2/3$, respectively. When concatenated with the BCH code, the overall rates of DVB-S2 codes are 0.497 bits and 0.664 bits.

Fig. 3.6 shows that in the waterfall region, the PBRL codes from [72] outperform both the AR4JA codes and the DVB-S2 codes. Note that the PBRL codes outperform DVB-S2 even though the DVB-S2 codes have longer blocklength and benefit from concatenation with a BCH code.

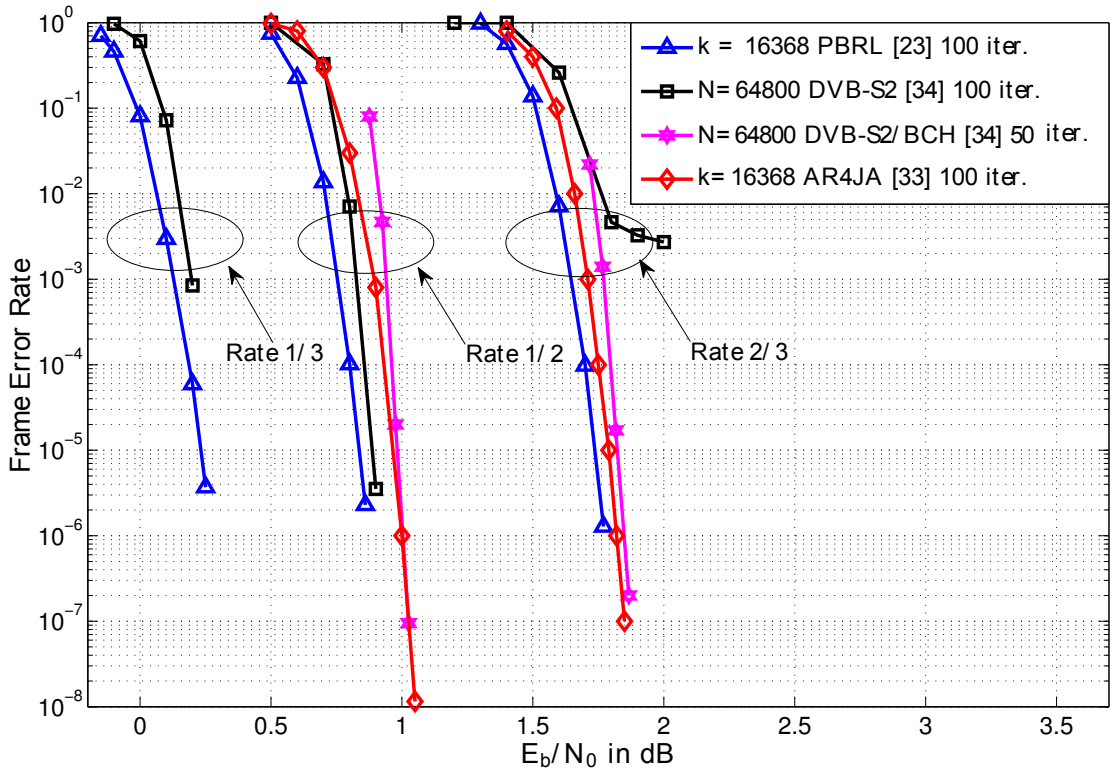


Figure 3.6: This figure is from [72]. FER comparison for PBRL codes from [72], codes in the DVB-S2 standard [83], and AR4JA LDPC codes in the CCSDS standard [82].

Fig. 3.7 presents the BER and FER performance for five of the rates supported by the PBRL code family based on (3.13)-(3.14).

Fig. 3.8 compares the FER between our $k = 16384$ PBRL code using (3.13)-(3.14), our $k = 16368$ PBRL code from [72], and the RC protograph codes proposed in [73] at rates $1/3$ and $1/2$. The simulation results in [73] used an 8-bit quantized decoder with 200 iterations whereas our simulations used floating point decoder with 200 iterations. The simulations of the codes based on (3.13)-(3.14) in Fig. 3.8 used layered belief propagation, which generally performs better than flooding. Using flooding in our simulation results in less than 0.05dB of degradation at 100 iterations in our PBRL code example.

Comparing at FER around 10^{-6}, this PBRL code is 0.2 dB better than Nguyen et al.'s code at rate $1/3$ and about 0.1 dB better at rate-$1/2$. Note that the code family using (3.13)-(3.14) performs

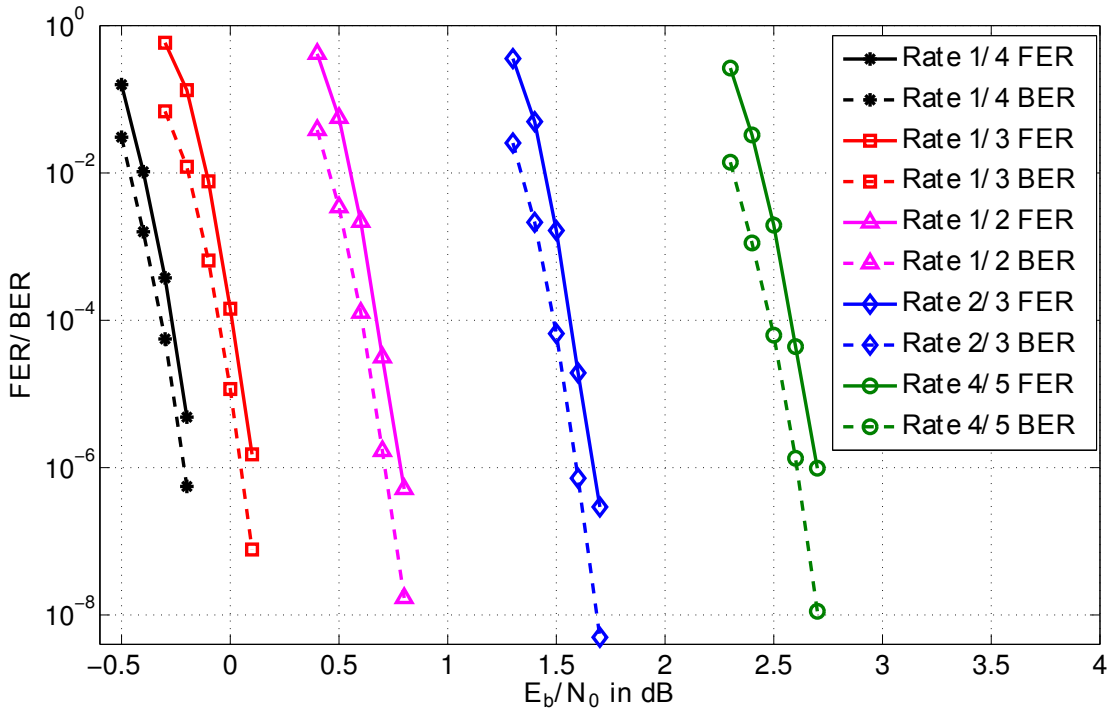


Figure 3.7: Frame error rates and bit error rates for the protographs defined by (3.13)-(3.14). The lifting number is 2048, resulting in $k = 16384$.

better even though the code family from [73] uses a larger protograph and is not constrained to the Raptor-like structure.

3.4.3 SNR Gap Analysis of PBRL Codes

Fig. 3.9 shows the SNR gaps in E_b/N_0 between the Shannon limit of the BI-AWGN channel and the required SNR to achieve FER of 10^{-5} in simulations of PBRL code families we have designed and the codes from [73] and [74]. Gaps from simulated PBRL codes are shown for the $k = 192$ PBRL codes from [71], $k = 1032$ PBRL codes from Section 3.4.1, $k = 16368$ PBRL codes from [72], and $k = 16384$ PBRL codes from Sec. 3.4.2. Gallager's random coding union bound and the refined normal approximation of [102], both computed for FER =10^{-5} and the specified rates and blocklengths are provided for reference.

Examining the differences between the average SNR gaps of the code families and the average

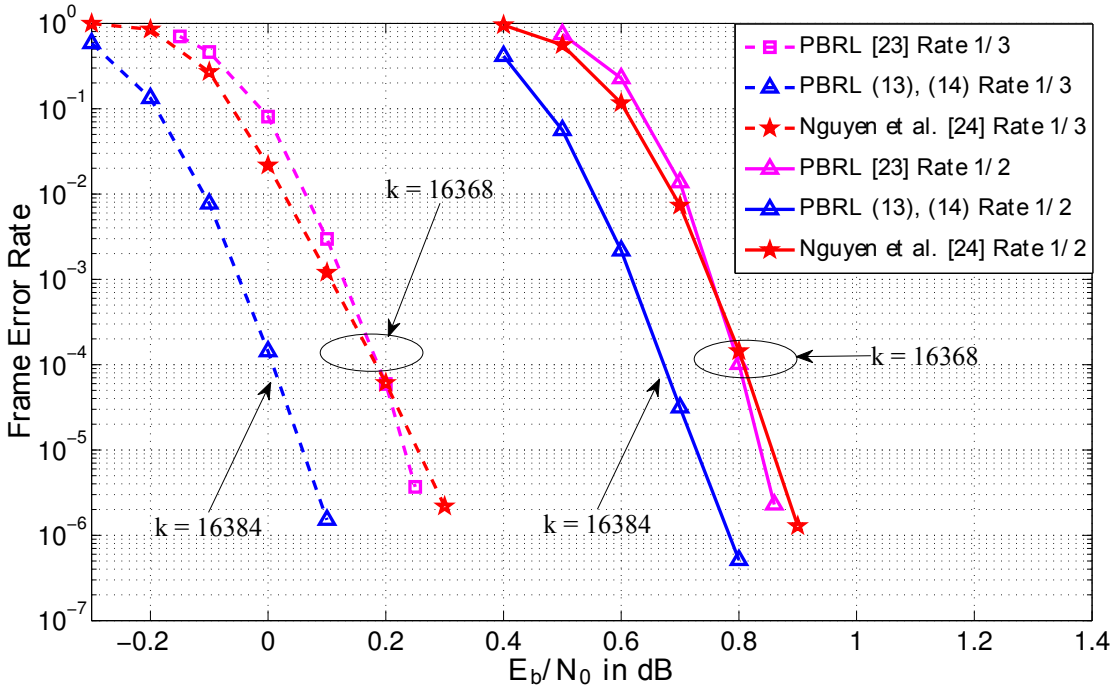


Figure 3.8: Frame error rates comparison between the PBRL codes with 19 variable nodes in the protograph (k = 16368), the PBRL codes with 33 variable nodes in the protograph (k = 16384), and the RC protograph codes proposed in [73] (k = 16368).

SNR gaps of the computed points for the Gallager bound at the comparable blocklengths, The difference is 1.053 dB for k = 192 family from [71], 0.658 dB for the k = 1032 family from Sec. 3.4.1, 0.660 dB for the k = 1024 family from [74], 0.395 dB for the k = 16386 code from [72], 0.354 dB for the k = 16384 code from [73], and 0.255 dB for the k = 16384 code from Sec. 3.4.1.

The k = 192 example from [71] was designed without the benefit of the fast RCA algorithm and without the use of ACE in the lifting so that more improvement is likely possible at this shortest blocklength. Still, this k = 192 PBRL code family outperforms the RCPT codes in the 3GPP-LTE standard [103] at high SNRs and does not have error floors up to the highest SNRs studied in [71]. Other rate-compatible product codes and high-order non-binary turbo codes are designed in [104, 105]. These codes perform well in the short blocklength regime, but have much higher decoding complexity. Their blocklengths are shorter than the codes explored in this chapter.

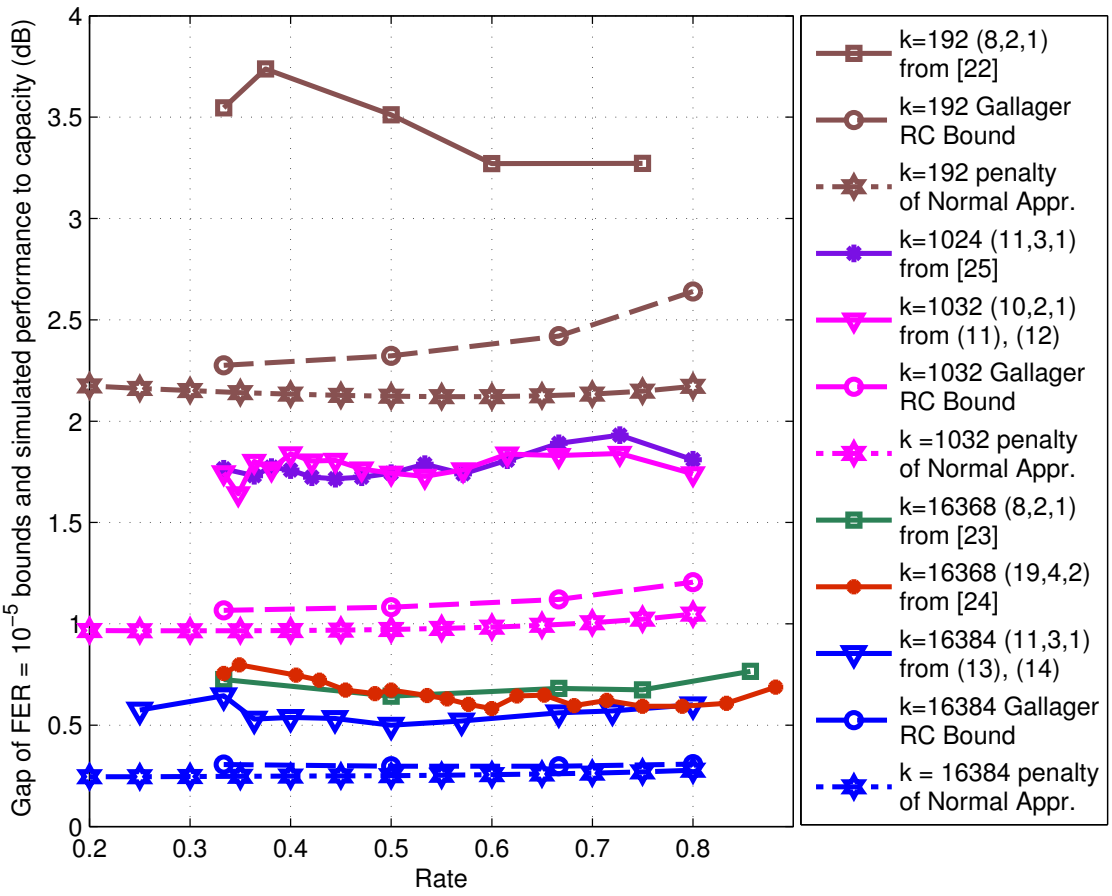


Figure 3.9: SNR gaps in E_b/N_0 between the Shannon limit of the BI-AWGN channel and the required SNR to achieve FER of 10^{-5} for rates in the PBRL code families for five different PBRL code families and the codes from [73] and [74]. The code from [74] has the PBRL structure although it was not a constraint in its design.

3.5 Conclusions

In this chapter, we study the construction and optimization of protograph-based Raptor-like (PBRL) LDPC codes by designing a highest-rate code (HRC) and sequentially adding degree-one variable nodes whose neighboring check node is connected to the variable nodes of the HRC so as to minimize the density evolution threshold. The new connections must obey constraints that control the error floor. Puncturing a single variable node in the HRC improves the threshold performance of PBRL codes.

Instead of the original density evolution, a modified reciprocal channel approximation (RCA) is used to obtain a fast and accurate approximation for the thresholds of PBRL codes to result in reasonable code-design complexity. Once the HRC is designed, the remaining code design including the ACE+CPEG lifting takes a few hours on a typical personal computer.

Controlling the error floor is especially important for short-blocklength PBRL code families. For an example short-blocklength PBRL code family designed for $k = 1032$ we sacrificed threshold in order to improve error floor performance by increasing the number of degree-4 variable nodes in the HRC. Even in our long-blocklength ($k = 16384$) PBRL design example, a small amount of threshold performance is sacrificed in order to ensure good error floor performance by allowing only half of the connections to the punctured node to involve parallel edges.

For both long- and short-blocklength designs, we found that using CPEG and ACE jointly led to a more successful lifting of the protograph so that more threshold-reducing parallel edges could be introduced before causing a problematic error floor.

In summary, this chapter provides a complete design procedure for constructing rate-compatible LDPC code families that perform uniformly close to finite blocklength performance limits for both short ($k = 1032$) and long ($k = 16384$) blocklengths.

$$H_{\mathrm{IRC}}^{(16384)} = \begin{bmatrix}
2 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
2 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 \\
2 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 \\
2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix} \tag{3.14}$$

CHAPTER 4

PBRL Codes for Binary Symmetric and Erasure Channels

This chapter discusses design of protograph-based Raptor-like (PBRL) codes as a class of rate-compatible (RC) LDPC codes for binary erasure channels (BEC) and binary symmetric channels (BSC). Similar to the Raptor codes, the RC property is achieved by X-OR operations of the pre-coded bits. The additional parity bits, which lower the rate, are selected such that their connections in the protograph optimize the density evolution threshold. In order to avoid problematic graphical objects in the lifted bipartite graph and guarantee the linear growth distance property some constraints are imposed in the threshold optimization algorithm. Simulation results are presented for information block sizes of $k = 1032$ and $k = 16384$. These results are compared with finite blocklength bounds of Polyanskiy, Poor, and Verdu (PPV) as well as several asymptotic bounds. The $k = 1032$ code family optimized for BEC operates at various rates in the range of $8/9$ to $8/48$ and has an average mutual information gap of 0.087 bits from the capacity at the frame error rate (FER) of 10^{-5}. The $k = 16384$ code family operates at rates $8/10$ to $8/32$ and has an average mutual information gap of 0.0326 bits from the capacity at FER = 10^{-5}. For BSC, the optimized $k = 1032$ and $k = 16384$ code families operate within an average mutual information gap of 0.147 and 0.073 bits from the BSC capacity at FER = 10^{-5}.

4.1 Introduction

Low-Density Parity-Check (LDPC) codes were first introduced by Gallager in his dissertation in 1963 [54]. These codes are identified by their parity-check matrices. Irregular LDPC codes have parity-check matrices with various number of ones (weights) in each column and row. By optimiz-

ing the variable-nodes' column weights and check-nodes' row weights (degree distributions), Luby et al. [58] showed that irregular LDPC codes can achieve rates very close to capacity. Richardson et al. [59] invented an algorithm called density evolution (DE) to design and analyze the optimal degree distributions of infinitely long LDPC codes. There are many papers analyzing LDPC codes over BEC. For examples see [106], [107], [108], [109], [110], [111], and [112].

Traditionally, rate-compatible (RC) codes are designed with optimized puncturing patterns with good error-rate performance across a family of rates. In general, any RC code may be considered as a low-rate code that is punctured to produce higher rates. However, in this chapter we use the method of extending a protograph [67, 73], where the higher-rate codes are designed first and the lower-rate codes are designed based on their higher-rate counterparts. We use DE and extrinsic information transfer (EXIT) threshold maximization as the main design criterion to select the connections between the variable and check nodes. After using DE thresholds to optimize the protograph, a copy-and-permute operation, often referred to as "lifting", is applied to the protograph to obtain larger parity check matrices of the desired sizes. Refer to [75, 76] for a thorough discussion on photographs and lifting algorithms.

In [72] the PBRL approach is applied to the design of RC LDPC families over additive white Gaussian noise (AWGN) channels. Also, reciprocal channel approximation (RCA) for AWGN channels replaces the standard DE algorithm of [59] to provide a fast and accurate approximation for DE threshold to speed up the optimization process. In previous chapter, the results of [71] and [72] are unified and extended; Error floor performance is improved both with edge constraints added to the RCA-based greedy optimization and by using approximate cycle extrinsic message degree (EMD) [101] constraints in addition to CPEG [113] in the lifting process.

Similar to [71, 72], this work restricts the code family to have the basic structure of Raptor codes [77] and this class of RC LDPC codes is called protograph-based raptor-like (PBRL) LDPC codes [71, 72]. In this chapter, we discuss the construction and optimization of PBRL codes specifically for the binary erasure channel (BEC) and binary symmetric channel (BSC) and present simulation results for codes with information lengths of $k = 1032$ and $k = 16384$. These results are compared with finite-blocklength approximations and asymptotic bounds. Constraining the design

to enforce Raptor code structure makes the construction and optimization of PBRL codes manageable while these codes still provide outstanding performance and extensive rate-compatibility. As in the previous chapters, lifting is performed using a combination of approximate cycle EMD (ACE) and circulant progressive edge growth (CPEG) liftings. We refer to this lifting algorithm that combines ACE and CPEG as an ACE+CPEG lifter.

The rest of the chapter is organized as follows: Sec. 4.2 presents the PBRL code structure. Sec. 4.3 provides the design procedure to construct PBRL codes for BEC. Sec. 4.4 provides a summary of the existing finite-blocklength bounds and approximations for BEC. Sec. 4.5 constructs two examples of PBRL code families for information block size of $k = 1032$ and $k = 16384$, and presents the analysis and simulation results. Sec. 4.6 proposes a fast and accurate alternative technique to DE algorithm for the BSC. Sec. 4.7 presents two code families with information block sizes of $k = 1032$ and $k = 16384$ optimized for the BSC. Finally Sec. 4.9 concludes the chapter.

4.2 Protograph-Based Raptor-Like LDPC Code

This section introduces the structure of PBRL codes. The parity check matrix of a protograph is called protomatrix and can be presented by a bipartite graph. Let $\mathbf{0}$ be the all-zeros matrix and $\mathbf{I}$ be the identity matrix, the protomatrix of PBRL codes has a general form of

$$H = \begin{bmatrix} H_{\text{HRC}} & \mathbf{0} \\ H_{\text{IRC}} & \mathbf{I} \end{bmatrix}, \tag{4.1}$$

where HRC describes the highest-rate protograph and IRC corresponds to the incremental redundancy protograph.

Fig. 4.1 shows the protograph structure of a PBRL code with HRC part on the left and IRC part on the right. The IRC part provides lower rates as gradually more of its variable nodes are transmitted, starting from the first variable node on the top. The use of the punctured node as shown in the protogaph of Fig. 4.1 improves the iterative decoding threshold.

The protomatrix of the protograph shown in Fig. 4.1 has HRC and IRC parts of

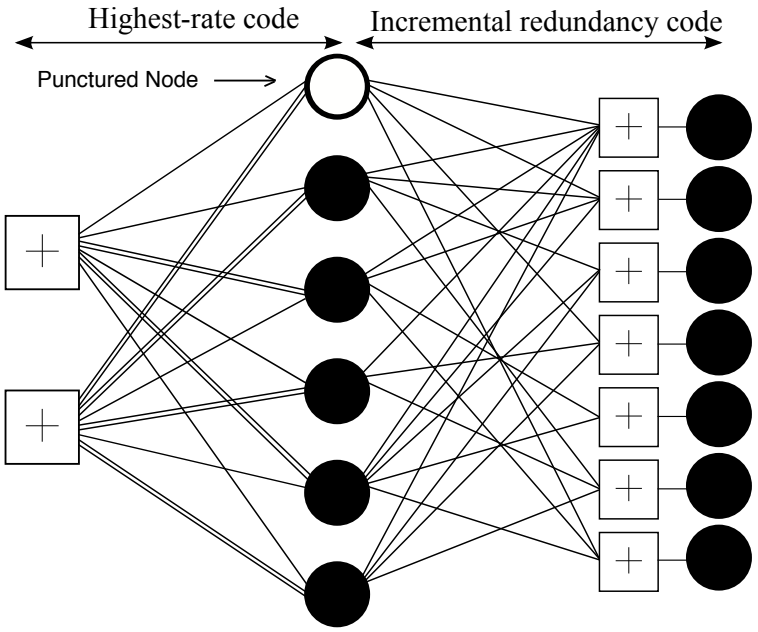


Figure 4.1: Protograph for a PBRL code with a punctured node and a highest-rate code (HRC) with rate $4/5$ followed by an incremental redundancy code (IRC) that uses only degree-one variable nodes. The IRC provides lower rates as more of its variable nodes are included, starting from the top.

$$H_{\text{HRC}} = \begin{bmatrix} 1 & 1 & 2 & 1 & 2 & 1 \\ 2 & 2 & 1 & 2 & 1 & 2 \end{bmatrix} \tag{4.2}$$

and

$$H_{\text{IRC}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}. \tag{4.3}$$

The IRC is created by the addition of variable nodes that are exclusively degree-one, which correspond to the identity matrix in (4.1). The first column of the above matrices corresponds to the

punctured node. The final protograph determined by HRC and IRC protographs is lifted to produce the actual code. The lifting algorithm uses circulant matrices in the lifting process to guarantee a minimum number of extrinsic connections (ACE number) for each cycle as well as a minimum girth.

The Raptor-like structure is quite restrictive; however, the structural constraints do not limit the performance of PBRL codes compared to less-restrictive RC LDPC codes obtained by extension. One of the main conclusions of our analysis is that we obtain Raptor-like protographs with very good iterative decoding thresholds despite these constraints. Similar observations for AWGN channels have been made in previous chapter where the resulting finite-length PBRL codes outperform existing RC LDPC codes that have been designed without the constraint of a Raptor-like structure. Furthermore, PBRL codes facilitate efficient encoding and decoding as discussed in the previous chapter.

4.3 Optimization of PBRL LDPC Codes for BEC

This section presents the optimization procedures for finding the HRC and IRC components that comprise a good PBRL code family for the BEC. The optimization criteria for both long and short blocklength codes is primarily based on maximizing the iterative decoding threshold (channel erasure probability) over the BEC at each rate, while enforcing constraints on the connections to avoid problematic error floors. These constraints are more stringent in design of short blocklength codes. For the iterative decoding threshold computations, we use the reciprocal channel approximation (RCA) algorithm as an efficient and accurate alternative to DE algorithm to calculate the iterative decoding thresholds. Note that the RCA for BEC represents the exact DE algorithm.

4.3.1 Density Evolution with Reciprocal Channel Approximation

The asymptotic iterative-decoding threshold characterizes the performance of an ensemble of the LDPC codes with identical protographs [91]. This threshold for BEC indicates the maximum channel-erasure probability p_{it} to have the expected bit-error rate go to zero as the blocklength

grows to infinity. The RCA algorithm, originally proposed in [47] for regular LDPC codes uses a single parameter to approximately characterize the distribution of the messages exchanged between variable and check nodes over a BEC or an AWGN channel.

4.3.2 Density Evolution in Protographs for BEC

For BEC a single real-valued parameter, the probability of erasure p, serves as a stand-in for full density evolution. Alternatively, we can use the RCA algorithm and track the self-information of an erasure, $s = -\ln p$, which is additive at variable nodes. The reciprocal parameter, $r = -\ln(1-p)$, the self-information of a non-erasure, is additive at the check nodes. The parameter r satisfies $C(s) + C(r) = 1$, which implies $e^{-s} + e^{-r} = 1$ where C is the capacity of the channel. Note that $r = R(s)$ and $s = R(r)$. The parameters r and s are related to each other by the self-inverting function $R(s) = C^{-1}(1 - C(s)) = -\ln(1 - e^{-s})$ where C^{-1} is the inverse of C. With the above definitions for BEC, the RCA algorithm represents the exact one-dimensional density evolution. RCA in [47] was used to find the optimal degree distribution for unstructured regular LDPC codes over BEC. In this section, we extend the idea of using RCA to efficiently calculate the DE threshold to protographs over BEC.

To calculate the DE threshold of a protograph using RCA, we first identify all transmitted variable nodes and select a target channel erasure probability $p_{ch} = e^{-s_{ch}}$. As shown in Fig. 4.2, the transformed messages obtained from $\vec{s}_e$ are passed along edges leaving the variable nodes ($\vec{s}_e = s_{ch}$ for the transmitted nodes and $\vec{s}_e = 0$ for the punctured nodes). The transformation $R(\vec{s}_e)$ is applied and an extrinsic return message, $\overleftarrow{r}_e$ is determined by computing the sum of all incoming messages except for the one along edge e. Transformation $R(\cdot)$ is then reapplied to produce $\overleftarrow{s}_e$. The process continues and the iterative decoding threshold $p_{it} = e^{-s_{ch}}$ is determined by the smallest value of s_{ch} such that an unbounded growth of all messages $\vec{s}_e$ can be achieved.

Algorithm 3 (RCA) *Let e_v (e_c) be the set of edges connected to the variable node v (check node c). For iterations $n = 0, \ldots, N$, and for all edges e in the protograph, the RCA computes the messages as follows:*

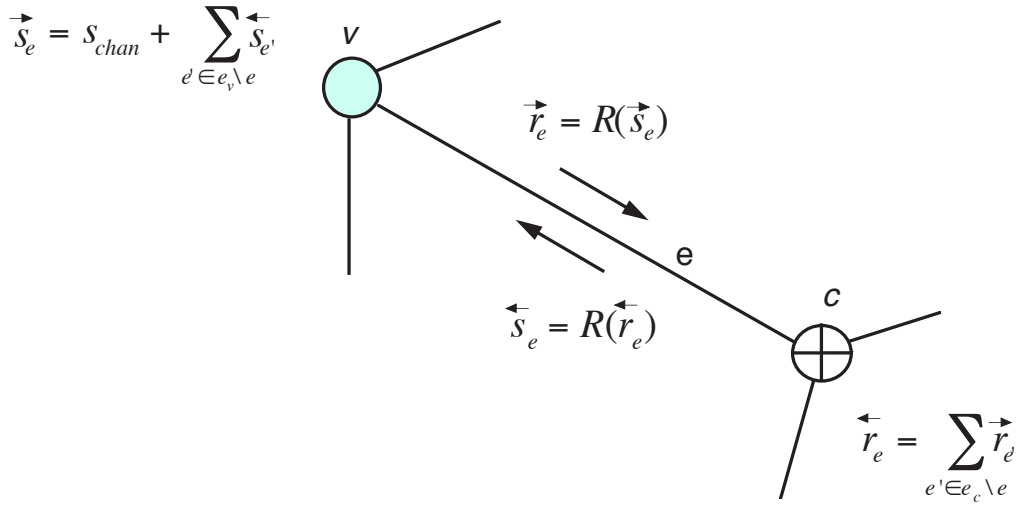


Figure 4.2: RCA algorithm to calculate the density evolution threshold in protographs for BEC.

0) For edges e connected to punctured variable nodes, set $\vec{s}_e^{(0)} = 0$. For all other edges set $\vec{s}_e^{(0)} = s_{ch} = -\ln(p_{ch})$.

1) For $n = 1, \ldots, N$, compute the following:

$$\vec{r}_e^{(n-1)} = R\left(\vec{s}_{e'}^{(n-1)}\right) \tag{4.4}$$

$$\overleftarrow{r}_e^{(n)} = \sum_{e' \in e_c \setminus e} \vec{r}_{e'}^{(n-1)}, \tag{4.5}$$

$$\overleftarrow{s}_e^{(n)} = R\left(\overleftarrow{r}_{e'}^{(n)}\right) \tag{4.6}$$

$$\vec{s}_e^{(n)} = \vec{s}_e^{(0)} + \sum_{e' \in e_v \setminus e} \overleftarrow{s}_{e'}^{(n)}, \tag{4.7}$$

2) When $n = N$ compute $S_v^{(N)}$ for all variable nodes v

$$S_v^{(N)} = \sum_{e \in e_v} \overleftarrow{s}_e^{(N)}, \tag{4.8}$$

For large enough N, and some large T, the maximum p_{ch} that produces $S_v^{(N)} > T$ for all variable nodes represents the iterative decoding threshold p_{it} for BEC.

4.3.3 Maximum a Posteriori Probability (MAP) Threshold of Protographs for the Erasure Channel

To compute an upper bound to the maximum a posteriori probability (MAP) threshold we use the area theorem [114]. We use the results from the final step of RCA algorithm described in the previous subsection for binary erasure channel by computing $S_v^{(N)}$ for each transmitted variable node for range of $p_{ch} \in [p_{it}, 1]$. Starting with $p_{ch} = 1$ and decreasing it, for each value of p_{ch}, we compute $h^{BP}(p_{ch}) = \frac{1}{|v|-|v_p|} \sum_{v' \in v \setminus v_p} e^{-S_{v'}^{(N)}}$ where $h^{BP}(p_{ch})$ is the output extrinsic erasure probability when the input channel erasure probability is p_{ch}. In the equation for $h^{BP}(p_{ch})$, v is the set of all variable nodes in the photograph and v_p is the set of all punctured variable nodes. As discussed in [106, 112, 115], by using the area theorem, p_{MAP}^{*} is obtained as the solution to

$$R_c = \int_{p_{MAP}^{*}}^{1} h^{BP}(p_{ch}) dp_{ch} \,, \tag{4.9}$$

where p_{MAP}^{*} is an upper bound to the MAP threshold p_{MAP}. For some ensembles (e.g. regular LDPC), $h^{BP}(p_{ch}) = h^{MAP}(p_{ch})$ for $p_{MAP} \leq p_{ch} \leq 1$, in such case $p_{MAP}^{*} = p_{MAP}$ [112, 115]. Let $p_{cap} = 1 - R_c$, where R_c is the coding rate in BEC, then $p_{it} \leq p_{MAP} \leq p_{cap}$.

4.3.4 Optimizing the Highest-Rate Code (HRC)

Primarily, the HRC photograph is simply the protograph of a good code at the highest desired rate. Our protograph code design mainly follows the guidelines in work of Divsalar et al. [76, Sec. III] but uses the RCA optimization algorithm for fast and efficient threshold calculation. The main conclusions of [76] are that protograph ensembles with a minimum variable-node degree of 3 or higher are guaranteed to have linear minimum distance growth with the blocklength. Furthermore, as explained in [76], puncturing a node in the HRC can improve the threshold performance. The HRC protographs designed in Secs. 4.5.1, 4.5.2, 4.7.1, and 4.7.2 ensure the linear minimum distance growth property and have a punctured node.

4.3.5 Optimizing the Incremental-Redundancy Code (IRC)

The IRC part is optimized by using the RCA algorithm of Sec. 4.3.2 and choosing the new rows that optimize the iterative thresholds. The other considerations that are taken into account are the linear minimum distance growth and low error-floor properties enforced by the additional constraints.

PBRL code families retain linear minimum distance growth by requiring that each new row in H_{IRC} has at least two nonzero values. As long as the HRC has linear minimum distance growth, ensuring non-trivial connections for each new check node in the IRC in this way preserves linear minimum distance growth for all rates.

There are two key features that dramatically improve protograph thresholds at low rates. The first is that parallel edges improve the threshold and the second is that the punctured node of the precode should connect to all (or almost all) of check nodes in the IRC part with at least a single edge. The increased connectivity to the punctured node through single and parallel edges improves the thresholds significantly for the lower rates. The threshold improvement, however, diminishes as the number of parallel edges increases. Moreover, even at long blocklengths lifting cannot overcome the resulting damage to the graphical structure of the code if the parallel edges are added too aggressively. Thus, an important aspect of IRC design is controlling the number of parallel edges to give the best possible thresholds for which lifting can successfully translate the excellent thresholds into excellent frame error rate (FER) and bit error rate (BER) performance.

As a result, the greedy optimization of Algorithm 3 is applied with additional constraints that require full connectivity to the punctured node with at least single edges and that control the number of parallel edges. For very short blocklengths and small protographs, only one pair of parallel edges connecting to the punctured node may be allowed as in the most effective designs in [71]. As the blocklength of the lifted codes and/or the size of the protograph increase, more parallel edges can be utilized.

4.4 upper and lower bounds for BEC

Gallager [54] proposed an upper bound on the probability of error for discrete memoryless channels. The upper bound can be expressed in terms of Kullback-Leibler distance. The relative entropy of p^* with respect to p, also called the Kullback-Leibler distance, is defined by

$$D(p^*, p) = p^* \ln \frac{p^*}{p} + (1 - p^*) \ln \frac{1 - p^*}{1 - p}. \tag{4.10}$$

If $p^* = p_{cap} = 1 - R_c$, which is maximum erasure probability at capacity of BEC, and $p < p^*$ is the channel erasure probability, the FER for a random (n, k) code with code rate $R_c = \frac{k}{n}$ is given by

$$P_e \leq e^{-nD(p_{cap}, p)}. \tag{4.11}$$

The PPV [102] achievable upper bound on the probability of error is

$$\begin{aligned} \bar{P}_e \quad \leq \quad & 1 - \sum_{j=0}^{n} \binom{n}{j} (1-p)^j p^{n-j} \sum_{m=0}^{2^{nR}-1} \frac{1}{m+1} \binom{2^{nR-1}}{m} \\ & \times 2^{-jm} (1 - 2^{-j})^{2^{nR}-1-m}. \end{aligned} \tag{4.12}$$

The PPV [102] lower bound using meta converse is given by

$$P_e \geq \sum_{e=[n-k]+1}^{n} \binom{n}{e} (1-p)^{n-e} p^e (1 - 2^{n-k-e}). \tag{4.13}$$

The following theorem is also due to PPV [102].

Theorem 4 *For the BEC with erasure probability p,*

$$nR_c = n(1-p) - \sqrt{np(1-p)} Q^{-1}(\epsilon) + O(1), \tag{4.14}$$

or $\epsilon \approx Q((1 - p - R_c)\sqrt{\frac{n}{p(1-p)}})$, regardless of whether ϵ is maximal or average probability of error.

The above PPV approximation is used to compare with the simulation results for $k = 1032$ and $k = 16384$ raptor-like codes for various code rates discussed later in Fig. 4.6.

4.5 Design Examples For BEC

This section presents two families of RC PBRL codes. Subsection 4.5.1 presents the design procedure for a short-blocklength (k = 1032) code and subsection 4.5.2 presents the design procedure for a long-blocklength (k = 16384) code.

4.5.1 Short-Blocklength PBRL Design Example

This subsection provides an example PBRL protograph family designed for information blocklength k = 1032. The HRC uses the protograph

$$H_{\text{HRC}}^{(1032)} = \begin{bmatrix} 3 & 3 & 3 & 1 & 1 & 1 & 3 & 1 & 2 & 1 \\ 1 & 1 & 1 & 3 & 3 & 3 & 1 & 2 & 1 & 2 \end{bmatrix}. \tag{4.15}$$

The IRC adds up to 39 degree-one variable nodes to provide a range of rates of the form $8/(9+i)$ from $8/9$ to $8/48$. After the design of the HRC protomatrix, IRC protomatrix is obtained by optimizing the threshold for each successive rate under the constraints on edges to ensure low error floor.

Fig. 4.3 shows an example of EXIT function using RCA to compute the iterative decoding threshold p_{it}. As discussed earlier, p^*_{MAP} is calculated using the area theorem as an upper bound on MAP threshold. The shaded region under the iterative BP curve has an area equal to the code rate R_c.

For the protomatrices of (4.15) and (4.16), Table 4.1 shows p_{it}, p^*_{MAP}, and p_{cap} . The parameter p_{it} is the highest channel-erasure probability that an iterative BP decoder can support as the blocklength of the LDPC code grows to infinity. p^*_{MAP} is the highest channel-erasure probability that an MAP decoder supports asymptotically. $p_{cap} = 1 - R_c$ is the maximum erasure probability at capacity for BEC. The "Gap" column of Table 4.1 shows the mutual information gap between the iterative threshold and the capacity in number of bits. For k = 1032 code family the average gap is 0.020 bits.

$$H_{\mathrm{IRC}}^{(1032)} = \begin{bmatrix}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
2 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
2 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
2 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix} \tag{4.16}$$

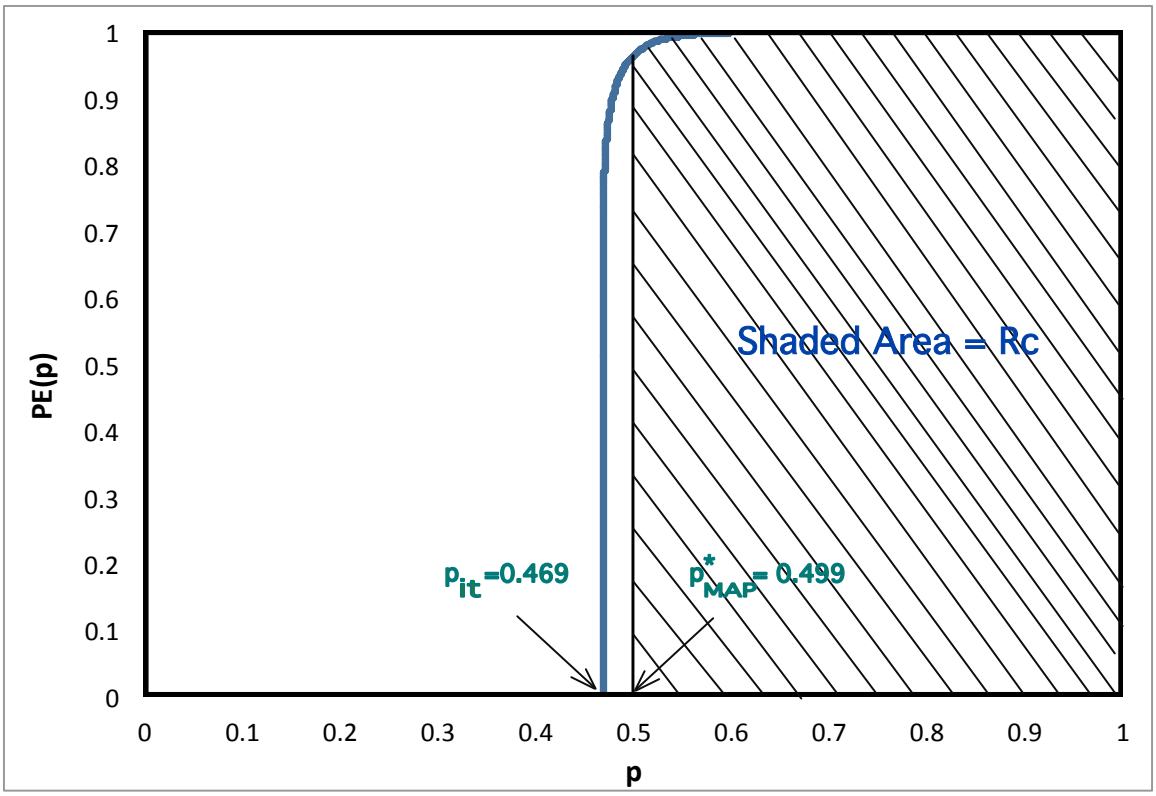


Figure 4.3: EXIT function using RCA for the rate-1/2 code of (4.15) and (4.16).

4.5.2 Long-Blocklength PBRL Design Example

In this subsection, we design a PBRL code family with information blocklength $k = 16384$. The protograph family provides a range of rates of the form $8/(10+i)$ from $8/10$ to $8/32$. The HRC protograph for the $k = 16384$ PBRL code is

$$H_{\text{HRC}}^{(16384)} = \begin{bmatrix} 3 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \end{bmatrix}. \tag{4.17}$$

Similar to the design of $k = 1032$ code family, RCA algorithm optimizes the threshold corresponding to the connections of the IRC check nodes to the variable nodes of the HRC part of (4.18). In the optimization process, each connection to the punctured node may have zero, one, or two edges leading to values of 0, 1, and 2 in the first column of (4.18). Each connection to a

non-punctured node may have zero or one edge in the other columns of (4.18). Table 4.2 shows p_{it}, $p^{\star}_{MAP}$, and p_{cap} for the $k = 16384$ code family with an average gap of 0.0138 across various rates.

4.5.3 Short and Long-Blocklength PBRL Results for BEC

After obtaining the HRC and IRC parts, two-step ACE+CPEG lifting as in [76] is used to lift the lowest rate code with a required girth of six and eight for the $k = 1032$ and $k = 16384$ families. For the $k = 1032$ family, the pre-lifting step has a lifting number of 3 to remove the parallel edges. The lifting number in the second stage is 43, resulting in an information blocklength of $k = 1032$. For the $k = 16384$ family, the pre-lifting step has a lifting number of 4 and the lifting number in the second stage is 512.

Figs. 4.4 and 4.5 show the frame-error rate performance of the PBRL codes designed for $k = 1032$ and $k = 16384$, respectively over BEC. For a fixed rate, the $k = 16384$ PBRL code family in Fig 4.5 achieves a particular FER at a higher BEC erasure probability than the codes designed for $k = 1032$ in Fig 4.4. The BEC erasure probability obtained at $FER = 10^{-5}$ is used for the analysis in Fig 4.6.

Fig. 4.6 shows the mutual information gap from capacity in number of bits for the PBRL codes constructed for $k = 1032$ and $k = 16384$ at a frame error rate of 10^{-5}. These simulations use floating-point iterative decoders with a flooding schedule for message-passing. Decoding terminates early if all parity checks are satisfied before reaching the maximum number (300) of iterations. A more efficient decoder for BEC is the peeling decoder. If the number of iterations in BP decoder is high enough, e.g. 300, the performance of these two decoding methods is the same. For BEC, it is also possible to use maximum likelihood decoding at the expense of higher complexity.

For the $k = 1032$ family, the simulations show a gap from the capacity of 0.0873 bits while the gap to the PPV finite blocklength analysis is only 0.0494 bits. For the $k = 16384$ family, the gaps from the capacity and PPV finite blocklength analysis are 0.0326 and 0.0236 bits, respectively. It is surprising that a greedy threshold optimization algorithm alone can produce a code family with

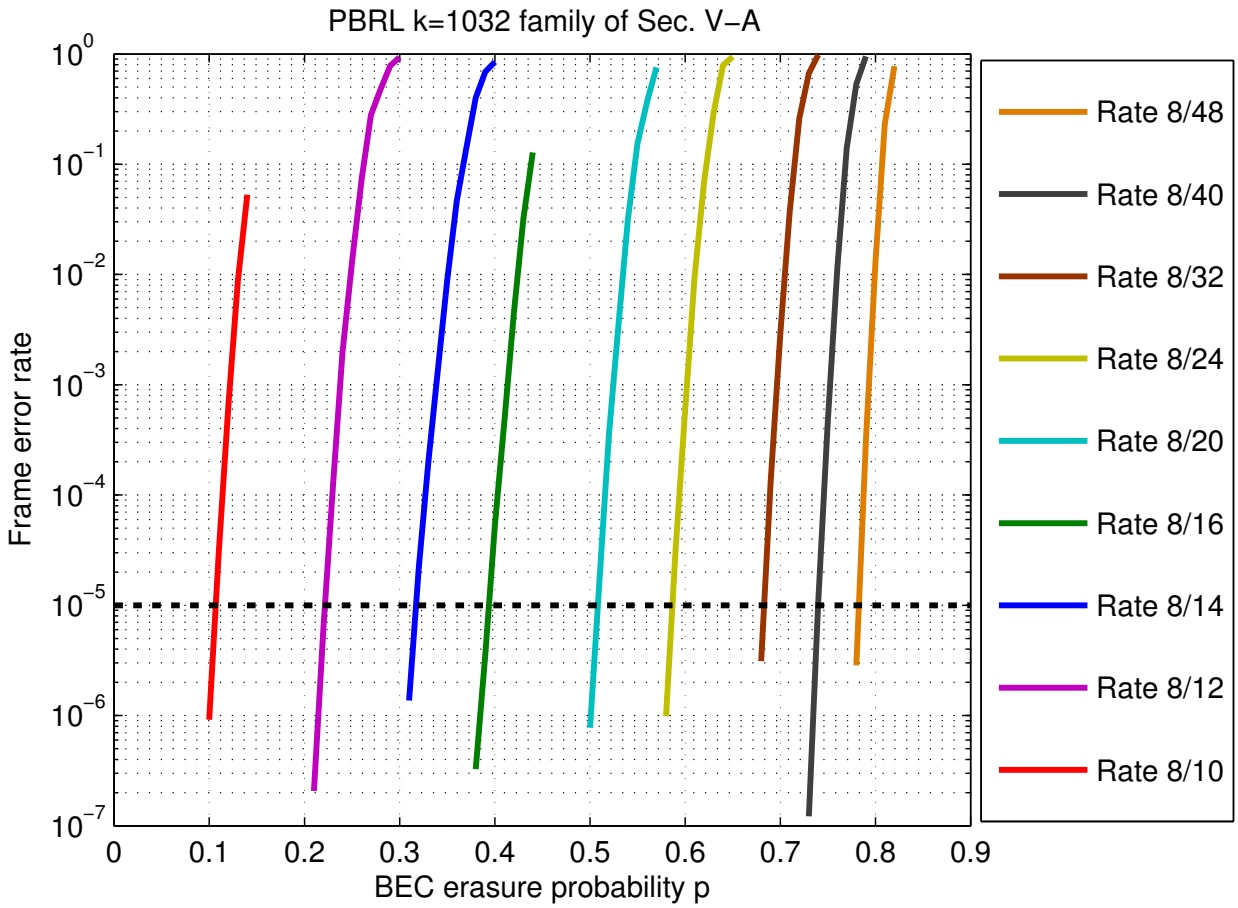


Figure 4.4: Frame error rates for the $k = 1032$ protographs defined by (4.15) and (4.16) over BEC.

such an outstanding performance.

Fig. 4.7 illustrates the ACE number η values for each rate. A pair (d_{ACE}, η) implies that every cycle consisting of up to d_{ACE} variable nodes, i.e. every cycle of length up to $2d_{\text{ACE}}$, is connected to at least η extrinsic check nodes. For more discussion of the ACE algorithm see [101]. All the cycles with length 10 or less in $k = 1032$ code have at least 6 extrinsic connections for the rate-1/2 code. Similarly, all the cycles with length 12 or less in $k = 16384$ code have at least 7 extrinsic connections for the rate-1/2 code.

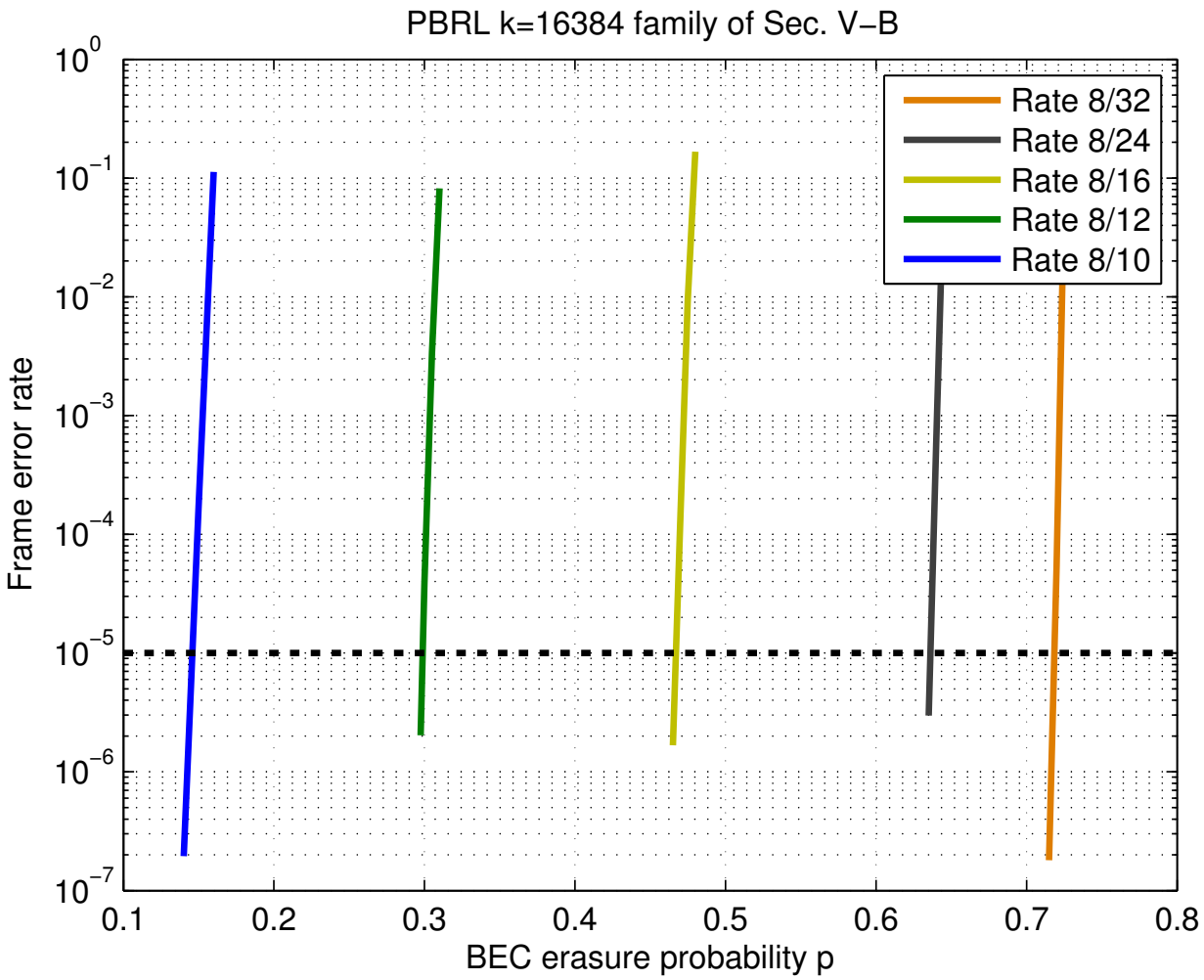


Figure 4.5: Frame error rates for the $k = 16384$ protographs defined by (4.17) and (4.18) BEC.

4.6 Density Evolution in Protographs for BSC

In this section we present two RCA algorithms to approximate the DE threshold for LDPC codes transmitted over binary symmetric channels (BSC). In both approximations, we assume a binary symmetric channel with the transition probability p. Furthermore, due to the symmetry of the channel, we assume that the all-zeros codeword is transmitted. The goal of these approximations is to find simple alternative algorithms to calculate the density evolution threshold by tracking a single parameter.

The first method is based on the similar ideas used in the analysis of BI-AWGN channels. RCA

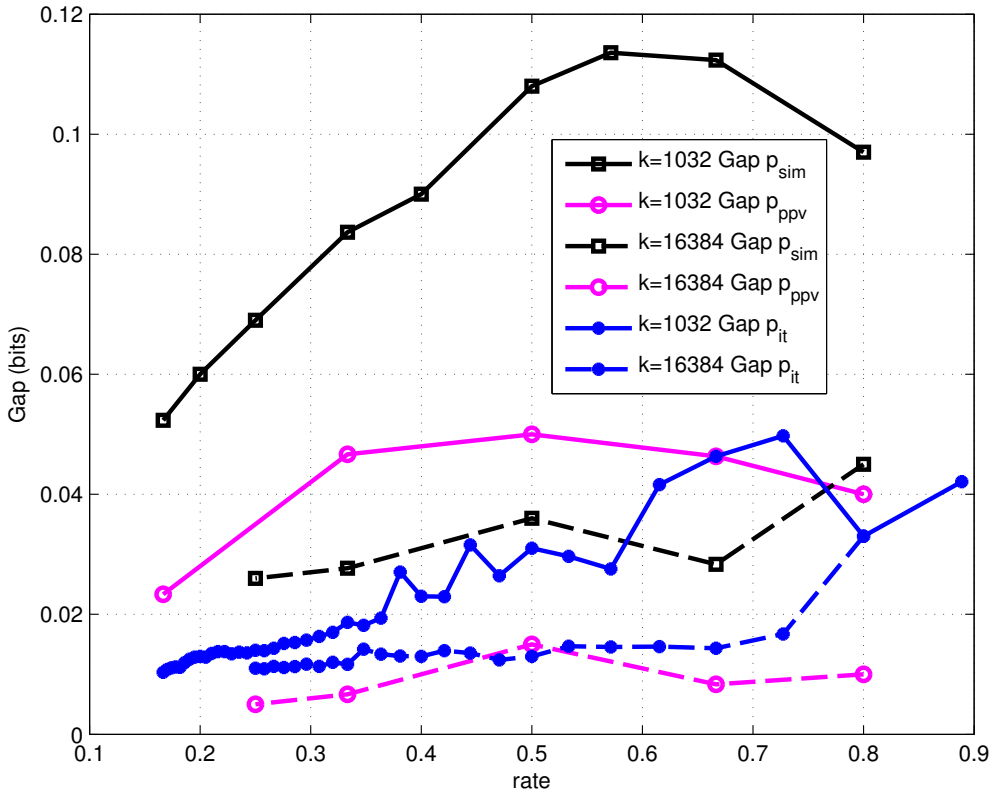


Figure 4.6: Mutual information gap from capacity and PPV finite-blocklength analysis in number of bits for the PBRL codes constructed for $k = 1032$ and $k = 16384$ at a frame error rate of 10^{-5}. The protograph for the $k = 1032$ PBRL code is based on (4.15,4.16,4.17, and 4.18).

for BI-AWGN channels uses a real-valued parameter s, the SNR, to approximate the probability density function of the LLR values transmitted between variable and check nodes. The function used in RCA algorithm for BI-AWGN channels is $C(s)$ which gives the capacity of the BI-AWGN channel with SNR s. For computational simplicity and numerical precision we prefer to express $C(s)$ as

$$C(s) = 1 - \int_{-\infty}^{\infty} \log_2\left(1 + e^{-(2\sqrt{2s}u+2s)}\right) \frac{e^{-u^2}}{\sqrt{\pi}} du, \tag{4.19}$$

which is obtained from [49, (15)] through a change of variables.

For the threshold analysis of BSC, as shown in Fig. 4.8, the variable-node update operation includes a summation of LLR distribution between an approximated normal distribution and the probability mass function corresponding to the LLR distribution of the messages that the variable

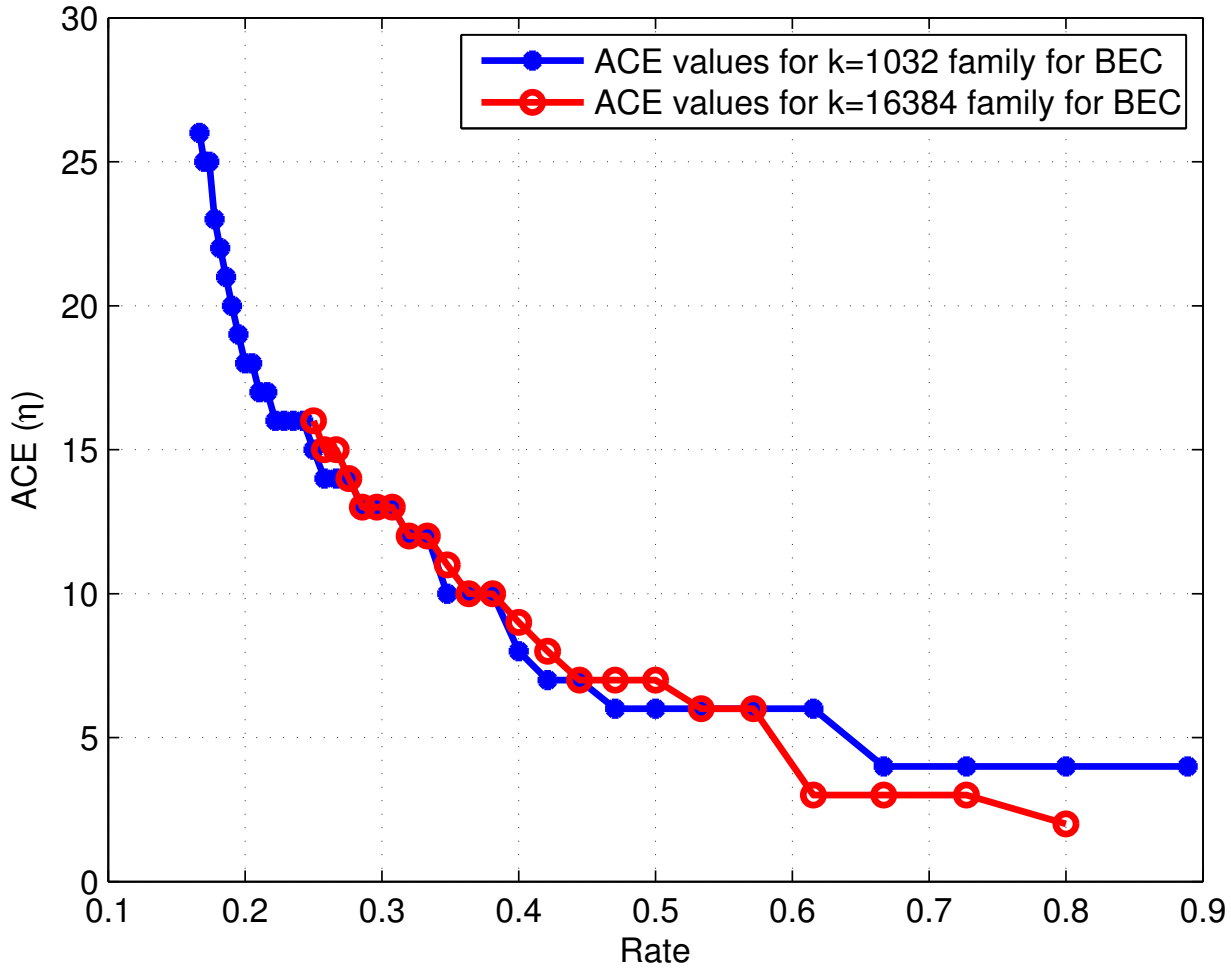


Figure 4.7: ACE value (η) versus rate for the short blocklength (k = 1032) with d_{ACE} = 5 and long blocklength (k = 16384) with d_{ACE} = 6 codes designed for BEC.

nodes receive from the channel. The resulting distribution of the messages has a bi-modal distribution. We approximate this bi-modal distribution with a normal distribution that carries the same amount of mutual information (capacity). In order to calculate the amount of information that the bi-modal LLR distribution of Fig. 4.8 carries for $\mu = 2s$, we can modify [4.19] to take into account the shift, $L = log(\frac{1-p}{p})$. The parameters L and $-L$ are the discrete support values for the LLR distribution of messages that the variable nodes receive from the channel. It needs to be noted that the variance of each of the modes in the bi-modal distribution individually is 2μ.

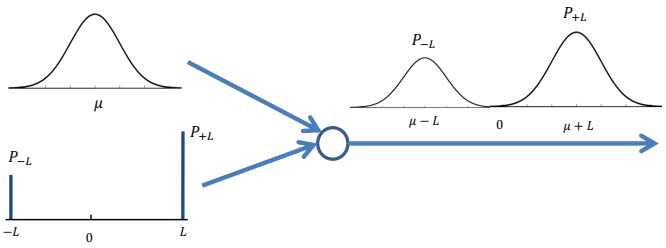


Figure 4.8: Variable-node update is the summation of the input LLR messages. Considering a simple case of variable-node update with two input LLR messages, the resulting LLR distribution is the convolution of the input LLR distributions, which result in a bi-modal distribution.

The modified version of 4.19 which includes the parameter L is given by

$$C(s,L) = 1 - p\int_{-\infty}^{\infty} \log_2\left(1 + e^{-(2\sqrt{2s}u+2s+L)}\right)\frac{e^{-u^2}}{\sqrt{\pi}}du$$
$$-(1-p)\int_{-\infty}^{\infty} \log_2\left(1 + e^{-(2\sqrt{2s}u+2s-L)}\right)\frac{e^{-u^2}}{\sqrt{\pi}}du, \tag{4.20}$$

.

As explained earlier, for each given raw error probability of the channel (p) RCA algorithm can be used to approximate the density evolution threshold by only tracking a single parameter s. The major complexity in using this approximation is in calculating the integrals in [4.20] in each iteration for variable node updates. The complexity of calculating the integrals can be high enough that may hinder designing large LDPC codes over BSC. In order to simply this calculation, we introduce another approximation which avoids the complicated integral operations. The threshold values obtained from the second method is compared with the threshold values from the first RCA algorithm and the true threshold from the density evolution algorithm. The threshold values obtained from the second method RCA_2 are used to design the LDPC codes for BSC in this work.

For the second method, we use the capacity of BSC ($C(p) = 1 - H(p)$) where $H(p)$ is binary entropy function given by

$$H(p) = -p\log_2(p) - (1-p)\log_2(1-p). \tag{4.21}$$

.

The RCA method is based on a transformation $R(s)$ of one dimensional message from variable nodes (VN) to check nodes (CN) such that the operations at VN and CN are additive. Similar to the RCA algorithm

for BEC, The reciprocal self-inverting function $R(s)$ is defined as

$$R(s) = C^{-1}(1 - C(s)). \tag{4.22}$$

We define the parameter r satisfying $C(s) + C(r) = 1$. This implies that $r = R(s)$ and $s = R(r)$. It is necessary to relate the parameter s to the parameter p, the crossover probability of BSC, such that using the transformation $R(s)$ provides accurate results for iterative decoding threshold close enough to the true density evolution threshold for BSC. The problem of choosing such transformation $R(s)$ and relating s to p has remained unsolved since the introduction of RCA for BI-AWGN channels. Here we propose $s = [Q^{-1}(p)]^2$, which results in defining the channel capacity as

$$C(s) = 1 - H(Q(\sqrt{s})). \tag{4.23}$$

With this definition for the BSC, the RCA represents an approximation to the one-dimensional density evolution. To apply density evolution to a protograph, define $s_{chl} = [Q^{-1}(p)]^2$. The remaining parts of the RCA algorithm for BSC is identical to the ones discussed in Sec. 4.3.2 for BEC. We tried other transformation candidates based on matching the mutual information or using the stability conditions for BSC and AWGN channels, but the transformation $s = [Q^{-1}(p)]^2$ gave us the best results very close to the true thresholds obtained from the full DE algorithm. Note that using exact DE for BSC to optimize the codes may take at least 100 times longer than the method proposed in this work.

4.7 Design Examples for BSC

This section presents two families of RC PBRL codes. Subsection 4.7.1 presents the design procedure for a short-blocklength codes ($k = 1032$) and subsection 4.7.2 presents the design procedure for a long blocklength code ($k = 16384$).

4.7.1 Short-Blocklength PBRL Design Example for BSC

This subsection provides an example PBRL protograph family designed for information blocklength $k = 1032$. The HRC uses the protograph

$$H_{\text{HRC}}^{(1032)} = \begin{bmatrix} 3 & 3 & 3 & 1 & 1 & 1 & 3 & 1 & 2 & 1 \\ 1 & 1 & 1 & 3 & 3 & 3 & 1 & 2 & 1 & 2 \end{bmatrix}. \tag{4.24}$$

The IRC adds up to 39 degree-one variable nodes to provide a range of rates of the form $8/(9+i)$ from $8/9$ to $8/48$. The first variable node in the HRC part is always punctured.

After the design of the HRC protomatrix, IRC protomatrix is obtained by optimizing the threshold for each successive rate in a greedy fashion under the constraints on edges to ensure low error floor.

The finite blocklength discussion in this sections follows the analysis from [102] for BSC. For equiprobable input distributions with BSC crossover probability p, blocklength n, and the number of information messages $M = 2^k$, there exists an (n, M, ϵ) code with a maximal probability of error of ϵ such that

$$\epsilon \leq \sum_{t=0}^{n} \binom{n}{t} p^t (1-p)^{n-t} \min \left\{ 1, (M-1) \sum_{i=0}^{t} \binom{n}{i} 2^{-n} \right\}. \tag{4.26}$$

For a given k and for each rate R_C, we use (4.26) as an approximation on the maximum cross over probability p to achieve a desired frame error probability ϵ.

For the protomatrices of (4.24) and (4.25), Table 4.3 shows p_{it}, p_{ppv}, and p_{cap} . The parameter p_{it} is the highest channel transition probability that an iterative BP decoder can support as the blocklength of the LDPC code grows to infinity. p_{ppv} is the highest channel transition probability obtained from the PPV finite blocklength analysis. The solution to $R_c = 1 - H(p_{cap})$ for p_{cap} is the maximum transition probability at capacity of BSC. The "Gap" column of Table 4.3 shows the mutual information gap between the iterative threshold and the capacity in bits. For $k = 1032$ code family, the average gap is 0.021.

After obtaining the HRC and IRC parts, similar to the codes designed for BEC, two-step ACE+CPEG lifting is used to lift the lowest-rate code with a required girth of six for the $k = 1032$ family. The pre-lifting step has a lifting number of 3 and the lifting number in the second stage is 43.

4.7.2 Long-Blocklength PBRL Design Example for BSC

This subsection designs a PBRL code family with information blocklength $k = 16384$. The protograph family provides a range of rates of the form $8/(10+i)$ from $8/10$ to $8/32$. The HRC protograph for the $k = 16384$ PBRL code is

$$H_{\text{HRC}}^{(16384)} = \begin{bmatrix} 3 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \end{bmatrix}. \tag{4.27}$$

$$H_{\mathrm{IRC}}^{(1032)} = \begin{bmatrix}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
2 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
2 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{4.25}$$

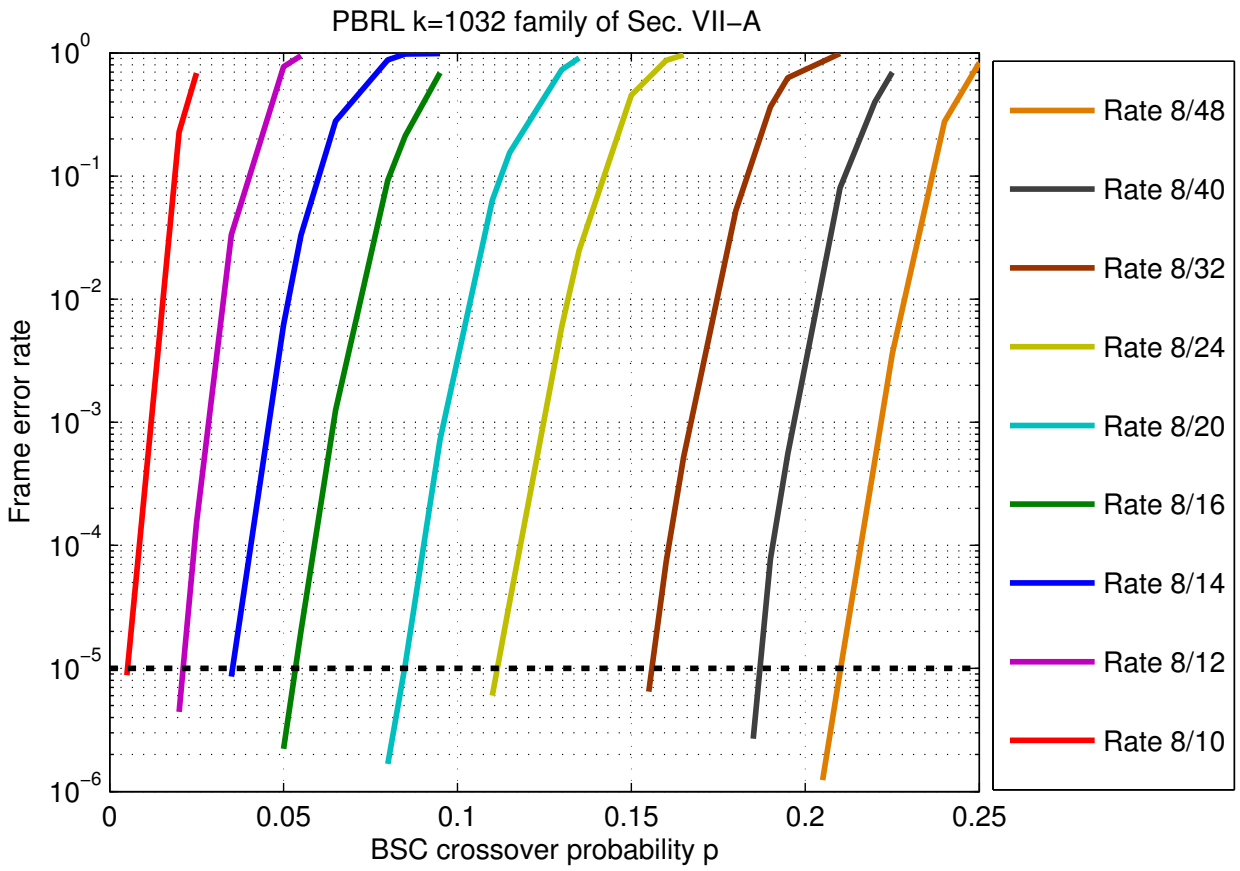


Figure 4.9: Frame error rates for the $k = 1032$ protographs defined by (4.24) and (4.25) over BSC.

Similar to the design of $k = 1032$ code family, RCA algorithm optimizes the threshold corresponding to the connections of the IRC check nodes to the variable nodes of the HRC part of (4.28). In the optimization process, each connection to the punctured node may have zero, one, or two edges in the first column of (4.28). Each connection to a non-punctured node may have zero or one edge in the other columns of (4.28). Table 4.4 shows p_{it}, p_{ppv}, and p_{cap} for the $k = 16384$ code family with an average gap of 0.0215 bits across various rates.

Two step ACE+CPEG lifting which guarantees a girth of eight is used to lift the lowest rate code. The pre-lifting step has a lifting number of 4 and the lifting number in the second stage is 512.

Figs. 4.9 and 4.10 show the frame-error rate performance of the PBRL codes designed for $k = 1032$ and $k = 16384$, respectively over BSC. For a fixed rate, the $k = 16384$ PBRL code

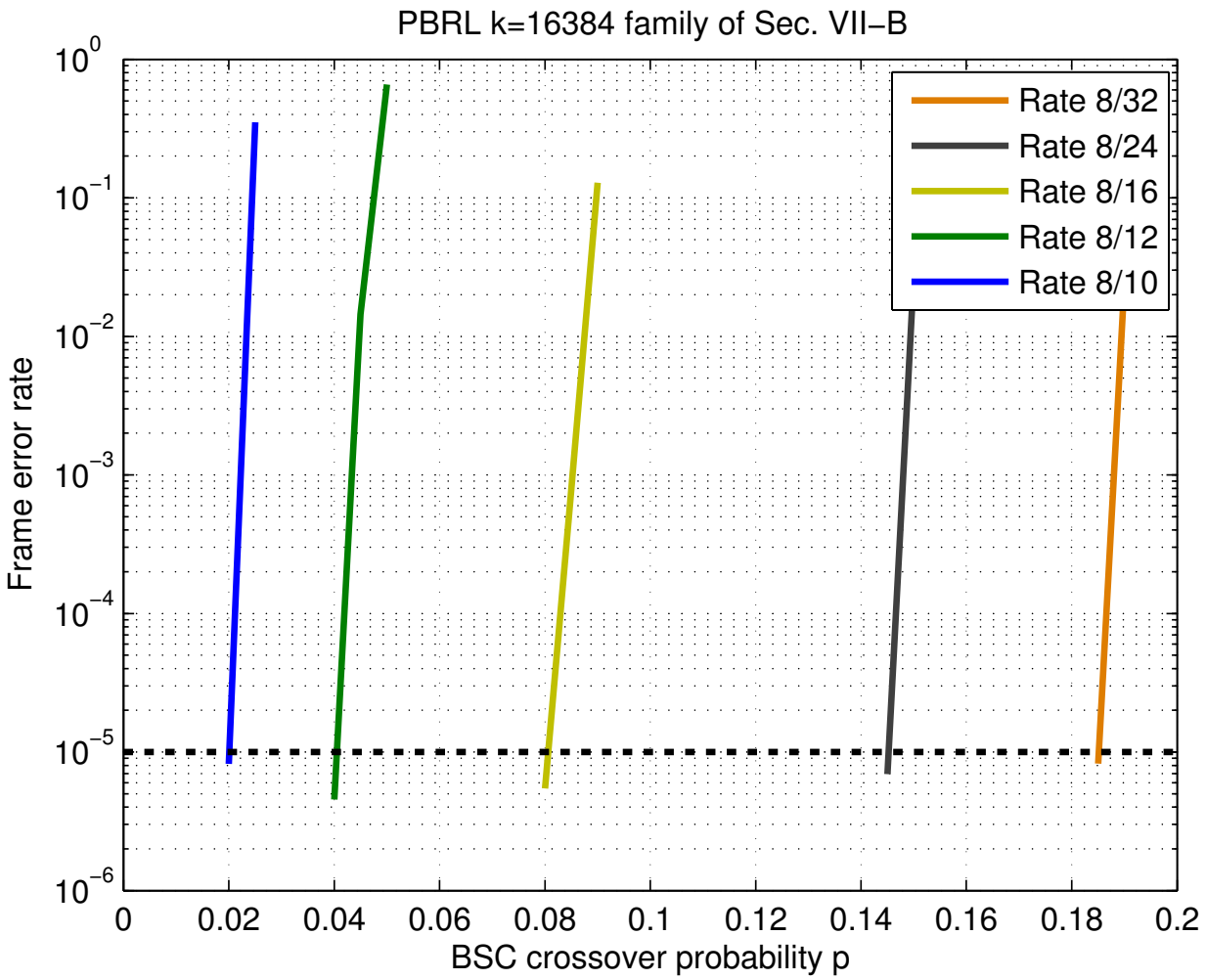


Figure 4.10: Frame error rates for the $k = 16384$ protographs defined by (4.27) and (4.28) over BSC.

family in Fig. 4.10 achieves a particular FER at a higher BSC crossover probability than the codes designed for $k = 1032$ in Fig 4.9. The BSC crossover probability obtained at $FER = 10^{-5}$ is used for the analysis in Fig 4.11.

Fig. 4.11 shows the gap to capacity in bits for $k = 1032$ and $k = 16384$ code families at $FER = 10^{-5}$. These simulations use floating-point iterative decoders with a flooding schedule for message-passing. Decoding terminates early if all parity checks are satisfied before reaching the maximum number (300) of iterations. The average mutual information gap to capacity for $k = 1032$ and $k = 16384$ codes are 0.1470 and 0.0730 bits. The gaps to PPV finite blocklength analysis are

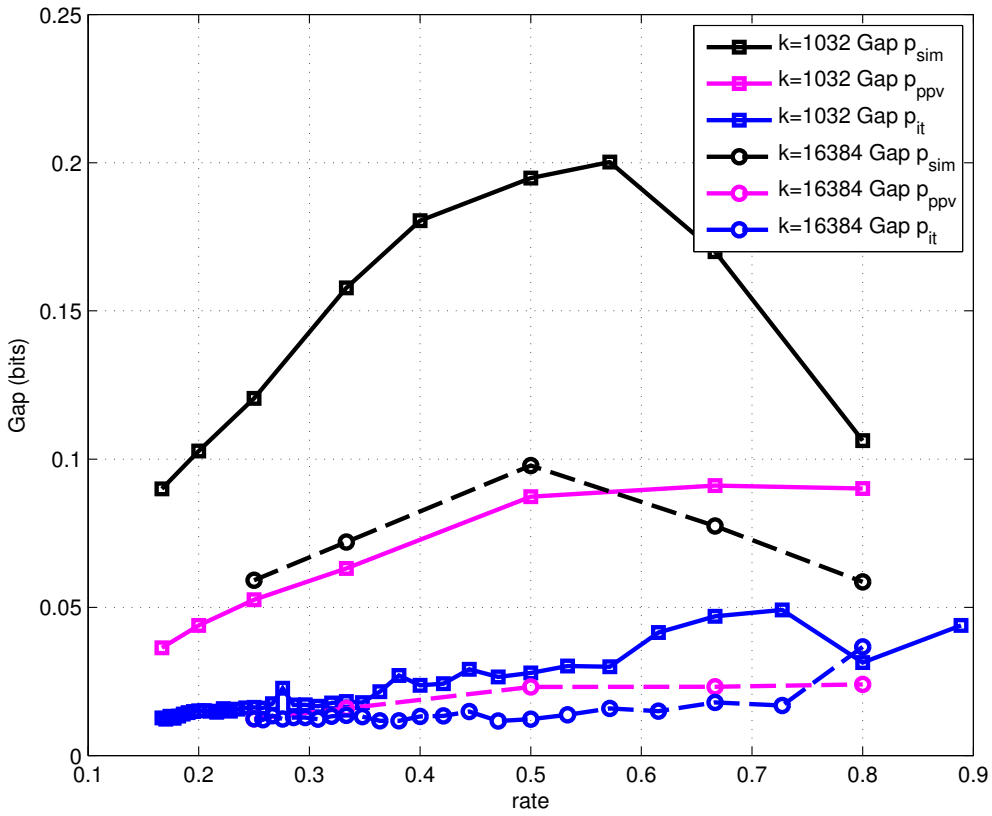


Figure 4.11: Mutual information gap from capacity and PPV finite-blocklength analysis in number of bits for the PBRL codes constructed for $k = 1032$ and $k = 16384$ at a frame error rate of 10^{-5}. The protograph for the $k = 1032$ PBRL code is based on (4.24,4.25,4.27, and 4.28).

even smaller at 0.0807 and 0.0466 bits respectively.

Fig. 4.12 illustrates the ACE number η values for each rate for the two families of short and long-blocklengths codes designed for BSC. As explained in [101], the ACE number is important ensuring that the cycles in the bipartite graph of the LDPC codes are well connected. As illustrated in the previous chapter, high ACE values help with the undesired graphical structures that cause error floors.

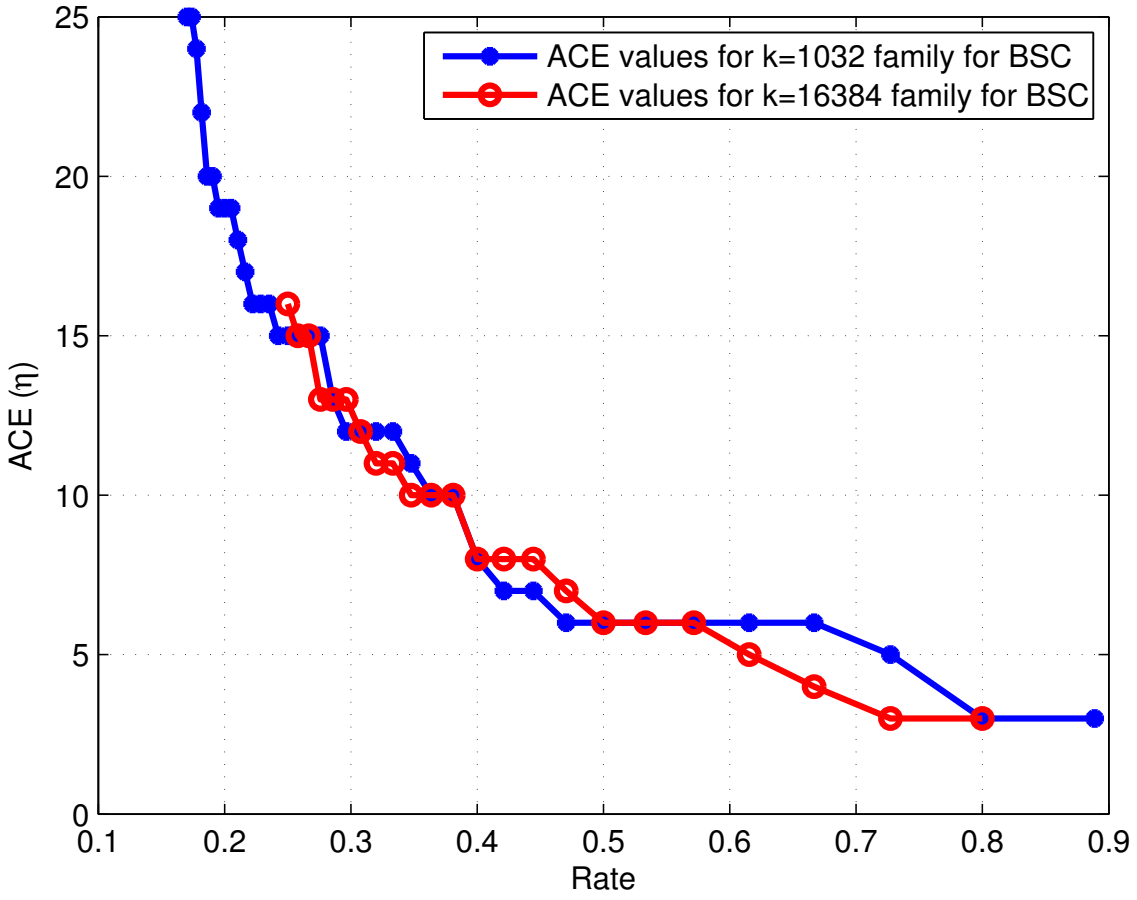


Figure 4.12: ACE value (η) versus rate for the short blocklength (k = 1032) with d_{ACE} = 5 and long blocklength (k = 16384) with d_{ACE} = 6 codes designed for BSC.

4.8 Mixed BEC-AWGN Channel

In this section we present an RCA algorithm to approximate the DE threshold for a mixed channel where some variable nodes of the LDPC code are absolutely correct or erased (with probability ϵ) while some other variable-nodes are transmitted over an AWGN channel. This sort of mixed channel is observed over networks where some bits are either decoded correctly with a very high probability or completely erased while some other bits only experience normally distributed noise. RCA for mixed channels can be developed efficiently by keeping track of three variables:

1. The mean (μ) of the approximated Gaussian distribution which relates to the SNR (s) by $\mu = 2s$,

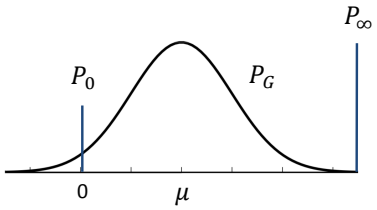


Figure 4.13: The probability density function for the general form of the LLR values (messages) passed between the variable nodes and check nodes.

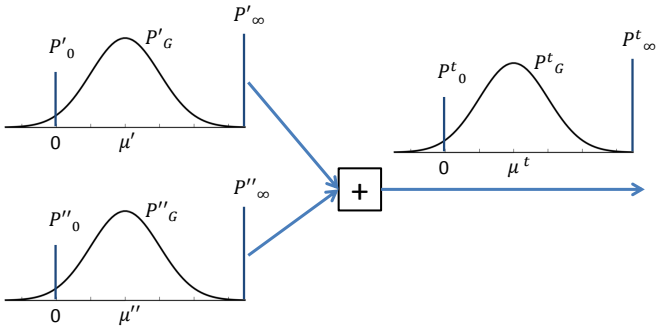


Figure 4.14: Check-node update with two input messages. The messages have three main parameters of P_∞, P_0, and μ. The last parameter of P_G can be calculated easily by $P_G = 1 - P_0 - P_\infty$.

2. The probability of decoding the message correctly (P_∞),

3. The probability that a bit is erased (P_0).

Fig. 4.13 shows the probability density function for the general form of the LLR values (messages) passed between the variable nodes and check nodes. The two probability mass functions at P_0 and P_∞ correspond to the probability that a variable node is either erased or decoded correctly, respectively. Furthermore, $P_G = 1 - P_0 - P_\infty$ and it corresponds to the total probability of the noisy part of the message. Fig. 4.14 shows an example of check-node updates with two messages received from two neighboring variable nodes. The output message also has three components corresponding to the erasure probability P_0^t, correct probability P_∞^t, and the expected value of the Gaussian approximation calculated using the capacity formula of $C(s)$ according to [4.19] for AWGN channels. Additionally the last parameter is P_G^t, which corresponds to the total probability associated with the noisy part of the LLR distribution.

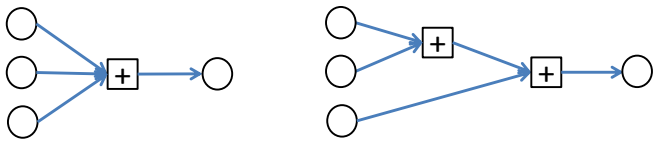


Figure 4.15: The check-node update in LDPC codes can be broken up into check-node operation with two inputs at a time. This process along with the updating algorithm shown in Fig. 4.14 make the implementation of RCA algorithm for mixed channels linear in complexity.

Considering Fig. 4.14, the only case where the output message from the check node is definitely correct is when both input messages correspond to the correct messages P'_∞ and P''_∞. Therefore, $P^t_\infty = P'_\infty \times P''_\infty$. As long as one of the input messages to the check node is erased, the out message is erased, hence, $P^t_0 = 1-(1-P'_0)(1-P''_0)$. The probability corresponding to LLR values following a Gaussian distribution is $P^t_G = 1 - P^t_0 - P^t_\infty$. Using the $C(s)$ and $R(s)$ functions as discussed in [4.19] and a dummy variable s_3, the check-to-variable node message follows

$$s_3 = R\left(R(s') + R(s'')\right), \tag{4.29}$$

$$s^t = C^{-1}\left(\frac{P'_G P''_\infty C(s') + P''_G P'_\infty C(s'') + P''_G P'_G C(s_3)}{P'_G P''_\infty + P''_G P'_\infty + P''_G P'_G}\right). \tag{4.30}$$

If the degree of a check node is more than three, by using the property of breaking up each check-node into multiple check-nodes with two inputs from variable nodes, as shown in Fig. 4.15, the check-node operation can be performed with linear complexity. The naive extension of the check node operation shown in Fig. 4.14 with more than two input messages by simultaneously considering the messages from all neighboring variable nodes has an exponential complexity.

The variable node update is similar to the check node update with only difference that a variable node is certain about its value as long as it receives at least one input from any neighboring check-node with P_∞. In addition, a variable node is erased with the probability that all the messages from the neighboring check nodes are erased. Similarly, the variable node operations can be performed with linear complexity.

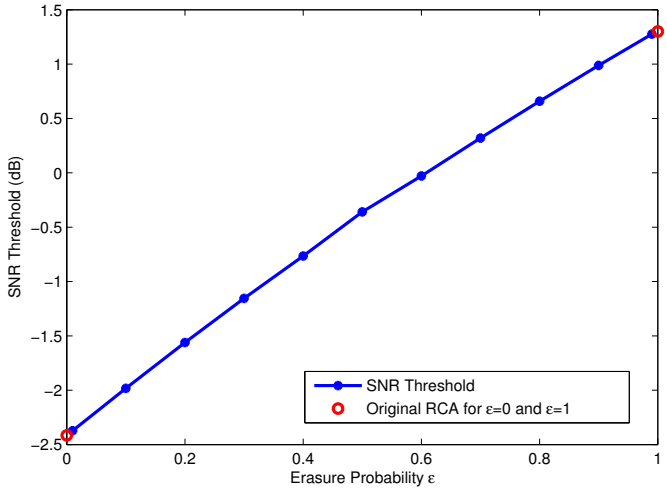


Figure 4.16: RCA threshold values corresponding to the highest σ or the smallest SNR value for the Gaussian noise channel in combination with different erasure channels at which the LDPC code of [4.31] is decoded correctly. The first and last variable nodes are assumed to be transmitted over a BEC with erasure probability ϵ while the other variable nodes are transmitted over an AWGN channel.

$$H_{\text{Mixed}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.31}$$

To illustrate the results for one matrix example in form of PBRL codes, consider the matrix of [4.31]. In this example, we assume the first and last variable nodes are transmitted over an erasure channel with erasure probability ϵ, while the other five variable nodes are transmitted over an AWGN channel with channel variance σ.

Fig. 4.16 shows the RCA threshold values corresponding to the smallest SNR value for the Gaussian noise channel in combination with different erasure channels at which the LDPC code can be decoded with zero probability of error. In Fig. 4.16 the extreme points for erasure probability of 1 (the first and last variable nodes are always erased) and erasure probability of 0 (the first and

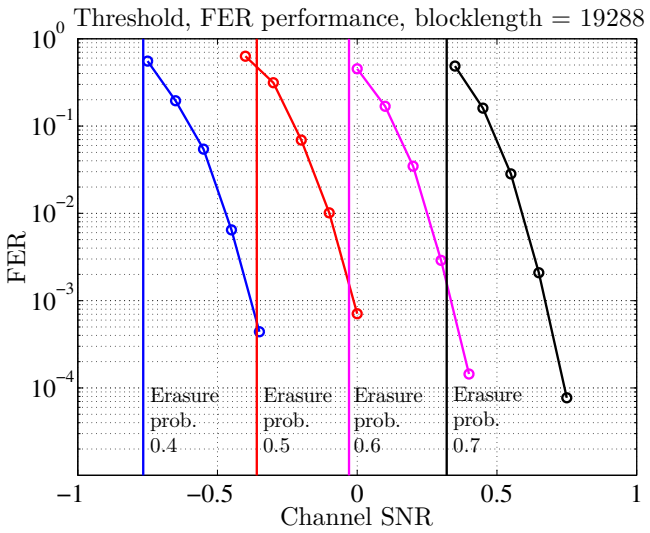


Figure 4.17: Frame error rates for the $n = 19288$ protographs and RCA threshold values corresponding to the smallest SNR value for the Gaussian noise channel in combination with different erasure channels for the LDPC code of [4.31]. The first and last variable nodes are assumed to be transmitted over a BEC with erasure probability ϵ while the other variable nodes are transmitted over an AWGN channel.

variable nodes are known) can be computed using the original RCA algorithm. These extreme points confirm the performance of RCA algorithm for the other cases where the erasure probability $0 < \epsilon < 1$.

Fig. 4.17 shows the simulation results for the LDPC code in [4.31] for different erase probability values. The vertical lines illustrate the RCA threshold values corresponding to the smallest SNR values for the Gaussian noise channel at which the LDPC code of [4.31] is decoded correctly. The first and last variable nodes are assumed to be transmitted over a BEC with erasure probability ϵ while the other variable nodes are transmitted over an AWGN channel.

4.9 Concluding Remarks

This chapter studies the construction and optimization of protograph-based Raptor-like (PBRL) LDPC codes over BEC and BSC. This work designs PBRL codes by designing a highest-rate

code (HRC) and sequentially adding degree-one variable nodes whose neighboring check node is connected to the variable nodes of the HRC while maximizing the density evolution threshold (erasure and crossover probability for BEC and BSC respectively). Puncturing a single variable node in the HRC improves the threshold performance of PBRL codes. Instead of the original density evolution, the RCA algorithm (which is exact for BEC) is used to obtain a fast and accurate thresholds of PBRL codes to result in reasonable code-design complexity. For BSC, we propose an efficient and precise RCA algorithm based on a Gaussian approximation as an alternative the density evolution algorithm.

In summary, the analysis in this chapter provides a complete design procedure for constructing rate-compatible LDPC code families that perform uniformly close to the capacity and finite block-length performance limits for both short (k = 1032) and long (k = 16384) blocklengths over BEC and BSC.

Table 4.1: Thresholds for the k=1032 PBRL code family over BEC

Rate	Threshold p_{it}	Threshold p^*_{MAP}	Capacity p_{cap}	Gap $1 - p_{it} - R_c$
8/9	0.069	0.111	0.111	0.045
8/10	0.167	0.200	0.200	0.033
8/11	0.223	0.264	0.273	0.064
8/12	0.287	0.327	0.333	0.046
8/13	0.343	0.380	0.385	0.064
8/14	0.401	0.427	0.429	0.028
8/15	0.437	0.465	0.467	0.053
8/16	0.469	0.499	0.500	0.031
8/17	0.503	0.528	0.529	0.053
8/18	0.524	0.554	0.556	0.066
8/19	0.556	0.578	0.579	0.052
8/20	0.577	0.598	0.600	0.019
8/21	0.592	0.617	0.619	0.066
8/22	0.617	0.635	0.636	0.050
8/23	0.634	0.651	0.652	0.051
8/24	0.648	0.665	0.667	0.016
8/25	0.663	0.678	0.680	0.051
8/26	0.676	0.691	0.692	0.051
8/27	0.688	0.702	0.704	0.051
8/28	0.699	0.713	0.714	0.051
8/29	0.709	0.723	0.724	0.052
8/30	0.719	0.732	0.733	0.052
8/31	0.728	0.741	0.742	0.052
8/32	0.736	0.749	0.750	0.013
8/33	0.744	0.757	0.758	0.054
8/34	0.751	0.764	0.765	0.054
8/35	0.758	0.771	0.771	0.055
8/36	0.764	0.777	0.778	0.057
8/37	0.770	0.783	0.784	0.058
8/38	0.776	0.789	0.790	0.059
8/39	0.782	0.794	0.795	0.060
8/40	0.787	0.799	0.800	0.013
8/41	0.792	0.804	0.805	0.063
8/42	0.797	0.809	0.810	0.063
8/43	0.802	0.813	0.814	0.060
8/44	0.807	0.817	0.818	0.058
8/45	0.811	0.821	0.822	0.059
8/46	0.815	0.825	0.826	0.060
8/47	0.819	0.829	0.830	0.060
8/48	0.823	0.832	0.833	0.011

$$H_{\mathrm{IRC}}^{(16384)} = \begin{bmatrix}
2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{4.18}$$

Table 4.2: Thresholds for the k=16384 PBRL code family over BEC

Rate	Threshold	Threshold	Capacity	Gap
	p_{it}	p^*_{MAP}	p_{cap}	$1 - p_{it} - R_c$
$8/10$	0.167	0.200	0.200	0.040
$8/11$	0.256	0.273	0.273	0.022
$8/12$	0.319	0.332	0.333	0.021
$8/13$	0.370	0.384	0.385	0.023
$8/14$	0.414	0.428	0.429	0.025
$8/15$	0.452	0.466	0.467	0.027
$8/16$	0.487	0.499	0.500	0.025
$8/17$	0.517	0.529	0.529	0.026
$8/18$	0.542	0.555	0.556	0.030
$8/19$	0.565	0.578	0.579	0.032
$8/20$	0.587	0.599	0.600	0.031
$8/21$	0.606	0.618	0.619	0.033
$8/22$	0.623	0.636	0.636	0.035
$8/23$	0.638	0.651	0.652	0.039
$8/24$	0.655	0.666	0.667	0.034
$8/25$	0.668	0.679	0.680	0.036
$8/26$	0.681	0.691	0.692	0.035
$8/27$	0.692	0.703	0.704	0.038
$8/28$	0.703	0.713	0.714	0.038
$8/29$	0.713	0.723	0.724	0.039
$8/30$	0.722	0.732	0.733	0.041
$8/31$	0.731	0.741	0.742	0.041
$8/32$	0.739	0.749	0.750	0.042

Table 4.3: Thresholds and finite-blocklength PPV analysis for k=1032 PBRL family over BSC

Rate	Threshold	Threshold	Capacity	Gap
	p_{it}	p_{ppv}	p_{cap}	$1 - H(p_{it}) - R_c$
8/9	0.008	0.0049	0.015	0.044
8/10	0.025	0.0146	0.031	0.031
8/11	0.036	0.0264	0.047	0.049
8/12	0.050	0.0400	0.061	0.047
8/13	0.064	0.0518	0.075	0.041
8/14	0.079	0.0635	0.088	0.029
8/15	0.090	0.0732	0.099	0.030
8/16	0.101	0.0830	0.110	0.028
8/17	0.111	0.0928	0.120	0.026
8/18	0.119	0.1025	0.129	0.029
8/19	0.129	0.1104	0.138	0.024
8/20	0.137	0.1182	0.146	0.024
8/21	0.143	0.1260	0.154	0.027
8/22	0.152	0.1338	0.161	0.021
8/23	0.160	0.1396	0.168	0.018
8/24	0.166	0.1475	0.174	0.018
8/25	0.172	0.1533	0.180	0.018
8/26	0.178	0.1592	0.186	0.017
8/27	0.183	0.1650	0.191	0.017
8/28	0.188	0.1689	0.196	0.017
8/29	0.190	0.1748	0.201	0.023
8/30	0.197	0.1787	0.206	0.017
8/31	0.202	0.1846	0.210	0.016
8/32	0.206	0.1885	0.215	0.016
8/33	0.210	0.1924	0.219	0.015
8/34	0.214	0.1963	0.222	0.015
8/35	0.218	0.2002	0.226	0.016
8/36	0.221	0.2041	0.230	0.015
8/37	0.225	0.2080	0.233	0.015
8/38	0.228	0.2119	0.237	0.015
8/39	0.231	0.2158	0.240	0.015
8/40	0.234	0.2178	0.243	0.015
8/41	0.237	0.2217	0.246	0.014
8/42	0.240	0.2256	0.249	0.014
8/43	0.243	0.2275	0.252	0.013
8/44	0.246	0.2314	0.254	0.012
8/45	0.249	0.2334	0.257	0.013
8/46	0.251	0.2354	0.260	0.012
8/47	0.254	0.2393	0.262	0.013
8/48	0.256	0.2412	0.264	0.013

$$H_{\mathrm{IRC}}^{(16384)} = \begin{bmatrix}
2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
2 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{4.28}$$

Table 4.4: Thresholds and finite-blocklength PPV analysis for the k=16384 PBRL code family over BSC

Rate	Threshold	Threshold	Capacity	Gap
	p_{it}	p_{ppv}	p_{cap}	$1 - H(p_{it}) - R_c$
$8/10$	0.024	0.0264	0.031	0.037
$8/11$	0.043	0.0420	0.047	0.017
$8/12$	0.057	0.0557	0.061	0.018
$8/13$	0.071	0.0693	0.075	0.015
$8/14$	0.083	0.0811	0.088	0.016
$8/15$	0.095	0.0928	0.099	0.014
$8/16$	0.106	0.1025	0.110	0.012
$8/17$	0.116	0.1123	0.120	0.012
$8/18$	0.124	0.1221	0.129	0.015
$8/19$	0.133	0.1299	0.138	0.013
$8/20$	0.141	0.1396	0.146	0.013
$8/21$	0.149	0.1455	0.154	0.011
$8/22$	0.156	0.1533	0.161	0.011
$8/23$	0.162	0.1611	0.168	0.013
$8/24$	0.168	0.1670	0.174	0.013
$8/25$	0.174	0.1729	0.180	0.013
$8/26$	0.180	0.1787	0.186	0.012
$8/27$	0.185	0.1846	0.191	0.013
$8/28$	0.190	0.1885	0.196	0.013
$8/29$	0.195	0.1943	0.201	0.012
$8/30$	0.199	0.1982	0.206	0.013
$8/31$	0.204	0.2041	0.210	0.012
$8/32$	0.208	0.2080	0.215	0.012

CHAPTER 5

Coding for Flash Memory Systems

Multiple reads of the same Flash memory cell with distinct word-line voltages provide enhanced precision for LDPC decoding. In this chapter, the word-line voltages are optimized by maximizing the mutual information (MI) of the quantized channel. The enhanced precision from a few additional reads allows frame error rate (FER) performance to approach that of full-precision soft information and enables an LDPC code to significantly outperform a BCH code.

For a well-designed LDPC code, the quantization that maximizes the mutual information also minimizes the FER in our simulations. The chapter also identifies a trade-off in LDPC code design when decoding is performed with multiple precision levels; the best code at one level of precision will typically not be the best code at a different level of precision.

5.1 Introduction

Flash memory can store large quantities of data in a small device that has low power consumption and no moving parts. The original NAND Flash uses only two levels. This is called single-level-cell (SLC) Flash because there is only one actively written charge level. Devices currently available using four levels are called multi-level-cell (MLC) Flash.

Error control coding for Flash memory is becoming more important as the storage density increases. The increasing number of levels increases sensitivity to variations in signal-to-noise ratio (SNR) from cell to cell and over time due to wear-out. This makes stronger error-correction codes necessary. Also, the wear-out effect is time varying, introducing a need for adaptive coding or modulation to maximize the potential of the system.

Low-density parity-check (LDPC) codes are well-known for their capacity-approaching ability for AWGN channels [116] and are the subject of recent interest for application to the Flash memory read channel. For example, in [117] LDPC codes without access to enhanced precision are shown to provide a performance improvement over BCH codes, but that improvement becomes small at high code rates. Also in [117], an alternative error correction scheme is introduced that takes into account the dominant cell-level errors found in eight-level cells. This scheme provides improvement for eight-level cells without using enhanced precision.

Important work related to codes that consider the dominant cell-level error is that of Gabrys et al. on graded bit error correcting codes [118]. In contrast to codes designed for dominant errors, our work focuses on the use of enhanced precision to improve performance. While we explore the improvement in terms of standard LDPC codes, enhanced precision should also improve the performance of alternative error correction schemes that focus on the dominant cell-level errors as long as the decoders can utilize soft information.

Another approach for using LDPC codes in Flash memories [119] is to design the codes for use with rank modulation. Rank modulation [120–122] stores information in the cell using the *relative* value (or ordering) of cell charge levels rather than the absolute value. LDPC codes for rank modulation require the cell charge-level ordering at the decoder.

As observed in [120], rank modulation eliminates the need for discrete cell levels, overcomes overshoot errors when programming cells, and mitigates the problem of asymmetric errors. This is an exciting approach for future Flash architectures. However, current Flash systems use the same word-line voltage to read all cells on the page and thus would require a large number of page reads to learn the charge-level ordering. Our work focuses on the traditional approach of coding with fixed target charge levels and assumes that when reading each page, the same word-line voltage is used for all cells.

We note that an alternative to using multiple reads to enhance precision is to perform a single read with a dynamic threshold as introduced by [123] to adapt to time-varying channel degradations such as the mean shift that occurs due to retention loss. We note that dynamic thresholds are complementary to the use of enhanced precision, and a combined approach could be especially

effective.

As also discussed in [124], this chapter uses mutual information maximization as the objective function that drives the optimization of the word-line voltages (thresholds) used for the multiple reads that provide enhanced precision. Mutual information maximization is also explored in [125] for the design of memory-efficient decoding of LDPC codes and in [126] for quantization of binary-input discrete memoryless channels and the design of the message-passing decoders of LDPC codes used on such channels.

Another aspect of current research follows from the fact that Flash memory systems must erase an entire block of data at once. Each block consists of numerous pages and each page contains thousands of bits. Even to change a small amount of data on a single page, the entire block must be erased. Moreover, the process of erasing and re-writing a block of data degrades performance. Each time electrons are written and then erased from the floating gate, the integrity of the floating gate degrades in a process known as "cell wear-out".

In [127], coding is used to minimize the frequency with which a block must be erased and the number of auxiliary blocks required for moving pages of data in a Flash memory system. Efficient wear-leveling and data movement in Flash is an important problem, but our work addresses the complementary problem of improving the ability to reliably read a page by using enhanced precision.

LDPC codes have typically been decoded with soft information (a relatively high-precision representation of a real or complex number describing a received symbol value) while Flash memory systems have typically provided only hard reliability information (a single bit representing the output of a sense-amp comparator) to their decoders. This work demonstrates that enhanced precision through multiple reads is crucial to successfully reaping the benefits of LDPC coding in Flash memory. We explore how to select the word-line voltages used for additional reads, how many such reads are necessary to provide most of the LDPC performance benefits, and how varying levels of precision can impact code design.

Section 5.2 focuses on the NAND architecture for Flash memory, shows the configuration of

a NAND Flash memory cell and introduces the NAND Flash memory read channel model. Section 5.3 shows how to obtain word-line voltages by maximizing the mutual information (MI) of the equivalent read channel using a simple Gaussian model of SLC (two-level) Flash as an example. The soft information is obtained by reading from the same sense-amp comparator multiple times with different word-line voltages. Section 5.4 shows that a few additional reads provide most of the benefit of enhanced precision through both a mutual information analysis and an LDPC simulation example. Section 5.5 describes the LDPC codes used in this chapter to illustrate that the relative performance of LDPC codes can depend on the level of quantization. This section also demonstrates a code design trade-off as follows: the best code in terms of both density evolution threshold and empirical performance at one precision level is not the best according to either density evolution threshold or empirical performance at another precision level. This is a practically important issue because the same code may well be decoded with varying levels of precision. In a practical system it is likely that additional page reads to enhance precision will be used only if the page could not be decoded without them.

Section 5.6 extends the discussion to MLC (four-level) Flash. The Gaussian Approximation and Reciprocal Channel Approximation ideas are used as alternatives to the DE analysis of LDPC codes tracking a single parameter μ, representing the mean, as a surrogate for the entire LLR distribution in each iteration. Section 5.7 discusses the degree-distribution optimization algorithm for a rate-r code and finally, Section 5.8 concludes this chapter.

5.2 The Read Channel of NAND Flash Memory

This section focuses on the NAND architecture for Flash memory. Fig. 5.1 shows the configuration of a NAND Flash memory cell. Each memory cell in the NAND architecture features a transistor with a control gate and a floating gate. To store information, a charge level is written to the cell by adding a specified amount of charge to the floating gate through Fowler-Nordheim tunneling by applying a relatively large voltage to the control gate [128].

To read a memory cell, the charge level written to the floating gate is detected by applying a

specified word-line voltage to the control gate and measuring the transistor drain current. When reading a page, a constant word-line voltage is applied to all cells in the page. A sense amp comparator compares the drain current to a threshold. If the drain current is above this threshold, then the word-line voltage was sufficient to turn on the transistor. This indicates that the charge written to the floating gate is below a certain value. The sense amp comparator provides only one bit of information about the charge level present in the floating gate.

A bit error occurring at this threshold-comparison stage is a *raw bit error* and the phrase *channel bit error probability* refers to the probability of a raw bit error given a specified amount of distortion in the process of writing to the cell, retaining the charge level over a period of time, and reading the cell. We refer to this overall process as the *read channel*.

The word-line voltage or reference voltage required to turn on a particular transistor (called the threshold voltage) can vary from cell to cell for a variety of reasons. For example, the floating gate can be overcharged during the write operation, the floating gate can lose charge due to leakage in the retention period, or the floating gate can receive extra charge when nearby cells are written [129]. The variation of threshold voltage from its intended value is the *read channel noise*.

We initially assume an i.i.d. Gaussian threshold voltage for each level of an SLC (i.e., two-level) Flash memory cell. This is equivalent to binary phase-shift keying (BPSK) with additive white Gaussian noise (AWGN), except that the threshold voltage cannot be directly observed. Rather, at most one bit of information about the threshold voltage may be obtained by each cell read.

More precise models such as the model in [129], in which the lowest and highest threshold voltage distributions have a higher variance, and the model in [130], in which the lowest threshold voltage (the one associated with zero charge level) is Gaussian and the other threshold voltages have Gaussian tails but a uniform central region, are sometimes used. The model in [131] is similar to [130], but is derived by explicitly accounting for real dominating noise sources, such as inter-cell interference, program injection statistics, random telegraph noise and retention noise.

In the next section, we present a general quantization approach for selecting word-line voltages

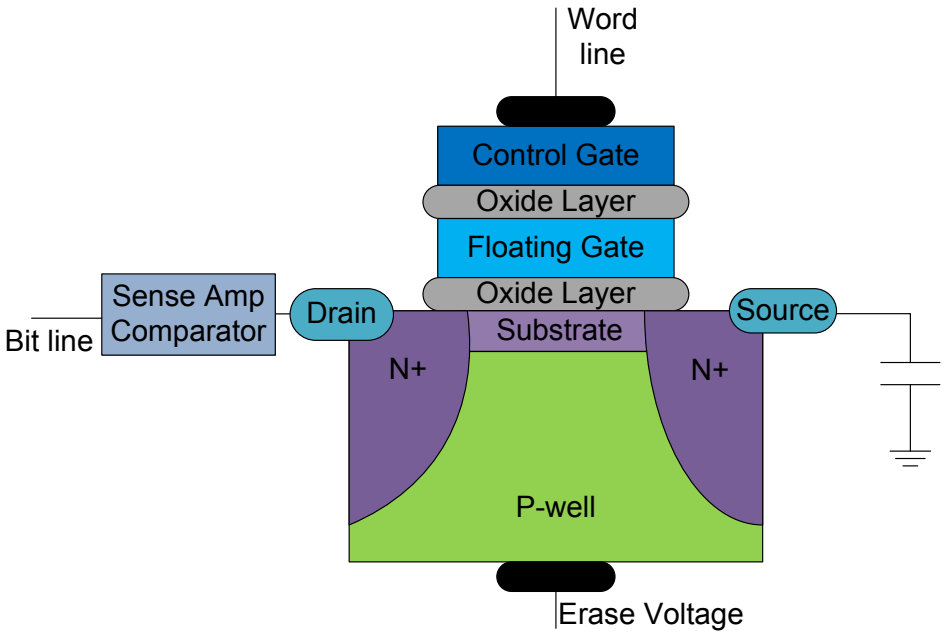


Figure 5.1: A NAND Flash memory cell.

for reading Flash memory cells and apply it to the specific example of SLC (two-level) Flash using a simple identically distributed Gaussian channel model.

5.3 Soft Information Via Multiple Cell Reads

Because the sense-amp comparator provides one bit of information about the threshold voltage (or equivalently the amount of charge present in the floating gate), decoders for error control codes in Flash have historically relied on hard decisions.

5.3.1 Obtaining Soft Information

However, soft information can be obtained in two ways: either by reading from the same sense-amp comparator multiple times with different word-line voltages (as is already done to read multi-level Flash cells) or by equipping a Flash cell with multiple sense-amp comparators on the bit line, which is essentially equivalent to replacing the sense amp comparator (a one-bit A/D converter)

with a higher-precision A/D converter.

These two approaches are not completely interchangeable in how they provide information about the threshold voltage. If the word-line voltage and floating gate charge level place the transistor in the linear gain region of the drain current vs. word-line-voltage curve (the classic I-V transistor curve), then valuable soft information is provided by multiple sense amp comparators. However, multiple comparators may not give much additional information if the I-V curve is too nonlinear. If the drain current has saturated too low or too high, the outputs from more sense-amp comparators are not useful in establishing precisely how much charge is in the floating gate.

In contrast, each additional read of a single sense amp comparator can provide additional useful information about the threshold voltage if the word-line voltages are well-chosen. Our work focuses on obtaining soft information from multiple reads using the same sense-amp comparator with different word-line voltages. This approach was studied in [132], and the poor performance of uniformly spaced word-line voltages was established.

The fundamental approach of this work is to choose the word-line voltages for each quantization by maximizing the MI between the input and output of the equivalent discrete-alphabet read channel. This approach has been taken in other work (not in the context of Flash memory) such as [125, 126]. Theoretically, this choice of word-line voltages maximizes the amount of information provided by the quantization. This section explores the simplest possible case, SLC (two-level) Flash using an identically distributed Gaussian model, which is equivalent to BPSK transmission with Gaussian noise.

5.3.2 Quantizing Flash to Maximize Mutual Information

This subsection describes how to select word-line voltages to achieve maximum mutual information (MMI) for two reads and three reads for the identically distributed Gaussian model.

For SLC Flash memory, each cell can store 1 bit of information. Fig. 5.2 shows a simplistic model of the threshold voltage distribution as a mixture of two identically distributed Gaussian random variables. When a “0” or “1” is written to the cell, the threshold voltage is modeled as a

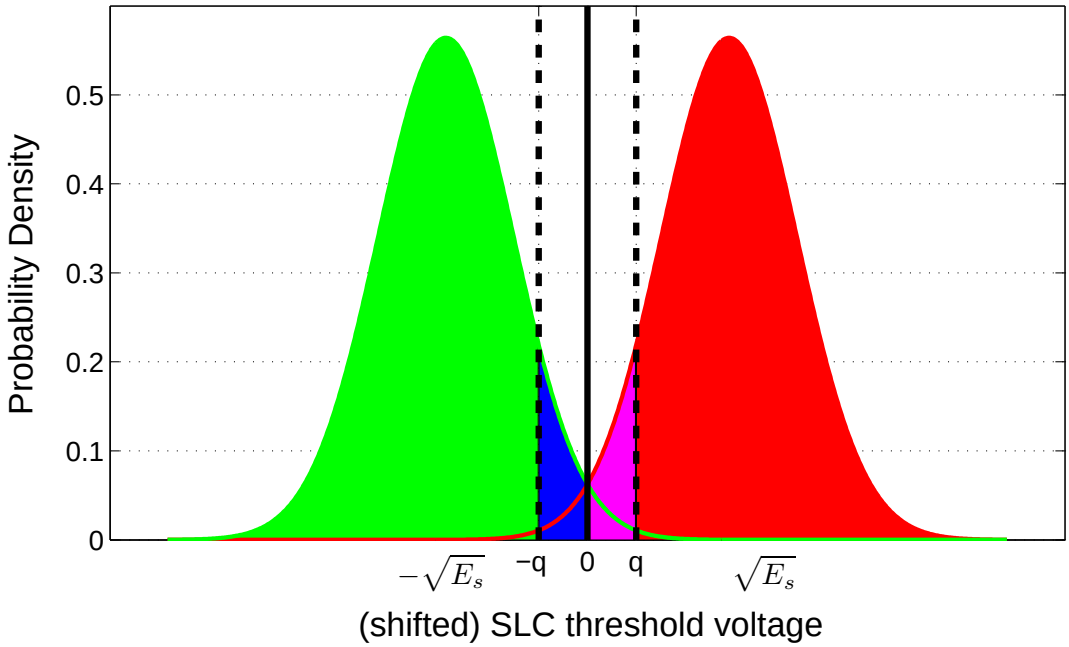


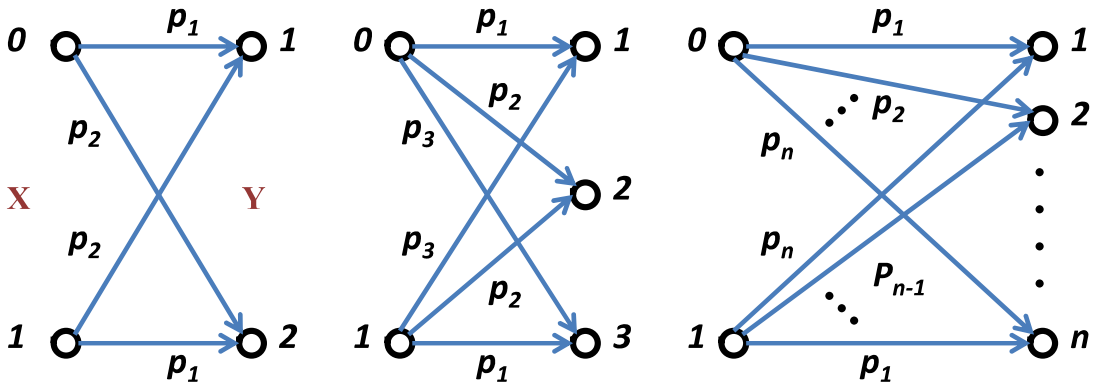
Figure 5.2: Identically distributed Gaussian model for SLC threshold voltages. Also shown are word-line voltages for two reads (the dashed lines) and three reads (all three lines). The quantization regions are indicated by shading with the middle region for two reads being the union of the blue and purple regions.

Gaussian random variable with variance $N_0/2$ and mean $-\sqrt{E_s}$ (for "1") or mean $+\sqrt{E_s}$ (for "0"), respectively.

5.3.3 Two reads per cell

For SLC with two reads, Fig. 5.2 shows symmetric word-line voltages q and $-q$. The threshold voltage is quantized into three regions as shown in Fig. 5.2: the green region, the red region, and the union of the blue and purple regions (which essentially corresponds to an erasure e). This quantization produces the effective discrete memoryless channel (DMC) model shown in Fig. 5.3(a) with input $X \in \{0, 1\}$ and output $Y \in \{0, e, 1\}$.

Assuming X is equally likely to be 0 or 1, the MI $I(X;Y)$ between the input X and output Y



(a) One read (b) Two reads (c) n-1 reads

Figure 5.3: SLC equivalent discrete read channels

of the resulting DMC can be calculated [133] as

$$I(X;Y) = H(Y) - H(Y|X) \tag{5.1}$$

$$= H\left(\frac{p_{13}}{2}, p_2, \frac{p_{13}}{2}\right) - H\left(p_1, p_2, p_3\right), \tag{5.2}$$

where H is the entropy function [133], $p_{ij} = p_i + p_j$, and the crossover probabilities shown in Fig. 5.3(a) are $p_1 = 1 - Q^-$, $p_2 = Q^- - Q^+$, and $p_3 = Q^+$ with

$$Q^- = Q\left(\frac{\sqrt{E_s} - q}{\sqrt{N_0/2}}\right) \text{ and } Q^+ = Q\left(\frac{\sqrt{E_s} + q}{\sqrt{N_0/2}}\right), \tag{5.3}$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-u^2/2} du$.

For fixed SNR $\frac{E_s}{N_0/2}$, the MI in (5.1-5.2) for the identically distributed Gaussian model is a quasi-concave function of q with a zero derivative only at the q that delivers the maximum MI and at $q = \infty$. The MI can be maximized analytically by setting $dI/dq = 0$. Let $f(x)$ be the probability density function of a standard normal distribution. The derivative is computed as

$$\frac{dI}{dq} = f\left(T_q^+\right) \log_2\left(\frac{p_{13}}{2p_3}\right) + f\left(T_q^-\right) \log_2\left(\frac{p_{13}}{2p_1}\right), \tag{5.4}$$

where $T_q^+ = \sqrt{E_s} + q$ and $T_q^- = \sqrt{E_s} - q$.

Note that dI/dq is continuous on $\mathbb{R}^+$. At $q = 0$ we have $p_{13} = p_1 + p_3 = 1$. Applying this to (5.4),

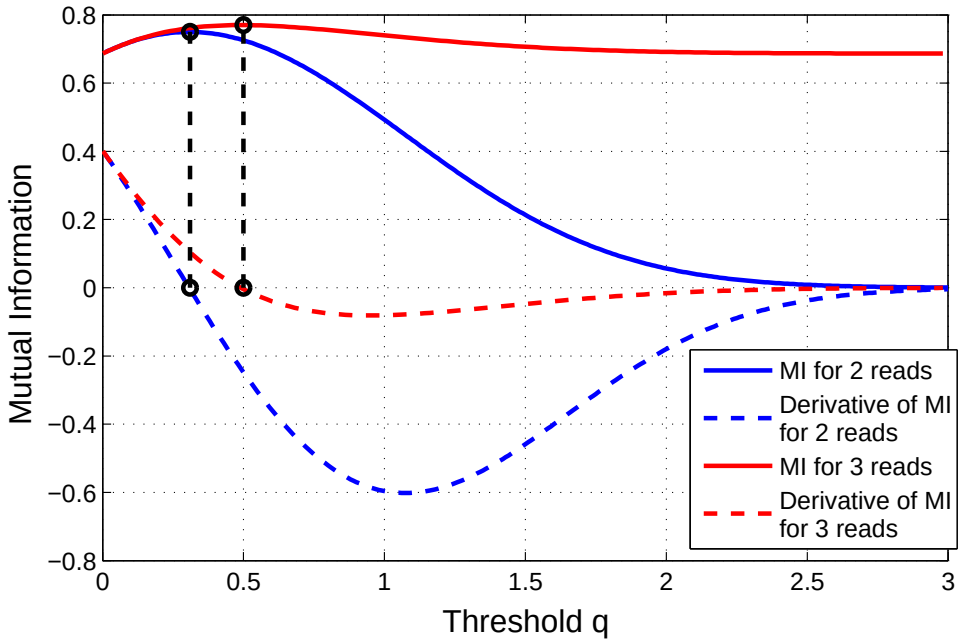


Figure 5.4: MI and its derivative vs. q (for $E_s = 1$) for SNR= $\frac{E_s}{N_0/2} = 4$ dB for SLC (two-level) Flash with two reads and with three reads under the identically distributed Gaussian model.

we have

$$\frac{dI}{dq} = -f\left(\sqrt{E_s}\right)\log_2\left(4p_1(1-p_1)\right) \geq 0\,, \tag{5.5}$$

at $q = 0$ by the inequality of arithmetic and geometric means. Equality holds only when $p_1 = 1/2$, which also causes $I(X;Y) = 0$. It can also be shown that dI/dq becomes negative for sufficiently large q and then increases monotonically, approaching zero as q approaches infinity.

These properties, illustrated in the example of Fig. 5.4, ensure that there is a single zero derivative for finite q corresponding to the desired maximum MI. Because (5.4) involves the Q function, solving for the q that sets $\frac{dI}{dq} = 0$ requires a numerical approach such as the bisection search [134].

The blue curves in Fig. 5.4 show how the MI for two reads and its derivative vary as a function of q for an SNR of 4 dB. Note that when $q = 0$ there is no erasure region, which is equivalent to a single read. As q increases so does the erasure region. MI is concave in q between $q = 0$ and the

point of inflection. Note that when $q = \infty$ the channel always produces the output e and the MI is zero.

5.3.4 Three reads per cell

Now consider SLC with three reads for each cell. The word-line voltages should again be symmetric (shown as q, 0, and $-q$ in Fig. 5.2). The threshold voltage is quantized according to the four differently shaded regions shown in Fig. 5.2. This quantization produces the DMC model as shown in Fig. 5.3 with input $X \in \{0, 1\}$ and output $Y \in \{00, 01, 10, 11\}$.

Assuming X is equally likely to be 0 or 1, the MI between the input and output of this DMC can be calculated as

$$\begin{aligned} I(X;Y) =& H(Y) - H(Y|X) \\ =& H\left(\frac{p_{14}}{2}, \frac{p_{23}}{2}, \frac{p_{23}}{2}, \frac{p_{14}}{2}\right) - H(p_1, p_2, p_3, p_4), \end{aligned} \tag{5.6}$$

where $p_{ij} = p_i + p_j$ with $p_1 = 1 - Q^-$, $p_2 = Q^- - Q^0$, $p_3 = Q^0 - Q^+$, and $p_4 = Q^+$. Q^- and Q^+ are as in (5.3) and

$$Q^0 = Q\left(\frac{\sqrt{E_s}}{\sqrt{N_0/2}}\right). \tag{5.7}$$

The derivative of MI with respect to the threshold q is

$$\frac{dI}{dq} = \sum_{j=1}^{4} p_j' \log_2(p_j) - p_{14}' \log_2(p_{14}) - p_{23}' \log_2(p_{23}), \tag{5.8}$$

where $-p_1' = p_2' = f(T_q^-)$ and $p_3' = -p_4' = f(T_q^+)$. When $q = 0$, (5.5) still applies.

The red curves in Fig. 5.4 show how the MI for three reads and its derivative vary as a function of q for an SNR of 4 dB. Both at $q = 0$ and $q = \infty$ the channel is equivalent to the binary symmetric channel (BSC) produced by a single read with the threshold at zero. Thus the MI for both of these extreme choices is identical. Fig. 5.4 shows a single zero derivative corresponding to the desired maximum MI occurring between these two extremes. Again, solving for the q that sets $\frac{dI}{dq} = 0$ requires a numerical approach such as the bisection search [134].

For the relatively simple identically distributed Gaussian model, dI/dq for the two-read and three-read cases can be identified analytically as described above. However, even in realistic models in which the distributions are described numerically, the optimum q can usually be found by a bisection search. Also, when the distributions are not identically distributed, the constant-ratio approach of [132], which is introduced in Section 5.6 for the four-level MLC case, can be used to produce a single-parameter optimization that again can be solved with quasi-convex optimization methods [134].

5.3.5 Five Reads per cell

In this work we assume i.i.d. Gaussian threshold voltages for each charge level in SLC or MLC Flash memory cells. These models are equivalent to 2-level or 4-level Pulse-Amplitude Modulation with additive white Gaussian ($\mathcal{N}(0, \sigma^2)$) noise (AWGN). Fig. 5.5 shows the model of the threshold voltage distribution as a mixture of the two identically distributed Gaussian random variables that comprise the SLC model. Since the levels are at $+1$ and -1, the average energy (E_{avg}) of this model is 1.

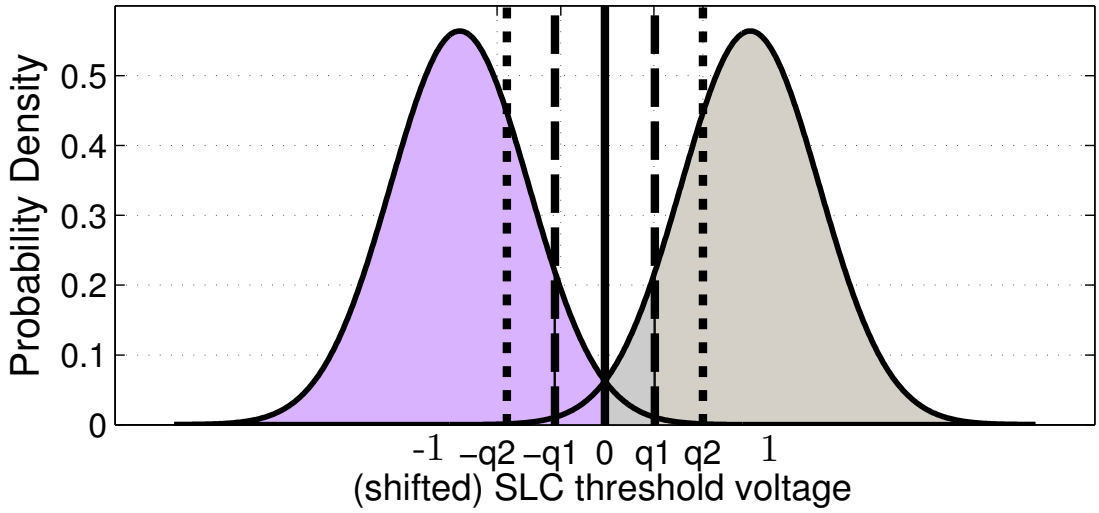


Figure 5.5: SLC threshold voltage model

Similar to Fig. 5.5, the shifted 4-level MLC threshold voltage distribution is a mixture of four

identically distributed Gaussian random variables. The voltage levels are at $\{+\frac{3}{\sqrt{5}}, +\frac{1}{\sqrt{5}}, -\frac{1}{\sqrt{5}}, -\frac{3}{\sqrt{5}}\}$ and correspond to the Gray labeled assignment $\{00, 01, 11, 10\}$ respectively. In this model E_{avg} = 1, similar to the SLC model.

5.3.6 Progressive Quantization on Read Channels

Each time a cell is read, the result is a single bit indicating whether the threshold voltage is above the word-line voltage at the time of the read. As described in [135,136], reading the same SLC cell $n-1$ times (for $n \geq 2$) with different word-line voltages effectively produces an equivalent channel with two inputs and n outputs as shown in Fig. 5.3.

In [135,136], the word-line voltages for each quantization are chosen by maximizing the mutual information (MI) between the input and output of the equivalent discrete read channels. Let's consider the MI for the three equivalent channels corresponding to a single read, three reads, and five reads, respectively. This set of channels might be seen by three LDPC decoding attempts with progressively more reads.

Define $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-u^2/2} du$, $Q_\sigma^-(x) = Q(\frac{x-1}{\sigma})$, and let $p_{ij} = p_i + p_j$.

For the 1-read SLC in Fig. 5.3a, the MI can be expressed as

$$I(X;Y) = 1 - H(p_1, p_2), \tag{5.9}$$

where H is the entropy function. $p_1 = Q_\sigma^-(q)$ and $p_2 = 1 - p_1$. The quantization threshold (q) that maximizes the MI is $q = 0$.

For the 3-read SLC model in Fig. 5.3c with $n = 4$,

$$I(X;Y) = H(p_{14}, p_{23}) + 1 - H(p_1, p_2, p_3, p_4), \tag{5.10}$$

where $p_1 = Q_\sigma^-(q_1)$, $p_2 = Q_\sigma^-(0) - p_1$, $p_3 = Q_\sigma^-(-q_1) - p_{12}$, and $p_4 = 1 - Q_\sigma^-(-q_1)$ with q_1 the word-line voltage shown in Fig 5.5. As described in [136], for 3-read SLC, the optimal q_1 satisfies $\frac{dI}{dq_1} = 0$. Since the MI is quasi-concave in q_1, the optimal q_1 can be found by a bisection search algorithm.

For the 5-read SLC (SLC_5) model in Fig. 5.3c with $n = 6$,

$$I(X;Y) = H(p_{16}, p_{25}, p_{34}) + 1 - H(p_1, p_2, \ldots, p_6), \tag{5.11}$$

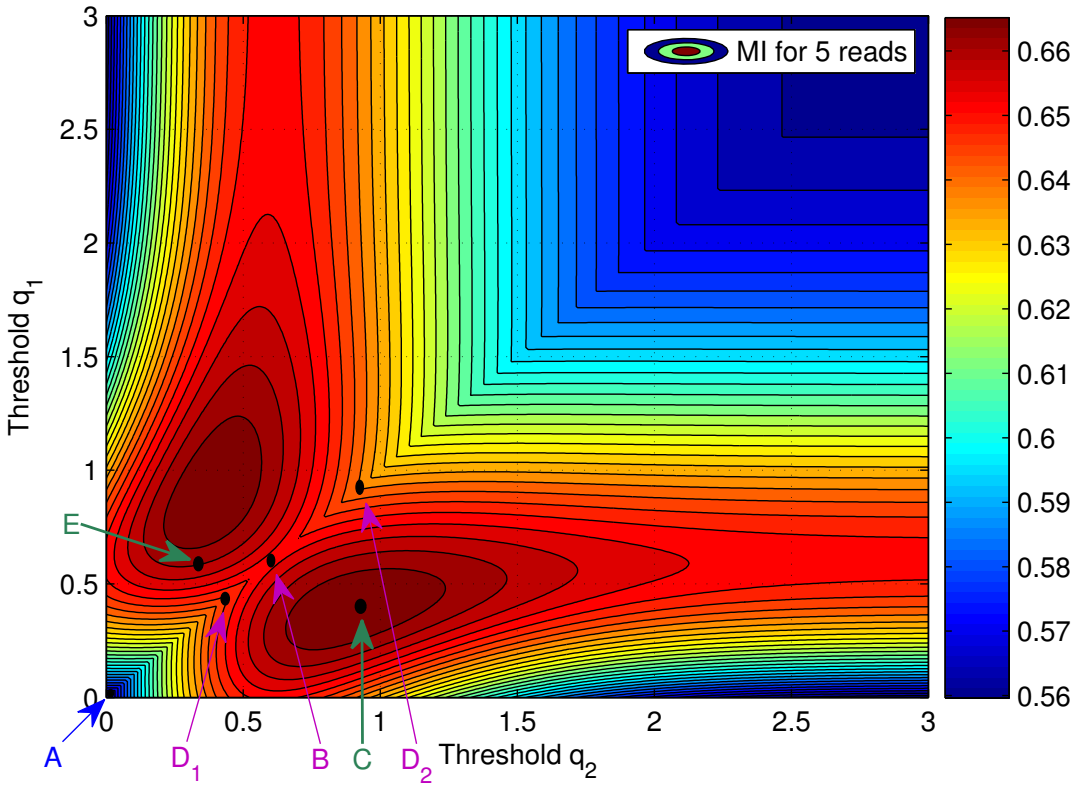


Figure 5.6: Mutual information vs. q_1 and q_2 for SLC_5

where $p_1 = Q_\sigma^-(q_2)$, $p_2 = Q_\sigma^-(q_1) - p_1$, $p_3 = Q_\sigma^-(0) - p_{12}$, $p_4 = Q_\sigma^-(-q_1) - Q_\sigma^-(0)$, $p_5 = Q_\sigma^-(-q_2) - Q_\sigma^-(-q_1)$, and $p_6 = 1 - Q_\sigma^-(-q_2)$ for q_1 and q_2 shown in Fig 5.5.

Fig. 5.6 shows the contour plot of mutual information (MI) vs. (q_1,q_2) for the 5-read case. The maximum MI points can be obtained by solving the two partial differential equations $\frac{dI}{dq_1} = 0$ and $\frac{dI}{dq_2} = 0$. Assuming $q_2 \geq q_1$, the MI is quasi-concave in q_1 for a fixed value of q_2 and vice-versa (in each dimension) and can be maximized by bisection search.

Let's use Fig. 5.6 to explore the effect of multiple decoding attempts with progressive reads on word-line voltage optimization. Note that the MI along the $q_1 = q_2$ line shows the three-read MI performance as a function of q_1. The optimal single-read word-line voltage of zero (Point A) is also optimal as the first read of three reads or five reads. However, The optimal position of q_1 for three reads is $q_1 = 0.61$ (point B), while the optimal position of q_1 for five reads is $q_1 = 0.4$ (with $q_2 = 0.9$, point C). We note that the MI obtained with three reads is 0.6524 when q_1 is optimized

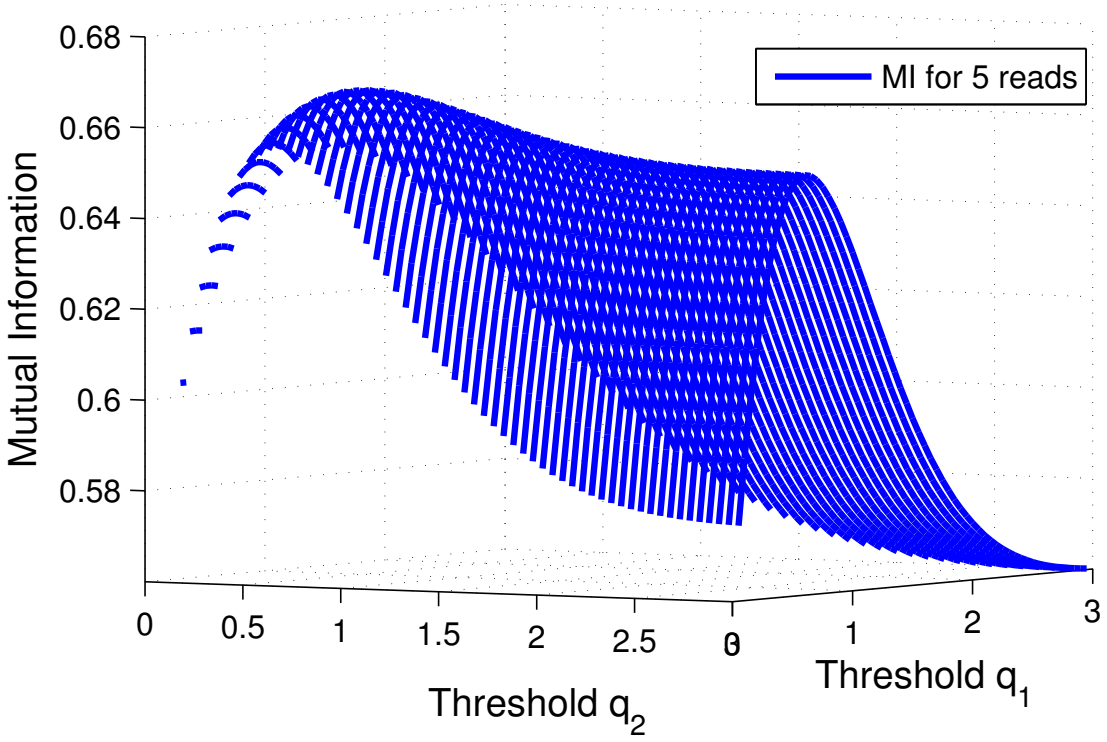


Figure 5.7: Mutual information vs. q_1 and q_2 for SLC_5

for three reads, 0.6439 when q_1 is optimized for five reads with $q_1 = 0.4$ (point D_1), and 0.6414 when q_1 is optimized for five reads with $q_1 = 0.9$ (point D_2). Also, the MI obtained with five reads is 0.6634 when q_1 is optimized for three reads (point E, with $q_1 = 0.61$ and $q_2 = 0.28$) and 0.6687 when q_1 is optimized for five reads (point C), assuming that q_2 is chosen optimally in each case. The MI differences are small in either case, but large enough to have noticeable effects on frame error rate according to [136]. A greedy approach (optimizing q_1 for three-read performance) leads to the smallest degradation and would lower decoding time by increasing the probability of decoding successfully after three reads, but optimizing q_1 for the five-read scenario will produce the lowest probability of losing a page.

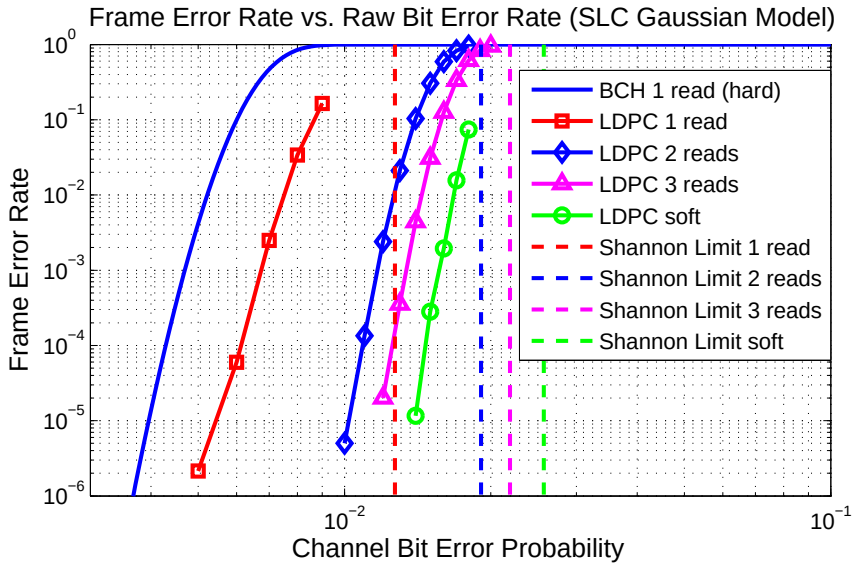


Figure 5.8: Simulation results of FER vs. channel bit error probability using the Gaussian channel model for SLC (two-level) Flash comparing LDPC Code 2 with varying levels of soft information and a BCH code. Both codes have rate 0.9021. The BCH and LDPC 1-read curves correspond to hard decoding.

5.4 Performance vs. Number of Reads Per Cell

The MMI optimization approach generalizes to more than three reads per cell, but the optimization becomes more complex. In these cases. For these cases we use bisection optimization performed on promising small regions of the space until the optimal set of thresholds (within a small tolerance) was identified.

Fig. 5.8 shows how the performance of an LDPC code (Code 2 described in Section 5.5 below) improves as more soft information is made available to the decoder using MMI-optimized thresholds. This simulation uses the Gaussian model of the SLC Flash memory cell shown in Fig. 5.2.

Fig. 5.8 plots FER versus channel bit error probability computed as $Q\left(\sqrt{\frac{2E_s}{N_0}}\right)$. For reference, the FER performance of a binary BCH code capable of correcting up to 64 bit errors (using one read per cell) is also shown. Both the LDPC code and the BCH code have rate 0.9021. The LDPC

code has a frame size of $k = 8225$ bits and the BCH code has a frame size of $k = 8256$ bits. Also for reference, dashed vertical lines show the Shannon limit (worst channel that could theoretically support reliable transmission) for each level of quantization at the target rate of 0.9021.

Consistent with the mutual information analysis, this plot illustrates that each additional read improves the FER performance of the LDPC code, but the performance improvement is diminishing. Using three reads places the LDPC code performance within a relatively small gap from the limit of the performance achieved by that code with full soft information (essentially, an infinite number of reads). Note that the LDPC code outperforms the BCH code even with a single read, but one or two additional reads significantly improve performance.

5.5 LDPC Code Descriptions

LDPC codes [137] are linear block codes defined by sparse parity-check matrices. By optimizing the degree distribution, it is well-known that LDPC codes can approach the capacity of an AWGN channel [116]. Several algorithms have been proposed to generate LDPC codes for a given degree distribution, such as the ACE algorithm [138] and the PEG algorithm [139].

In addition to their powerful error-correction capabilities, another appealing aspect of LDPC codes is the existence of low-complexity iterative algorithms used for decoding. These iterative decoding algorithms are called belief-propagation algorithms. Belief-propagation decoders commonly use soft reliability information about the received bits, which can greatly improve performance. Conversely, a quantization of the received information which is too coarse can degrade the performance of an LDPC code.

Traditional algebraic codes, such as BCH codes, commonly use bounded-distance decoding and can correct up to a specified, fixed number of errors. Unlike these traditional codes, it can be difficult for LDPC codes to guarantee a specified number of correctable errors. However the average bit-error-rate performance can often outperform that of BCH codes in Gaussian noise.

5.5.1 Description of LDPC Codes

In this chapter we consider three irregular LDPC codes, which we will refer to as Code 1, Code 2, and Code 3. These codes were selected to illustrate two points about LDPC codes in the context of limited-precision quantization. The first point, illustrated later in this section, is that the relative performance of LDPC codes (i.e., which one is better) can depend on the level of quantization. Codes 2 and 3 were selected so that Code 2 has a better density evolution threshold than Code 3 for a single read while Code 3 has a better density evolution threshold than Code 2 for the full-precision AWGN channel.

The second point is that MMI quantization does not provide the right threshold for every code but should provide the right threshold as long as the code is good enough. This point is explored in Section 5.6. Code 1 provides an example of a code that is bad enough (because of small absorbing sets) that MMI quantization does not provide the correct quantization thresholds. Code 2 is a well-designed code that avoids these absorbing sets and for which the MMI quantization minimizes the frame error rate.

Codes 1 and 3 have degree distributions that optimize the density evolution threshold for the full-precision AWGN channel with maximum variable node degrees of 19 and 24 respectively. The degree distribution for Code 2 is a modification of the Code 1 degree distribution. It was not explicitly designed to optimize any density evolution threshold, but has a better density evolution threshold for the single-read AWGN channel than either Code 1 or Code 3.

The LDPC matrices were constructed according to their respective degree distributions using the ACE algorithm [138] and the stopping-set check algorithm [140]. All of the simulations were performed using a maximum of 50 iterations of a sequential belief propagation decoder. Decoding stops as soon as all check nodes are satisfied. The frame size is $k = 8225$ bits for each of the three

LDPC codes. The degree distributions of the three codes are as follows:

$$\begin{aligned}
\lambda_1(x) =& 2.0054 \times 10^{-5} + 3.5776 \times 10^{-2} x + 0.39869 x^2 \\
&+ 8.4827 \times 10^{-3} x^8 + 3.7701 \times 10^{-2} x^9 + 0.51933 x^{18} \\
\rho_1(x) =& 0.15662 x^{54} + 0.84338 x^{55} \\
\lambda_2(x) =& 1.7701 \times 10^{-5} + 3.1579 \times 10^{-2} x + 0.46923 x^3 \\
&+ 7.4877 \times 10^{-3} x^8 + 3.3278 \times 10^{-2} x^9 + 0.45841 x^{18} \\
\rho_2(x) =& 1.0975 \times 10^{-3} x^{61} + 0.73267 x^{62} + 0.26623 x^{63} \\
\lambda_3(x) =& 3.2172 \times 10^{-2} x + 2.681 \times 10^{-3} x^2 \\
&+ 0.55764 x^3 + 0.40751 x^{23} \\
\rho_3(x) =& 0.10366 x^{57} + 0.89634 x^{58} ,
\end{aligned}$$

where $\lambda(x)$ is the left (variable-node) degree distribution and $\rho(x)$ is the right (check-node) degree distribution. A term of ax^{d-1} in $\lambda(x)$ indicates that a is the fraction of edges connecting to variable nodes with degree d. The asymptotic threshold analysis of an LDPC code depends on the degree distribution of its variable and check nodes. $\lambda(x) = \sum_i \lambda_i x^{i-1}$ represents the variable-node degree distribution where λ_i is the fraction of the total number of edges connected to degree-i variable nodes. Similarly, $\rho(x) = \sum_j \rho_j x^{j-1}$ represents the check-node degree distribution where ρ_j is the fraction of all edges that are connected to degree-j check nodes. An LDPC code with variable-node and check-node degree distributions $\lambda(x)$ and $\rho(x)$ has a rate $r = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}$. For a particular code rate r, the code design optimization consists of finding the variable-node and check-node degree distributions that minimize the E_b/N_0 threshold.

5.5.2 Quantization-based Design Trade-off

Because reading a page of bits from the sense-amp comparators is a time-intensive operation, it is likely that enhanced precision will be added progressively only if needed to facilitate successful decoding. Hence, a single LDPC code will be decoded at a variety of precision levels, which

Table 5.1: Density evolution thresholds for three LDPC codes. Full-precision threshold are in terms of both noise variance σ and SNR. Single-read thresholds are in terms of channel bit error probability ϵ and the corresponding SNR.

	Full-precision AWGN		Single-Read AWGN	
Code	σ	$SNR = 2E_s/N_0$	ϵ	$SNR(\epsilon)$
1	0.499	6.04 dB	9.29×10^{-3}	7.44 dB
2	0.483	6.32 dB	1.05×10^{-2}	7.26 dB
3	0.492	6.16 dB	9.61×10^{-3}	7.39 dB

introduces a design trade-off that can be illustrated with two LDPC codes.

Table 5.1 shows the density evolution thresholds for these three codes for the extremes of a full-precision SLC channel and a single-read SLC channel assuming the Gaussian model of Fig. 5.2. Table 5.1 reveals a trade-off between full precision performance and single-read performance. For example, Code 2 has a lower (in dB) single-read threshold than Code 3, but a higher full-precision threshold than Code 3. The density evolution differences indicate that the different channels produced by the different quantizations will typically have different optimal LDPC codes under iterative belief propagation decoding.

Fig. 5.9 shows FER vs. SNR simulation results consistent with the density evolution threshold results shown in Table 5.1. Code 3 outperforms Code 2 when full soft information is available, but Code 2 outperforms Code 3 when only a single read is available. Additional FER vs. channel BER simulations for Codes 2 and 3 (that were omitted from Fig. 5.9 for clarity of presentation) demonstrate that for 2 reads the codes have essentially the same performance, but for three reads Code 3 has better performance than Code 2.

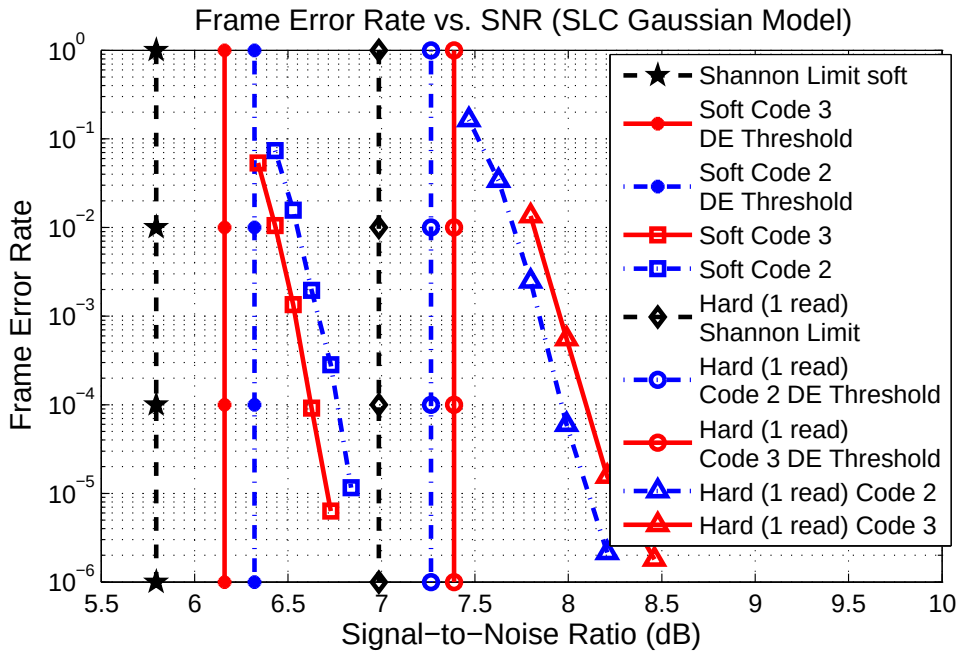


Figure 5.9: Simulation results of FER vs. SNR= $2E_s/N_0$ using the Gaussian channel model for SLC (two-level) Flash comparing LDPC Code 2 and LDPC Code 3 with hard decoding (1 read) and full soft decoding (essentially an infinite number of reads). Also shown are the Shannon limits for hard and soft decoding and the density evolution thresholds for the two codes under the two quantization scenarios. Both the density evolution results and simulation results show a trade-off between performance under hard decoding and performance under soft decoding.

5.6 RCA-based EXIT Function Analysis

In their density evolution analysis of binary-input AWGN channels, Chung et al. [141] observed that the distribution of the LLR messages at any iteration is approximately Gaussian. They showed that based on a stability condition, the variance of the Gaussian distribution has to be twice its mean. Therefore, the Gaussian distribution could be fully characterized by a single parameter. They also observed that due to a duality property (reciprocal approximation) the LLR messages from the variable to check nodes become additive at check nodes. This property is similar to the LLR messages from check nodes to variable nodes that are additive at variable nodes. The Gaussian Approximation and Reciprocal Channel Approximation simplified the DE analysis of

LDPC codes over binary-input (BI) AWGN channels by tracking a single parameter μ, representing the mean, as a surrogate for the entire LLR distribution in each iteration.

The references [142] and [143] applied similar ideas by using the EXtrinsic mutual Information Transfer (EXIT) functions to simplify the DE threshold analysis. EXIT functions in iterative coding schemes characterize how *a priori* input information to a decoder check or variable node converts into *extrinsic* output information from the decoder check or variable node. The EXIT functions track the evolution of the MI between the variable-to-check messages and the variable node true value (I_v^{out}) and the MI between check-to-variable messages and the variable node true value (I_v^{in}).

We now describe the use of EXIT functions with the Gaussian approximation and RCA to compute LDPC decoding thresholds for different number of reads. In the following sections, similar to Fig. 5.3, p_is represent the transition probabilities. The LLRs are given by $L_{ij} = \log \frac{p_i}{p_j}$. $f_L^{ch}(l)$ represents the apriori channel LLR probability distribution function (p.d.f.). The notation $\oplus$ represents the convolution operator. $\delta(.)$ represents the Dirac delta function.

5.6.1 Binary LDPC Thresholds from RCA-based EXIT Functions

By the assumption of uniform input, symmetric channel, and transmission of the all-zero codeword the initial channel LLR ($\log \frac{P(Y|X=0)}{P(Y|X=1)}$) distribution at a variable node for the single-read SLC channel in Fig. 5.3a is

$$f_L^{ch}(l) = p_1\delta(l - L_{12}) + p_2\delta(l - L_{21}), \tag{5.12}$$

where $L_{ij} = \log \frac{p_i}{p_j}$. This p.d.f. is effectively a p.m.f. with support at $L_{12} = \log(\frac{P(Y=1|X=0)}{P(Y=1|X=1)})$ and $L_{21} = \log(\frac{P(Y=2|X=0)}{P(Y=2|X=1)})$. Since the all-zeros codeword is assumed to be transmitted, the channel LLR takes the values of L_{12} and L_{21} with probabilities p_1 and p_2 respectively. Similar expressions hold for more reads, e.g. with five reads the channel LLR p.d.f. is

$$\begin{aligned} f_L^{ch}(l) &= p_1\delta(l - L_{16}) + p_2\delta(l - L_{25}) + p_3\delta(l - L_{34}) \\ &+ p_4\delta(l - L_{43}) + p_5\delta(l - L_{52}) + p_6\delta(l - L_{61}), \end{aligned} \tag{5.13}$$

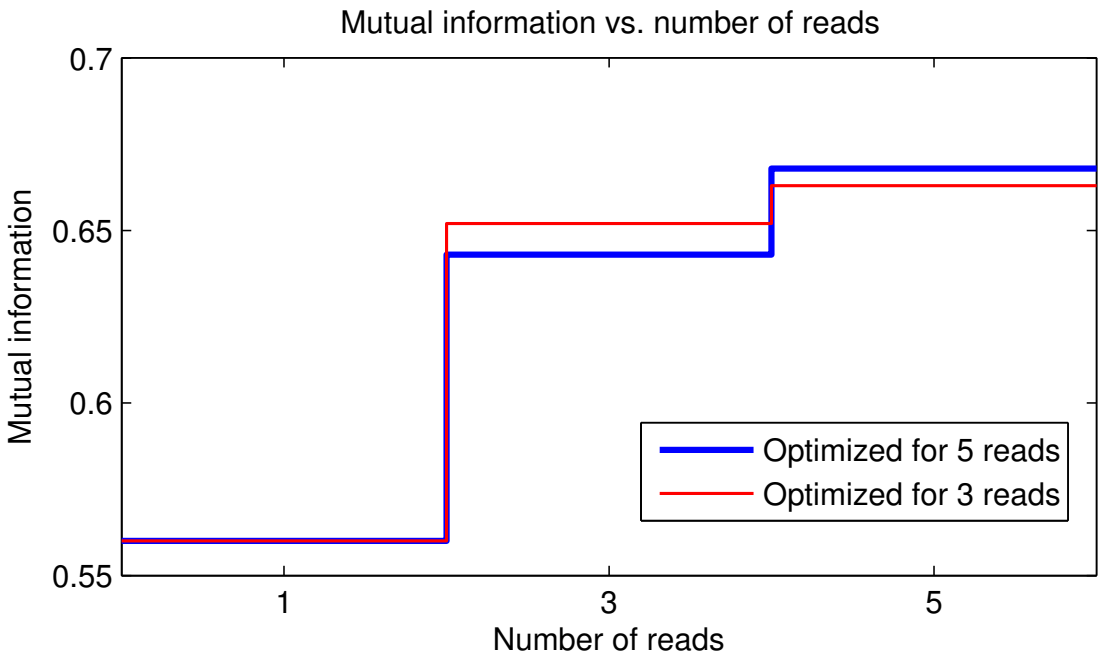


Figure 5.10: Mutual information vs. q_1 and q_2 for SLC_5

with the p_is as defined previously for the five-read channel. For the full soft-information (i.e. in the limit of infinitely many reads of the flash cell) BI-AWGN channel, $LLR = \log(\frac{P(Y=y|X=0)}{P(Y=y|X=1)}) = \frac{2}{\sigma^2}y$ where σ^2 is the variance of the Gaussian noise. Therefore, the LLR is normally distributed with a mean of $\frac{2}{\sigma^2}$ and variance of $\frac{4}{\sigma^2}$:

$$f_L^{ch}(l) = \mathcal{N}(2/\sigma^2, 4/\sigma^2)\,. \tag{5.14}$$

In the first iteration (initialization step) of the threshold calculation algorithm, the extrinsic mutual information from a variable node is calculated by $I_v^{out} = J(f_L^{ch}(l))$ where

$$J(f_L(l)) = 1 - \int_L \log_2(1+e^{-l}) f_L(l) dl \tag{5.15}$$

$J(f_L(l))$ is a function of the LLR distribution $f_L(l)$ which gives the mutual information of a binary-input symmetric-output channel with the LLR distribution of $f_L(l)$. For instance, for the 1-read SLC channel we can alternatively calculate the mutual information of (5.9) as $J(f_L^{ch}(l))$ where $f_L^{ch}(l)$ is given in (5.12). If the LLR distribution is in form of $\mathcal{N}(\mu, 2\mu)$, $J_{\mathcal{N}}(\mu) = J(\mathcal{N}(\mu, 2\mu))$.

After initialization, (5.16-5.21) are performed. This process is repeated until $H(X) - I_v^{out} < \epsilon'$ (e.g. 10^{-7}), where $H(X)$ is the entropy of the input in Fig. 5.3. This is the convergence or the stopping criterion of the algorithm. If the threshold-calculation algorithm fails at a particular noise level, the noise level is decreased until the maximum noise level (minimum E_b/N_0) at which the algorithm converges is identified.

$$I_c^{in} = 1 - I_v^{out} \tag{5.16}$$

$$\mu_c^{in} = J_{\mathcal{N}}^{-1}(I_c^{in}) \tag{5.17}$$

$$I_c^{out} = \sum_{j=1}^{m} \rho_i J_{\mathcal{N}}((j-1)\mu_c^{in}, 2(j-1)\mu_c^{in}) \tag{5.18}$$

$$I_v^{in} = 1 - I_c^{out} \tag{5.19}$$

$$\mu_v^{in} = J_{\mathcal{N}}^{-1}(I_v^{in}) \tag{5.20}$$

$$I_v^{out} = \sum_{i=1}^{n} \lambda_i J(f_L^{ch}(l) \oplus \mathcal{N}((i-1)\mu_v^{in}, 2(i-1)\mu_v^{in}) \tag{5.21}$$

Eq. (5.16) follows from the duality property between the extrinsic information from a variable node (I_v^{out}) and the apriori information to a neighboring check node (I_c^{in}). Eq. (5.17) follows from the monotonicity of J, hence the inverse exists, as well as the assumption of normally distributed LLR messages at each iteration. μ_c^{in} is the mean of the approximately normally distributed LLR messages at check nodes. Eq. (5.18) follows from the reciprocal approximation of the messages exchanged between variable and check nodes, resulting in additive messages at check nodes. Eq. (5.19) follows from the duality property between I_v^{in} and I_c^{out}. Eq. (5.20) follows the monotonicity of J and the assumption of normally distributed LLR messages. μ_v^{in} is the mean of the approximately normally distributed LLR messages at variable nodes. Eq. (5.21) follows from the additive property of messages at variable nodes due to the independence assumption of apriori LLR messages.

For the SLC channels with one or more reads, such as the single-read and five-read cases of (5.12) and (5.13), the convolution of the channel LLR p.d.f. ($f_L^{ch}(l)$) and the LLR p.d.f. of the incoming messages ($\mathcal{N}(\mu_v^{in}, 2\mu_v^{in})$) from the neighboring check nodes to variable node v in (5.21) results in a mean shift LLR distribution of the messages from the check nodes. For example, for the single-read case, $f_L^{ch}(l) \oplus \mathcal{N}(\mu_v^{in}, 2\mu_v^{in}) = p_1\mathcal{N}(\mu_v^{in} - L_{12}, 2\mu_v^{in}) + p_2\mathcal{N}(\mu_v^{in} - L_{21}, 2\mu_v^{in})$. However, for

BI-AWGN (5.14) the convolution results in the Gaussian distribution $\mathcal{N}(\mu_v^{in} + 2/\sigma^2, 2\mu_v^{in} + 4/\sigma^2)$.

5.6.2 RCA-EXIT for Non-binary LDPC codes for SLC

For non-binary LDPC codes over $GF(q)$ for SLC with uniform input, the LLR messages are $(q-1)$ dimensional probability distribution vectors. We only consider $GF(4)$ where the LLR vectors correspond to $\{01, 10, 11\}$. The steps in calculating the threshold are similar to (5.16) to (5.21). However, the J functional and the initialization steps are modified as follows:

$$\boldsymbol{l} = \begin{pmatrix} LLR_{01}^{ch} \\ LLR_{10}^{ch} \\ LLR_{11}^{ch} \end{pmatrix} = \begin{pmatrix} \log \frac{P(Y|X=00)}{P(Y|X=01)} \\ \log \frac{P(Y|X=00)}{P(Y|X=10)} \\ \log \frac{P(Y|X=00)}{P(Y|X=11)} \end{pmatrix} \tag{5.22}$$

$$J(f_{\boldsymbol{L}}(\boldsymbol{l})) = 1 - \frac{1}{2}\int_{\boldsymbol{L}} \log_2(1 + \sum_{i=1}^{3} e^{-l_i}) f_{\boldsymbol{L}}(\boldsymbol{l}) d\boldsymbol{l} \tag{5.23}$$

$$J_{\mathcal{N}}(\boldsymbol{\mu}) = J(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) \tag{5.24}$$

$$\boldsymbol{\mu} = \begin{pmatrix} \mu \\ \mu \\ \mu \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} 2\mu & \mu & \mu \\ \mu & 2\mu & \mu \\ \mu & \mu & 2\mu \end{pmatrix}. \tag{5.25}$$

Eq. (5.22) shows the 3-dimensional LLR probability distribution vector with the assumption that the all-zero codeword is transmitted. The Eq. (5.23) follows from the assumption that each bit is independently sent over the channel. For a $GF(4)$ NB-LDPC code for SLC Flash systems the two bits are independently combined to give a $GF(4)$ element and the MI is the average of the MI for each bit. The Eq. (5.24) is an extension of the binary $J_{\mathcal{N}}$ to $GF(4)$. The Eq. (5.25) follows from the stability condition of DE along with the Gaussian approximation which requires the mean and co-variance matrix of the normally distributed messages to be in form of the Eq. (5.25) under the assumption that the $GF(4)$ labels are chosen at random and uniformly [144]. This property is in compliance with $GF(2)$ where $\sigma^2 = 2\mu$.

The channel LLR p.d.f. for a $GF(4)$ NB-LDPC code for SLC over BI-AWGN channel is a

jointly Gaussian vector with the mean vector and co-variance matrix as in the Eq. (5.27).

$$f_{\boldsymbol{L}}^{ch}(\boldsymbol{l}) = \mathcal{N}(\boldsymbol{\mu}^{ch}, \boldsymbol{\Sigma}^{ch}) \tag{5.26}$$

$$\boldsymbol{\mu}^{ch} = \begin{pmatrix} 2/\sigma^2 \\ 2/\sigma^2 \\ 4/\sigma^2 \end{pmatrix} \quad \boldsymbol{\Sigma}^{ch} = \begin{pmatrix} 4/\sigma^2 & 0 & 4/\sigma^2 \\ 0 & 4/\sigma^2 & 4/\sigma^2 \\ 4/\sigma^2 & 4/\sigma^2 & 8/\sigma^2 \end{pmatrix} \tag{5.27}$$

By assuming that the all-zero codeword is transmitted, y_1 and y_2 in Eq. (5.28) are i.i.d $\mathcal{N}(1, \sigma^2)$ random variables.

$$l\begin{pmatrix} LLR_{01}^{ch} = \log(\frac{P(Y=y_1y_2|X=00)}{P(Y=y_1y_2|X=01)}) \\ LLR_{10}^{ch} = \log(\frac{P(Y=y_1y_2|X=00)}{P(Y=y_1y_2|X=10)}) \\ LLR_{11}^{ch} = \log(\frac{P(Y=y_1y_2|X=00)}{P(Y=y_1y_2|X=11)}) \end{pmatrix} = \begin{pmatrix} \frac{2y_2}{\sigma^2} \\ \frac{2y_1}{\sigma^2} \\ \frac{2(y_1+y_2)}{\sigma^2} \end{pmatrix} \tag{5.28}$$

Eq. (5.29) shows the channel LLR p.d.f. vector for NB_1^{SLC}. It also shows that $\text{LLR}_{11} = \text{LLR}_{01} + \text{LLR}_{10}$ due to the independence of noise affecting each transmitted bit.

$$\begin{aligned} f_{\boldsymbol{L}}^{ch}(\boldsymbol{l}) &= p_1^2\delta(\boldsymbol{l} - \begin{bmatrix} L_{12} \\ L_{12} \\ 2L_{12} \end{bmatrix}) + p_1p_2\delta(\boldsymbol{l} - \begin{bmatrix} L_{12} \\ L_{21} \\ 0 \end{bmatrix}) \\ &+ p_2p_1\delta(\boldsymbol{l} - \begin{bmatrix} L_{21} \\ L_{12} \\ 0 \end{bmatrix}) + p_2^2\delta(\boldsymbol{l} - \begin{bmatrix} L_{21} \\ L_{21} \\ 2L_{21} \end{bmatrix}) \end{aligned} \tag{5.29}$$

5.6.3 RCA-EXIT for Non-binary LDPC Codes for MLC

In this chapter we assume i.i.d. Gaussian threshold voltages for each charge level in SLC or MLC Flash memory cells. These models are equivalent to 2-level or 4-level Pulse-Amplitude Modulation with additive white Gaussian ($\mathcal{N}(0, \sigma^2)$) noise (AWGN). For the SLC model, since the levels are at $+1$ and -1, the average energy (E_{avg}) of this model is 1. Similar to Fig. 5.5, the shifted 4-level MLC threshold voltage distribution is a mixture of four identically distributed Gaussian random variables. The voltage levels are at $\{+\frac{3}{\sqrt{5}}, +\frac{1}{\sqrt{5}}, -\frac{1}{\sqrt{5}}, -\frac{3}{\sqrt{5}}\}$ and correspond to the Gray labeled assignment $\{00, 01, 11, 10\}$ respectively. In this model $E_{avg} = 1$, similar to the SLC model.

We use $GF(4)$ NB-LDPC threshold analysis to find the best code for the MLC Flash models.

The J functional for MLC under the assumption of uniform input is

$$J(f_{\boldsymbol{L}}(\boldsymbol{l})) = 2 - \int \log_2(1 + \sum_{i=1}^{3} e^{-l_i}) f_{\boldsymbol{L}}(\boldsymbol{l}) d\boldsymbol{l}\,. \tag{5.30}$$

The channel LLR p.d.f. is described as follows:

$$\begin{aligned} f_{\boldsymbol{L}}^{ch}(\boldsymbol{l}) = \frac{1}{4}(&\mathcal{N}(\boldsymbol{\mu}_{00}^{ch}, \boldsymbol{\Sigma}_{00}^{ch}) + \mathcal{N}(\boldsymbol{\mu}_{01}^{ch}, \boldsymbol{\Sigma}_{01}^{ch}) \\ &+\mathcal{N}(\boldsymbol{\mu}_{10}^{ch}, \boldsymbol{\Sigma}_{10}^{ch}) + \mathcal{N}(\boldsymbol{\mu}_{11}^{ch}, \boldsymbol{\Sigma}_{11}^{ch})) \end{aligned} \tag{5.31}$$

$$\begin{aligned} \boldsymbol{\mu}_{00}^{ch} &= \begin{bmatrix} 2/5\sigma^2 \\ 18/5\sigma^2 \\ 8/5\sigma^2 \end{bmatrix} \boldsymbol{\Sigma}_{00}^{ch} = \begin{pmatrix} 4/\sigma^2 & 12/\sigma^2 & 8/\sigma^2 \\ 12/\sigma^2 & 36/\sigma^2 & 24/\sigma^2 \\ 8/\sigma^2 & 24/\sigma^2 & 16/\sigma^2 \end{pmatrix} \\ \boldsymbol{\mu}_{01}^{ch} &= \begin{bmatrix} 2/5\sigma^2 \\ 2/5\sigma^2 \\ 8/5\sigma^2 \end{bmatrix} \boldsymbol{\Sigma}_{01}^{ch} = \begin{pmatrix} 4/\sigma^2 & -4/\sigma^2 & -8/\sigma^2 \\ -4/\sigma^2 & 4/\sigma^2 & 8/\sigma^2 \\ -8/\sigma^2 & 8/\sigma^2 & 16/\sigma^2 \end{pmatrix} \\ \boldsymbol{\mu}_{10}^{ch} &= \boldsymbol{\mu}_{00}^{ch} \quad \boldsymbol{\mu}_{11}^{ch} = \boldsymbol{\mu}_{01}^{ch} \quad \boldsymbol{\Sigma}_{10}^{ch} = \boldsymbol{\Sigma}_{00}^{ch} \quad \boldsymbol{\Sigma}_{11}^{ch} = \boldsymbol{\Sigma}_{01}^{ch} \end{aligned} \tag{5.32}$$

The p.d.f. vectors in Eq. (5.32) are obtained by using the idea in [144] for cosets over $GF(q)$ and the assumption that the non-binary labels are selected at random and uniformly.

5.7 LDPC Degree Distributions Optimization for Flash

By assuming that the LLR values at any iteration of the DE algorithm are normally distributed with the variance (σ^2) twice the absolute value of the mean (μ), the approximation of the DE is obtained by considering the μ as a surrogate of the entire LLR distribution.

The degree-distribution optimization algorithm for a rate-r code starts with finding the threshold for an initial (e.g. regular) degree distribution that results in a rate-r code. While the rate is kept constant, the parameters of $\lambda(x)$ and $\rho(x)$ are slightly changed and the new threshold is calculated. If the new threshold is smaller than the previous threshold, the new degree distribution is selected and slightly changed to find a degree distribution with lower threshold; otherwise, the previously lowest degree distribution is slightly changed to find a degree distribution which gives a lower

threshold. Once there is no more improvement in threshold by changing the degree distribution, the degree distribution with the lowest threshold is considered to be optimal. Table 5.2 shows the degree distributions we obtained using this approach.

5.7.1 MMI vs. RCA-EXIT for Word-Line-Voltage Optimization

For a fixed degree distribution, RCA-EXIT analysis can determine the word-line voltages that minimize the E_b/N_0 threshold for a multiple-read Flash channel. This is an alternative approach to MMI word-line voltages. Fig. 5.11 shows the plot of the threshold E_b/N_0 for each word-line voltage $q_1 = q$ for the 2-read SLC model in Fig. 5.3b for two degree distributions. Also shown is the curve showing the MMI word-line voltage (q_{MMI}) for each E_b/N_0. For a fixed degree distribution and q, we find the minimum required E_b/N_0 such that the threshold calculation algorithm converges as was explained in 5.6.1. The optimal value of q and its corresponding E_b/N_0 are shown in Fig. 5.11 by the pair $(q^*, E_b/N_0^*)$. E_b/N_0^* is the absolute minimum required E_b/N_0 for the RCA-EXIT to converge for at least one quantization threshold q. The threshold E_b/N_0 at q_{MMI} is less than 1% away from the optimal E_b/N_0^*.

It is difficult to simultaneously optimize both degree distribution and word-line voltage. Thus, even if the final word-line voltage will be selected to explicitly minimize the threshold, the MMI word-line voltage at a given E_b/N_0 is an excellent approximation when optimizing degree distributions. We use the q_{MMI} to optimize the degree distribution until the RCA-EXIT no longer converges at an E_b/N_0 and then adjust the q to obtain the final small improvement in E_b/N_0.

5.7.2 Optimized Degree Distributions for Each Precision Level

Table 5.3 shows the thresholds achieved by the various SLC degree distributions with various levels of precision. As expected, for a specified number of reads, the degree distributions optimized for that number of reads has the lowest threshold. However, this table quantifies the relatively small performance loss in threshold that is required for the same code to be used with multiple decoding attempts using increased precision.

Table 5.2: Optimized degree distributions for the Gaussian model of SLC Flash with 1,2,3, and 5 reads. MLC Flash with 3 reads, and both SLC and MLC with full soft information.

# Reads	Coefficients					
SLC	λ_2	λ_3	λ_7	λ_8	λ_{27}	ρ_{61}
1 read	0.07	0.25	0.11	0.13	0.44	1
SLC	λ_2	λ_3	λ_7	λ_{25}		ρ_{57}
2 reads	0.1	0.21	0.25	0.44		1
SLC	λ_2	λ_3	λ_6	λ_7	λ_{26}	ρ_{56}
3 reads	0.1	0.21	0.11	0.12	0.46	1
SLC	λ_2	λ_3	λ_5	λ_8	λ_{25}	ρ_{56}
5 reads	0.11	0.21	0.09	0.14	0.45	1
SLC	λ_2	λ_3	λ_6	λ_8	λ_{26}	ρ_{56}
soft	0.11	0.21	0.21	0.14	0.47	1
MLC	λ_2	λ_3	λ_4	λ_7	λ_8	λ_{11}
	0.16	0.31	0.1	0.18	0.1	0.15
3 reads	ρ_5	ρ_6	ρ_8	ρ_{22}		
	0.45	0.16	0.1	0.29		
MLC	λ_2	λ_3	λ_4	λ_7	λ_9	λ_{11}
	0.15	0.3	0.1	0.09	0.22	0.14
soft	ρ_5	ρ_6	ρ_8	ρ_9	ρ_{22}	
	0.44	0.07	0.1	0.11	0.27	

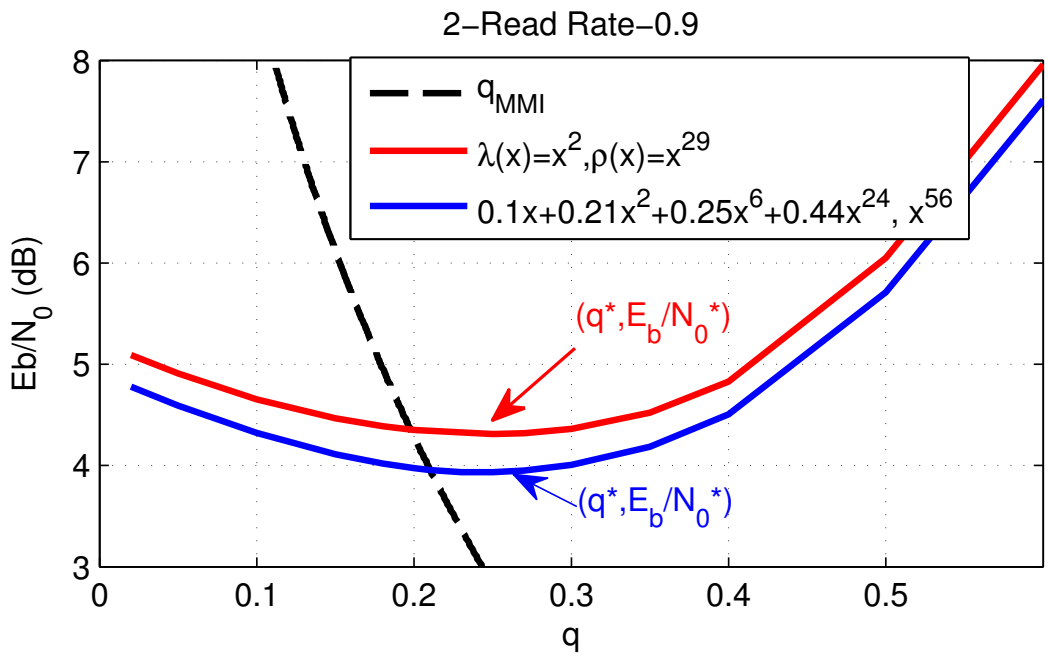


Figure 5.11: Threshold E_b/N_0 vs q for a regular and the optimized rate-0.9 binary LDPC codes for 2-read SLC model

The largest performance loss occurs when using a degree distribution optimized for 5 reads on the SLC 1-read channel. Here, the loss is 0.17 dB. Any degree distribution except the one designed for the single-read channel experiences the 0.17 dB loss. In contrast, if the degree distribution optimized for the single-read SLC channel is used on the 5-read channel, only 0.11 dB of loss is incurred. Thus there is reason to consider using the code designed for a single read. However, using the degree distribution optimized for 5 reads would minimize the probability of losing a page at the expense of additional decoder latency for those times when additional reads might have been avoided by using the degree distribution optimized for one read. If it becomes feasible to switch among codes as the Flash cells deteriorate over time, an optimized code for a higher number of reads may replace the previously optimized code with lower amount of precision.

Fig. 5.12 compares the threshold of regular LDPC degree distributions with the optimized irregular degree distribution $(\lambda(x), \rho(x))$ for each number of reads. There is roughly a gain of 0.4 dB for the optimized codes compared to the regular (x^2, x^{29}) and (x^3, x^{39}) codes.

Table 5.3: Thresholds for optimized degree distributions on SLC flash with 1, 2, 3, and 5 reads as well as soft information.

Target	1 read	2 reads	3 reads	5 reads	Soft
1 read	**4.752**	3.995	3.728	3.542	3.398
2 reads	4.922	**3.943**	3.658	3.470	3.324
3 reads	4.923	3.958	**3.640**	3.441	3.295
5 reads	4.926	3.973	3.649	**3.437**	3.288
Soft	4.926	3.982	3.662	3.443	**3.275**
Shannon-Limit	4.400	3.733	3.495	3.328	3.198

For the same average energy, the optimized rate-0.45 non-binary codes used in the MLC model have a much lower threshold compared to the rate-0.9 codes for the SLC model. MLC Flash degree distributions provide thresholds more than 0.5 dB lower than degree distributions for rate-0.9 binary LDPC codes on SLC Flash with the same number of reads (i.e. three reads that would provide hard decisions for MLC and limited soft information for SLC). The MLC approach has a potential threshold reduction of about 1.5 dB over the SLC when both systems have access to full soft information. This is consistent with the MI analysis of [145] showing that a spectral efficiency of 0.9 is too high for binary PAM.

5.8 Conclusions

This work explores the benefit of using soft information in an LDPC decoder for NAND Flash memory. Using a small amount of soft information improves the performance of LDPC codes significantly and demonstrates a clear performance advantage over conventional BCH codes.

In order to maximize the performance benefit of the soft information, we present an approach

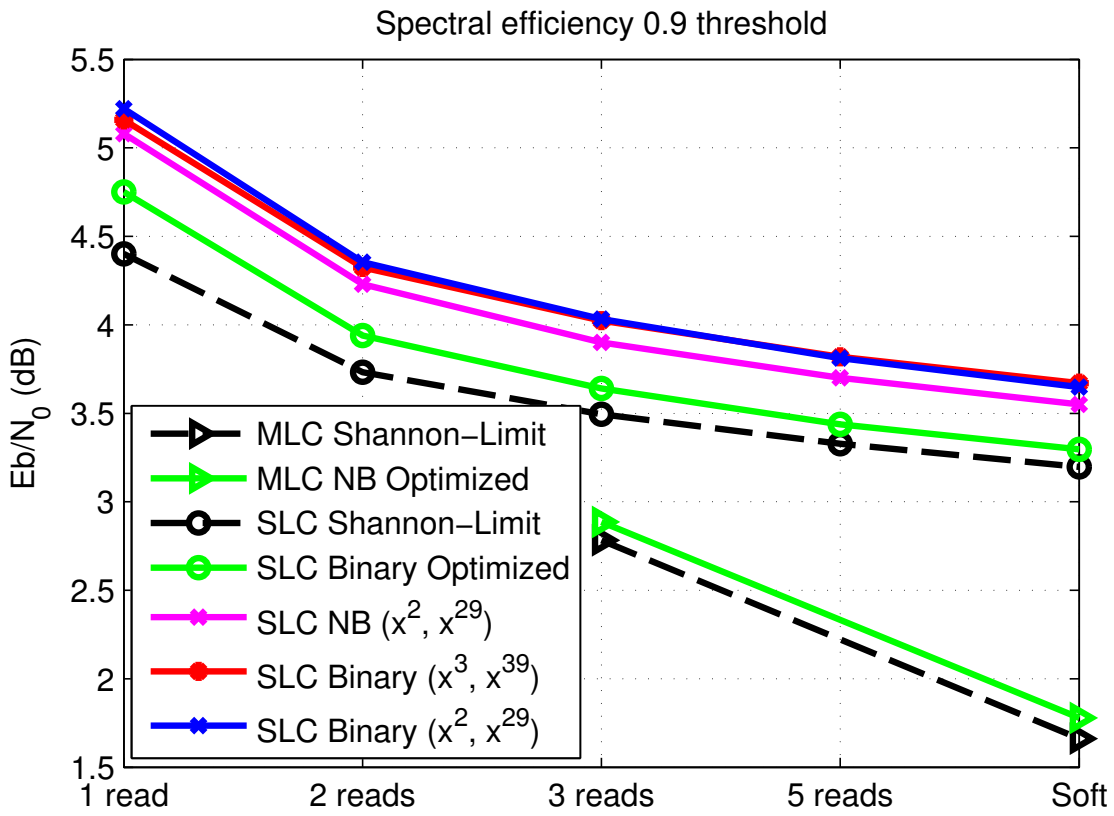


Figure 5.12: Threshold E_b/N_0 vs number of reads for spectral efficiency of 0.9 bits per cell.

for optimizing word-line-voltage selection so that the resulting quantization maximizes the mutual information between the input and output of the equivalent read channel. Furthermore, only a few additional reads can harvest most of the performance improvement available through enhanced precision.

Furthermore, in this work, the channel information has been quantized with various levels of precision. However, the messages passed between the variable nodes and check nodes of the decoder have been represented as floating point numbers in our simulations. It is an interesting area of further investigation to consider limited-precision representations within the LDPC decoder in conjunction with the limited-precision channel information that is available in the Flash memory setting.

We have used RCA-EXIT analysis to optimize the degree distribution of binary and non-binary LDPC codes with E_b/N_0 thresholds of about 0.1 dB away from the Shannon-Limit for multiple-

read models. While regular codes are usually used in Flash systems due to their simple decoder implementation, the optimized irregular binary codes have 0.4 dB lower threshold than the regular codes across different number of reads.

We have examined and validated the use of MMI word-line voltages in threshold-based degree distribution optimization. For the same spectral efficiency, low-rate non-binary LDPC codes for MLC Flash systems have lower thresholds compared to high-rate LDPC codes used in SLC systems. In addition, it is easier to design low-rate (e.g. rate-0.45) than high-rate (e.g. rate-0.9) LDPC codes.

REFERENCES

[1] C. E. Shannon, "The zero error capacity of a noisy channel," *IRE Trans. Inf. Theory*, vol. 2, no. 3, pp. 8–19, Sep. 1956.

[2] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Feedback in the non-asymptotic regime," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 4903–4925, Aug. 2011.

[3] T.-Y. Chen, N. Seshadri, and R. Wesel, "A sphere-packing analysis of incremental redundancy with feedback," in *2011 IEEE Int. Conf. Commun. (ICC)*, June 2011, pp. 1–5.

[4] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[5] S. Lin and P. Yu, "A hybrid arq scheme with parity retransmission for error control of satellite channels," *IEEE Trans. Commun.*, vol. 30, no. 7, pp. 1701–1719, July 1982.

[6] J. Costello, D.J., J. Hagenauer, H. Imai, and S. Wicker, "Applications of error-control coding," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2531–2560, Oct. 1998.

[7] E. Visotsky, Y. Sun, V. Tripathi, M. Honig, and R. Peterson, "Reliability-based incremental redundancy with convolutional codes," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 987– 997, June 2005.

[8] C. Lott, O. Milenkovic, and E. Soljanin, "Hybrid ARQ: Theory, state of the art and future directions," in *2007 IEEE Inf. Theory Workshop for Wireless Networks*, Bergen, Norway, July 2007.

[9] J. Fricke and P. Hoeher, "Reliability-based retransmission criteria for hybrid ARQ," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2181–2184, Aug. 2009.

[10] M. Heindlmaier and E. Soljanin, "Isn't hybrid ARQ sufficient?" *CoRR*, vol. abs/1411.4061, 2014. [Online]. Available: http://arxiv.org/abs/1411.4061

[11] A. Roongta and J. Shea, "Reliability-based hybrid ARQ using convolutional codes," in *Proc. 2003 IEEE Int. Conf. Commun. (ICC)*, vol. 4, May 2003, pp. 2889–2893.

[12] ——, "Reliability-based hybrid ARQ and rate-compatible punctured convolutional (RCPC) codes," in *Proc. 2004 IEEE Wireless Commun. and Networking Conf. (WCNC)*, vol. 4, Mar. 2004, pp. 2105–2109.

[13] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[14] A. Raghavan and C. Baum, "A reliability output Viterbi algorithm with applications to hybrid ARQ," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1214–1216, May 1998.

[15] J. C. Fricke, H. Schoeneich, and P. A. Hoeher, "Reliability-based HARQ using word error probabilities," in *Proc. 2006 NEWCOM-ACoRN Joint Workshop (NAW)*.

[16] J. Fricke and P. Hoeher, "Word error probability estimation by means of a modified Viterbi decoder," in *Proc. 66th IEEE Veh. Technol. Conf. (VTC)*, Oct. 2007, pp. 1113–1116.

[17] A. R. Williamson, M. J. Marshall, and R. D. Wesel, "Reliability-output decoding of tail-biting convolutional codes," *IEEE Trans. Commun.*, vol. 62, no. 6, pp. 1768–1778, June 2014.

[18] E. Soljanin, N. Varnica, and P. Whiting, "Incremental redundancy hybrid ARQ with LDPC and Raptor codes," 2005. [Online]. Available: ftp://netlib.bell-labs.com/cm/ms/who/emina/papers/hybridarq_final.pdf

[19] ——, "Punctured vs rateless codes for hybrid ARQ," in *Proc. 2006 IEEE Inf. Theory Workshop (ITW)*, Punta del Este, Uruguay, Mar. 2006, pp. 155–159.

[20] I. Andriyanova and E. Soljanin, "IR-HARQ schemes with finite-length punctured LDPC codes over the BEC," in *Proc. 2009 IEEE Inf. Theory Workshop (ITW)*, Taormina, Sicily, Oct. 2009, pp. 125 –129.

[21] ——, "Optimized IR-HARQ schemes based on punctured LDPC codes over the BEC," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6433–6445, Oct. 2012.

[22] J. Perry, H. Balakrishnan, and D. Shah, "Rateless spinal codes," in *Proc. 10th ACM Workshop on Hot Topics in Networks*, 2011.

[23] J. Perry, P. A. Iannucci, K. E. Fleming, H. Balakrishnan, and D. Shah, "Spinal codes," in *Proc. 2012 ACM SIGCOMM Conf. on Applicat., Tech., Arch., and Protocols for Comput. Commun.*, Helsinki, Finland, 2012, pp. 49–60.

[24] D. L. Romero, "A comparative analysis of physical-layer rateless coding architectures," Master's thesis, MIT, Cambridge, MA, 2014.

[25] K. Chen, K. Niu, and J.-R. Lin, "A hybrid ARQ scheme based on polar codes," *IEEE Commun. Lett.*, vol. 17, no. 10, pp. 1996–1999, Oct. 2013.

[26] K. Chen, K. Niu, Z. He, and J.-R. Lin, "Polar coded HARQ scheme with Chase combining," in *Proc. 2014 IEEE Wireless Commun. and Network Conf. (WCNC)*, Apr. 2014, pp. 474–479.

[27] S. Pfletschinger, D. Declercq, and M. Navarro, "Adaptive HARQ with non-binary repetition coding," *IEEE Trans. on Wireless Commun.*, vol. 13, no. 8, pp. 4193–4204, Aug. 2014.

[28] A. Williamson, T.-Y. Chen, and R. Wesel, "A rate-compatible sphere-packing analysis of feedback coding with limited retransmissions," in *2012 IEEE Int. Symp. Inf. Theory (ISIT)*, July 2012, pp. 2924–2928.

[29] ——, "Variable-length convolutional coding for short blocklengths with decision feedback," *IEEE Trans. Commun.*, vol. 63, no. 7, pp. 2389–2403, July 2015.

[30] B.-Y. Chang, D. Divsalar, and L. Dolecek, "Non-binary protograph-based LDPC codes for short block-lengths," in *2012 IEEE Inf. Theory Workshop (ITW)*, Sep. 2012, pp. 282–286.

[31] A. R. Williamson, "Reliability-output decoding and low-latency variable-length coding schemes for communication with feedback," Ph.D. dissertation, Dept. Elec. Eng., UCLA, Los Angeles, CA, 2014.

[32] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular $(2, d_c)$-LDPC codes over GF(q) using their binary images," *IEEE Trans. Commun.*, vol. 56, no. 10, pp. 1626–1635, Oct. 2008.

[33] S. Ranganathan, K. Vakilinia, D. Divsalar, and R. Wesel, "Design of high-rate irregular non-binary LDPC codes using algorithmic stopping-set cancellation," in *Proc. 2014 IEEE Int. Symp. Inf. Theory (ISIT)*, July 2014.

[34] T.-Y. Chen, A. R. Williamson, N. Seshadri, and R. D. Wesel, "Feedback communication systems with limitations on incremental redundancy," Available: http://arxiv.org/abs/1309.0707.

[35] K. Vakilinia, T.-Y. Chen, S. V. S. Ranganathan, A. R. Williamson, D. Divsalar, and R. D. Wesel, "Short-blocklength non-binary LDPC codes with feedback-dependent incremental transmissions," in *Proc. 2014 IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, USA, July 2014.

[36] K. Vakilinia, A. Williamson, S. Ranganathan, D. Divsalar, and R. Wesel, "Feedback systems using non-binary LDPC codes with a limited number of transmissions," in *Inf. Theory Workshop (ITW), 2014 IEEE*, Nov 2014, pp. 167–171.

[37] M. Naghshvar, T. Javidi, and M. A. Wigger, "Extrinsic jensen-shannon divergence: Applications to variable-length coding," *CoRR*, vol. abs/1307.0067, 2013.

[38] F. Leduc-Primeau, S. Hemati, S. Mannor, and W. Gross, "Dithered belief propagation decoding," *IEEE Trans. Commun.*, vol. 60, no. 8, pp. 2042–2047, 2012.

[39] T. Chen, K. Vakilinia, D. Divsalar, and R. Wesel, "Protograph-based raptor-like LDPC codes," *IEEE Trans. Commun.*, vol. 63, no. 5, pp. 1522–1532, May 2015.

[40] C.-Y. Lou, B. Daneshrad, and R. Wesel, "Convolutional-code-specific crc code design," *IEEE Trans. Commun.*, to be published. Available: http://arxiv.org/abs/1506.02990.

[41] A. Williamson, T.-Y. Chen, and R. Wesel, "Firing the genie: Two-phase short-blocklength convolutional coding with feedback," in *2013 Inf. Theory and Applications Workshop (ITA)*, 2013, pp. 1–6.

[42] J. Anderson and K. Balachandran, "Decision depths of convolutional codes," *IEEE Trans. Inf. Theory*, vol. 35, no. 2, pp. 455–459, Mar. 1989.

[43] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *CoRR*, vol. abs/0807.3917, 2008. [Online]. Available: http://arxiv.org/abs/0807.3917

[44] B. Serbetci and A. Pusane, "On the selection of generator matrices for polar coded communication systems," in *Signal Processing and Communications Applications Conference (SIU), 2012 20th*, April 2012, pp. 1–4.

[45] S. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *Information Theory, IEEE Transactions on*, vol. 56, no. 12, pp. 6253–6264, Dec 2010.

[46] E. Arikan and I. Telatar, "On the rate of channel polarization," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, June 2009, pp. 1493–1495.

[47] S. Y. Chung, "On the construction of some capacity-approaching coding schemes," Ph.D. dissertation, MIT, Cambridge, MA, 2000.

[48] T. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like LDPC codes," *CoRR*, vol. abs/1403.2111, 2014. [Online]. Available: http://arxiv.org/abs/1403.2111

[49] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2657–2673, Nov 2004.

[50] T.-Y. Chen, A. Williamson, N. Seshadri, and R. Wesel, "Feedback communication systems with limitations on incremental redundancy," *IEEE Trans. Inf. Theory*, (submitted) 2013.

[51] J. Hagenauer, "Rate-compatible punctured convolutional codes (rcpc codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[52] D. Rowitch and L. Milstein, "On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes," *Communications, IEEE Transactions on*, vol. 48, no. 6, pp. 948–959, 2000.

[53] R. Liu, P. Spasojevic, and E. Soijanin, "Punctured turbo code ensembles," in *Information Theory Workshop, 2003. Proceedings. 2003 IEEE*, Mar. 2003, pp. 249–252.

[54] R. G. Gallager, "Low-density parity-check codes," Ph.D. dissertation, MIT, Cambridge, MA, 1963.

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

[56] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[57] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit errorcorrecting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Commun.*, Geneva, Switzerland, May 1993.

[58] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, pp. 585–598, Feb. 2001.

[59] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," vol. 47, No.2, pp. 618–637, Feb. 2001.

[60] S. Sesia, G. Caire, and G. Vivier, "Incremental redundancy hybrid ARQ schemes based on low-density parity-check codes," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1311–1321, 2004.

[61] J. Ha, J. Kim, and S. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2824–2836, 2004.

[62] J. Ha, J. Kim, D. Klinc, and S. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 728–738, 2006.

[63] M. El-Khamy, J. Hou, and N. Bhushan, "Design of rate-compatible structured LDPC codes for hybrid ARQ applications," *Selected Areas in Communications, IEEE Journal on*, vol. 27, no. 6, pp. 965–973, 2009.

[64] J. Kim, A. Ramamoorthy, and S. McLaughlin, "The design of efficiently-encodable rate-compatible LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 2, pp. 365–375, 2009.

[65] B. Vellambi and F. Fekri, "Finite-length rate-compatible LDPC codes: a novel puncturing scheme," *Communications, IEEE Transactions on*, vol. 57, no. 2, pp. 297–301, 2009.

[66] M. Yazdani and A. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *Communications Letters, IEEE*, vol. 8, no. 3, pp. 159–161, 2004.

[67] S. Dolinar, "A rate-compatible family of protograph-based LDPC codes built by expurgation and lengthening," in *Proc. International Symposium on Information Theory*, 2005, pp. 1627–1631.

[68] J. Li and K. Narayanan, "Rate-compatible low density parity check codes for capacity-approaching ARQ schemes in packet data communications," in *Proc. Int. Conf. on Comm., Internet, and Info. Tech. (CIIT)*, Nov. 2002.

[69] N. Jacobsen and R. Soni, "Deign of rate-compatible irregular LDPC codes based on edge growth and parity splitting," in *Proc. IEEE Vehicular Technology Conference (VTC)*, Baltimore, MD, USA, Oct. 2007.

[70] S. Song, D. Hwang, S. Seo, and J. Ha, "Linear-time encodable rate-compatible punctured LDPC codes with low error floors," in *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, 2008, pp. 749–753.

[71] T.-Y. Chen, D. Divsalar, J. Wang, and R. D. Wesel, "Protograph-based raptor-like LDPC codes for rate-compatibility with short blocklengths," in *Proc. IEEE Global Communications Conference*, Houston, TX, Dec. 2011.

[72] T.-Y. Chen, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like LDPC codes with low thresholds," in *Proc. IEEE International Conference of Communications*, Ottawa, Canada, Jun. 2012.

[73] T. Nguyen, A. Nosratinia, and D. Divsalar, "The design of rate-compatible protograph LDPC codes," *IEEE Trans. Commun.*, vol. 60, no. 10, pp. 2841–2850, 2012.

[74] T. V. Nguyen and A. Nosratinia, "Rate-compatible short-length protograph LDPC codes," *Communications Letters, IEEE*, vol. 17, no. 5, pp. 948–951, 2013.

[75] J. Thorpe, "Low Density Parity Check (LDPC) codes constructed from protographs," *JPL IPN Progress Report*, vol. 42–154, Aug. 2003.

[76] D. Divsalar, S. Donlinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, No. 6, pp. 876–888, Aug. 2009.

[77] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[78] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002.

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

[80] W. Nitzold, M. Lentmaier, and G. Fettweis, "Spatially coupled protograph-based LDPC codes for incremental redundancy," in *Turbo Codes and Iterative Information Processing (ISTC), 2012 7th International Symposium on*, 2012, pp. 155–159.

[81] A. Jimenez Felstrom and K. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *Information Theory, IEEE Transactions on*, vol. 45, no. 6, pp. 2181–2191, 1999.

[82] "Low density parity check codes for use in near-earth and deep space. orange book." *CCSDS standard, Issue No.2*, Sep. 2007.

[83] "Digital video broadcasting; second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications," *ETSI EN 302 307 V1.2.1*, Aug. 2009.

[84] Z. Li and B. V. Kumar, "A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2. IEEE, 2004, pp. 1990–1994.

[85] J. Garcia-Frias and W. Zhong, "Approaching shannon performance by iterative decoding of linear codes with low-density generator matrix," *IEEE Commun. Lett.*, vol. 7, no. 6, pp. 266–268, June 2003.

[86] A. Eckford, J. Chu, and R. Adve, "Low-complexity cooperative coding for sensor networks using rateless and LDGM codes," in *IEEE International Conference on Communications, 2006. ICC '06.*, vol. 4, June 2006, pp. 1537–1542.

[87] K. Vakilinia, T.-Y. Chen, S. V. S. Ranganathan, A. R. Williamson, D. Divsalar, and R. D. Wesel, "Short-blocklength non-binary LDPC codes with feedback-dependent incremental transmissions," in *Proc. 2014 IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, USA, July 2014.

[88] ——, "Feedback systems using non-binary ldpc codes with a limited number of transmissions," in *Proc. 2014 IEEE Inf. Theory Workshop (ITW)*, Hobart, Tasmania, Australia, November 2014.

[89] E. Soljanin, "Punctured vs. rateless codes for hybrid ARQ," in *Proc. IEEE Inform. Theory Workshop '06*, Punta del Este, Uruguay, Mar 2006.

[90] A. Venkiah, C. Poulliat, and D. Declercq, "Analysis and design of raptor codes for joint decoding using information content evolution," in *Proc. International Symposium on Information Theory (ISIT)*, Jun. 2007, pp. 421–425.

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

[92] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge University Press, Mar. 2008.

[93] R. G. Gallager, "Low density parity check codes," Ph.D. dissertation, Massachusetts Institute of Technology, 1960.

[94] C. Di, T. Richardson, and R. Urbanke, "Weight distribution of low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 52, no. 11, pp. 4839–4855, Nov 2006.

[95] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of ldpc code ensembles," *Information Theory, IEEE Transactions on*, vol. 51, no. 3, pp. 929–953, March 2005.

[96] D. Burshtein and G. Miller, "Asymptotic enumeration methods for analyzing ldpc codes," *Information Theory, IEEE Transactions on*, vol. 50, no. 6, pp. 1115–1131, June 2004.

[97] ——, "Asymptotic enumeration methods for analyzing ldpc codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1115–1131, Jun. 2004.

[98] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 929–953, March 2005.

[99] C.-H. Hsu and A. Anastasopoulos, "Capacity-achieving codes with bounded graphical complexity and maximum likelihood decoding," *IEEE Trans. Inf. Theory*, vol. 56, no. 3, pp. 992–1006, March 2010.

[100] D. Divsalar, C. Jones, S. Dolinar, and J. Thorpe, "Protograph based LDPC codes with minimum distance linearly growing with block size," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 3, Nov 2005.

[101] T. Tian, C. Jones, J. Villasenor, and R. Wesel, "Selective avoidance of cycles in irregular ldpc code construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247, Aug 2004.

[102] Y. Polyanskiy, H. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," vol. 56, no. 5, pp. 2307 –2359, May. 2010.

[103] 3rd Generation Partnership Project, "3GPP TS 36.212 Multiplexing and channel coding," *(http://www.3gpp.org)*, vol. Release 8, Mar. 2008.

[104] G. Liva, E. Paolini, and M. Chiani, "On optimum decoding of certain product codes," *Communications Letters, IEEE*, vol. 18, no. 6, pp. 905–908, June 2014.

[105] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, "Short turbo codes over high order fields," *Communications, IEEE Transactions on*, vol. 61, no. 6, pp. 2201–2211, 2013.

[106] G. Liva, B. Matuz, E. Paolini, and M. Chiani, "Achieving a near-optimum erasure correction performance with low-complexity ldpc codes," *Int. J. Satell. Commun. Network.*, vol. 28, no. 3-4, pp. 236–256, Oct. 2010.

[107] E. Paolini, G. Liva, B. Matuz, and M. Chiani, "Maximum likelihood erasure decoding of ldpc codes: Pivoting algorithms and code design," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3209–3220, Nov. 2012.

[108] H. Pfister and I. Sason, "Accumulate-repeat-accumulate codes: Capacity-achieving ensembles of systematic codes for the erasure channel with bounded complexity," *Information Theory, IEEE Transactions on*, vol. 53, no. 6, pp. 2088–2115, June 2007.

[109] M. Lentmaier, M. Tavares, and G. Fettweis, "Exact erasure channel density evolution for protograph-based generalized ldpc codes," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, June 2009, pp. 566–570.

[110] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *Information Theory, IEEE Transactions on*, vol. 50, no. 3, pp. 439–454, March 2004.

[111] C. Measson, A. Montanari, and R. Urbanke, "Maxwell construction: The hidden bridge between iterative and maximum a posteriori decoding," *Information Theory, IEEE Transactions on*, vol. 54, no. 12, pp. 5277–5307, Dec 2008.

[112] C. W. Wang and H. Pfister, "Upper bounds on the map threshold of iterative decoding systems with erasure noise," in *Turbo Codes and Related Topics, 2008 5th International Symposium on*, Sept 2008, pp. 7–12.

[113] Z. Li and B. Kumar, "A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2, Nov 2004, pp. 1990–1994 Vol.2.

[114] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *Information Theory, IEEE Transactions on*, vol. 50, no. 11, pp. 2657–2673, Nov 2004.

[115] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge University Press, 2008.

[116] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 616–637, Feb. 2001.

[117] E. Yaakobi, L. Grupp, P. Siegel, S. Swanson, and J. Wolf, "Characterization and error-correcting codes for tlc flash memories," in *Proc. of ICNC*, Jan. 2012, pp. 486–491.

[118] R. Gabrys, E. Yaakobi, and L. Dolecek, "Graded bit error correcting codes with applications to flash memory," *IEEE Trans. Inform. Theory*, vol. 59, no. 4, pp. 2315–2327, Apr. 2013.

[119] F. Zhang, H. Pfister, and A. Jiang, "Ldpc codes for rank modulation in flash memories," in *Proc. of IEEE ISIT*, 2010, pp. 859–863.

[120] A. Jiang, M. Schwartz, and J. Bruck, "Error-correcting codes for rank modulation," in *Proc. of IEEE ISIT*, 2008, pp. 1736–1740.

[121] A. Mazumdar, A. Barg, and G. Zemor, "Constructions of rank modulation codes," in *Proc. of IEEE ISIT*, 2011, pp. 869–873.

[122] M. Qin, A. Jiang, and P. Siegel, "Parallel programming of rank modulation," in *Proc. of IEEE ISIT*, 2013, pp. 719–723.

[123] F. Sala, R. Gabrys, and L. Dolecek, "Dynamic threshold schemes for multi-level non-volatile memories," *IEEE Trans. Comm*, vol. 61, no. 7, pp. 1660–1673, May 2013.

[124] J. Wang, T. Courtade, H. Shankar, and R. Wesel, "Soft Information for LDPC Decoding in Flash: Mutual-Information Optimized Quantization," in *Proc. IEEE GLOBECOM*, Houston, TX, Dec. 2011.

[125] J. K.-S. Lee and J. Thorpe, "Memory-Efficient Decoding of LDPC Codes," in *Proc. of IEEE ISIT*, 2005, pp. 459–463.

[126] B. M. Kurkoski and H. Yagi, "Quantization of Binary-Input Discrete Memoryless Channels, with Applications to LDPC Decoding," *Submitted to IEEE Trans. Inform. Theory. Available http://arxiv.org/abs/1107.5637.*

[127] J. Bruck, A. Vardy, E. Yaakobi, J. Bruck, P. Siegel, A. Vardy, and J. Wolf, "Storage coding for wear leveling in flash memories," in *Proc. of IEEE ISIT*, 2009, pp. 1229–1233.

[128] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to Flash Memory," *Proc. IEEE*, vol. 91, no. 4, pp. 489–502, April 2003.

[129] Y. Maeda and K. Haruhiko, "Error Control Coding for Multilevel Cell Flash Memories Using Nonbinary Low-Density Parity-Check Codes," in *24th IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, Oct. 2009, pp. 367–375.

[130] S. Li and T. Zhang, "Improving Multi-Level NAND Flash Memory Storage Reliability Using Concatenated BCH-TCM Coding," *IEEE Trans. VLSI Systems*, vol. 18, no. 10, pp. 1412–1420, Oct. 2010.

[131] Q. Wu, G. Dong, and T. Zhang, "Exploiting Heat-Accelerated Flash Memory Wear-Out Recovery to Enable Self-Healing SSDs," in *USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)*, June 2011.

[132] G. Dong, N. Xie, and T. Zhang, "On the Use of Soft-Decision Error-Correcting Codes in NAND Flash Memory," *IEEE Trans. Circ. and Sys.*, vol. 58, no. 2, pp. 429–439, Feb. 2011.

[133] T. M. Cover and J. A. Thomas, *Elements of Information Theory, second Edition*. Wiley, Jul. 2006.

[134] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[135] J. Wang, T. Courtade, H. Shankar, and R. Wesel, "Soft information for ldpc decoding in flash: Mutual-information optimized quantization," in *GLOBECOM*, Dec 2011, pp. 1–6.

[136] J. Wang, K. Vakilinia, T.-Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel, "Enhanced precision through multiple reads for LDPC decoding in flash memories," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 5, pp. 880–891, May 2014.

[137] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[138] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Selective Avoidance of Cycles in Irregular LDPC Code Construction," *IEEE Trans. Comm.*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.

[139] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE GLOBECOM*, San Antonio, TX, Feb. 2001, pp. 995–1001 vol.2.

[140] A. Ramamoorthy and R. D. Wesel, "Construction of Short Block Length Irregular LDPC Codes," in *Proc. IEEE ICC*, Paris, France, June. 2004.

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

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

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

[144] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 549–583, Feb 2006.

[145] G. Ungerboeck, "Channel coding with multilevel/phase signals," *Information Theory, IEEE Transactions on*, vol. 28, no. 1, pp. 55–67, Jan 1982.