2019

Raptor Codes for BIAWGN Channel: SNR Mismatch and the Optimality of the Inner and Outer Rates

Hussein Fadhel
University of Mississippi

RAPTOR CODES FOR BIAWGN CHANNEL: SNR MISMATCH AND OPTIMALITY OF THE INNER AND OUTER RATES

A Dissertation
presented in partial fulfillment of requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering
The University of Mississippi

by

HUSSEIN FADHEL

May 2019

ABSTRACT

Fountain codes are a class of rateless codes with two interesting properties, first, they can generate potentially limitless numbers of encoded symbols given a finite set of source symbols, and second, the source symbols can be recovered from any subset of encoded symbols with cardinality greater than the number of source symbols. Raptor codes are the first implementation of fountain codes with linear complexity and vanishing error floors on noisy channels. Raptor codes are designed by the serial concatenation of an inner Luby transform (LT) code, the first practical realization of fountain codes, and an outer low-density parity-check (LDPC) code. Raptor codes were designed to operate on the binary erasure channel (BEC), however, since their invention they received considerable attention in order to improve their performance on noisy channels, and especially additive white Gaussian noise (AWGN) channels. This dissertation considers two issues that face Raptor codes on the binary input additive white Gaussian noise (BIAWGN) channel: inaccurate estimation of signal to noise ratio (SNR) and the optimality of inner and outer rates. First, for codes that use a belief propagation algorithm (BPA) in decoding, such as Raptor codes on the BIAWGN channel, accurate estimation of the channel SNR is crucial to achieving optimal performance by the decoder. A difference between the estimated SNR and the actual channel SNR is known as signal to noise ratio mismatch (SNRM). Using asymptomatic analysis and simulation, we show the degrading effects of SNRM on Raptor codes and observe that if the mismatch is large enough, it can cause the decoding to fail. Using the discretized density evolution (DDE) algorithm with the modifications required to simulate the asymptotic performance in the case of SNRM, we determine the decoding threshold of Raptor codes for different values of SNRM ratio. Determining the threshold under SNRM enables us to quantify its effects which in turn can be used to reach important conclusions about the effects

of SNRM on Raptor codes. Also, it can be used to compare Raptor codes with different designs in terms of their tolerance to SNRM. Based on the threshold response to SNRM, we observe that SNR underestimation is slightly less detrimental to Raptor codes than SNR overestimation for lower levels of mismatch ratio, however, as the mismatch increases, underestimation becomes more detrimental. Further, it can help estimate the tolerance of a Raptor code, with certain code parameters when transmitted at some SNR value, to SNRM. Or equivalently, help estimate the SNR needed for a given code to achieve a certain level of tolerance to SNRM. Using our observations about the performance of Raptor codes under SNRM, we propose an optimization method to design output degree distributions of the LT part that can be used to construct Raptor codes with more tolerance to high levels of SNRM. Second, we study the effects of choosing different values of inner and outer code rate pairs on the decoding threshold and performance of Raptor codes on the BIAWGN channel. For concatenated codes such as Raptor codes, given any instance of the overall code rate R, different inner (R_i) and outer (R_o) code rate combinations can be used to share the available redundancy as long as $R = R_i R_o$. Determining the optimal inner and outer rate pair can improve the threshold and performance of Raptor codes. Using asymptotic analysis, we show the effect of the rate pair choice on the threshold of Raptor codes on the BIAWGN channel and how the optimal rate pair is decided. We also show that Raptor codes with different output degree distributions can have different optimal rate pairs, therefore, by identifying the optimal rate pair we can further improve the performance and avoid suboptimal use of the code. We make the observation that as the outer rate of Raptor codes increases the potential of achieving better threshold increases, and provide the reason why the optimal outer rate of Raptor codes cannot occur at lower values.

Finally, we present an optimization method that considers the optimality of the inner and outer rates in designing the output degree distribution of the inner LT part of Raptor codes. The designed distributions show improvement in both the decoding threshold and performance compared to other code designs that do not consider the optimality of the inner and outer rates.

ACKNOWLEDGEMENTS

First and most I would like to thank Dr. Lei Cao for his guidance and much appreciated support throughout the program. His knowledge and intelligence always shed light on the path in my journey to learning. For his kindness, generous guidance and extraordinary patient encouragement I am genuinely grateful.

My sincerest thanks go to my dissertation prospectus defense committee; Dr. Ramanarayanan "Vish" Viswanathan, Dr. John N. Daigle, and Dr. Feng Wang. For your time and kind support I am always grateful.

This work was supported in part by NASA EPSCoR program under the grants NNX13AB31A and NNX14AN38A.

Last but not the least, I am deeply grateful for all the support and love my family have graced me with over the years. Their love has always enlightened my path through life and for that I am forever thankful.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AWGN additive white Gaussian noise

BEC binary erasure channel

BER bit error rate

BPA belief propagation algorithm

BPSK binary phase shift keying

BIAWGN binary input additive white Gaussian noise

CN check node

DE density evolution

DDE discretized density evolution

EXIT extrinsic information transfer

FEC forward error correction

GA Gaussian approximation

GF Galois field

LDPC low-density parity-check

LR likelihood ratio

LLR log-likelihood ratio

LT Luby transform

PMF probability mass function

RSD robust Soliton distribution

SNR signal to noise ratio

SNRM signal to noise ratio mismatch

SPA sum-product algorithm

VN variable node

CHAPTER 1

INTRODUCTION

1.1 Background

Fountain codes [1] are a class of rateless codes. For a given block of data of k input symbols, a rateless code can generate a potentially limitless number of encoded symbols n, where, the source symbols can be bits or blocks of data. Fountain codes, in general, have simple encoding and decoding algorithms based on the sparse nature of their coding graphs, and on the binary erasure channel (BEC) the decoder can recover the original data by receiving any subset of encoded symbols with cardinality slightly higher than the number of the source symbols k. Raptor codes [2] are a class of fountain codes, they are an extension of Luby transform (LT) codes [3] which are the first practical realization of fountain codes. While LT codes have nonlinear encoding and decoding costs [3] and suffer from relatively high error floors on noisy channels, Raptor codes overcome these issues by using a linear block code as a precode to encode source symbols before the inner LT encoder [2]. The precode can be a concatenation of multiple codes, usually, an LDPC code [4] [5] coupled with another linear code such as Hamming code [6]. Due to their rateless nature, fountain codes have the ability to adapt to changes in the channel condition that block codes may fail in adapting to. Raptor codes combine the advantages of both types of codes, namely block codes and fountain codes, to produce a class of fountain codes with linear encoding and decoding costs and the flexibility of rateless codes in adapting channel conditions.

The main structure of Raptor code is a serial concatenation of an outer LDPC code and an inner LT code as shown in Figure 1.1 below.

Given an instance of the code rate R, the two codes share the available redundancy

Figure 1.1. Structure of Raptor codes.

such that any inner code rate R_i and the outer code rate R_o can be chosen as long as the relationship $R = R_i R_o$ is held. However, careful design of the inner and outer codes and selecting the optimal code rate pair (R_i, R_o) must be accomplished in order to optimize the overall performance of Raptor codes.

Raptor codes were originally designed to operate on the BEC channel [2] and usually as a part of the application layer. The code offers an impressive universally capacity achieving performance on the BEC channel. Due to their desired rateless nature and linear complexity, since the introduction of Raptor code, there have been continuous and numerous efforts to improve and optimize their performance on noisy channels and in particular the binary input additive white Gaussian noise (BIAWGN) channel [7–12].

Raptor codes encounter different issues that affect their performance on additive white Gaussian noise (AWGN) channels which need to be considered in order to achieve successful and efficient use of Raptor codes. signal to noise ratio mismatch (SNRM) is one issue that affects codes operating on the AWGN channel. SNRM describes the condition when on the receiver side of the communication system the value of the signal to noise ratio (SNR) of the received information is estimated incorrectly. If the estimated channel SNR is higher than the actual value, this condition is known as SNR overestimation, and if the estimated channel SNR is less than the actual value, this condition is known as SNR underestimation. Codes such as LDPC and Raptor codes that use a belief propagation algorithm (BPA), also known as sum-product algorithm (SPA), are negatively impacted by SNRM because the incorrectly estimated SNR information is used by the BPA in the decoding process. SNRM can degrade the decoding process and if the difference between the estimated and actual SNR is large enough, then, the decoding process will fail completely.

Studying and determining the tolerance limits of codes to SNRM and implementing modifications in their design or algorithms in order to make the codes more tolerant to SNRM is crucial for successful decoding in cases where mismatch is expected.

Codes that are defined by Tanner graphs [13], such as LDPC and Raptor codes, can counteract the effects of SNRM through two approaches: (a) using modified versions of the BP decoding algorithm that do not use the channel SNR information in decoding such as min-sum [14] or one of its derivatives [15–17], however, this leads to degrade the performance of such decoding algorithms compared to BP decoding in terms of decoding threshold, i.e. the minimum SNR value below which the decoder cannot correct all errors and the decoding performance, (b) using degree distributions that are designed to have higher tolerance to SNRM, however, although such distributions show higher tolerance to SNRM compared to other distributions, this causes their performance to be less than optimal when SNRM is not present [18].

Another factor that affects the performance of Raptor codes on the BIAWGN channel is the choice of the inner and outer code rates (R_i, R_o). Given an instance of the overall code rate R, different combinations of the inner and outer rates can be used to produce $R = R_i R_o$. However, due to the specific nature of the error floor of the inner LT code, different code rate pairs lead to varying performance. Determining the optimal rate pair (R_i, R_o) is another factor that can aid in optimizing the performance of Raptor codes.

1.2 Motivation and Objective

Considering the desire in the research community and industry to apply Raptor codes in a wide range of communication systems and scenarios, when operating on the AWGN channel Raptor codes can face the problem of SNRM. It is known that for codes that use BP decoding SNRM can considerably degrade the performance, especially when codes operate close to their threshold, and if the SNRM is high enough, the decoding process can completely fail. Therefore, the need to study the performance of Raptor codes in the presence of SNRM

is evident. One way to quantify the effects of SNRM on codes is to determine the SNR threshold for each value of SNRM ratio, where the SNRM ratio is defined as the ratio of estimated SNR to the true channel SNR. This helps identify while operating at a given SNR how much tolerance to SNRM to be expected. From the other perspective, aiming for a certain level of SNRM tolerance for the code in use, we can assign the needed SNR. Also, being able to determine the threshold of Raptor codes for a range of SNRM ratio values can be used to compare Raptor codes whose output degree distributions are different in terms of the tolerance to SNRM.

In order to determine the threshold of codes that use the SPA in decoding, including Raptor codes, the density evolution (DE) algorithm [19] or less complex, but also less accurate, algorithms such as Gaussian approximation (GA) [20] or extrinsic information transfer (EXIT) charts [21] can be used. These algorithms can simulate the asymptotic behavior of the SPA and are used to determine the thresholds of the considered codes. Due to the high complexity of the DE algorithm, GA or EXIT charts are often used in the analysis. However, in order to use GA or EXIT chart algorithms to simulate the performance of codes under SNRM, the algorithms must be modified to accommodate the changes that SNRM introduces to the distribution of the codeword received from the channel [18, 22, 23], while DE algorithm does not require such large scale modification.

The quantified version of the DE algorithm called discretized density evolution (DDE). DDE was developed for LDPC codes [24] and Raptor codes [12], and successfully used to analyze the asymptotic performance and design optimized LDPC and Raptor codes that operate close to the Shannon limit. Analyzing the suitability of the DDE algorithm to simulate the asymptotic performance of concatenated codes such as Raptor codes under the condition of SNRM and identifying the necessary modifications in the algorithm to work accurately, and then implementing them are crucial steps towards analyzing the asymptotic performance of Raptor codes under SNRM and designing optimized Raptor codes that are more tolerant to SNRM.

Another aspect of Raptor codes that can be used to optimize the performance is choosing the optimal inner and outer code rate pair (R_i, R_o). Generally, in the literature, whether in using or design of Raptor codes, the outer code rate is fixed as a constant value, usually 50/51, or ignored in the process of searching for optimized output degree distributions. Fixing the outer rate equal to 50/51 is equivalent to setting the inner and outer rate pair as $(R/(50/51), 50/51)$, however, it is not always the case that this rate pair provides the optimal performance for the particular Raptor code in use, or that the optimization process will return the optimal code possible at it.

Studying the effects of choosing a certain rate pair on the threshold of Raptor codes helps in determining the optimal inner and outer rates for a given Raptor code, and therefore, more efficient utilization of the code. On the other hand, designing output degree distributions for the inner code that consider the optimality of the inner and outer rates in the optimization process produces Raptor codes that exhibit further improvement in performance and threshold compared to codes designed without considering this issue.

The main objective of our work is to address two issues that Raptor codes encounter on the BIAWGN channel: (a) SNR mismatch, (b) the optimality of the inner and outer code rates. We study the asymptotic performance of Raptor codes under SNRM and present our observations and conclusions about the effects of this condition on Raptor codes. Then, taking advantage of what we learned about the behavior of Raptor codes in the presence of SNRM, we propose an optimization approach to design output degree distributions of the inner LT code that can be used to construct Raptor codes with more tolerance to higher levels of SNRM. The second issue we consider is the optimality of the inner and outer rate pair of Raptor codes. Using asymptotic analysis, we demonstrate the effect of inner and outer rates on determining the decoding threshold. We observe that different Raptor codes have different optimal rate pairs, therefore, identifying the optimal rate pair for the code in use is more efficient than using a fixed rate pair for all Raptor codes. Finally, we propose an optimization method that considers the optimality of the inner and outer rates in the design

process and test the performance of our optimized distributions.

1.3 Contributions and Organization

In chapter 2 we discuss the structure of LDPC and LT codes and their encoding and decoding algorithms. These two codes are serially concatenated to design Raptor codes. Understanding their structure and properties will help in introducing Raptor codes. Chapter 3 is assigned to discuss Raptor codes, how they are designed, their properties, and their advantages over other fountain codes.

In chapter 4, we define the concept of SNRM, how it occurs, and how it is mathematically formulated so that its effects on channel codes can be studied. We present a brief literature review of the effects of SNRM on Turbo and LDPC codes and take a look at the proposed solutions to design these codes such that they become more tolerant to SNRM. Then, we turn our attention to study the effects of SNRM on Raptor codes. We review the DDE algorithm of Raptor codes and show that the DDE algorithm can be used to correctly simulate the asymptotic performance of BP decoding of Raptor codes under SNRM condition. We show that BP decoding does not affect the SNRM ratio of the decoded data, and explain the effect of this property on serially concatenated codes such as Raptor codes. Having established the groundwork to use the DDE algorithm to study the effects of SNRM on Raptor codes, we apply few modifications to the algorithm and use it to determine the decoding threshold of a given Raptor code for a range of SNRM ratio values. The threshold performance of Raptor codes under SNRM shows that for lower values of mismatch ratio, SNR underestimation and overestimation have a similar effect on Raptor codes with overestimation being slightly more detrimental. For higher values of SNRM, SNR underestimation is more detrimental compared to overestimation, a property that was previously observed in LDPC and Turbo codes. Determining the threshold under SNRM of a Raptor code can be used to estimate how much SNRM tolerance to expect for a given channel SNR value. Equivalently, it can help in estimating the SNR needed to ensure a certain level of tolerance

to SNRM. Also, comparing the thresholds of different Raptor codes for a range of SNRM ratios, we can recognize which codes are comparably more tolerant to SNRM.

In chapter 5 we formulate a DDE based optimization approach to design Raptor codes that are more tolerant to SNRM by optimizing the output degree distribution of the inner LT code. Using our knowledge about the performance and properties of Raptor codes under SNRM, we devise an optimization program for the special case of SNRM. Our optimized distributions show improved thresholds and better tolerance at higher levels of SNRM, however, that comes with a loss in the threshold for lower levels of SNRM and when no SNRM exists.

In chapter 6 we study the effects of inner and outer code rates choice on Raptor codes. We study how each rate pair can affect the threshold, and how to determine the optimal code rate pair for a given Raptor code at a certain instance of the overall code rate. For a Raptor code with a given output degree distribution, we show how to determine the optimal rate pair for two cases: assuming the use of a regular LDPC code outer code, and a capacity achieving outer code. This shows whether the use of a regular LDPC code as it is customary in literature is optimal or further improvement can be made through the use of better outer code. We study the effect of relatively high error floors LT codes have on the threshold of Raptor codes and conclude that choosing higher values of outer rates can help achieve better thresholds. Finally, we design an optimization algorithm that incorporates choosing the optimal code rate pair in the optimization process.

CHAPTER 2

LT and LDPC CODES

In this chapter we will study the structure of LDPC [4] and LT codes [3]. These two codes are serially concatenated to form Raptor codes [2]. Introducing these two codes will help us understand the design and operating principles of Raptor codes discussed in the next chapter. LT codes are the first practical realization of fountain codes [1]. LT codes have nonlinear encoding and decoding costs and suffer from relatively high error floors on noisy channels. Raptor codes overcome these issues by using a series concatenation of a linear precode $\mathcal{C}$, usually of high rate, with an inner LT code [2]. Fountain codes have the ability to adapt to changes in the channel characteristics that block codes may fail in adapting to because of the rateless nature of fountain codes. Raptor codes combine the advantages of both block and fountain codes to produce a class of rateless codes with linear encoding and decoding costs and vanishing error floors [8]. The precode of Raptor codes can be a concatenation of multiple codes, usually, the prominent class of linear block codes known as LDPC codes [4]- [5] coupled with another code such as Hamming code [6].

2.1 LT Codes

Fountain codes [1] are a class of rateless codes, for a given k source or input symbols, the code can generate potentially a limitless number n of encoded output symbols, the source symbols can be bits or block of data of equal size. Fountain codes have simple encoding and decoding algorithms based on sparse graphs and the decoder can recover the original data after receiving any subset of encoded symbols with cardinality slightly higher than k, i.e. the size of the source block, on the BEC channel. Fountain code applications, e.g. LT and

Raptor codes, are universally capacity approaching i.e. they approach Shannon's limit for any erasure channel.

The name of fountain codes originates from the fact that their encoder can be visualized as a fountain with endless water drops (output symbols), and a receiving end can continue collecting these drops until the original file is recovered [25].

Traditional block codes can be inefficient in terms of their use of the available resources, if the communicating devices are unable to detect the channel parameters continuously and choose the code rate accordingly, then, the code rate used can become inefficient, either using unnecessary redundancy, or using a higher rate than what is necessary to achieve reliable transmission. This can be more advent with large scale communication such as multicast or satellite communications where monitoring the condition of every channel to assign the proper rate is unrealistic. Therefore, the worst case error rate is assumed which may be wasteful on some channels and still may fail on others due to sudden changes in the channel characteristics. Fountain codes can deal with such issues more efficiently due to their rateless nature that can be utilized to adopt to channel condition more dynamically.

LT codes [3] can be used to encode a given source data of size k symbols, where each symbol can be one bit or an arbitrary l-bit long symbol, such that "each output symbol can be generated, independently of all other output symbols, on average by $O(\ln(k/\delta))$ symbol operations, and the k original input symbols can be recovered from any $k + O(\sqrt{k} \cdot \ln^2(k/\delta))$ output symbols with probability $1 - \delta$ by on average $(k \cdot \ln(k/\delta))$ symbol operations" [3]. Where, δ is the failure probability of the decoder to recover the original data from any k output symbols, and a symbol operation is either an exclusive-or operation between two symbols or copying one symbol to another.

2.1.1 LT Encoding

The LT encoding process is fairly easy to describe:

1. Randomly choose the degree d of the output symbol from the output degree distri-

bution. The degree of an encoded symbol is the number of input symbols used in generating the output symbol.

2. Uniformly at random choose d distinct input symbols.

3. The value of the output symbol is exclusive-or of the d input symbols when $(d > 1)$ and is simply a copy of the input symbol when $(d = 1)$.

The output degree distribution sampled to choose the degree of the output symbols has a major role in the failure or success of the decoding process of LT codes, and its design and effects will be further discussed later.

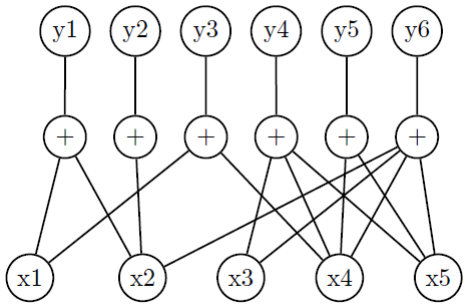

Figure 2.1. LT encoding.

Figure 2.1 is a bipartite graph illustrating a toy example of encoding 5 input symbols x1, x2, . . , x5 to generate 6 output symbols y1, y2, . . ,y6. For example, output symbol y1 = x1⊕x2 is a degree-2 output symbol that is connected to two input nodes known as "neighbors". y2 = x2 is degree-1 and has only one neighbor.

After transmission, the decoder receives the encoded symbols with added distortion based on the channel type and its characteristics. The decoder needs to know the degree and set of neighbors of each output symbol. This information can be delivered to the decoder by:

- Attaching the degree and set of neighbors of output symbols to the packet carrying them.

- The decoder computing the required information for each output symbol implicitly based on the timing of reception, or the position of the output symbol relative to the other symbols.

- Passing a key which can be used as a seed to a pseudo-random generator in order to reconstruct the degree and set of neighbors of each output symbol. The software used in our simulations uses this method.

When decoding starts, the decoder knows the degree and set of input symbols used to construct each output symbol. The decoding objective is finding the correct value of each input symbol.

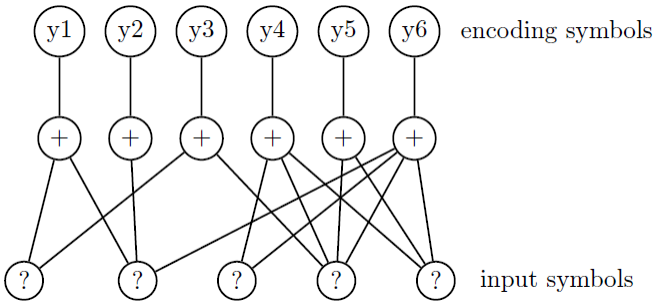

Figure 2.2. LT decoding.

2.1.2 LT Decoding

To understand the mechanism in which LT decoding operates we need to keep in mind that each output symbol of degree $d > 1$ is the XOR-sum, or addition on Galois field (GF) $\mathbb{F}_2$, of its input symbol neighbors and generating a degree one output symbol is simply the process of copying a randomly chosen input symbol to the output symbol. Also, for the sake of simplicity, we consider working on the BEC channel.

When decoding starts, degree one output symbols are released to cover the input symbol they contains. Now, these input symbols can be subtracted from all the output symbols that they are a neighbor of, and the degree of each neighbor can be decremented

by one since the decoder identified the value of this input symbol. The subtraction is done by XORing between the corresponding symbols since subtraction is equivalent to addition on $\mathbb{F}_2$. One way to visualize the decoding process from the perspective of bipartite graphs of LT codes, decoding is the process of removing edges connecting output symbols to source symbols that have been recovered until an output symbol has one edge left, i.e. it is degree-1, in which case it contains the value of the source symbol it is connected to and can be released to the ripple (the set of output symbols reduced to degree one). In Figure 2.2, y2 is a degree one output symbol which can be released to cover x2, which in turn can reduce y2 to degree-1. This process continues for every output symbol of degree $d > 1$ until it decreases to degree 1, in which case it can be released to cover the input symbol it contains if it has not been covered yet. Having that clarified, we can present two definitions from Luby's paper "LT Codes" [3].

Definition 1 (decoder recovery rule): If there is at least one output symbol that has exactly one neighbor, then, the neighbor can be recovered immediately since it is a copy of the output symbol. The value of the recovered input symbol is exclusive-ORed into any remaining output symbols that also have that input symbol as a neighbor, the recovered input symbol is removed as a neighbor from each of these output symbols and the degree of each such output symbol is decreased by one to reflect this removal.

Definition 2 (LT process): All input symbols are initially not covered (decoded). At the first step, all output symbols with one neighbor are released to cover their unique neighbor. The set of covered input symbols that have not yet been processed is called the ripple, and thus at this point all covered input symbols are in the ripple. At each subsequent step, once an input symbol in the ripple is processed, it is removed as a neighbor from all output symbols it is connected to. Gradually, the output symbols that become reduced to have exactly one remaining neighbor, will be released to cover the remaining neighbor. Some of these covered neighbors (input symbols) may have not been previously covered, and will cause the ripple to grow. On the other hand, other covered input symbols may have already

been in the ripple, therefore, will not contribute to any growth in the ripple. The process stops when the ripple is empty and will be considered failed, if there is at least one input symbol that could not be covered. The process succeeds if all input symbols are covered by the end.

Definition 3 introduces the term (*ripple*) which is the set of degree one released output symbols, or covered input symbols which have not been processed yet, i.e., have not been subtracted from neighboring output symbols yet. From the definition, we can understand the significance of the ripple in the success of the decoding process. If the ripple is empty before recovering all input symbols, then the decoding process fails. At the same time, it is advantageous to keep the size of the ripple small in order to avoid having redundant input symbols covered in the ripple. These requirements can be satisfied by proper analysis and careful design of the output degree distribution.

2.1.3 Design and Analysis of the Degree Distribution

Definition 4 (degree distribution): For all d, $\rho(d)$ is the probability that an output symbol has degree d [3].

The design of the degree distribution has the following two objectives:

1. Ensure the success of the LT process using as few output symbols as possible.

2. Keep the average degree of output symbols as small as possible.

Using the degree distribution $\rho(1) = 1$ which corresponds to choosing degree one for all output symbols, and encoding is done by simply choosing an input symbol at random and copying its value to the output symbol. The probability analysis of this case shows that an average of $k \cdot \ln \frac{k}{\delta}$ output symbols are needed to cover all input symbols at the decoder side with a probability of success $1 - \delta$. From this we can infer that for any distribution the sum of the degrees needed to recover k input symbols needs to be in the order of $k \cdot \ln \frac{k}{\delta}$, i.e. the average degree of the output symbols needs to be in the order of $\ln \frac{k}{\delta}$.

Before going into the details of the degree distribution $\rho(i)$, we first analyze the probability of releasing an output symbol of degree i when L input symbols remain unprocessed. Below, we state the definition of the degree release probability as given in [3].

Definition 5 (degree release probability): Let $q(i, L)$ be the probability that an output symbol of degree i is released when L input symbols remain unprocessed, then $q(i, L)$ is :

- $q(1, k) = 1$ for $i = 1$
- For $i = 2, ...k$, for all $L = k - i + 1, ..., 1$,

$$\frac{i(i-1) \cdot L \cdot \prod_{j=0}^{i-3} k - (L+1) - j}{\prod_{j=0}^{i-1} k - j} \tag{2.1}$$

- For all other i and L, $q(i, L) = 0$.

Definition 5 (overall release probability): Let $r(i, L)$ be the probability that an output symbol is chosen to be of degree i and is released when L input symbols remain unprocessed, i.e., $r(i, L) = \rho(i)\Delta q(i, L)$. Let $r(L)$ be the overall probability that an output symbol is released when L input symbols remain unprocessed, i.e., $r(L) = \sum_i r(i, L)$. Where, $\rho(i)$ is the output degree distribution.

The significance of the release probability will become evident later when we discuss the importance of the size of the ripple. Also, we will be able to value the importance of the degree distribution $\rho(i)$ in the LT process in terms of keeping the ripple at a desirable size.

2.1.4 Ideal Soliton Distribution

The name Soliton distribution comes from the analogy to the Soliton wave, a wave that travels at a constant speed while maintaining its amplitude due to its unique property of perfect balance between dispersion and refraction. A similar property is required in the output degree distribution in terms of keeping the ripple at a desirable size by ensuring that input symbols are added to the ripple at the same rate they are processed.

When designing the degree distribution there are two properties that impose the constraints in light of which the degree distribution is chosen :

1. Keep the ripple size small enough in order to avoid covering input symbols already in the ripple to avoid unnecessary redundancy.

2. Keep the ripple size large enough in order to ensure the ripple does not disappear before recovering all input symbols.

Analytically, the ideal Soliton distribution performs ideally in terms of the total number of output symbols needed to recover the source data and desirable ripple size, however, this distribution performs very poorly in practice for reasons we will be able to understand after some analysis.

Recall that the ripple is of the set of released output symbols which have not been processed yet, hence, it is directly affected by the release probability of the encoding process, which depends on the degree distribution $\rho(i)$. The release probability is equal to $r(i, L) = \rho(i) \cdot q(i, L)$

Using the argument, above we can use the analyses of the release probability as a tool to determine the expected size of the ripple using the Ideal Soliton distribution.

uniform release probability $r(L)$:

For the Ideal Soliton distribution, $r(L) = 1/k$ for all $L = k, \ldots, 1$.

Using the value of the release probability given above we can calculate the expected number of output symbols released, i.e., the ripple size, of the LT process for the Ideal Soliton distribution as:

$$k \cdot r(L) = k \cdot \sum_{i=1}^{k-L+1} r(i, L) = 1. \tag{2.2}$$

Assuming an ideal case, one output symbol is released for every input symbol processed, and exactly k output symbols are sufficient to recover k input symbols. However,

the ideal case is far from the actual case. The Ideal Soliton distribution performs very poorly in practice because the expected ripple size is one, and any variance can cause the ripple to decrease to zero, and cause the decoding process to fail. However, the ideal Soliton distribution gives an insight into the type of behavior required from the output degree distribution, and in fact, the Ideal Soliton distribution is modified to produce the robust Soliton distribution (RSD).

2.1.5 Robust Soliton Distribution

The modified distribution is designed based on two objectives: (1) the expected ripple size remains large enough throughout the decoding process, and (2) ripple size is not larger than necessary in order to avoid redundancy. The Robust Soliton distribution introduces two new parameters, δ, and c. Where, δ is the allowable failure probability of the decoder to recover the source data from any K output symbols, and c is a constant $c > 0$.

The Robust Soliton distribution modifies the release probability so that the expected size of the ripple is about $\ln(\frac{k}{\delta}\sqrt{k})$ by analogy to the case of a random walk of length k, where, the probability of deviating from the mean by more than $\ln(\frac{k}{\delta}\sqrt{k})$ has an upper bound of δ.

Definition 7 (Robust Soliton distribution): The Robust Soliton distribution $\mu(i)$ is defined as follows. Let $S = c \cdot \ln(k/\delta)\sqrt{k}$ for some suitable constant $c > 0$. Define

$$\tau(i) = \begin{cases} S/ik & \text{for } i = 1, ..., k/S - 1 \\ S\ln(S/\delta)/k & \text{for } i = k/S \\ 0 & \text{for } i = k/S + 1, ..., k \end{cases}$$

To obtain the robust Soliton distribution $\mu(i)$, add $\tau(i)$ to the Ideal Soliton distribution $\rho(i)$ and normalize as follows:

- $\beta = \sum_{i=1}^{k} \rho(i) + \tau(i)$

- for all $i = 1, ..., k, \mu(i) = (\rho(i) + \tau(i))/\beta$.

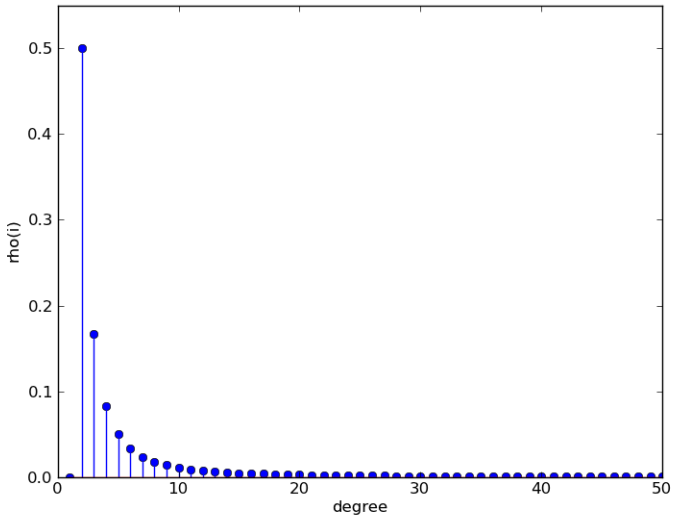

Figure 2.3. Ideal Soliton distribution for $k = 10000$.

Figure 2.3 illustrates the probability mass function of the Ideal Soliton distribution for $k = 10000$. The additional parameter $\tau(i)$ is shown in Figure 2.4 below. The small elevation in the beginning ensures that there are enough degree-1 output symbols for decoding to start, and the spike at k/S increases the probability that every source symbol is covered. Figure 2.5 illustrates the Robust Soliton distribution for $k = 10000, c = 0.3$, and $\delta = 0.05$.

The discussion above gives us an idea about the importance of the output degree distribution used and its role in the success or failure of the recovering process of input symbols. The output degree distribution of LT and Raptor codes play a significant role in determining their asymptotic and finite performance. Careful design of the output degree distribution is needed in order to achieve the desired performance.

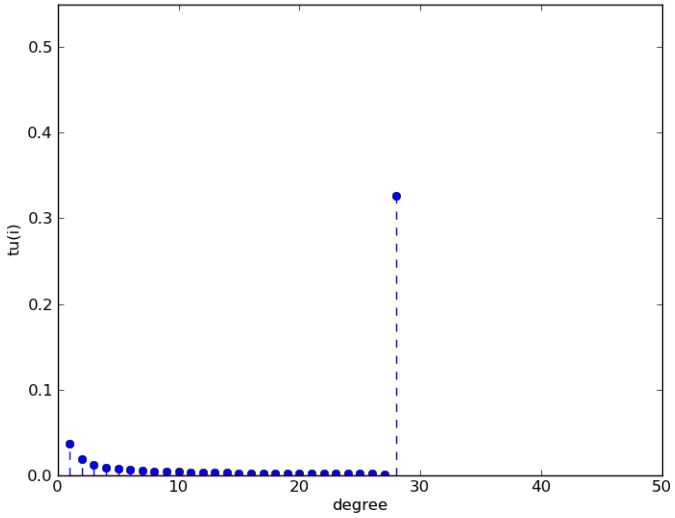

Figure 2.4. $\tau(i)$ for $c = 0.3$ and $\delta = 0.05$.

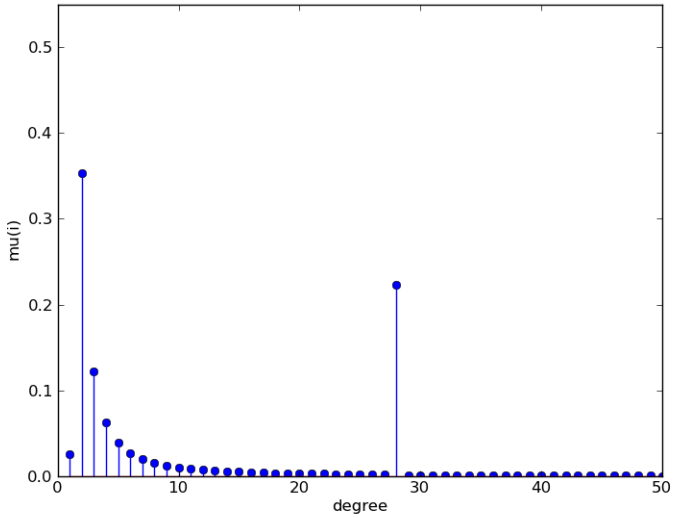

Figure 2.5. Robust Soliton Distribution.

2.2 Low-Density Parity-Check Codes

Low-density parity-check (LDPC) codes are a class of linear block codes with near-capacity performance on a large set of data transmission channels, surpassing Turbo codes on many channels [19]. LDPC codes were discovered by Gallager and presented in detail in his doctoral dissertation in 1960 [4] but remained ignored until 1981 when Tanner used bipartite graphs to generalize and introduce a graphical representation of LDPC codes, and this representation was eventually named as Tanner graph [13]. LDPC codes were rediscovered in the 1990s by MacKay and Neal [26], [5]. The LDPC codes suggested by Mackay and Neal are slightly different from Gallager's. Mackay and Neal reported based on empirical results that Gallager's work is superior [27].

2.2.1 Representations of LDPC Codes

Introducing the representations of LDPC codes can help understand both the mathematical basis, and encoding and decoding algorithms used to implement LDPC codes. Without loss of generality we will consider only binary LDPC codes with arithmetic on $\mathbb{F}_2$.

2.2.1.1 Matrix Representation:

We start our discussion with a brief definition of linear codes:
Linear Codes: A linear error-correcting code can be represented by an $K \times N$ binary matrix G called the generator matrix, such that a K-bit binary message s can be encoded as a N-bit vector:

$$t = sG \; mod \, 2.$$

LDPC codes are usually represented by their parity-check matrices H of size $M \times K$, where, $M = N - K$ and H is the null space of the $K \times N$ Generator matrix G. A quick look at the dimensions of the two matrices can reveal that their row dimensions sum up to N which is the dimension of the vector space to which the vector subspaces H and G belong. LDPC codes are linear block codes with sparse or low-density parity-check matrices, and this

property is an important attribute when it comes to finding good implementable decoders. Before going forward we give a few definitions first:

Definition 1: The weight of a binary vector or matrix is equal to the number of 1's in it. The overlap between the two vectors is the number of 1s in common between them. The density of a source of random bits is the expected fraction of 1's it contains. A vector is considered to be sparse if its density is less than 0.5. A vector is very sparse if its density vanishes as its length increases. Low-density can be a vague term but usually, a density of or less than 0.01 would be considered low.

The design code rate of the LDPC code is usually calculated as:

$$R = 1 - \frac{M}{N} = \frac{K}{N}.$$

A source message s of size $1 \times K$ is encoded as a $1 \times N$ codeword, $t = sG \; mod \, 2$, where $G_{(K \times N)}$ is the generator matrix of the LDPC code. When the codeword t is transmitted, the channel noise n is added, and the received codeword will be:

$$r = (sG + n) \; mod \, 2.$$

The task of the decoder is to deduce s given; the received codeword r, the assumed noise properties of the channel, and the code structure which is assumed known to both the encoder and decoder. An optimal decoder is supposed to return the message s which maximizes the posterior probability:

$$\frac{p(r|s,G)p(s|G)}{p(r|G)} = \frac{p(r|s,G)p(s)}{p(r|G)}.$$

2.2.1.2 Graphical Representation

LDPC codes can be graphically represented by Tanner graphs [13]. Tanner graphs are bipartite graphs. A bipartite graph denoted by $\mathcal{G} = (U, V, E)$,0 is a graph whose vertices

(nodes) are decomposed into two disjoints sets U and V, such that every edge e in the set of edges E connects a vertex in U and a vertex in V, and no two nodes in the same set are connected.

The two types of vertices or nodes in a Tanner graph are the variable node (VN) which corresponds to the coded bits, and the check node (CN) which correspond to the constraint bits. Before going further into the details of Tanner graphs, we present some information about the construction of the Parity-check matrix H which a Tanner graph provides a graphical representation of.

The parity-check matrix H is a $(N - K) \times N$ low density matrix on which the actual decoding is performed. When designing LDPC codes usually the parity-check matrix is constructed first with the desirable characteristics in the code, and then, the generator matrix G is generated based on the fact that the H matrix is a vector subspace on $\mathbb{F}_2$, and G is its dual subspace

$$GH^T = 0 \mod 2.$$

There are different methods to generate parity-check matrices, and below we present a number of easy to follow methods as given in [27]:

1. Matrix H is generated by starting from an all-zero matrix, and then randomly flipping g not necessarily distinct bits in each column.

2. Matrix H is generated by randomly creating weight g columns.

3. Matrix H is generated with weight g per column, and (as near as possible) uniform weight per row.

4. Matrix H is generated with a weight of g per column, and uniform weight per row, and no two columns having overlap greater than 1.

5. Matrix H is further constrained so that its bipartite graph has large girth.

6. Matrix $H = [H1|H2]$ is further constrained, or slightly modified so that $H2$ is an invertible matrix.

More on the new terms and concepts introduced above will be presented later.

Back to our main discussion, a Tanner graph is constructed as follows: an edge connects a CN i to a VN j if the element h_{ij} in the parity-check matrix H is equal to 1. Suppose that we have the parity-check matrix below with $R = \frac{K}{N} = \frac{1}{2}$, column weight $w_c = 2$ and row weight $w_r = 4$:

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix},$$

then the Tanner graph corresponding to this H matrix would be as shown in Figure 2.6:

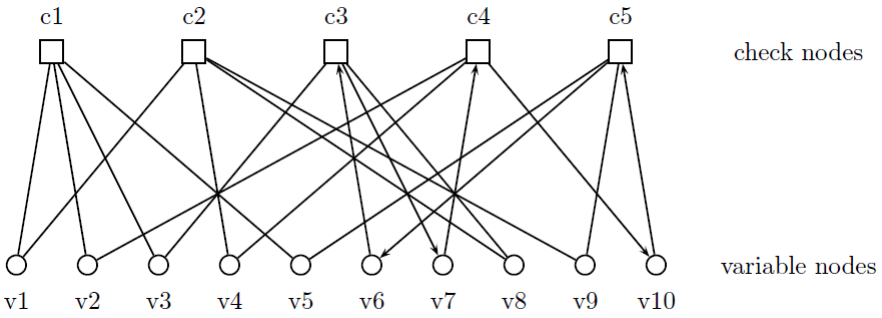

Figure 2.6. Tanner graph of the H matrix.

In Figure 2.6, if we look at variable node 1 (v1) we can see that it is connected to check nodes 1 and 2, now, if we examine the H matrix we can observe that elements h_{11} and h_{21} are equal to 1. Also if we check node 1 (c1), we can see it has four edges connected to v1, v2, v3 and v4, now, if we go back to the H matrix we can observe that $h_{11} = h_{12} = h_{13}$

$= h_{14} = 1$.

Tanner graphs of LDPC codes can help in visualizing and understanding the iterative decoding process of LDPC codes as follows:
Each of the nodes acts as a locally operating processor and each edge acts as a channel to communicate information from a given node to each of its neighbors (neighbors of a node are the nodes which share an edge with it). The exchanged information is generally of probabilistic nature describing the amount of certainty about the values of the bits assigned to variable nodes, this information is generally represented as a likelihood ratio (LR), or the numerically more stable log-likelihood ratio (LLR). The decoding is initiated by N LLRs computed from the $(1 \times N)$ received codeword and passed to the variable nodes. Then, the decoder works in an iterative fashion, where, at each iteration check nodes receive LLRs passed from neighboring variable nodes, process the information, then, pass the appropriate information back to each of the neighboring variable nodes which utilize the new information from their different neighboring check nodes to reach a decision about their corresponding bit values. The iterations continue until a correct codeword is found, or a predetermined maximum number of iterations is reached.

Now, we turn our attention to another topological aspect of LDPC codes that affect the performance of iterative decoders of LDPC codes. Cycles are a structural characteristic of LDPC parity-check matrices. A cycle in Tanner graph is a sequence of edges which form a closed path that begins and ends at the same node. Cycles can degrade the performance of the iterative decoder. As shown in Figure 2.6, we can see a cycle starting and ending at v10 by following the arrows. The length of a cycle equals the number of edges forming the cycle, and the minimum cycle length in a graph is called the girth of the graph. A desired property in Tanner graphs of LDPC codes is that the girth approaches infinity. However, for most practical code lengths such condition is not attainable, e.g., the girth of the graph in Figure 2.6 is equal to 6. The shortest cycle has a length equal to four and manifests itself in a parity-check matrix as four 1's in the four corners of a rectangular sub-matrix. Notice that

steps three and four in our example method given in the beginning to create an H matrix are an effort to eliminate or reduce the number of cycles in the parity-check matrix. Applying cycle-eliminating algorithms to H matrices is a common practice, and the software used in our simulations uses a procedure to eliminate cycles of length four in the H matrix. Cycles force the decoder to operate locally in some portions of the graph (around the elements of the cycle), and therefore, a globally optimum solution is impossible [28].

LDPC codes can be regular, irregular, or partially regular. Regular LDPC codes have a parity-check matrix with constant row and column weights. The Tanner graph shown in Figure 2.6 is regular, each VN has degree 2, i.e., it is connected to two neighbors via two edges, and each CN is degree 4. Irregular LDPC codes have varying row and column weights, and their row and column or CN and VN degree distributions can be expressed by degree distribution polynomials $\rho(x)$ and $\lambda(x)$, as shown below, respectively.

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1},$$

where, ρ_d is equal to the fraction of edges connected to degree-d CNs, and d_c is the maximum CN degree in the graph.

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1},$$

where, λ_d is equal to the fraction of edges connected to degree-d variable nodes, and d_v is the maximum VN degree in the graph.

The regular Tanner graph in Figure 2.6 can be represented by the polynomials: $\rho(x) = x^3$, and $\lambda(x) = x$.

The number of edges in a graph can be calculated as :

$$E = \frac{N}{\int_0^1 \lambda(x)dx} = \frac{M}{\int_0^1 \rho(x)dx}.$$

Finally, partially regular LDPC codes have column and row weights that can be divided into sets of different weights.

Irregular LDPC codes can outperform regular LDPC codes, and exhibit performance extremely close to the Shannon limit [19]. It was also shown in [19] that the performance of irregular LDPC codes further approaches channel capacity as the block length increases. The same paper also, shows that irregular LDPC codes can outperform the state-of-the-art Turbo codes for different code lengths.

As Shannon suggested, finding good codes which can achieve or approach the channel capacity is a problem of finding codes with random nature and large block lengths. However, finding optimal decoders is also a requirement, and all these requirements have complexity as an inevitable byproduct.

2.2.2 Encoding of LDPC Codes

The encoding process of LDPC codes is generally straightforward. Given a soure message s of size $(1 \times K)$, and a $(K \times N)$ Generator matrix G, then the encoded codeword:

$$t = sG \; mod \, 2.$$

LDPC codes can be systematic, or non-systematic. In systematic codes, the source message s is a part of the encoded word, and the other part is the added redundancy. In systematic LDPC codes the Generator matrix has the form:

$$G = [I_K | P_{K \times M}].$$

The above form can be reached by transforming the parity-check matrix H into the form $H = [h_{M\times K}|I_M]$ using Gaussian elimination and matrix manipulation, where, $M = N - K$. Knowing the property that $GH^T = 0$ we can follow:

$$\begin{bmatrix} I_K & P \end{bmatrix} \begin{bmatrix} h^T \\ I_M^T, \end{bmatrix} = 0,$$

$$I_K h^T + P I_M = 0 \; mod\, 2,$$

$$P I_M = -I_K h^T \; mod\, 2,$$

$$P = h^T.$$

The last step comes as the additive inverse of an element a on $\mathbb{F}_2$ is itself. For the procedure above, matrix rank M is assumed for H.

As the Generator matrix $G = [I_K \; P]$, then, it is trivial that the first K symbols or bits of the codeword will be the $(1 \times K)$ source message s. The added $N - K$ redundancy is equal to sP, where, P is the $K \times M$ parity-check matrix. In non-systematic LDPC codes, G is simply the dual space of the H matrix, and after decoding the original source message s of size $(1 \times K)$ needs to be extracted from the decoded $(1 \times N)$ block.

The complexity of the encoding process depends on the density of (i.e., the number of 1's) in the G matrix. The complexity grows in $O(N^2)$, where, N is the block length of the encoded message. LDPC codes further approach the capacity of the channel as the block length N increases. Considering the desire for larger block lengths and the dense nature of the G matrix, the encoding complexity can become an issue when designing LDPC codes, therefore, there have been many attempts to design LDPC codes with lower encoding and decoding complexities [14] [29] [30].

2.2.3 Decoding of LDPC Codes

The decoding problem as stated in [26] is: given the received message $r = (t+n)\, mod\, 2$, where, $t = sG$ is the transmitted message, and n is a sparse random vector with density equal to the channel error rate, and given that H is the dual space of G, then, the decoding problem becomes finding the most probable n that satisfies:

$$nH^T = z\ mod 2,$$

where, $z = Hr$. In other words, taking advantage of the fact that H and G are the bases of two dual subspaces, then, if $t = sG$ is a vector in the vector subspace whose basis is G, then, $tH^T = 0$. A received vector r such that $rH^T \neq 0$ implicates that the message was distorted by an added noise vector n,

$$rH^T = (t+n)H^T = tH^T + nH^T = nH^T,$$

then, the decoding problem is obviously finding t by finding the error vector n.

2.2.3.1 The Sum-Product Algorithm

The SPA was developed to perform iterative decoding on the parity-check matrix H. It was introduced by Gallager along with LDPC codes in his doctoral dissertation in 1963 [4], and it was rediscovered by Mackay and Neal in 1990s [5]. The sum-product algorithm is also called belief propagation algorithm BPA, the name comes from Bayesian inference literature, where, the algorithm was derived independently [31]. We can think of the Tanner graph as a belief or Bayesian network where every bit is the parent of g check nodes and every check node is the child of r bits.

SPA is a symbol wise MAP decoding algorithm. The objective of the SPA is to compute APP that a specific bit in the transmitted codeword $t = [t_0\, t_1\, \ldots\, t_{n-1}]$ is equal to 1, given, the received word $y = [y_0\, y_1\, \ldots\, y_{n-1}]$. Without loss of generality, we focus on

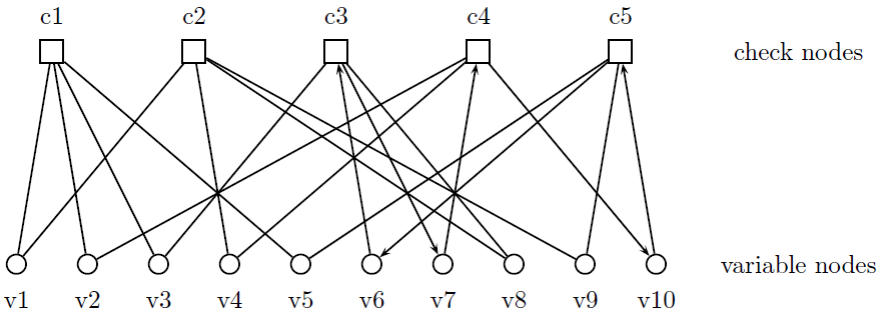

Figure 2.7. Decoding on Tanner graph.

decoding a bit t_j by calculating the APP:

$$Pr(t_j = 1|y).$$

Considering that we have two competing hypotheses for the value of t_j, namely, 1 and 0, we turn our attention to compute the likelihood ratio (LR):

$$l(t_j|y) = \frac{Pr(t_j = 0|y)}{Pr(t_j = 1|y)},$$

which can be put in the more numerically stable log-likelihood ratio (LLR) form:

$$L(t_j|y) = \log\left(\frac{Pr(t_j = 0|y)}{Pr(t_j = 1|y)}\right),$$

where, log is assumed to be the natural logarithm.

SPA is a message passing algorithm that operates by conveying information between check nodes and variable nodes of a Tanner graph. In order to understand the SPA we need to be able to answer the following questions:

- What are the roles of check nodes and variable nodes in the decoding process?
- What is the nature of the information exchanged between check nodes and variable nodes?

- What effects do the structural properties of the Tanner graph impose on the BPA?

In a Tanner graph, there are N variable nodes and each one of these nodes corresponds to a bit in the $(1 \times N)$ received codeword to be decoded. VNs act as local processors that add the LLRs values passed to them from the channel, and the neighboring CNs to reach the most plausible decision about the value of the bit it represents.

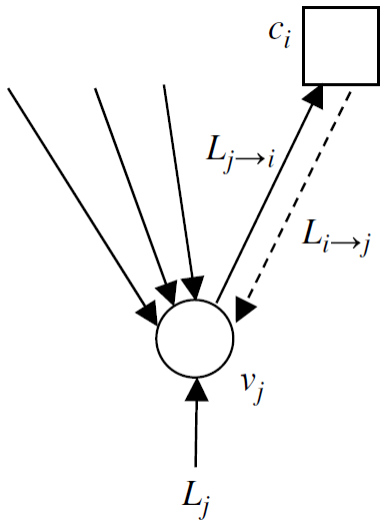

Figure 2.8. A Variable Node.

In a Tanner graph, there are also M CNs that correspond to the M parity bits added by the encoder. Each CN acts as a local processor that receives LLRs form its neighboring VNs, processes the information, and then, passes the appropriate conclusion to each of its neighbors.

SPA algorithm is a message passing decoding algorithm. The term "message passing" refers to the process of a collection of low complexity decoders working in a distributed fashion. In SPA, VNs and CNs work cooperatively in an iterative fashion to reach estimates of the likelihood ratio $L(t_j|y)$ for $j = 0$ to $N-1$. In the first half of every iteration, each VN processes the LLRs it received from neighboring CNs and passes the extrinsic information to each of its neighboring CNs. In the second half of the iteration, every CN processes the LLRs it received from neighboring VNs and passes the extrinsic information to every neighboring

VN.

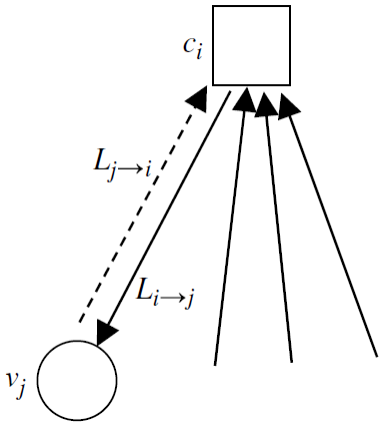

Figure 2.9. A Check Node.

Message passing algorithm introduces the concepts of extrinsic and intrinsic information. MPA imposes the constraint of exchanging only extrinsic information between nodes, the concept of extrinsic information is that a node does not pass any information to another if the other node possesses the information. In the case of SPA, a node N_i does not pass LLR_q to N_j, if LLR_q was originally passed from N_j, because LLR_q is considered intrinsic information to N_j. A message passing decoding algorithm cannot be claimed to be optimal if cycles exist in the Tanner graph. Cycles provide a path through which intrinsic information can travel and reach the originating nodes, i.e., the nodes that sent them. Although there exist many procedures to remove cycles from graphs in general, or from Tanner graphs in the case of parity-check matrices, most practical codes are suboptimal in the sense that they contain cycles. However, message passing decoding algorithms perform very well for properly designed codes and reach error rates deemed as acceptable for the majority of applications. From the discussion above, we can understand the reason to desire Tanner graphs with the largest girth possible.

Before discussing the SPA decoding used in LDPC codes, we study the operations of VNs and CNs with more detail so that we can follow the steps of the algorithm more thoroughly.

VNs as Repetition Decoders

A repetition decoder bases its decision on the majority rule. Given a repetition code that repeats every bit d times, then, every bit b_i in the source word is transmitted n times, and the decoder decision will be 1 or 0 based on the majority in the received d bits. Similarly, the decoder can reach the same decision with soft information, i.e., probabilistic information. For a binary repetition code that transmits every bit t for n times over a memory-less channel, and r is the received n-vector. A MAP on t can be reached by computing:

$$L(t|r) = \log\left(\frac{Pr(c=0|r)}{Pr(c=1|r)}\right).$$

Assuming $Pr(t=0) = Pr(t=1)$:

$$L(t|r) = \log\left(\frac{Pr(t=0,r)Pr(r)}{Pr(t=1,r)Pr(r)}\right),$$

$$L(t|r) = \log\left(\frac{Pr(r|t=0)Pr(t=0)}{Pr(r|t=1)Pr(t=1)}\right),$$

$$L(t|r) = \log\left(\frac{Pr(r|t=0)}{Pr(r|t=1)}\right),$$

$$L(t|r) = \log\left(\frac{\prod_{i=0}^{n-1} Pr(r_i|t=0)}{\prod_{i=0}^{n-1} Pr(r_i|t=1)}\right),$$

$$L(t|r) = \sum_{i=0}^{n-1} \log\left(\frac{Pr(r_i|t=0)}{Pr(r_i|t=1)}\right),$$

$$= \sum_{i=0}^{n-1} L(r_i|x).$$

Where, $L(r_i|x)$ is the LLR of the received bit or r_i. The MAP decoder of the repetition

code adds the individual LLRs, and a majority decision is reached such that $t_l = 1$ if $L(t_l|r) < 0$, and 0 otherwise.

A VN node in a Tanner graph is a repetition decoder in the sense that, at the last iteration of decoding it adds the LLRs it receives from all its neighboring CNs, and the channel according to the equation:

$$L_{j\to i} = L_j + \sum_{i'\in N(j)} L_{i'\to j}, \tag{2.3}$$

and depends on the value of $L_{j\to i}$ to reach a decision on the value of t_j.

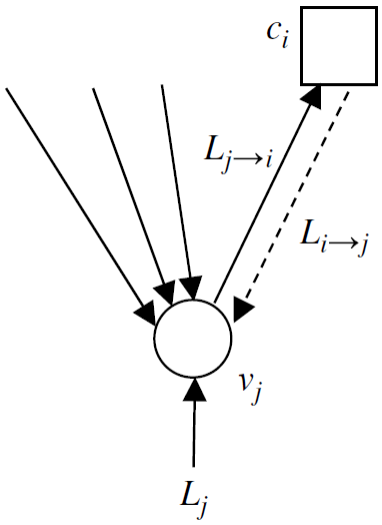

Figure 2.10. VN as a repetition decoder.

During decoding a variable-node VN_j passes extrinsic information $L_{j\to i}$ to a CN_i according to the equation:

$$L_{j\to i} = L_j + \sum_{i'\in N(j)-(i)} L_{i'\to j} \tag{2.4}$$

In equation 2.4 above, we can notice that VN_j excludes $L_{i\to j}$ when it is calculating $L_{j\to i}$ so that no intrinsic information is passed. L_j is the LLR computed from y_j which is the received symbol from the channel using the equation:

$$L_j = L(t_j|y) = \log\left(\frac{Pr(t_j = 0|y_j)}{Pr(t_j = 1|y_j)}\right). \tag{2.5}$$

Below, we give the probabilities and channel LLRs used for the BEC, BSC, and BIAWGN channels.

BEC Channel:

Assuming t is the transmitted vector, $t_j \in \{0, 1\}$, y is the received codeword, and $y_j \in \{1, 0, e\}$, where, e stands for an *erasure*, and $b \in 0, 1$:

$$Pr(t_j = b|y_j) = \begin{cases} 1 & \text{when } \mathrm{y}_j = b, \\ 0 & \text{when } \mathrm{y}_j = b', \\ 1/2 & \text{when } \mathrm{y}_j = \mathrm{e}. \end{cases}$$

Where, b' is the complement of b, applying the above probabilities to equation 2.5 gives:

$$L(t_j|y_j) = \begin{cases} \infty & \mathrm{y}_j = 0, \\ -\infty & \mathrm{y}_j = 1, \\ 0 & \mathrm{y}_j = \mathrm{e}. \end{cases}$$

BSC Channel:

Assuming $y_j \in \{1, 0\}$, and the channel error rate or transition probability $\epsilon = Pr(y_j = b'|t_j = b)$:

$$Pr(t_j = b|y_j) = \begin{cases} \text{1-}\epsilon & \mathrm{y}_j = b, \\ \epsilon & \mathrm{y}_j = b'. \end{cases}$$

And

$$L(t_j|y_j) = (-1)^{y_j} \log\left(\frac{1-\epsilon}{\epsilon}\right).$$

BIAWGN Channel:

Let $y_j = x_j + n_j$, where, $x_j = (-1)^{v_j}$ such that $x_j = +1$ and (-1) for $v_j = 0$ and (1), respectively. n_j is Gaussian channel noise $\mathcal{N}(0, \sigma^2)$, and the value of σ or an estimate must be known to the decoder in order to calculate the LLR as:

$$L(t_j|y_j) = \frac{2y_j}{\sigma^2}.$$

CNs as Single Parity Codes

A single parity code (SPC) of length n on $\mathbb{F}_2$ is a linear code which consists of $n-1$ information bits, and a single parity-check bit. The value of the parity check bit is equal to the $mod\,2$ sum of the $n-1$ information bits, thus, the generated codeword always sums to $0\,mod\,2$, and has an even number of 1's.

Parity-check bits added by the LDPC code follow the same logic. Keeping this structure in mind, when we design SPC decoder. Suppose a codeword c of size $(1 \times n)$ is encoded using SPC, and we receive the codeword r which was transmitted over some channel. Without loss of generality, we choose bit c_b to decode by conditioning on the received vector r, and the fact that for the received vector r to be correct it must have an even weight:

$$\hat{c_b} = \arg\max_{x \in 0,1} Pr(c_b = x|r, SPC).$$

Therefore,

$$Pr(c_b = 0|r, \text{SPC}) = Pr(c_0, c_1, \ldots, c_{b-1}, c_{b+1}, \ldots, c_{n-1} \text{ have an even number of 1s}|r)$$

$$= \frac{1}{2} + \frac{1}{2} \prod_{l=1, \neq b}^{n-1} (1 - 2Pr(c_l = 1|r_l))$$

This can be rewritten as:

$$2Pr(c_b = 0|r, \text{SPC}) = 1 + \prod_{l=1, \neq b}^{n-1} (1 - 2Pr(c_l = 1|r_l)) \tag{2.6}$$

Where, $Pr(c_l = 0|r, \text{SPC}) + Pr(c_l = 1|r, \text{SPC}) = 1$, then, we can rewrite 2.6 as:

$$1 - 2Pr(c_b = 1|r, \text{SPC}) = \prod_{l=1,\neq b}^{n-1} (1 - 2Pr(c_l = 1|r_l)) \tag{2.7}$$

Using the relation between binary random variables $1 - 2p_1 = \tanh(\frac{1}{2}\log(\frac{p_0}{p_1}))$ we rewrite 2.7 as:

$$\tanh\left(\frac{1}{2}\frac{Pr(c_b = 0|r, \text{SPC})}{Pr(c_b = 1|r, \text{SPC})}\right) = \prod_{l=1,\neq b}^{n-1} \tanh\left(\frac{1}{2}\frac{Pr(c_l = 0|r_l)}{Pr(c_l = 1|r_l)}\right) \tag{2.8}$$

We already have defined $\log(\frac{p_0}{p_1})$ as the Log-Likelihood Ratio (LLR) and applying this to 2.8 gives:

$$\tanh\left(\frac{1}{2}L(c_b|r, \text{SPC})\right) = \prod_{l=0,\neq b}^{n-1} \tanh\left(\frac{1}{2}L(c_l|r_l)\right)$$

$$L(c_b|r, \text{SPC}) = 2\tanh^{-1}\left(\prod_{l=1,\neq b}^{n-1} \tanh\left(\frac{1}{2}L(c_l|r_l)\right)\right) \tag{2.9}$$

The MAP decoder of the SPC code makes decisions based on the outcome of 2.9 such that $\hat{c} = 1$, if $L(c_b|r, SCP) < 0$, and $\hat{c} = 0$ otherwise.

$$L_{i\to j} = 2\tanh^{-1}\left(\prod_{j`\in N(i)-(j)}^{n-1} \tanh\left(\frac{1}{2}L_{j`\to i}\right)\right) \tag{2.10}$$

The MAP decoder described above was developed for SPC codes. However, the CNs in a Tanner graph can apply (2.9) to process the LLR values it receives from the VNs it is connected to since every CN and the VNs connected to it adhere to the constraint of SPC code, i.e., their sum is equal to $0\, mod 2$. $L_{i\to j}$ in (2.10) is the value of the *extrinsic* LLR a CN_i sends to its neighboring VNs. The notation $j` \in N(i) - (j)$ refers to the fact that when computing $\text{LLR}_{i\to j}$ a CN_i processes the LLRs sent to form all neighboring VNs except j so that only extrinsic information is sent.

Sum-Product Algorithm

Having discussed the sum-product algorithm, and investigated the operations performed at each iteration in terms of: 1) type of messages communicated in the BPA, and the rules that control this message exchanging process between check nodes and variable nodes, 2) the local operations performed by the VNs and CNs to process these message, below we summarize the SPA as given in [28] p. 220:

1. Initialization: For all j, initialize L_j according to 2.5 for the appropriate channel model. Then, for all i, j for which $h_{ij} = 1$, set $L_{j\to i} =$ Lj.

2. CN update: Compute outgoing CN messages $L_{i\to j}$ for each CN using 2.10

$$L_{i\to j} = 2\tanh^{-1}\left(\prod_{j\text{`}\in N(i)-(j)}^{n-1}\tanh\left(\frac{1}{2}L_{j\text{`}\to i}\right)\right)$$

   and then transmit to the VNs.

3. VN update: Compute outgoing VN messages Lj→i for each VN using Equation 2.3

$$L_{j\to i} = L_j + \sum_{i'\in N(j)-(i)} L_{i'\to j}$$

   and then transmit to the CNs.

4. LLR total: For $j = 0, 1, \ldots, n-1$ compute:

$$L_{j\to i} = L_j + \sum_{i'\in N(j)} L_{i'\to j}$$

5. Stopping criteria: For $j = 0, 1, \ldots, n-1$, set

$$\hat{v}_j = \begin{cases} 1 & \text{if } L_j^{\text{total}} < 0, \\ 0 & \text{else,} \end{cases}$$

to obtain $\hat{v}$. If $\hat{v}H^T = 0$ or the number of iterations equals the maximum limit,stop; else, go to Step 2.

CHAPTER 3

RAPTOR CODES

Raptor codes are an extension of LT codes [3], they are the first implementation of fountain codes with linear encoding and decoding costs. LT codes like other fountain codes, have a non-linear encoding and decoding costs and suffer from high error floors. The motivation that led to the invention of Raptor codes was to provide fast encoding and decoding algorithms with linear complexity and vanishing error floor for rateless codes. Raptor codes are forward error correction (FEC) codes that provide application-layer protection against packet losses. They were proposed by Amin Shokrallahi in late 2000 and filed for a patent in 2001. All variations of Raptor codes outperform LT codes in terms of computational cost and decoding performance. Raptor codes, also, have better over-head failure performance in comparison to LT codes [32].

3.1 Design of Raptor Codes

In order to better understand the design of Raptor codes we need to understand the factors which led to proposing Raptor codes as an extended version of LT codes.
Let $\Omega_0, \Omega_1, \ldots, \Omega_k$ denote the probability distribution on $0, 1..., k$,such that, Ω_i is equal to the probability of the value i being chosen. This distribution is typically represented by a generator polynomial $\Omega(x)$:

$$\Omega(x) = \sum_{i=0}^{k} \Omega_i x^i = 1.$$

where, $\Omega_0 = 0$ but it is kept for notational convenience. This notation above can be used to represent the output degree distribution of LT codes in the following manner: the polynomial $\Omega(x)$ can be used to denote the probability distribution on a vector space $\mathbb{F}_2^k$. Given a

fountain code such as LT code, where, $\Omega(x)$ denotes the Robust Soliton Distribution, this code can be defined by its two parameters $(k, \Omega(x))$. Encoding is the process of the linear mapping $\mathbb{F}_2^k \rightarrow \mathbb{F}_2^N$, where, N can potentially be an infinite number, however, in practice it is some finite number. Considering the above notation, the encoding process proceeds as follows: for a given source block of k symbols, $(x_1, \ldots, x_k)$, every output symbol is generated independently by sampling the degree distribution $\Omega(x)$ for a degree d, and then, uniformly at random a vector v of weight d is chosen from the vector space $\mathbb{F}_2^k$, and the output symbol is computed as $\sum_{i=1}^{k} v_i x_i$.

An LT code $(k, \Omega(x))$ possesses a reliable decoding algorithm if the algorithm can recover all k input symbols from any set of size $n > k$ of output symbols, with error probability at most $1/k^c$, where c is a positive constant. The decoding graph of LT codes is a bipartite graph with k input (variable) nodes on one side representing the source symbols, and n output (check) nodes on the other sides representing the collected output symbols by the receiver side. The error probability of the decoder is lower-bounded by the probability that there exists uncovered input symbols since it is not possible to identify input symbols that did not participate in the encoding process. This leads to a lower bound on the number of edges in the graph in the order of $ck\log(k)$. Considering that the average encoding cost of LT codes is the expected number of operations needed to generate an output symbol, then, the encoding cost of LT codes is at least in the order of $\log(k)$, or as stated in [3] $O(\ln(k/\delta))$. For the desirable case of the number of output symbols sufficient for successful decoding (n) that approaches k, output symbols need to have an average weight of $\log(k)$.

In order to analyze the decoding cost, we assume a BEC channel, and Maximum Likelihood (ML) decoding, which would be equivalent to Gaussian elimination. With the assumptions above, the decoding process becomes equivalent to solving a consistent system (a system that has a solution) of n linearly independent equations corresponding to the collected output bits for the values of input bits $(x_1, \cdots, x_k)$. For the system above to be solvable its corresponding $(n \times k)$ matrix must have a full rank. The cost of solving such a

matrix is $O(nk^2)$ operations which means that the average decoding cost per source symbol is $O(nk)$. In [3] it is stated that LT codes the average decoding cost is $O(k \cdot \ln(k/\delta))$.

From the discussion above, we find that LT codes have a non-constant average encoding and decoding costs per source block. On the other hand, Raptor codes designed in [2] have the following properties; for a given source block of k symbols, and overhead $\varepsilon > 0$ Raptor codes have an average number of $O(\log(1/\varepsilon))$ symbol operations per generated output symbol, and an average decoding cost of $O(k \cdot \log(1/\varepsilon))$ per source block with failure probability of $1/k^a$, for a positive constant $a > 1$ that is independent of ε. The advantage of Raptor codes in terms of computational complexity is clear.

The reason it is difficult to construct LT codes with a constant average degree is that the decoding graph needs at least $k \log(k)$ edges in order to ensure all input symbols are covered otherwise there is a high probability that a fraction of the input symbols would not be covered. The solution proposed by Raptor codes to solve this issue is to precode the input symbols by a traditional erasure correcting code $\mathcal{C}$, and then, apply an LT code with a constant average output degree to the symbols generated by the precode $\mathcal{C}$.

Raptor codes can be parameterized by $(k, \mathcal{C}, \Omega(x))$, where, k denotes the number of input symbols, the precode $\mathcal{C}$ is linear block code of dimensions (k, n), and typically the precode used for Raptor codes is a regular high rate LDPC code, so henceforth we use the terms precode, and LDPC code interchangeably, and $\Omega(x)$ is the degree distribution of the LT part. The encoding process starts with the precode encoding the k input and adding a relatively small redundancy of $(m - k)$ symbols to produce $m > k$ symbols known as the *intermediate symbols*. The intermediate symbols are used to generate the *output symbols* by the LT code part.

Below, we give a toy example in order to describe the mechanism in which Raptor codes operate, and how the encoding process proceeds:

In Figure 3.2, the input symbols x1, ...,x6 are encoded using an LT code. We can

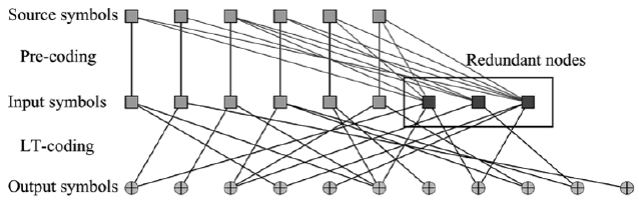

Figure 3.1. Graphical representation of Raptor codes.

observe that x2 was not covered by the LT code, and hence, it will not be recovered by the decoder.

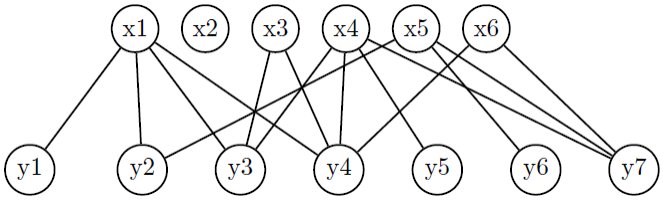

Figure 3.2. LT code.

Now, we apply Raptor code to the same input set, first, we encode the input symbols by a precode such that two redundancy symbols z1, and z2 are added as shown in Figure 3.3 below.

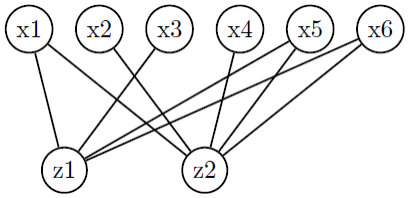

Figure 3.3. Precode of Raptor codes.

Next, we apply an LT code to the input symbols x1, . . . , x6 in addition to z1, and z2 which are collectively called intermediate symbols. The code graph will be as illustrated by Figure 3.4. Despite the fact that x2 was not covered by the LT code but it was covered by the precode, and it can be recovered in the LDPC decoding part of Raptor codes.

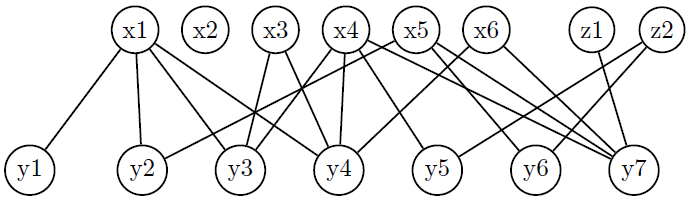

Figure 3.4. Raptor code.

3.2 Decoding of Raptor Codes

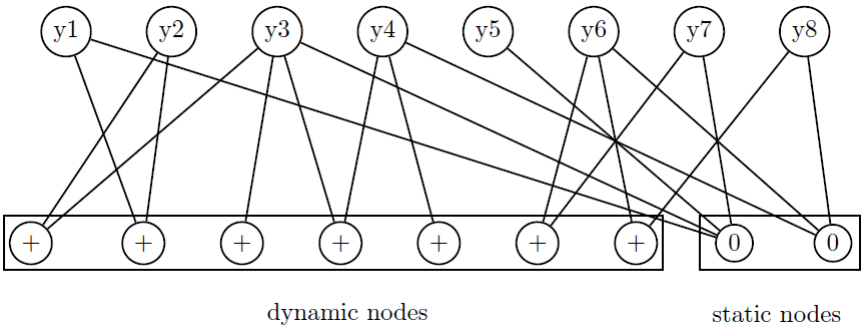

Figure 3.5. Decoding graph of Raptor code.

The decoding graph of a length n Raptor code is a bipartite graph that consists of n nodes on one side called the input nodes, they represent the n output symbols of the LDPC code, where the objective of the decoding process is to identify their correct values. On the other side, there are $m + n - k$ nodes, that are called the output nodes, and they can be divided into two sets.

One set consists of the m output bits or symbols collected from the channel and each node in this set is connected to the input nodes of which it is a neighbor, and its value is equal to the sum of their values. This set is named as the dynamic set. The name dynamic comes as this part of the graph depends on the particular set of m encoding symbols collected by the decoder. The dynamic set corresponds to the LT code part of Raptor codes.

The other set of $n - k$ nodes are called the static set and it corresponds to the $n - k$ parity-check bits added by the LDPC code. Remember that the decoding graph is

reconstructed at the decoder side based on information passed by the encoder.

Typically, the decoder starts the decoding process on the dynamic part and after a pre-fixed number of iterations, it shifts to decoding on the static part. Decoding on the dynamic part is actually an LT decoding process, and decoding on the static part is an LDPC decoding process on the parity-check matrix H of the precode of the Raptor code. The inner workings of each phase are governed by the rules, and procedures related to each code as given in sections 2.1 and 2.2.

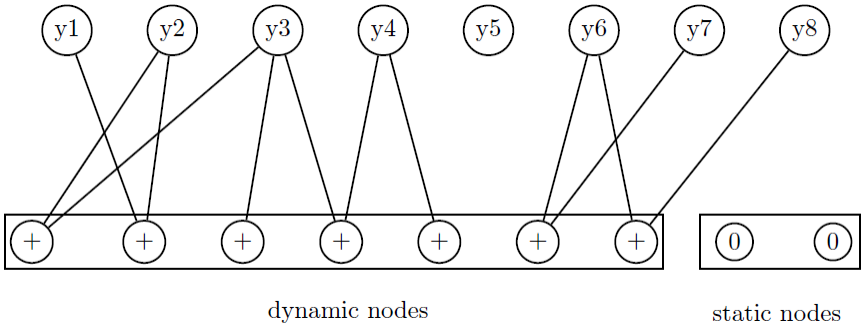

Figure 3.6. Decoding on the dynamic part.

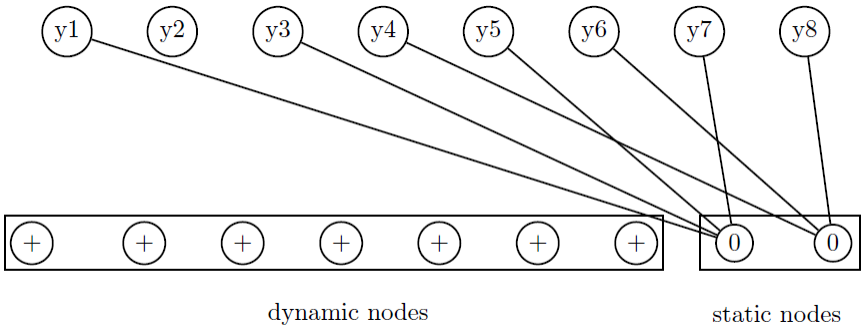

Figure 3.7. Decoding on the static part.

Raptor codes are a concatenation of two codes; LDPC, and LT codes. It is necessary for the two codes to be interfaced properly and in a careful manner. The n symbols we collect from the channel are output symbols of Raptor codes which were encoded by the LT code part before transmission. As we mentioned earlier, decoding starts on the dynamic part of the decoding graph which corresponds to the LT-decoding part of the Raptor decoder. As the

belief-propagation algorithm (BPA) proceeds, messages from output nodes to input nodes ($m^l_{o,i}$) and from input nodes to output node ($m^l_{i,o}$) are conveyed back and forth according to rules and constraints of BPA algorithm. For more on BPA check section 3.3.1. During the decoding process, the output nodes act as local single parity-check (SPC) decoders, they process $m^l_{i,o}$s, which are LLRs. They receive and send $m^l_{o,i}$ to their neighboring input nodes by applying the equation:

$$\tanh\left(\frac{m^l_{o,i}}{2}\right) = \tanh\left(\frac{Z_o}{2}\right) \cdot \prod_{i' \neq i} \tanh\left(\frac{m^l_{i',o}}{2}\right),$$

where, Z_o is the channel LLR of every output bit o. It is computed from the value of the corresponding symbol received from the channel using equation 2.5.

Input nodes act as local Repetition processors. They receive LLRs from neighboring output nodes ($m^l_{o,i}$), and apply equation 3.1 to compute the extrinsic information ($m^{l+1}_{i,o}$) they pass to the output nodes they are connected to,

$$m^{l+1}_{i,o} = \sum_{o' \neq o} m^l_{o',i} \tag{3.1}$$

After a preset maximum number of iterations decoding stops on the dynamic part of the graph, and every input node computes its total LLR as $\sum_o m_{o,i}$. Next decoding on the static, or LDPC part of the graph starts. Decoding is initiated by input nodes passing their LLRs computed in the dynamic phase of decoding as channel LLR to the LDPC decoder, and again input nodes act as repetition decoders while the nodes of the static set act as SPC decodes. After a maximum number of iterations has been reached, or a stopping criterion has been met, decoding stops, and the decoder declares a value for each bit of the input bits. The decoded m bits are the output of the LDPC code part, and the LDPC decoder part returns estimates of the k source bits.

3.3 The Output Degree Distribution

In Raptor codes the Robust Soliton distribution is not used as the output degree distribution for the LT code part, instead as shown in [2] a new degree distribution is developed based on heuristic analysis which leads to optimized degree distributions for different values of source data size k.

The analysis is performed on the dynamic (LT) part of the decoding graph. Before going further into the analysis we need to define few parameters of the graph that are used in the analysis. w_i is the probability that a randomly chosen edge is connected to a degree i output node, and $w(x) = \sum_i w_i x^{i-1}$. Ω_i is the probability that a randomly chosen output node is of degree i, and $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$. ι_i is the probability that a randomly chosen edge is connected to a degree i input node, and $\iota(x) = \sum_i \iota_i x^i$. The following relations exist between the parameters:

$$w(x) = \frac{\Omega'(x)}{\Omega'(1)},$$
$$\iota(x) \triangleq e^{\alpha(x-1)}. \tag{3.2}$$

Assuming belief propagation decoding, we view the decoding process from a binary perspective, i.e., the messages exchanged are 0 or 1 for the sake of simplifying the analysis. An input node sends a message 1 to a neighboring output node if its value has been recovered, and an output node sends a message 1 to an input node if it has identified its value and vice versa.

Let p_i denote the probability that a randomly chosen edge carries a message 1 from an output node at iteration i, then we have the recursion:

$$p_{i+1} = w(1 - \iota(1 - p_i)) \tag{3.3}$$

This comes from the and-or tree analysis argument [33] that an input node (VN) needs to receive at least one message of value 1 to be decoded which makes the probability to be decoded $1 - (1 - p_i)^{d_{in}}$, where, d_{in} is the degree of the input node, while, for an

output node (CN) to send a 1 to a neighbor it needs to receive message 1 from all the other neighbors. Then, the probability of a CN sending a message 1 becomes $(1-(1-p_i)^{d_{in}})^{d_{out}-1}$.

Now, we define u_i as the probability that an input symbol is recovered at iteration i. u_i is equal to $1-(1-p_i)^d$ since an input symbol is recovered if it receives a message 1 from any of its neighboring d output nodes. In terms of the decoding graph this probability is written as $1-\iota(1-p_i)$. Applying 3.2 we get:

$$1-\iota(1-p_i) = 1-e^{-\alpha p_i} = u_i, \tag{3.4}$$

and

$$p_i = \frac{-\ln(1-u_i)}{\alpha}, \tag{3.5}$$

where, $u_{i+1} = e^{-\alpha p_{i+1}}$. Taking a look at equations (3.3) and (3.4) we can express u_{i+1} as:

$$u_{i+1} = 1-e^{-\alpha w(u_i)}. \tag{3.6}$$

Equation 3.6 implies that having a fraction-x of recovered symbols at some iteration i, then, in the next iteration $(i+1)$ the fraction increase to $1-e^{-\alpha w(x)}$, i.e., an increase of $1-x-e^{-\alpha w(x)}$. If decoding is performed on $k(1+\epsilon)$ output symbols, then we have $w(x) = (1+\epsilon)\Omega'(x)/\alpha$, and the expected fraction of input symbols in the input ripple is:

$$1-x-e^{-\Omega'(x)(1+\epsilon)} \tag{3.7}$$

hence the expected input ripple size is equal to:

$$k(1-x-e^{-\Omega'(x)(1+\epsilon)}) \tag{3.8}$$

The derivation above is based on heuristic assumptions but it does not affect the findings because after reaching candidate degree distributions the error probability of the LT-decoder is computed for each distribution.

From Luby's analysis in [34] the concept of keeping the expected ripple size equal to, or larger than $c\sqrt{(1-x)k}$ for a positive constant c is adapted. Using this boundary condition on equation 3.8 for some ϵ, δ, and k leads to:

$$1 - x - e^{-\Omega'(x)(1+\epsilon)} \geq c\sqrt{(1-x)/k},$$

where, $x \in [0, 1-\delta]$ and:

$$\Omega'(x) \geq \frac{-\ln(1 - x - c\sqrt{(1-x)/k})}{1+\epsilon}. \tag{3.9}$$

where, ϵ is the overhead, and δ is the fraction of intermediate symbols that are not recovered.

By discretizing the interval $[0, 1-\delta]$ and requiring the above inequality to hold on the discretization points, we obtain linear inequalities in the unknown coefficients of $\Omega(x)$. The optimized degree distributions are for the values of k shown in Table 3.1, $\delta = 0.01$, the overhead ϵ used in the optimization process, and a is the average degree of output symbols.

Table 3.1. Degree distributions for given values of k source symbols.

k	65536	80000	100000	120000
Ω_1	0.007969	0.007544	0.006495	0.004807
Ω_2	0.493570	0.493610	0.495044	0.496472
Ω_3	0.166220	0.166458	0.168010	0.166912
Ω_4	0.072646	0.071243	0.067900	0.073374
Ω_5	0.082558	0.084913	0.089209	0.082206
Ω_8	0.056058		0.041731	0.057471
Ω_9	0.037229	0.043365	0.050162	0.035951
Ω_{18}				0.001167
Ω_{19}	0.055590	0.045231	0.038837	0.054305
Ω_{20}		0.010157	0.015537	
Ω_{65}	0.025023			0.018235
Ω_{66}	0.003135	0.010479	0.016298	0.009100
Ω_{67}		0.017365	0.010777	
ϵ	0.038	0.035	0.028	0.02
a	5.87	5.91	5.85	5.83

CHAPTER 4

SIGNAL TO NOISE RATIO MISMATCH

On AWGN channels, SNR mismatch is a condition that occurs when the estimated SNR on the receiver side is not equal to the actual channel SNR. Without loss of generality, we will restrict our discussion to the BIAWGN channel and binary codes. Also, we assume binary phase shift keying (BPSK) modulation with the mapping $0 \rightarrow +1$ and $1 \rightarrow -1$. Given a BIAWGN channel with input $x = \pm 1$, the output of the channel can be described as $y = x + n$, where, $n \sim \mathcal{N}(0, \sigma^2)$ is the channel noise with mean zero and variance (power) equal to σ^2, and such that $y \sim \mathcal{N}(x, \sigma^2)$. On the receiver side, ideally, the channel noise power will be estimated correctly as σ^2 and the SNR per bit is computed as:

$$\gamma = \frac{E_b}{N_o} = \frac{1}{2R\sigma^2},$$

where, R is the code rate. However, if the noise power is erroneously estimated as $\sigma_e \neq \sigma^2$, then, we will obtain

$$\gamma_e = \frac{E_b}{N_{o,e}} = \frac{1}{2R\sigma_e^2} \neq \gamma,$$

where, γ_e is the incorrectly estimated SNR. The difference or "mismatch" between the true γ and inaccurately estimated γ_e SNRs is the reason behind the term signal to noise ratio mismatch (SNRM).

On the BIAWGN channel, the ith bit is transmitted as x_i and received from the channel as y_i. For decoders using BP decoding, every channel output y_i is translated into its equivalent LLR as

$$\text{LLR}_i = \frac{p(y_i|x = 1)}{p(y_i|x = -1)} = \frac{2y_i}{\sigma^2}.$$

After that, the LLR values of the received codeword is relayed to the decoding algorithm. If the value of the channel noise power σ^2 is inaccurately estimated as $\sigma_e^2 \neq \sigma^2$, then this leads to giving the BP decoder incorrect LLR values of received information, i.e., for y_i the decoder receives $\text{LLR}_i = 2y_i/\sigma_e^2$ instead of $\text{LLR}_i = 2y_i/\sigma^2$. The BPA algorithm performs optimally when perfect knowledge of the channel characteristics is available, on the other hand, SNRM degrades the performance of BPA and can lead to complete decoding failure if the mismatch is high enough.

In order to study the topic of SNRM we need to quantify its parameters; the degree of mismatch, the changes in the LLR messages exchanged during BP decoding, and the effects SNRM has on the decoding threshold and performance parameters such as bit error rate (BER). We start by defining the SNRM ratio η:

$$\eta = \frac{\gamma_e}{\gamma} = \frac{\frac{1}{2R\sigma_e^2}}{\frac{1}{2R\sigma^2}} = \frac{\sigma^2}{\sigma_e^2}. \tag{4.1}$$

When η is expressed in dB, it is called SNR offset (Υ),

$$\Upsilon = 10\log_{10}(\gamma_e) - 10\log_{10}(\gamma) \tag{4.2}$$

SNRM can be classified into two types:

1. SNR overestimation which occurs when $\gamma_e > \gamma$, or equivalently $\sigma_e^2 < \sigma^2$.

2. SNR underestimation which occurs when $\gamma_e < \gamma$, or equivalently $\sigma_e^2 > \sigma^2$.

4.1 SNRM Effects on Channel Codes

SNRM can affect any code that uses the channel SNR information in the decoding process such as Turbo, LDPC, or Raptor codes. The performance of Turbo and LDPC codes with BP decoding under SNR mismatch has been studied [18,22,35,36] and it was found that both codes are affected adversely at both positive and negative values of SNR offset, namely,

under SNR overestimation and underestimation. For Turbo and LDPC codes, it has been found that SNR underestimation is more detrimental compared to overestimation. Different approaches were proposed to mitigate the effects of SNRM on Turbo codes operating on the BIAWGN channel as can be seen in [37–39]. For LDPC codes, to handle the issue of SNR mismatch, some methods depend on modifying the decoding mechanism such as min-sum [14] or one of its derivatives [15–17]. These codes do not use the channel SNR information in decoding and depend only on the value of each symbol (y) observed directly from the channel. Other LDPC codes are designed by searching for degree distributions that are more tolerant to SNR mismatch at the cost of some degradation in the performance of the code when no SNR mismatch is present [18].

It is not difficult to check the degrading effects of SNRM on Raptor codes. Figure 4.1 shows the BER performance of a Raptor code with $k = 8000$, a (4,204) LDPC precode, and output distribution 4.24 given in [2]. The BER performance is shown for overall code rates 1/3, 1/2, and 5/7 with $E_b/N_o = 0.5$ dB, 1.25 dB, and 2.5 dB, respectively. As the SNR offset varies from -5 dB of underestimation to 0 dB (perfect noise power estimation i.e. no SNRM), and to 5 dB of overestimation. The degrading effects of SNR mismatch are clear and we can observe that if SNR mismatch is high enough, the decoding process collapses.

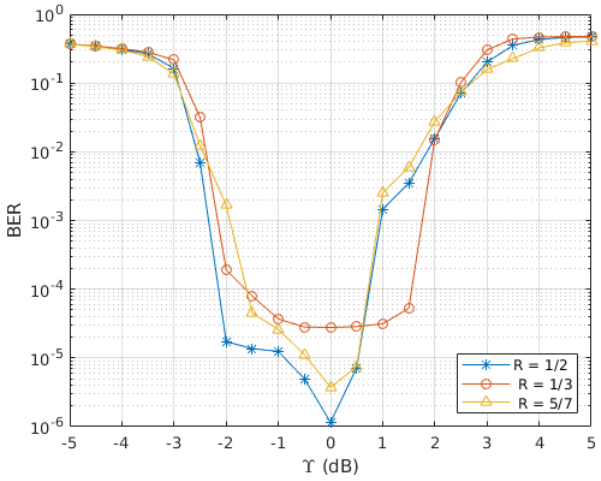

Figure 4.1. BER vs SNR offset.

Next, we turn our attention to study the effects of SNRM on Raptor codes, how it affects the performance, and how to design SNRM tolerant Raptor codes.

4.2 LLR Distribution Under SNRM

As it is customary for memoryless symmetric channels we assume an all-zero codeword was transmitted, i.e., $y_i \sim (1, \sigma^2)$ for the sake of simplifying the analysis. Since $\text{LLR} = 2y/\sigma^2$ we have:

$$\text{E[LLR]} = \frac{2}{\sigma^2}\text{E[y]} = \frac{2}{\sigma^2},$$

$$\text{Var(LLR)} = \text{Var}(\frac{2y}{\sigma^2}) = \frac{4\sigma^2}{\sigma^4} = \frac{4}{\sigma^2}.$$

The LLR values of received bits follow the Gaussian distribution $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$. This distribution has the property of the variance to mean ratio being equal to 2, and such a distribution is known as *consistent* in the literature [19]. However, if the channel noise power is incorrectly estimated as $\sigma_e^2 \neq \sigma^2$, the channel LLR will be computed as $\text{LLR} = 2y/\sigma_e^2$ and this will lead to

$$\text{E[LLR]} = \frac{2}{\sigma_e^2}\text{E[y]} = \frac{2}{\sigma_e^2},$$

$$\text{Var(LLR)} = \text{Var}(\frac{2y}{\sigma_e^2}) = \frac{4\sigma^2}{\sigma_e^4}.$$

With SNRM present, the LLR values will follow the distribution $\mathcal{N}(\frac{2}{\sigma_e^2}, \frac{4\sigma^2}{\sigma_e^4})$, instead of $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$, which is not consistent because the variance to mean ratio is not equal to 2 anymore, but rather, to $2\sigma^2/\sigma_e^2 = 2\eta$, and such a distribution is called *inconsistent*.

4.3 DDE of Raptor Codes Under SNR Mismatch

The performance and properties of codes that use the belief propagation algorithm (BPA), also known as sum-product algorithm (SPA), in the decoding process can be examined using the density evolution (DE) algorithm [19]. DE algorithm can simulate the evolution of an initial distribution in a bipartite graph on which BP decoding is run, where, the initial message distribution describes the message received from the channel. A quantized version of DE discretized density evolution (DDE) was introduced in [24] to offer a less complex yet still accurate version of DE. DDE simply assumes a discretized SPA algorithm, where, all the messages exchanged in decoding are quantized with quantization step equal to Δ and restricted to a range of values $[-L_q, -L_q + \Delta, \ldots, +L_q]$. The quantization step is defined as $\Delta = 2L_q/2^{n_b}$, where n_b is the number of quantization bits. In order to discretize the messages a quantization operator Q is used as follows. Let $Q(x)$ be the quantized value of x, then the quantization operator Q is defined as:

$$Q(x) = \begin{cases} \left\lfloor \frac{x}{\Delta} + \frac{1}{2} \right\rfloor \Delta, & \text{if } x \geq \frac{\Delta}{2} \\ \left\lceil \frac{x}{\Delta} - \frac{1}{2} \right\rceil \Delta, & \text{if } x \geq -\frac{\Delta}{2} \\ 0, & \text{otherwise;} \end{cases} \tag{4.3}$$

In order to study the performance of LT and Raptor codes in the presence of SNR mismatch, we will use DDE of Raptor codes proposed in [12], after implementing the needed modification as we will mention later. We assume tandem decoding for Raptor codes, where, the inner LT decoder receives the LLR values of output symbols collected from the channel and runs for I_{in} iterations, and then, passes the LLR values of intermediate symbols computed during this stage to be used as the initial LLRs in the outer LDPC decoder which will run for another I_o iterations.

4.3.1 Sum Product Decoding

As discussed earlier, in SP decoding the LLR message from jth check node to ith variable node is denoted as $L_{c_j v_i}$, and from ith VN to jth CN node as $L_{v_i c_j}$, and are calculated as

$$L_{c_j v_i} = 2 \tanh^{-1} \left(\left(\tanh(\frac{Lc_j}{2}) \right) \prod_{i' \neq i} \tanh(\frac{L_{v_{i'} c_j}}{2}) \right), \tag{4.4}$$

$$L_{v_i c_j} = Lv_i + \sum_{j' \neq j} L_{c_{j'} v_i}, \tag{4.5}$$

where, the product in (4.4) is over all the VNs connected to CN j other than VN i, and the sum in (4.5) is over all the CNs connected to VN i other than CN j. Lv_i and Lc_j are the channel LLR values of VN i and CN j, respectively.

Assuming a non-systematic Raptor code, for the inner decoder Lv_i is equal to zero $\forall\ i = 1, \ldots m$. The collected output symbols act as CNs and intermediate symbols as VNs and decoding is performed according to (4.4) and (4.5). After I_{in} iterations, the LLR of the ith VN of the inner LT decoder is computed as

$$L_{i,in} = \sum_{j} L_{c_j v_i}. \tag{4.6}$$

Then, the LLR values $L_{i,in}\ \forall\ i = 1, \ldots, m$ are passed as initial LLRs to the outer decoder. For the outer decoder, intermediate symbols act as VNs and the check nodes are given by the precode in use. $Lc_j = 0\ \forall\ 1, \ldots, m - k$, and iterations proceed according to (4.4) and (4.5) before concluding the decoding process by computing the final LLR for each VN of the outer decoder as

$$L_{i,out} = L_{i,in} + \sum_{j} L_{c_j v_i}. \tag{4.7}$$

At this point, the SP algorithm is finished and $L_{i,out}$ is used to decide on estimate for each of the k source bits.

4.3.2 DDE of Raptor Codes

According to the decoding sequence above, a DDE algorithm for Raptor codes can be devised which consists of two stages: inner and outer DDE [12]. Considering a Raptor code $(k, \mathcal{C}, \Omega(x))$, where k is the number of source symbols, $\mathcal{C}$ is the precode which is an LDPC code in our case, with the variable and check node edge distributions $\lambda^o(x)$ and $\omega^o(x)$, respectively. $\Lambda(x)$ and $\Omega(x)$ are the input and output node degree distributions of the inner LT decoder, while, $\lambda(x)$ and $\omega(x)$ are their respective edge distributions. The DDE algorithm starts by computing the initial or channel probability mass function (PMF) of the quantized LLR interval $[-L_q, +L_q]$ according to the distribution $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$, where, σ^2 is the noise power of the BIAWGN channel considered. Let $\bar{v}$ and $\bar{u}$ be the quantized versions of VN to CN $Q(L_{vc})$ and CN to VN $Q(L_{cv})$ randomly chosen messages, respectively. Also, let $p_{\bar{v}}$ and $p_{\bar{u}}$ be the PMFs of $\bar{v}$ and $\bar{u}$, respectively, i.e. the PMFs of exchanged quantized messages during our assumed discretized SPA decoding. The PMF $p_{\bar{v}}$ of degree d_v VN is computed as

$$p_{\bar{v}} = p_{\bar{u}_0} \overset{d_v - 1}{\bigotimes} p_{\bar{u}}, \tag{4.8}$$

where, $p_{\bar{u}_0}$ is the PMF of quantized channel LLR of VNs. The PMF $p_{\bar{u}}$ of degree d_c CN is computed as:

$$p_{\bar{u}} = \mathcal{R}(p_{\bar{v}_0}, \mathcal{R}(p_{\bar{v}}, \ldots, \mathcal{R}(p_{\bar{v}}, p_{\bar{v}}))) = \mathcal{R}(p_{\bar{v}_0}, \mathcal{R}^{d_c - 1} p_{\bar{v}}), \tag{4.9}$$

where, $p_{\bar{v}_0}$ is the PMF of quantized channel LLR of CNs. In order to compute 4.9 we define function $\Gamma(a, b) = Q\left(2 \tanh^{-1}(\tanh(a/2) \tanh(b/2))\right)$, where a and b are quantized values. Then, if $R(a, b) = \tau\Delta$, the PMF of $\mathcal{R}(a, b) = p_\tau$ is computed as:

$$p_\tau = \mathcal{R}(a, b) = \sum_{(i,j):\tau\Delta = \Gamma(i\Delta, j\Delta)} p_a[i] p_b[j] \tag{4.10}$$

As provided in [12], given a bipartite graph with edge distributions $\lambda(x)$ and $\omega(x)$ we

have $f_\omega(p) = \omega_1 p_{\bar{v}_0} + \sum_{j=2}^{d_c} \omega_j \left(\mathcal{R}(p_{\bar{v}_0}, \mathcal{R}^{(j-1)}p)\right)$ and $f_\lambda(p) = \lambda_1 p_{\bar{u}_0} + \sum_{i=2}^{d_v} \lambda_i \left(p_{\bar{u}_0} \bigotimes (\overset{i-1}{\bigotimes} p)\right)$ as the equations that define the evolution of PMFs of LLR messages at a CN following (4.4) and a VN following (4.5), respectively.

Assuming a non-systematic Raptor code $(k, \mathcal{C}, \Omega(x))$, the Raptor DDE algorithm starts by passing the PMF of channel LLR, i.e., $p_{\bar{v}_0}$ to the inner or LT part of DDE algorithm. The inner DDE runs (4.11) for I_{in} iterations:

$$p_{\bar{u},in}^{(l)} = f_\omega \left(f_\lambda(p_{\bar{u},in}^{(l-1)})\right). \tag{4.11}$$

After that, the PMF of the decision rule of the inner decoder is computed as

$$P_{\bar{D}} = \sum_{i=1}^{d_v} \Lambda_i \left(\overset{i}{\bigotimes} p_{\bar{u}}^{(I_{in})}\right).$$

Then, $P_{\bar{D}}$ is passed as initial PMF $p_{\bar{u}_0}$ to the outer or LDPC part of the DDE algorithm which runs (4.12) for $l = I_o$ iterations.

$$p_{\bar{v},o}^{l} = f_{\lambda^o}\left(f_{\omega^o}(p_{\bar{v},o}^{l-1})\right) \tag{4.12}$$

$p_{\bar{u},o}^{(I_o)}$ is extracted from 4.12 and used to compute the decision rule PMF of the outer decoder as $P_{\bar{D}} = \sum_{i=0}^{d_v} \Lambda_i \left(p_{\bar{u}_0} \overset{i}{\bigotimes} p_{\bar{u},o}^{(I_o)}\right)$. $P_{\bar{D}}$ is used to compute the final BER of decoded data as in (4.13), where t represents the quantized LLR values.

$$P_{\bar{e}} = \sum_{t} P_{\bar{D}}(t) \;\; \forall\, t \leq 0. \tag{4.13}$$

4.3.3 SPA Processing of Inconsistent Distributions

LLR values of channel output with SNRM are described by the distribution $\mathcal{N}(\frac{2}{\sigma_e^2}, \frac{4\sigma^2}{\sigma_e^4})$ instead of $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$. As we have discussed before, the LLR distribution $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ is called *consistent* because it has the property Var(LLR) = 2 E[LLR]. Such distributions describe the LLR values of all-zero codewords transmitted through a channel with perfect noise estimation. For channels with SNRM, LLR values follow the distribution $\mathcal{N}(\frac{2}{\sigma_e^2}, \frac{4\sigma^2}{\sigma_e^4})$ which is called *inconsistent* because:

$$\text{Var(LLR)} = \frac{\sigma^2}{\sigma_e^2} \times \text{E[LLR]} = 2\eta\text{E[LLR]}$$

A question that needs to be answered is whether BP decoding changes the variance to mean ratio as the decoder iterates between the VN and CN processors. The answer to this question tells us whether the output LLR values of the decoder still suffer from the same value of SNR mismatch as the input LLRs and the implications this has on serially concatenated codes such as Raptor codes, i.e., will the SNRM effects reach the outer decoder part?

In [19] it was proved that if the initial distribution of LLR is symmetric, then the evolution of distributions of LLR messages across the decoding graph is symmetric. The keyword here is symmetric, because the authors define a symmetric distribution as follows: Let u denote LLR values, then, the probability density function of u, i.e., $f(u)$ is symmetric if:

$$f(u) = e^u f(-u). \tag{4.14}$$

For a Gaussian random variable x with mean μ and variance θ, i.e. $x \sim \mathcal{N}(\mu, \theta)$, such as u, the relationship of (4.14) is a result of the fact that:

$$\frac{f(x)}{f(-x)} = \exp(\frac{2\mu x}{\theta}). \tag{4.15}$$

Then, for variables with *consistent* distributions, i.e. $\theta = 2\mu$, e.g. $u \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ we have:

$$\frac{f(x)}{f(-x)} = \exp(x),$$

$$\frac{f(u)}{f(-u)} = \exp(u). \tag{4.16}$$

Equation (4.16) is a test to check if an LLR distribution is symmetric or not, and as we will see below it is also a test to check whether a distribution is consistent, i.e., suffers from SNRM or not. The proof in [19] that a distribution that follows (4.16) will keep its symmetric properties as the BPA iterates between CNs and VNs can also be interpreted as a proof that the consistency properties will be held as well.

On the other hand, for *inconsistent* LLR distributions that we get from channels with SNRM we denote LLR values as $u_m \sim \mathcal{N}(\frac{2}{\sigma_e^2}, \frac{4\sigma^2}{\sigma_e^4})$. In this case we have:

$$\frac{f(u_m)}{f(-u_m)} = \exp\left(\frac{2\mu u_m}{\theta}\right) = \exp\left(\frac{2(\frac{2}{\sigma_e^2})u_m}{\frac{4\sigma^2}{\sigma_e^4}}\right) = \exp(\frac{u_m}{\frac{\sigma^2}{\sigma_e^2}}) = \exp(\frac{u_m}{\eta}). \tag{4.17}$$

From (4.17), we conclude that for inconsistent distributions we have the relationship:

$$f(u_m) = e^{\frac{u_m}{\eta}} f(-u_m). \tag{4.18}$$

Equation (4.18) creates a discrepancy with (4.16). It also invokes the question that if a distribution following (4.18) still keeps it symmetry and inconsistency (SNRM ratio value) properties through the BP decoding process. The authors in [19] provide a proof which can also be viewed as a test to check if an initial distribution of LLR, which we denote as $P_0(u)$, is symmetric and also prove that such a distribution keeps its symmetry and consistency properties throughout the BPA algorithm. We show the proof below because we will use it later for the inconsistent case.

Let u denote LLR values:

$$L(y) \quad = \quad u \quad = \quad \log\left(\frac{p(y|x=1)}{p(y|x=-1)}\right) \quad = \quad \log\left(\frac{p(-y|x=-1)}{p(-y|x=1)}\right) \quad = \quad -L(-y),$$

and the value of y can be extracted from u using the inverse function $L^{-1}(u)$.

$$e^u = \frac{p(y|x=1)}{p(y|x=-1)}.$$

Therefore,

$$\begin{aligned}
e^u P_0(-u) &= e^u p\left(y \in L^{-1}(-u)|x=1\right) \\
&= e^u p\left(-y \in L^{-1}(u)|x=1\right) \\
&= e^u p\left(y \in L^{-1}(u)|x=-1\right) \\
&= \frac{p\left(y \in L^{-1}(u)|x=1\right)}{p\left(y \in L^{-1}(u)|x=-1\right)} p\left(y \in L^{-1}(u)|x=-1\right) \\
&= p\left(y \in L^{-1}(u)|x=1\right) \\
&= P_0(u)
\end{aligned}$$

We follow the same steps to check if LLR distributions under SNRM which have the property $f(u_m) = e^{\frac{u}{\eta}} f(-u_m)$ exhibit the same symmetry properties as LLR distributions with no SNRM. However, before proceeding we need to clarify two points. First, since for LLR with no SNRM $u = 2y/\sigma^2$, and for LLR with SNRM $u_m = 2y/\sigma_e^2$, we have the relationship

$$\frac{u_m}{u} = \frac{\sigma^2}{\sigma_e^2} = \eta.$$

Second, it is crucial to understand that the probability of $u_m = 2y/\sigma_e^2$ is equal to the probability of $u = 2y/\sigma^2$ because the true probability of receiving u_m corresponds to $p(y|x=1)$ with $y \sim \mathcal{N}(1, \sigma^2)$ and not the miscalculated $y \sim \mathcal{N}(1, \sigma_e^2)$ due to mismatch.

Therefore, we also have:

$$P_0(u_m) = p(y|x = 1).$$

Taking all the above into consideration:

$$\begin{aligned} e^{u_m} P_0(-u_m) &= e^{\frac{u_m}{\eta}} p\left(y \in L^{-1}(-u)|x = 1\right) \\ &= e^{\frac{u_m}{\eta}} p\left(-y \in L^{-1}(\frac{u_m}{\eta})|x = 1\right), \end{aligned} \tag{4.19}$$

$$e^{u_m} P_0(-u_m) = e^{\frac{u_m}{\eta}} p\left(y \in L^{-1}(\frac{u_m}{\eta})|x = -1\right). \tag{4.20}$$

For simplicity, let $\alpha = \frac{u_m}{\eta}$ and then we can rewrite (4.20) as:

$$\begin{aligned} e^{u_m} P_0(-u_m) &= e^{\alpha} p\left(y \in L^{-1}(\alpha)|x = -1\right), \\ &= \frac{p\left(y \in L^{-1}(\alpha)|x = 1\right)}{p\left(y \in L^{-1}(\alpha)|x = -1\right)} p\left(y \in L^{-1}(\alpha)|x = -1\right), \\ &= p\left(y \in L^{-1}(\alpha)|x = 1\right), \end{aligned}$$

re-substituting α:

$$\begin{aligned} e^{u_m} P_0(-u_m) &= p\left(y \in L^{-1}(\frac{u_m}{\eta})|x = 1\right), \\ &= p\left(y \in L^{-1}(u)|x = 1\right), \\ &= P_0(u_m). \end{aligned}$$

From the discussion and analysis above we conclude that LLR values under SNRM satisfy the symmetry condition in the sense defined in [19], and therefore, keep their symmetry properties including SNRM ratio value throughout the BP decoding process. For example, considering the variable node processor where LLR values are added, and equivalently distributions are convoluted, the convolution of symmetric distributions is symmetric [19].

Let f and g be two symmetric distributions of LLR values with SNRM that satisfy

4.18, then:

$$\int_{-\infty}^{\infty} f(x-y)g(y)dy = \int_{-\infty}^{\infty} e^{\frac{x-y}{\eta}} f(y-x)e^{\frac{y}{\eta}} f(-y)dy = e^{\frac{x}{\eta}} \int_{-\infty}^{\infty} f(-x-y)g(y)dy. \tag{4.21}$$

A similar approach can be used to show that check node processors keep the symmetry properties of LLR values with SNRM. Based on the discussion above, we reach three conclusions:

1. For LLR densities $f(u) = e^{\frac{u}{\eta}} f(-u), u, \eta \in \mathbb{R}$ is a broader condition of symmetry for LLR densities on BIAWGN channel which includes LLR densities with SNRM as well.

2. For input LLR with SNRM ratio equal to η, the ratio is kept throughout the message passing algorithm and the output LLR has the same SNRM ratio η. This means that for serially concatenated codes, e.g. Raptor codes, SNRM effects reach the outer code.

3. For concatenated codes, this means that the degradation in performance or threshold comes from both the inner and outer codes.

Figure 4.2 below shows the empirically computed variance to mean ratio of LLR messages of the LT part of the DDE algorithm. The curves show the variance to mean ratio progress for three cases of initial SNRM: (a) $\Upsilon = -3$ dB i.e. $\eta = 0.5$, (b) $\Upsilon = 0$ dB i.e. $\eta = 1$, and (c) $\Upsilon = 3$ dB i.e. $\eta = 2$. As expected, once all the check nodes start to produce non-zero messages, the variance to mean ratio stabilizes to the same value as of the initial (channel) LLR distribution, and the same ratio is kept as decoding iterations proceed. Since the decision LLR values of the inner decoder are passed as initial LLR values to the outer LDPC decoder, this supports our earlier conclusion that the SNRM effects reach the outer decoder.

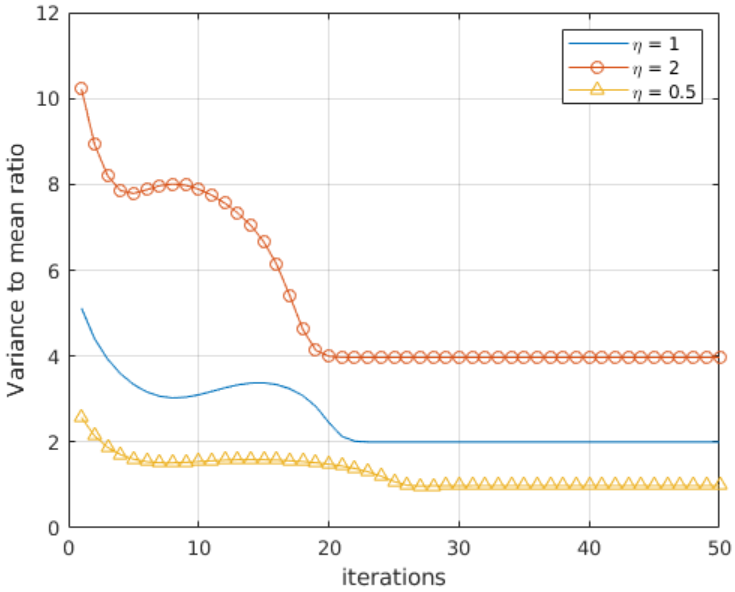

Figure 4.2. Variance to mean ratio of the inner decoder.

4.4 Modifying DDE for Distributions with SNRM

Having introduced the DDE algorithm of Raptor codes, we turn our attention to study how the asymptotic performance of BP decoding will react to input LLR values with SNRM ratio η. Examining the DDE algorithm we described, we observe that the algorithm can be modified to work with densities representing LLRs with SNRM ratio η by replacing the input distribution to the DDE algorithm from $(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ representing LLRs with no SNRM to $(\frac{2}{\sigma_e^2}, \frac{4\sigma^2}{\sigma_e^4})$ corresponding to input LLR with SNRM ratio $\eta = \sigma^2/\sigma_e^2$.

In order to understand the effects of SNR mismatch on a Raptor code $(k, \mathcal{C}, \Omega(x))$ we need to quantify its performance under a given value of SNR offset Υ. One way to quantify the performance of Raptor codes on the BIAWGN channel is to determine the SNR decoding threshold. SNR decoding threshold is defined as the infimum of all SNR values such that the BER converges to zero as the number of iterations approaches infinity. Using the DDE algorithm, we can determine the SNR threshold as the infimum of all SNR values such that 4.13 converges to zero.

Equivalently, as was shown in [12], for a serially concatenated code such as Raptor code, the threshold can be defined as infimum of all SNR values such that the decoded BER of the inner decoder converges to a value known as the critical BER of the outer decoder as the number of iterations approaches infinity. The critical BER of the outer decoder can be calculated by determining its threshold σ^o_{tsh}, then, the critical BER can be computed as $Q(\frac{1}{\sigma^o_{\text{tsh}}})$.

$$\sigma^o_{\text{tsh}} = \left(\frac{1}{2R_o E^o_{\text{tsh}}}\right)^{\frac{1}{2}} \tag{4.22}$$

$$P^o_{\text{e,tsh}} = Q\left(\frac{1}{\sigma^o_{\text{tsh}}}\right) \tag{4.23}$$

where, $E^o_{\text{tsh}}(R_o)$ is the E_b/N_o threshold of the outer LDPC code with rate R_o. Applying this approach to determine the threshold of Raptor codes in the case of SNRM requires further attention, because for each value of SNR offset the decoding threshold, and hence, the critical BER ($P^o_{\text{e,tsh}}$) of the outer code will be different. Assuming we use a regular $(4, 204)$ LDPC code as the outer code, in Table 4.1 we give the threshold and corresponding $P^o_{\text{e,tsh}}$ values for the SNR offset (Υ) values shown.

SNR offset (Υ)	Threshold	$P^o_{\text{e,tsh}}$
-5	0.3220	9.495×10^{-4}
-4	0.3401	1.6×10^{-3}
-3	0.3555	2.5×10^{-3}
-2	0.3681	3.3×10^{-3}
-1	0.3737	3.73×10^{-3}
0	0.3750	3.8×10^{-3}
1	0.3735	3.7×10^{-3}
2	0.3717	3.6×10^{-3}
3	0.37	3.45×10^{-3}
4	0.3695	3.4×10^{-3}
5	0.36711	3.2×10^{-3}

Table 4.1. Decoding thresholds and equivalent critical BERs of $(4, 204)$ LDPC code under SNRM.

We used both methods to determine the threshold of Raptor codes under SNRM and the results are almost identical with a maximum difference of 0.003 dB. From another perspective, this also further confirms our previous observation that SNRM affects both the inner and outer decoder of Raptor codes.

Next, we will use the modified DDE algorithm to study the asymptotic performance of Raptor codes under different values of SNRM ratios and investigate if it is possible to design SNRM tolerant Raptor codes.

4.4.1 SNRM Effects on the Threshold

As in [22] we define $\alpha(\Upsilon)$ to be the E_b/N_o threshold for an SNR offset value Υ. In order to reach more general conclusions, we considered four different example distributions to study the effects of SNRM on the threshold of Raptor codes. As shown below, the distributions are $\Omega_1(x)$, $\Omega_3(x)$ and $\Omega_4(x)$ from [2], [40] and [11], respectively. $\Omega_2(x)$ varies for each instance of the code rates tested and can be found in [12].

$$\begin{aligned}\Omega_1(x) = {} & 0.007969x + 0.49357x^2 + 0.1662x^3 + 0.072646x^4 + 0.082558x^5 + 0.056058x^8 \\ & + 0.037229x^9 + 0.05559x^{19} + 0.025023x^{65} + 0.003135x^{66}\end{aligned} \tag{4.24}$$

$$\begin{aligned}\Omega_3(x) = {} & 0.0082x + 0.5019x^2 + 0.043x^3 + 0.2365x^4 + 0.0067x^5 + 0.0911x^8 + 0.0398x^{14} \\ & + 0.0108x^{30} + 0.0273x^{33} + 0.0347x^{197}\end{aligned} \tag{4.25}$$

$$\begin{aligned}\Omega_4(x) = {} & 0.0791x + 0.4560x^2 + 0.1916x^3 + 0.0564x^4 + 0.0449x^5 + 0.0252x^8 + 0.0376x^9 \\ & + 0.0825x^{19} + 0.0165x^{65} + 0.0102x^{66}\end{aligned} \tag{4.26}$$

Figures 4.3, 4.4, 4.5,4.6,4.7, and 4.8 show the SNR threshold of the distributions above for SNR offset $\Upsilon = [-5, 5]$ dB, and for code rates 1/3, 2/5, 1/2, 5/9, 5/8, and 5/7, respectively.

Examining the figures below, we observe that as $|\Upsilon|$ increases the E_b/N_o threshold increases. Higher values of E_b/N_o help the SP decoding algorithm to encounter the effects of SNRM, however, this is accomplished through different mechanisms for underestimation ($\Upsilon < 0$) and overestimation ($\Upsilon < 0$). For $\Upsilon < 0$, the estimated noise power is larger than the true noise power i.e. $\sigma_e^2 > \sigma^2$ and the computed LLR values are less than their true values. Higher E_b/N_o helps reduce the fraction of received bits with flipped sign which is equal to $Q(1/\sigma)$, and also helps by reducing the fraction of lower LLR values, i.e. the values highly impacted by SNR underestimation, by concentrating the received channel output y around $+1$ and away from 0, leading to counteracting the effects of underestimation. For $\Upsilon > 0$, the estimated $\sigma_e^2 < \sigma^2$ and the computed LLR values are higher than their true values. This gradually turns the decoder into a BEC channel decoder with a fraction of $Q(1/\sigma)$ undetected erroneous bits. Therefore, higher E_b/N_o values help in counteracting the effects of SNR overestimation by mainly reducing the fraction $Q(1/\sigma)$ of undetected erroneous bits.

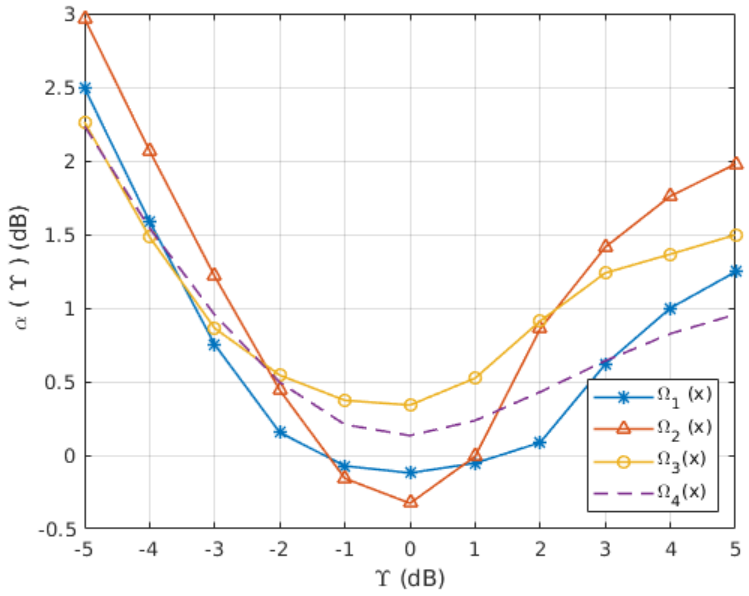

Figure 4.3. SNR threshold vs SNR offset with $R = 1/3$.

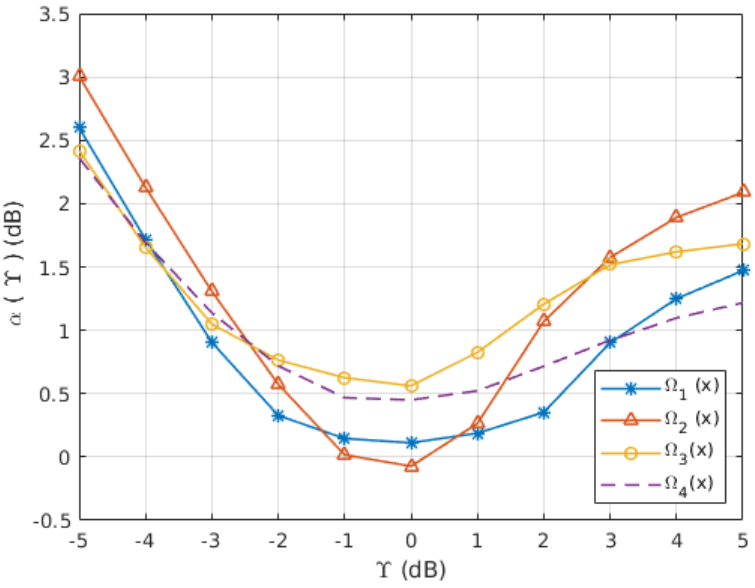

Figure 4.4. SNR threshold vs SNR offset with $R = 2/5$.

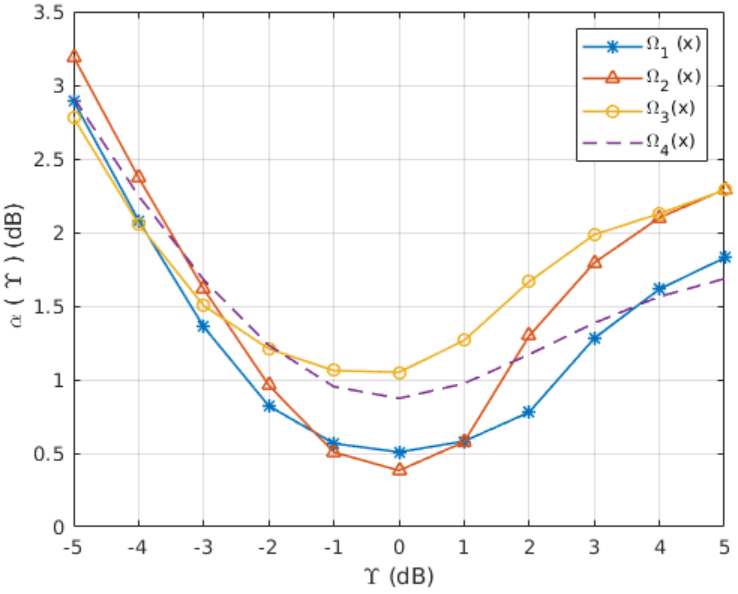

Figure 4.5. SNR threshold vs SNR offset with $R = 1/2$.

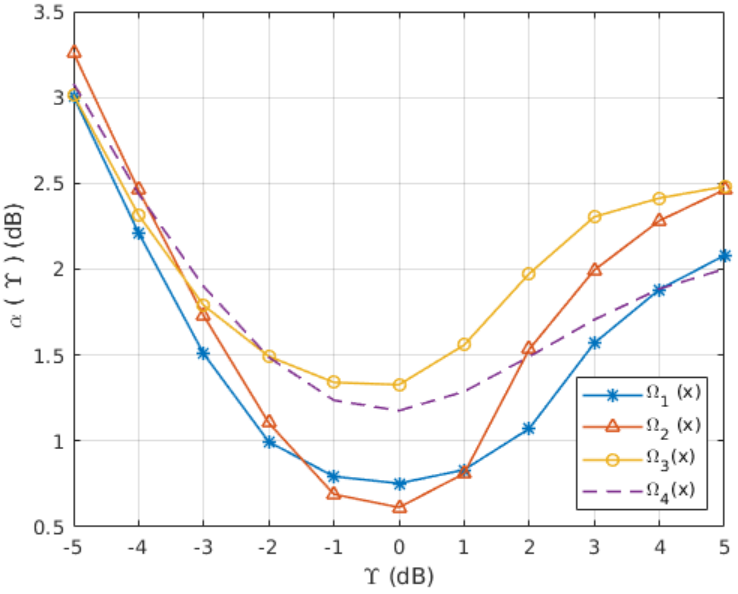

Figure 4.6. SNR threshold vs SNR offset with $R = 5/9$.

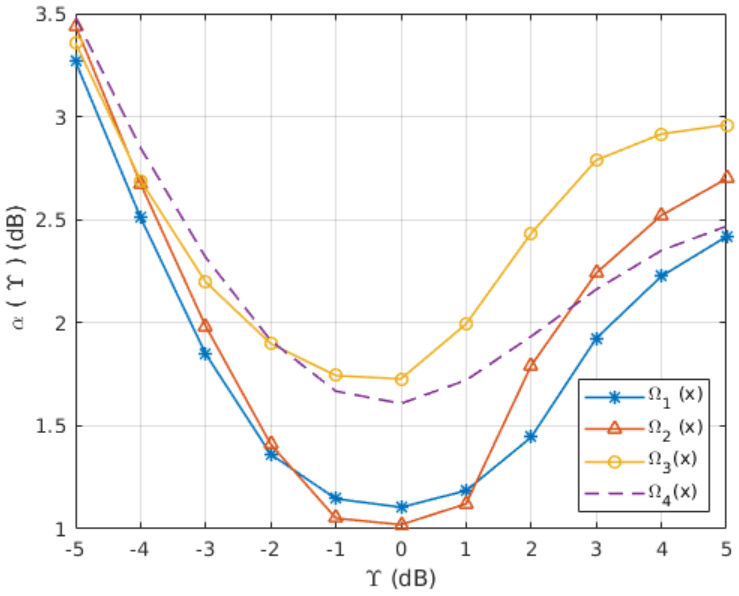

Figure 4.7. SNR threshold vs SNR offset with $R = 5/8$.

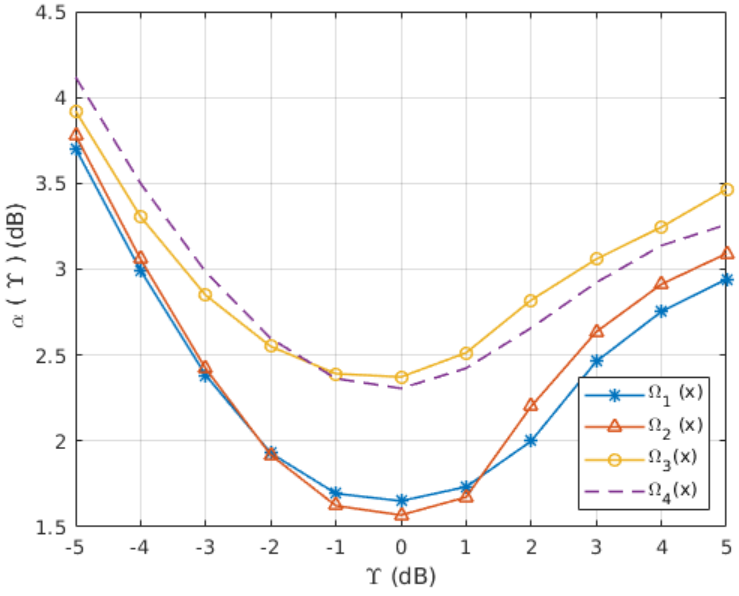

Figure 4.8. SNR threshold vs SNR offset with $R = 5/7$.

From what we learned so far, we also can observe the following:

- As $|\Upsilon|$ increases, SNR underestimation becomes more degrading compared to overestimation.
- In the region around $\Upsilon = 0$ and approximately within a distance of 3 dB in both directions, the response is almost symmetric with a slightly less degradation on the underestimation part.
- There is an inverse relationship between the performance of a code at $\Upsilon = 0$, and higher values of $|\Upsilon|$. This is more obvious for lower rates, i.e. 1/3 and 1/2.
- Output distribution 4.24 seems to provide the best average performance between the tested distributions. On the other hand, $\Omega_2(x)$ seems to offer the lowest SNR threshold for no mismatch and lower values of $|\Upsilon|$.

These observations will give helpful insights when designing SNRM tolerant output distributions for the inner LT part of Raptor codes.

CHAPTER 5

DESIGNING SNRM TOLERANT RAPTOR CODES

In the previous chapter, we studied the performance of Raptor codes in the presence of SNRM for different output degree distributions and code rates. We observed that the performance of Raptor codes under SNRM is considerably affected by the output distribution used by the inner (LT) decoder and that some distributions have better performance compared to the others. This suggests the question if it is possible to design output degree distributions that can be used to construct Raptor codes with higher tolerance to SNRM. To answer this question, we formulated our search for SNRM tolerant output degree distributions in the form of an optimization problem. Below, we give the details of setting up the optimization program.

5.1 Optimization for the SNRM Case

As we learned, for Raptor codes and high enough $|\Upsilon|$, SNR underestimation ($\Upsilon < 0$) is more detrimental compared to SNR overestimation ($\Upsilon > 0$). The threshold at some $\Upsilon = -a$, i.e., $\alpha(-a)$ has the property that $\alpha(-a) \geq \alpha(x), x \in (-a, a]$. Therefore, improving the threshold performance at $\Upsilon = -a$ is expected to result in improvement for $\Upsilon \in [-a, a]$. Therefore, if we are interested in designing a Raptor code that is more tolerant to SNRM within the range $[-a, a]$, we set up the optimization problem such that the channel is assumed to have SNR offset of $\Upsilon = -a$. A similar approach was successfully applied to LDPC codes in [18].

5.1.1 Objective

The objective is: Design an output degree distribution ($\Omega(x)$) s.t. for a given SNR offset Υ, $P_e^{(l)}$ defined in 4.13 goes to zero at the minimum E_b/N_o possible.

5.1.2 Formulating the Optimization

In order to optimize a Raptor code $(k, \mathcal{C}, \Omega(x))$ at an instance of the realized code rate R, we fix the outer code $\mathcal{C}$ as a regular LDPC code $(4, 204)$ with rate $R_o = 50/51$, and therefore, the inner code is assigned the rate $R_i = R/R_o$. Regarding the SNRM side of the problem, we fix the SNR offset (Υ) at the desired value, and compute the equivalent SNRM ratio $\eta = 10^{(\Upsilon/10)} = \sigma^2/\sigma_e^2$. Then, we incorporate into the optimization by using the initial distribution $\mathcal{N}(\frac{2}{\sigma^2}\eta, \frac{4}{\sigma^2}\eta^2)$ instead of the $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ used when perfect SNR estimation is assumed. This reduces the optimization problem to optimizing the output degree distribution $\Omega(x)$.

The optimization problem: for the fixed parameters given above, the output degree distribution of the inner LT code is optimized as described in the objective above. The constraints are: (a) $\Omega_i \geq 0 \forall i \in [1, k]$, (b) $\sum_{i=1}^{k} \Omega_i = 1$, (c) $\omega(1 - \lambda(1 - x)) > x \forall x \in (0, 1)$. The first and second constraints provide the conditions necessary for a valid distribution. The third condition ensures successful decoding on the LT part and recovery of all input symbols. Also, since $\Omega_1(x)$ shows the best performance on average considering all the tested distributions, we use it as the initial distribution.

5.1.3 Results

Running the optimization program for a code rate half, and at three points of SNR offset $\Upsilon = -5$ dB, -3 dB, and -2 dB, We obtained the distributions $\Omega_{o1}(x)$, $\Omega_{o2}(x)$, and $\Omega_{o3}(x)$ given below, respectively.

$$\begin{aligned}\Omega_{o1}(x) = {} & 0.071892x^1 + 0.41454x^2 + 0.16x^3 + 0.16322x^4 + 0.07298x^5 + 0.0079x^6 \\ & + 0.00765x^7 + 0.0055x^8 + 0.00307x^9 + 0.0028x^{10} + 0.00166x^{11} + 0.002x^{14} \\ & + 0.016547x^{15} + 0.025x^{19} + 0.0102x^{20} + 0.013785x^{30} + 0.008x^{40} + 0.009x^{50} \\ & + 0.00369x^{60}\end{aligned} \tag{5.1}$$

$$\begin{aligned}\Omega_{o2}(x) = {} & 0.007837x^1 + 0.512x^2 + 0.1905x^3 + 0.074644x^5 + 0.1067x^6 + 0.066017x^{15} \\ & + 0.015253x^{19} + 0.026983x^{70}\end{aligned} \tag{5.2}$$

$$\begin{aligned}\Omega_{o3}(x) = {} & 0.00799x^1 + 0.49123x^2 + 0.1812x^3 + 0.043467x^4 + 0.10232x^5 + 0.024349x^6 \\ & + 0.002872x^7 + 0.00101x^8 + 0.025133x^{10} + 0.041566x^{11} \\ & + 0.033736x^{12} + 0.04514x^{40}\end{aligned} \tag{5.3}$$

5.1.4 Performance Comparison

Figure 5.1 shows the E_b/N_o threshold for $\Upsilon = [-5, 5]$ dB of Raptor code with the distribution $\Omega_{o1}(x)$ compared to the threshold of the initial distribution (4.24) used in the optimization. The output distribution $\Omega_{o1}(x)$ is optimized for $\Upsilon = -5$, and the gain achieved at points $\Upsilon = -5$ dB, and 5 dB is 0.3 and 0.6 dB, respectively. However that comes at a cost for the threshold at $\Upsilon = 0$ dB of 0.2 dB increase. Also, we notice that the gain achieved at higher values of Υ i.e. around $\Upsilon = -5$, and 5 dB comes at a cost for the threshold in the region of $\Upsilon = 0$ dB, specifically for $\Upsilon \in [-3, 2.5]$ dB.

The E_b/N_o threshold response of $\Omega_{o1}(x)$ to SNR offset led us to try and improve the performance within the region $\Upsilon \in [-3, 3]$ dB. $\Omega_{o1}(x)$ is optimized for relatively high value of $\Upsilon = -5$ dB which leads to degrading the threshold at lower values of Υ. Therefore, $\Omega_{o2}(x)$ was optimized for $\Upsilon = -3$ dB. As can be observed in Figure 5.2, the gain in threshold at higher values values of Υ is less compared to $\Omega_{o1}(x)$, however, the loss for lower values of Υ is less as well.

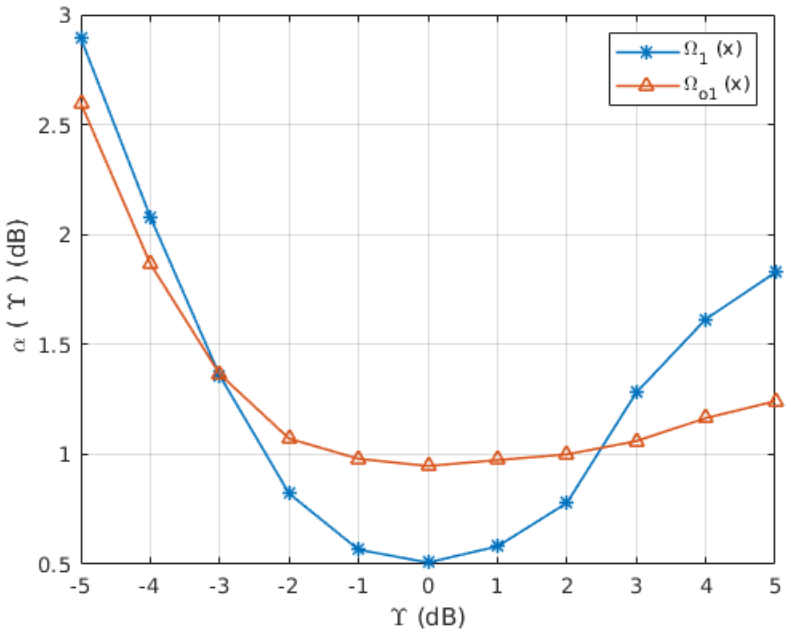

Figure 5.1. SNR threshold vs SNR offset of $\Omega_1(x)$ and $\Omega_{o1}(x)$.

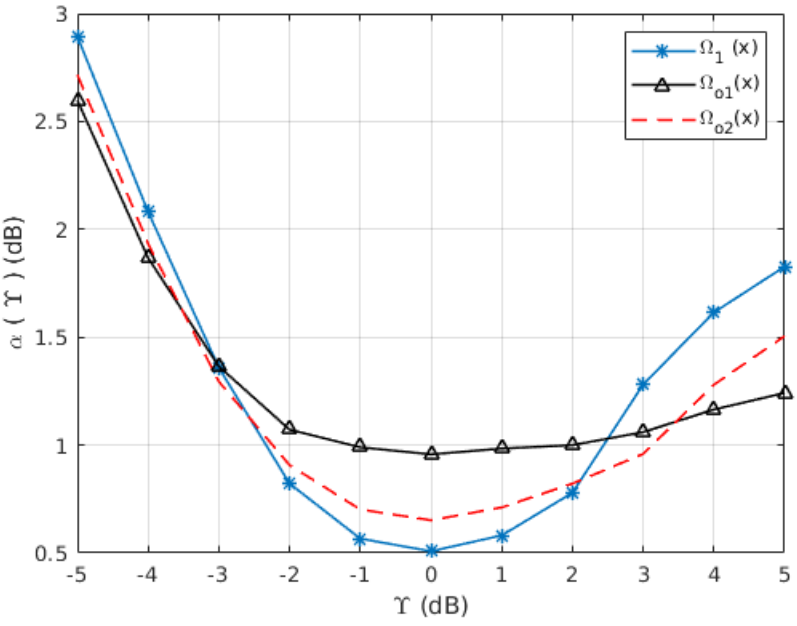

Figure 5.2. SNR threshold vs SNR offset of $\Omega_1(x)$, $\Omega_{o1}(x)$, and $\Omega_{o2}(x)$.

Trying to optimize the output degree distribution for lower values of SNR offset such as $\Upsilon = -2$, offers no considerable change in the performance, e.g., $\Omega_{o3}(x)$ was optimized for $\Upsilon = -2$ dB and the threshold response can be seen in Figure 5.3.

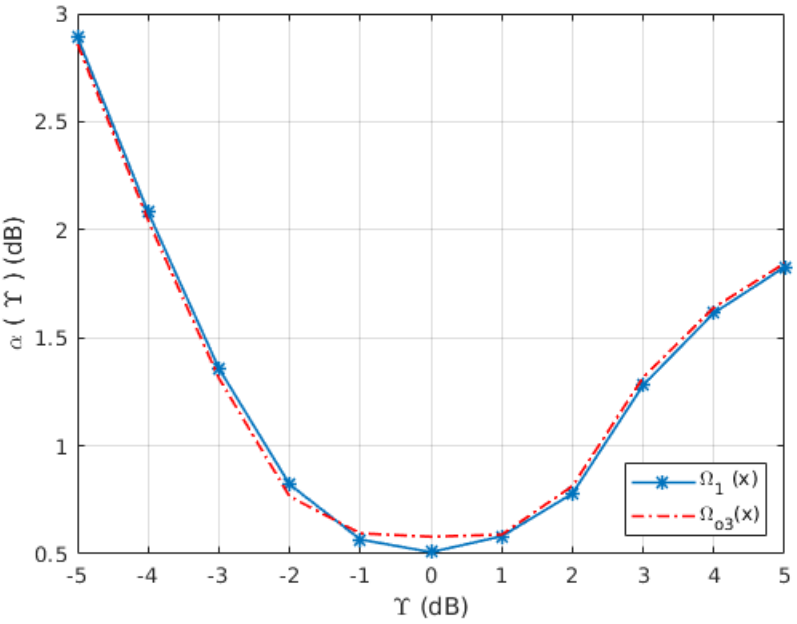

Figure 5.3. SNR threshold vs SNR offset of $\Omega_1(x)$ and $\Omega_{o3}(x)$.

The optimized output distributions $\Omega_{o4}(x)$ and $\Omega_{o5}(x)$ were obtained by optimizing at realized code rates 1/3 and 5/7, respectively, and with $\Upsilon = -5$ dB. Figures 5.4 and 5.5 show the threshold versus offset graphs of Raptor codes with the optimized distributions and $\Omega_1(x)$. The behavior of the codes is similar to that we previously saw with $\Omega_{o1}(x)$.

$$\begin{aligned}\Omega_{o4}(x) = {} & 0.010885x^1 + 0.5742x^2 + 0.2173x^3 + 0.011625x^4 + 0.008830x^5 \\ & + 0.004450x^7 + 0.020398x^8 + 0.024381x^9 + 0.032298x^{10} + 0.026729x^{11} \\ & + 0.020295x^{12} + 0.013272x^{13} + 0.029369x^{70}\end{aligned} \tag{5.4}$$

$$\Omega_{o5}(x) = 0.007674x^1 + 0.516220x^2 + 0.213390x^3 + 0.00082x^4 + 0.061620x^5 + 0.060650x^6 + 0.021417x^7 + 0.012584x^8 + 0.044843x^9 + 0.022504x^{20} + 0.038279x^{12} \tag{5.5}$$

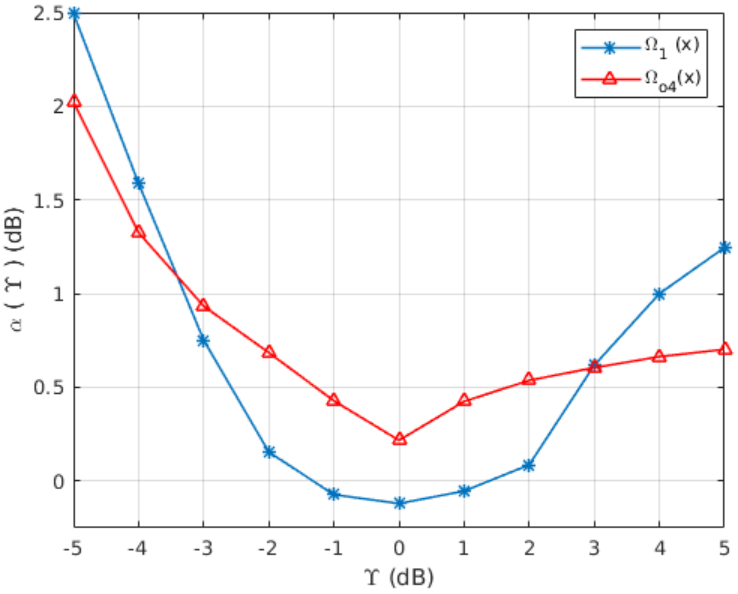

Figure 5.4. SNR threshold vs SNR offset of $\Omega_1(x)$ and $\Omega_{o4}(x)$.

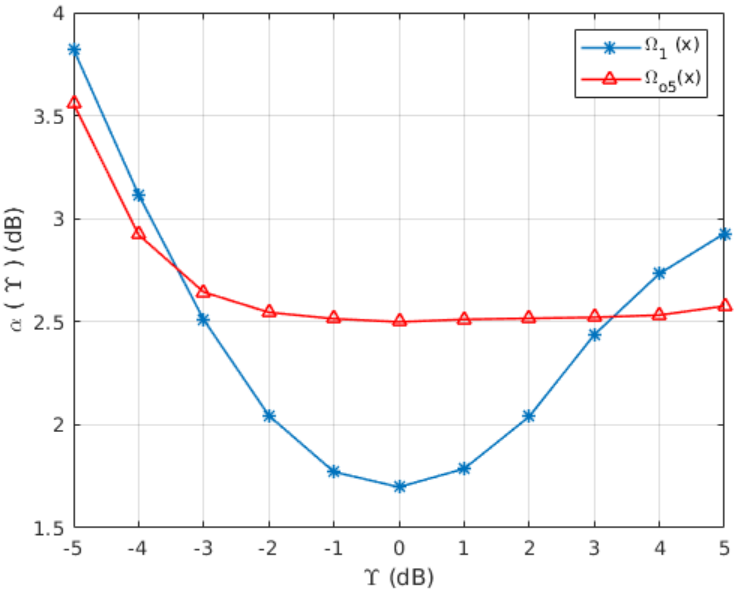

Figure 5.5. SNR threshold vs SNR offset of $\Omega_1(x)$ and $\Omega_{o5}(x)$.

The performance of the optimized output distributions at code rate instances other than the rate they are optimized at is another aspect we tested. This shows the impact of the variation in the code rate on the optimized distribution. Figures 5.6 and 5.7 show the performance of $\Omega_{o1}(x)$ optimized for $R = 1/2$ at code rates 1/3 and 5/7. As can be seen, for high values of SNR offset the optimized distribution still performs better than other distributions.

Figure 5.8 shows the comparison of BER performance between the distributions $\Omega_1(x)$, and $\Omega_{o1}(x)$. The code parameters are; $k = 10000$, a regular $(4, 204)$ LDPC precode, and realized code rate half. For high levels of SNRM, the optimized distribution outperforms other distributions which enables Raptor codes to have higher tolerance to SNRM.

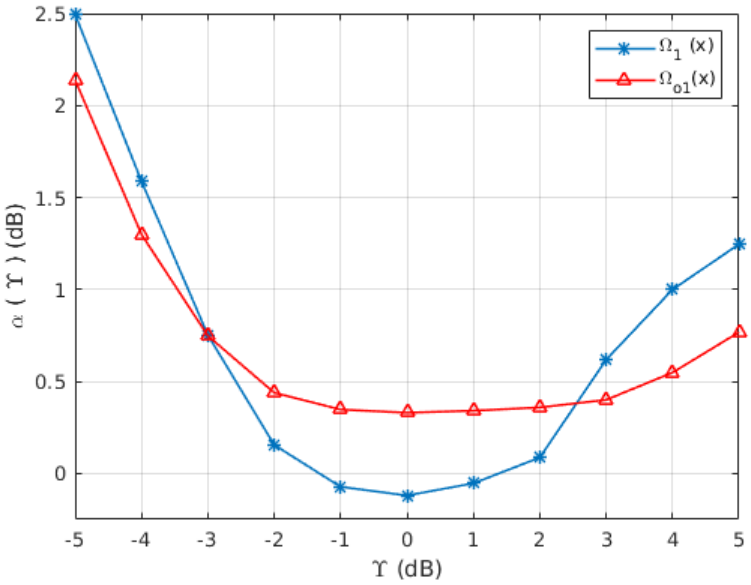

Figure 5.6. Testing the performance of $\Omega_{o1}(x)$ at R = 1/3.

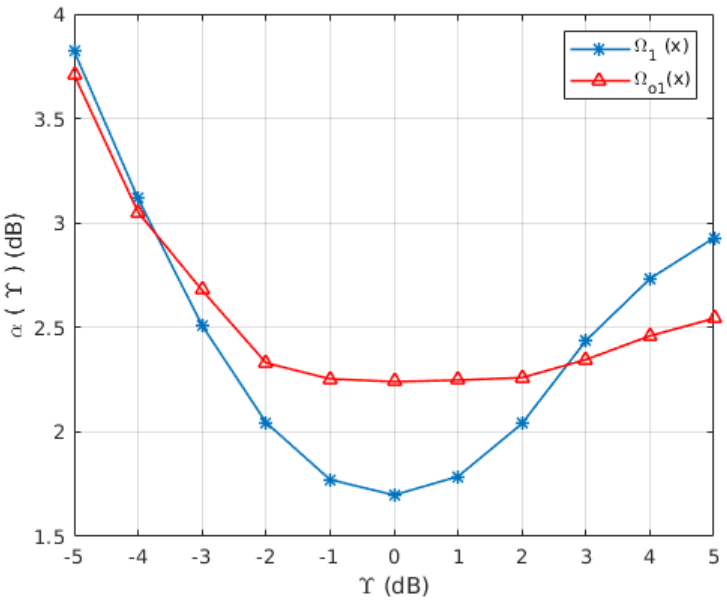

Figure 5.7. Testing the performance of $\Omega_{o1}(x)$ at R = 5/7.

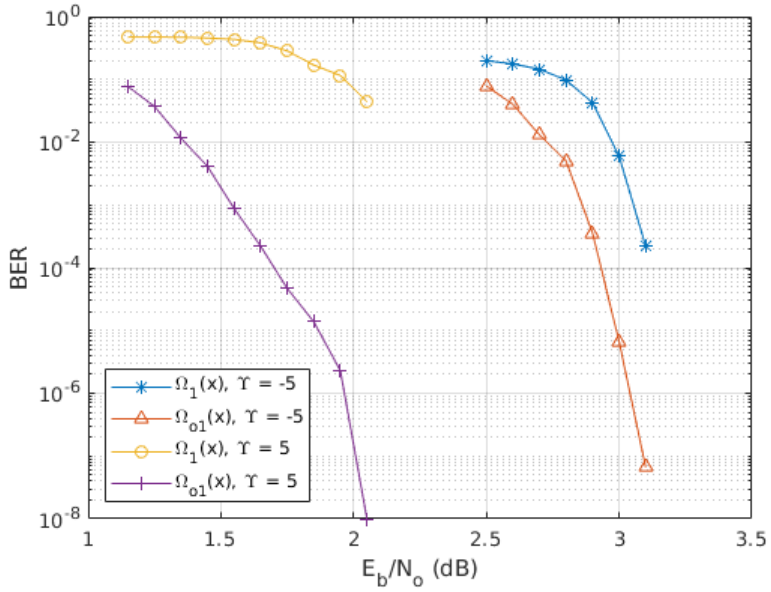

Figure 5.8. BER performance comparison between the optimized distribution and $\Omega_1(x)$.

5.1.5 Limits of the Decoding Threshold

As can be seen in Figure 5.3, when optimizing within the range $\Upsilon \in [-2, 2]$ dB, the achievable gain in decoding threshold is very limited. This suggests that a lower bound for the E_b/N_o threshold achievable at each value of Υ can be found by searching for valid output distributions that can be used to design Raptor codes capable of successful decoding, at the lowest E_b/N_o possible. The search process can be accomplished using the optimization program described earlier. For a Raptor code with outer code fixed as a $(4, 204)$ LDPC code, Figure 5.9 shows the lower bounds on E_b/N_o values we found by using this approach. Applying the E_b/N_o lower bound curves we obtained in Figure 5.9 to the tested and optimized distributions we previously obtained the decoding thresholds, we obtain the curves shown in Figures 5.11, 5.10, 5.12 and for the code rates $R = 1/3, 1/2$, and $5/7$, respectively.

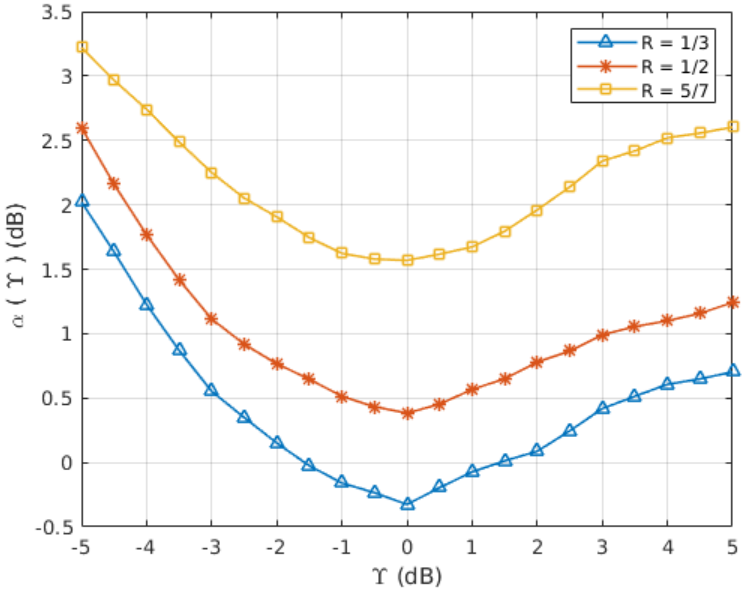

Figure 5.9. Lowest E_b/N_o values where valid output degree distributions could be found.

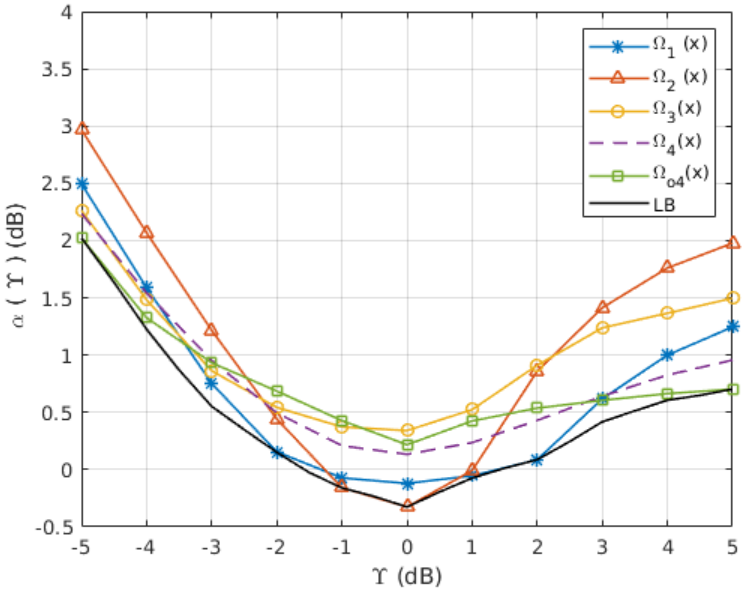

Figure 5.10. SNR threshold vs SNR offset with $R = 1/3$. LB: lower bound.

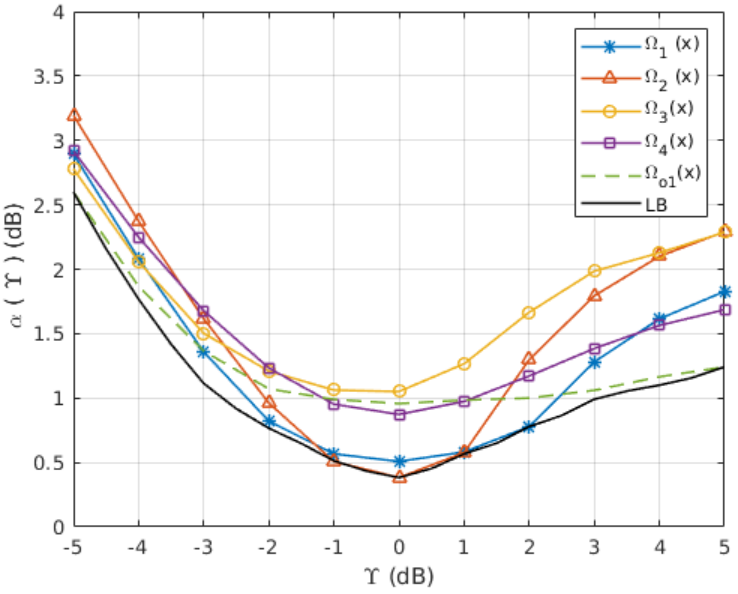

Figure 5.11. SNR threshold vs SNR offset with $R = 1/2$. LB: lower bound.

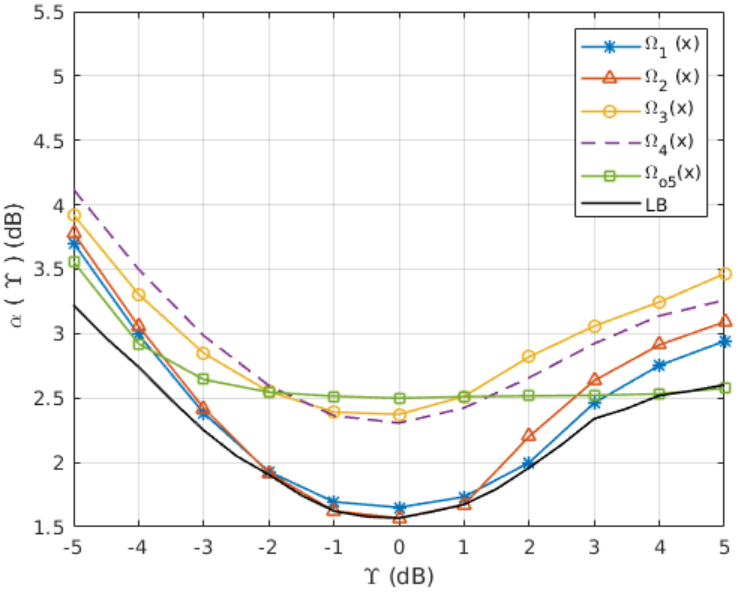

Figure 5.12. SNR threshold vs SNR offset with $R = 5/7$. LB: lower bound.

Determining such a lower bound gives us a general evaluation of the decoding thresholds of Raptor codes under SNRM.

Finally, considering the various output degree distributions of Raptor codes we examined, and the distribution we designed in order to offer better tolerance to high levels of SNRM, we can arrive at few conclusions. Optimizing the degree distributions for the purpose of improving the decoding threshold and performance at higher values of SNR offset (Υ) comes at a cost to the performance when no SNRM is present and lower values of Υ as can be seen in Figures 5.1, 5.2, and 5.4. In channels where high levels of SNRM are likely to appear our optimized distributions can outperform other distributions as can be seen in 5.1, 5.4. However, the output distribution given in [2] offers better performance on average among the distributions we tested. By comparing the asymptotic performance of Raptor code we provided in our work and the asymptotic performance of irregular LDPC codes given in [18], we observe that Raptor and irregular LDPC codes have a similar response to SNRM in the case of SNR underestimation, while, irregular LDPC codes offer better performance in the case of SNR overestimation.

CHAPTER 6

OPTIMAL INNER AND OUTER RATES OF RAPTOR CODES

Raptor codes are serially concatenated codes consisting of an inner LT code with output distribution $\Omega(x)$, and an outer code that is usually a regular LDPC code. The inner code is rateless and thus the inner rate R_i can potentially change to any value in the range $[0, 1]$. On the other hand, the outer code rate R_o is usually fixed and fulfilled by the outer code. At any instant, the overall realized rate of a Raptor code is $R = R_i R_o$.

Figure 6.1. Structure of Raptor codes.

Any instance of the overall code rate R can be realized by different inner and outer code rate pairs (R_i, R_o) as long as $R = R_i R_o$. In other words, for a Raptor code different outer codes can be used in the design. Given a Raptor code with output distribution $\Omega(x)$ of the inner code, the rate pair chosen play a significant role in determining the performance and threshold. For the sake of demonstration, given a Raptor code with inner code output distribution $\Omega_1(x)$ of (4.24), an LDPC outer code with node degrees (d_v, d_c), and overall code rate $R = 0.5$, if we determine the E_b/N_o thresholds of the code for the rate pairs (R_i, R_o) given below, such that each outer code rate is realized by the corresponding node degrees given in Table 6.1.

$$
\begin{aligned}
(R_i, R_o) = \{&(0.5/0.75, 0.75), (0.5/0.8, 0.8), (0.5/0.85, 0.85), (0.5/0.9, 0.9) \\
&, (0.5/0.925, 0.925), (0.5/0.95, 0.95), (0.5/0.96, 0.96), (0.5/0.9, 0.97) \\
&, (0.5/0.9, 0.98), (0.5/0.99, 0.99)\}.
\end{aligned}
$$

R_o	(d_v, d_c)
0.99	(4,400)
0.98	(4,204)
0.9699	(4,133)
0.96	(4,100)
0.95	(4,80)
0.9245	(4,53)
0.9	(4,40)
0.8519	(4,27)
0.8	(4,20)
0.75	(4,16)

Table 6.1. Regular LDPC codes and their respective rates.

Then, we obtain the decoding thresholds shown in Figure 6.2 for each rate pair. In Figure 6.2, for the sake of simplicity, only the outer code rate is indicated but since the overall realized code rate R is defined as $R = 0.5$, then, the inner rate can be inferred as $R_i = 0.5/R_o$.

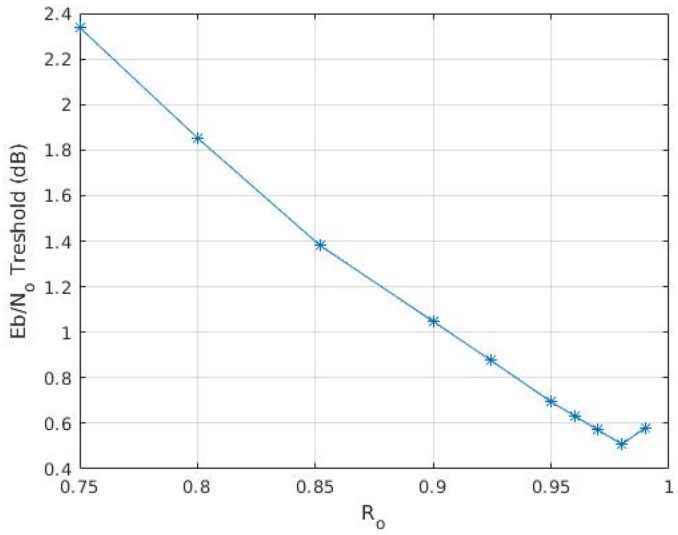

Figure 6.2. Decoding threshold vs code rate pair (R_i, R_o).

Figure 6.2 shows that for the Raptor code with parameters specified above, the optimal rate pair in the set given above is $(0.5/0.98, 0.98)$ since the minimum E_b/N_o threshold can be achieved with this pair. Therefore, a question to be asked is that given a Raptor code with a fixed output distribution, what is the best rate pair to use? Also for some instance

of R and given the set of all rate pairs (R_i, R_o) s.t. $R = R_i R_o$, what is the optimal output degree distribution $\Omega(x)$ and rate pair combination? Before trying to answer these questions we turn to a more restricted one, given an overall rate R and a fixed output distribution $\Omega(x)$, what is the effect of a given rate pair on the threshold and how the optimal rate pair that achieves the lowest E_b/N_o threshold is decided?

6.1 Effect of Rate Pair Choice on Raptor Codes

Assume we are given a Raptor code with output degree distribution $\Omega(x)$, an outer code $\mathcal{C}$, and some overall rate R. Choosing an outer code with rate R_o leads to fixing the E_b/N_o threshold value for the outer decoder. We denote the decoding threshold as $E^o_{\text{tsh}}(R_o)$ in order to signify that we consider as a function of the outer rate chosen. For concatenated codes such as Raptor codes, the threshold of the outer code can be translated into an equivalent BER known as the critical BER which we denote as ($\mathrm{P}^o_{\text{e,tsh}}$) [12]. The value of $\mathrm{P}^o_{\text{e,tsh}}$ can be calculated as follows

$$\sigma^o_{\text{tsh}}(R_o) = \left(\frac{1}{2R_o E^o_{\text{tsh}}(R_o)}\right)^{\frac{1}{2}},$$

$$P^o_{\text{e,tsh}} = Q\left(\frac{1}{\sigma^o_{\text{tsh}}(R_o)}\right).$$

where $\sigma^o_{\text{tsh}}(R_o)$ is the decoding threshold equivalent to $E^o_{\text{tsh}}(R_o)$ but expressed in terms of the channel noise parameter.

Unless the inner the decoder can reduce the BER of the received codeword at its output to a value $\mathrm{P}^i_{\text{e,out}} \leq \mathrm{P}^o_{\text{e,tsh}}$, the outer decoder cannot reduce the BER at its output below a constant value $c > 0$. Therefore, the E_b/N_o threshold value of a Raptor code is defined as the minimum E_b/N_o at which the output BER of the inner decoder $\mathrm{P}^i_{\text{e,out}}$ is equal to $\mathrm{P}^o_{\text{e,tsh}}$.

From the discussion above we can conclude that given a Raptor code, for every outer

code with rate R_o, there is a corresponding BER ($P^o_{e,tsh}$) that the inner decoder with rate ($R_i = R/R_o$) must produce at its output with as much channel E_b/N_o as necessary. This leads to different rate pairs having different E_b/N_o thresholds as we can see in Figure 6.2.

6.2 Decoding Threshold of Raptor Codes

The inner decoder of Raptor codes is an LT code. This code exhibits relatively high error floors and sharp waterfall region curves. These factors play an important role in determining the threshold of Raptor codes as we will see below. Figure 6.3 shows the asymptotic BER curves of the inner (LT) decoder with output distribution (4.24), overall rate $R = 0.5$, and for the inner rates shown on the figure.

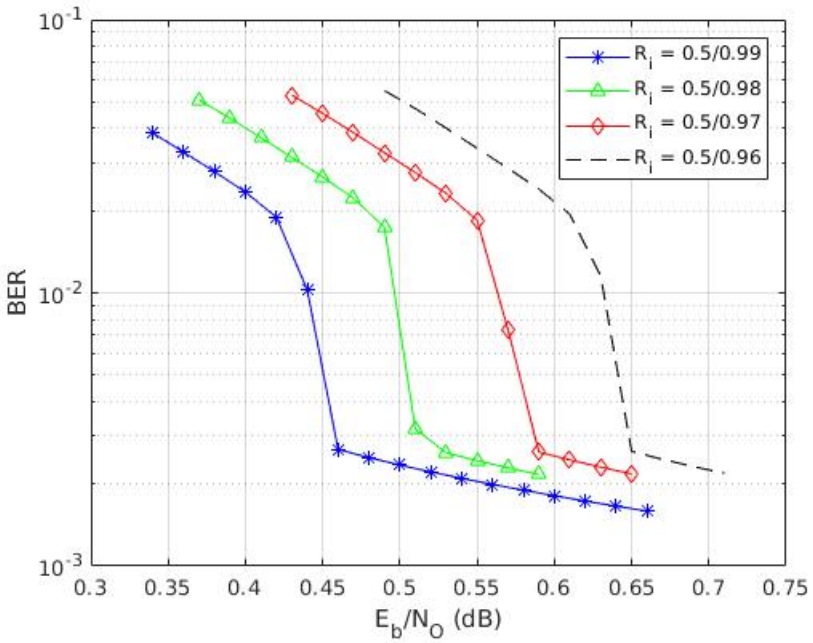

Figure 6.3. Asymptomatic BER curves of the inner decoder.

Assuming we couple the inner code corresponding to Figure 6.3 above with a regular LDPC outer code such that each outer rate $R_i = \{0.99, 0.98, 0.97, 0.96\}$ is realized by the corresponding code from Table 6.2. The E_b/N_o threshold value at each rate pair (R_i, R_o)

from the set:

$$\{(0.5/0.99, 0.99), (0.5/0.98, 0.98), (0.5/0.97, 0.97), (0.5/0.96, 0.96)\}$$

can be determined as shown in Figure 6.4. The threshold is reached when the output BER of the inner code ($P^{i}_{e,out}$) reaches (intersects with) the horizontal line representing the critical BER ($P^{o}_{e,tsh}$) of the corresponding outer code. Comparing the thresholds at the rate pairs tested, $(0.5/0.98, 0.98)$ returns the best (lowest) E_b/N_o threshold.

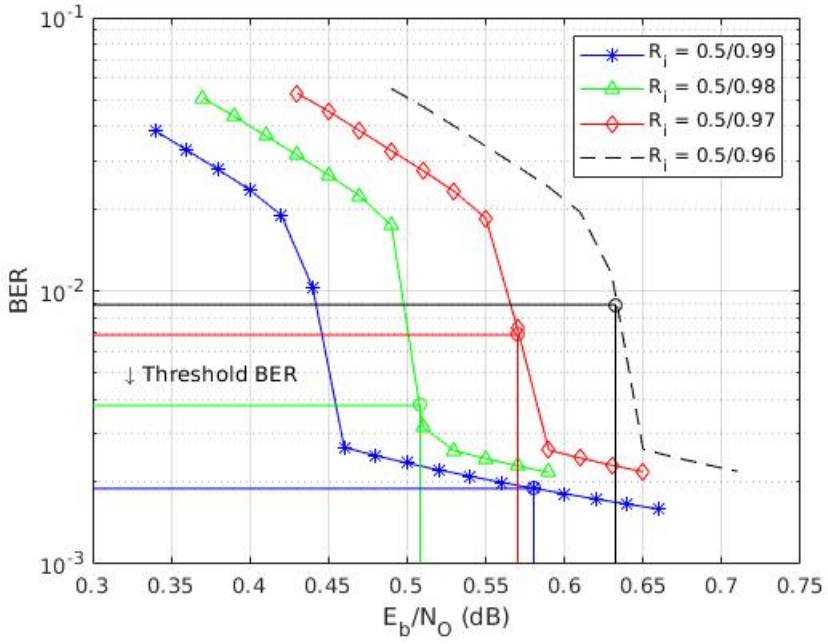

Figure 6.4. Intersection of $P^{o}_{e,tsh}$ with asymptomatic BER curves of the outer LDPC decoder.

R	(d_v, d_c)	$E^o_{\text{tsh}}(R_o)$ dB	$P^o_{\text{e,tsh}}$
0.99	(4,400)	6.2694	1.89×10^{-3}
0.98	(4,204)	5.595	3.81×10^{-3}
0.9699	(4,133)	5.1487	5.87×10^{-3}
0.96	(4,100)	4.8207	7.89×10^{-3}
0.95	(4,80)	4.5613	9.89×10^{-3}
0.9245	(4,53)	4.0433	1.52×10^{-2}
0.9	(4,53)	3.6700	2.03×10^{-2}
0.8519	(4,27)	3.6700	3.09×10^{-2}
0.8	(4,20)	2.6857	4.24×10^{-2}
0.75	(4,16)	2.3591	5.40×10^{-2}

Table 6.2. Regular LDPC codes and their respective rates and thresholds.

Examining a Raptor code with another output distribution, Figure 6.5 below shows the asymptomatic inner code BER curves of a rate half Raptor code with output distribution $\Omega_2(x)$ of (6.1), and their intersection points with the critical BER of the outer decoder for each of the (R_i, R_o) rate pairs shown. We can observe that the rate pair $(0.5/0.98, 0.98)$ proves to be the best choice.

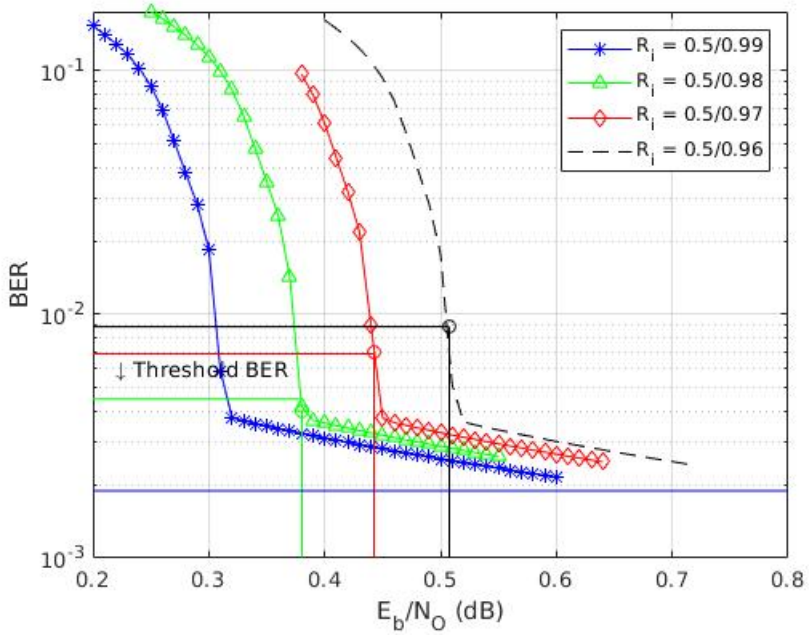

Figure 6.5. Intersection of $P^o_{\text{e,tsh}}$ with asymptomatic BER curves of the outer LDPC decoder.

$$\Omega_2(x) = 0.00967x + 0.45025x^2 + 0.20937x^3 + 0.02332x^4 + 0.14735x^5 + 0.11249x^{11} + 0.04755x^{40} \tag{6.1}$$

Figures 6.6, 6.7, and 6.8 illustrate the thresholds of Raptor codes with the distributions $\Omega_1(x)$ of (4.24), $\Omega_2(x)$ of (6.1), and $\Omega_4(x)$ of (4.26), respectively, for the rate pairs shown. The outer code rates are realized by the LDPC codes given in Table 6.2. Each figure shows the thresholds for three instances of the overall code rate $R = 1/3, 1/2$, and $5/7$. For the first two codes and for all three instances of the code rate R tested, the optimal pair is $(R/0.98, 0.98)$ since it is the rate pair that leads to the lowest SNR threshold, while, for the third code it is $(R/0.99, 0.99)$. Only the outer code rate is indicated on the figures since the inner rate can be inferred as $R_i = R/R_o$.

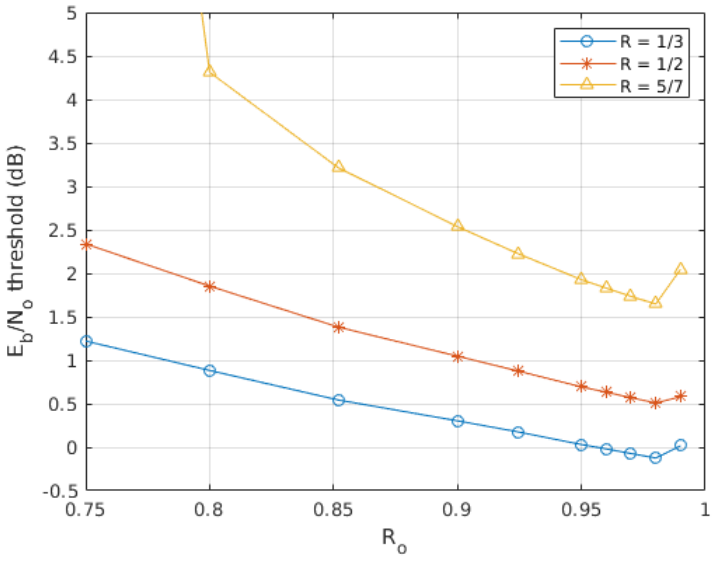

Figure 6.6. Decoding threshold vs code rate pair (R_i, R_o) with output distribution $\Omega_1(x)$.

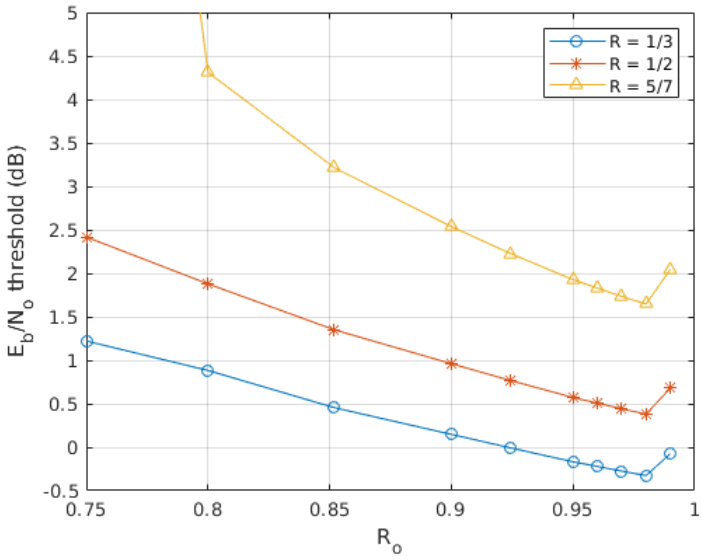

Figure 6.7. Decoding threshold vs code rate pair (R_i, R_o) with output distribution $\Omega_2(x)$.

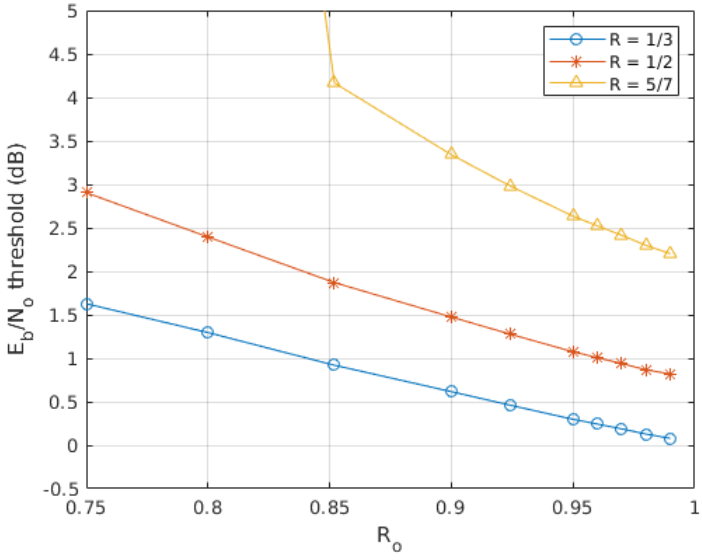

Figure 6.8. Decoding threshold vs code rate pair (R_i, R_o) with output distribution $\Omega_4(x)$.

6.3 The Optimal Rate Pair with a Capacity Achieving Outer Code

Given a Raptor code with a fixed output distribution $\Omega(x)$, overall rate R, and a set of rate pairs (R_i, R_o) such that each R_o is realized by a fixed regular LDPC code we were able to determine the optimal rate pair in a given set. Next, we add another degree of freedom to our problem by removing the condition that each R_o is realized by a specific LDPC code. Instead, we assume we can use a capacity-achieving outer code with a rate of R_o. We will restrict our search for the optimal rate pair to the discrete set $\{(R/0.99, 0.99), (R/0.98, 0.98) \ldots (R/0.96, 0.96)\}$, however, the same approach can be easily extended to more refined search space.

In order to find the optimal rate pair assuming a capacity achieving outer code, we start by extracting the critical BER for each rate R_o as follows

$$\sigma^o_{\text{cap}} = \left(\frac{1}{2R_o E^o_{\text{cap}}(R_o)} \right)^{\frac{1}{2}}, \tag{6.2}$$

$$P^o_{\text{e,cap}} = Q\left(\frac{1}{\sigma^o_{\text{cap}}} \right). \tag{6.3}$$

Where, $E^o_{\text{cap}}(R_o)$ is the Shannon SNR limit for rate R_o. Table 6.3 shows the Shannon E_b/N_o thresholds and the corresponding critical BERs ($P^o_{\text{e,cap}}$) for the rates listed.

R	$E^o_{\text{tsh}}(R_o)$ dB	$P^o_{\text{e,cap}}$
0.99	6.015	2.5×10^{-3}
0.98	5.31	4.9×10^{-3}
0.9699	4.85	7.5×10^{-3}
0.96	4.484	1.01×10^{-3}
0.95	4.195	1.27×10^{-3}
0.9245	3.628	1.95×10^{-2}
0.9	3.205	2.62×10^{-2}
0.8519	2.564	3.98×10^{-2}
0.8	2.042	5.48×10^{-2}
0.75	1.628	6.98×10^{-2}

Table 6.3. Critical BERs of capacity achieving codes.

Assuming a capacity achieving outer code, we can determine the asymptotically op-

timal code rate pair for a given output distribution. This answers the question whether for a given output distribution $\Omega(x)$ the rate pair we use is asymptotically optimal or improvement is possible. In order to determine the optimal rate pair (R_i, R_o), we rely on the same method we used earlier with the difference that we use critical BERs corresponding to the Shannon limit thresholds ($P^o_{\text{e, cap}}$) given in Table 6.3 rather than the LDPC code thresholds given in Table 6.2. We consider two example cases with an overall rate of $R = 1/2$. For the distribution (4.24) and as shown in Figure 6.9 below, we can see that the asymptotically optimal rate pair is $(0.5/0.99, 0.99)$ with a threshold $E_b/N_o = 0.48$ dB. This leaves some room for improvement from the earlier case, where, a regular LDPC outer code was used and the optimal rate pair was $(0.5/0.98, 0.98)$ with threshold $E_b/N_o = 0.51$ dB.

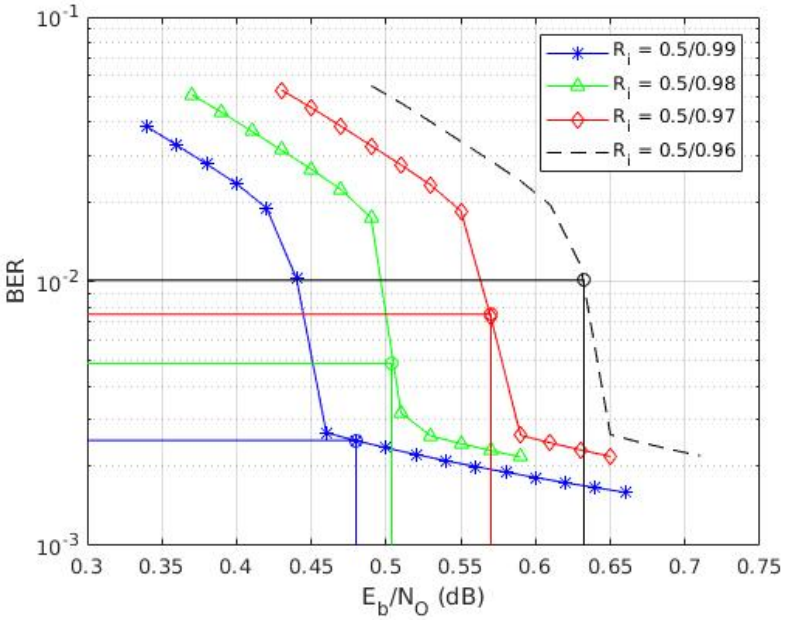

Figure 6.9. Intersection of $P^o_{\text{e,tsh}}$ with asymptomatic BER curves of capacity achieving outer decoder.

On the other hand, for a Raptor code with output distribution (6.1) as we can see in Figure 6.10 the asymptotically optimal code rate pair is $(0.5/0.98, 0.98)$ which is the same optimal rate pair when a regular LDPC code is used as the outer code.

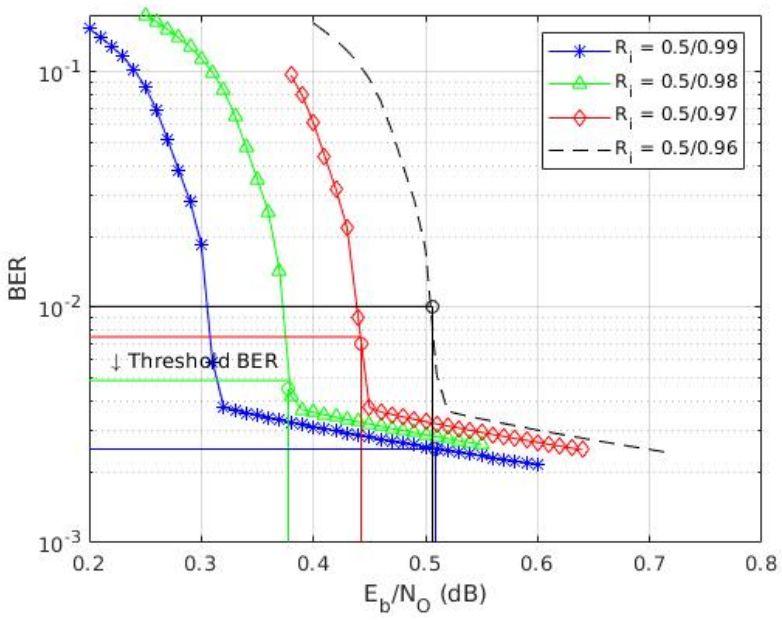

Figure 6.10. Intersection of $\mathrm{P}^{o}_{\mathrm{e,tsh}}$ with asymptomatic BER curves of capacity achieving outer decoder.

Figures 6.11, 6.12, and 6.13 below show the thresholds of Raptor codes with a capacity achieving outer code, and with the output distributions (4.24), (6.1), and (4.26), respectively, for the rate pairs shown. Each figure shows the thresholds for three instances of the overall code rate $R = 1/3, 1/2$, and $5/7$. As expected, for the tested distributions the threshold improves as the outer code improves. We also observe that the optimal rate for the output distributions $\Omega_2(x)$ and $\Omega_4(x)$ stays the same, while, for $\Omega_1(x)$ it changes to $(R/0.99, 0.99)$.

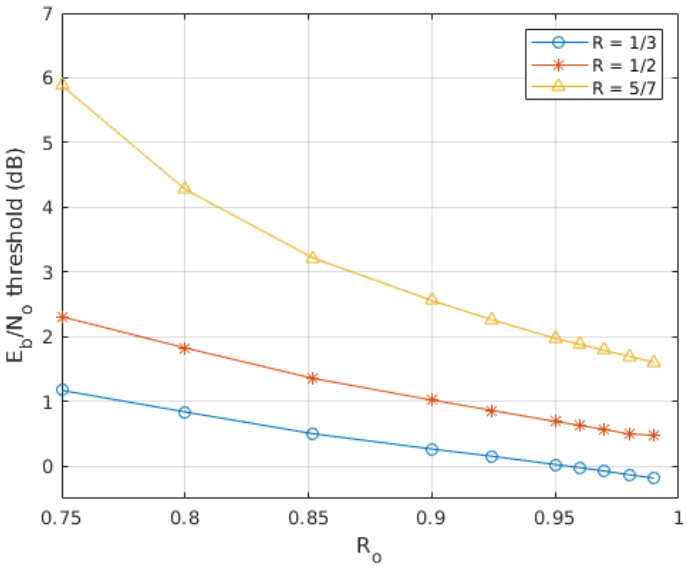

Figure 6.11. Decoding threshold vs code rate pair (R_i, R_o) with optimal outer code and $\Omega_1(x)$.

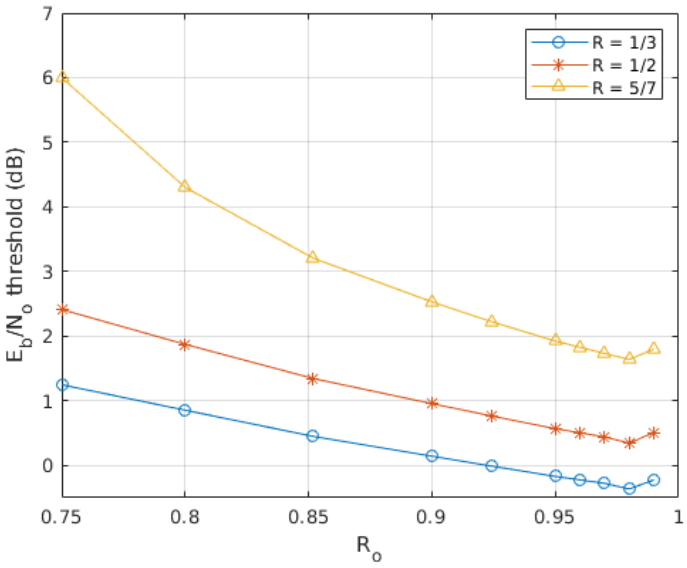

Figure 6.12. Decoding threshold vs code rate pair (R_i, R_o) with optimal outer code and $\Omega_2(x)$.

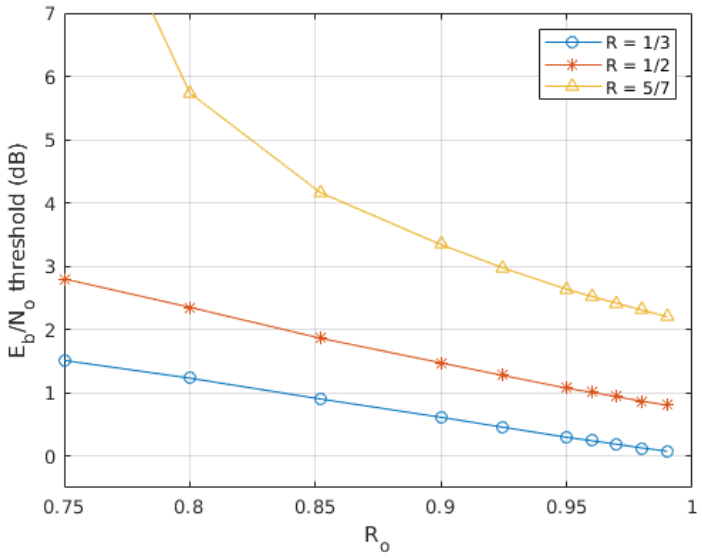

Figure 6.13. Decoding threshold vs code rate pair (R_i, R_o) with optimal outer code and $\Omega_4(x)$.

6.4 Asymptotically Optimal Rate Pairs

Raptor codes are generally designed based on the assumption that the outer code is a regular LDPC code despite the fact that irregular LDPC codes can offer better thresholds and performance. The reason for that can be explained using Figure 6.14 below, where the E_b/N_o thresholds of a capacity-achieving code and regular LDPC codes with node degrees $(4, \lfloor \frac{4}{1-R} \rceil)$ are compared.

As can be observed, for high rates the difference between the thresholds of a regular and a capacity-achieving code is much less than the difference for lower rates, which implies that for an irregular LDPC code the thresholds are even closer for high rates. Therefore, using a regular outer code comes with an acceptably small sacrifice in performance compared to the gain in computational complexity. For lower rates, the threshold of regular LDPC codes moves away from the channel capacity limit, while irregular codes can reach extremely close to the channel limit [24]. The implication of this when LDPC codes are used as outer codes is that as the outer rate decreases the criti-

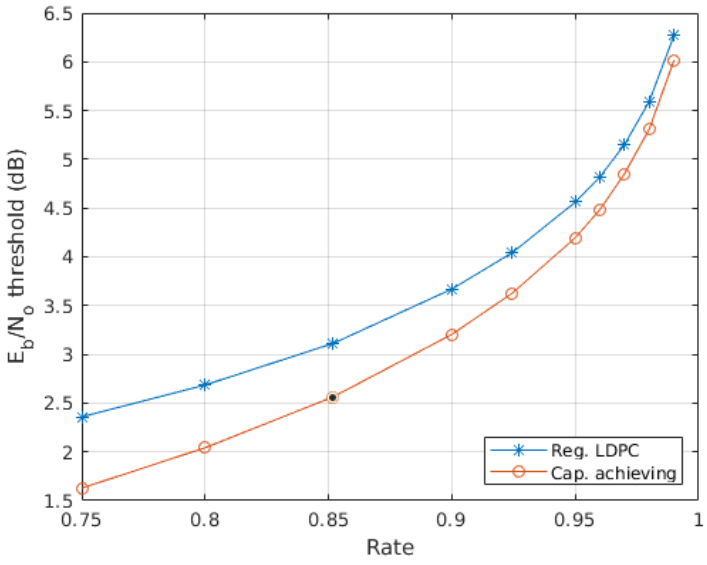

Figure 6.14. Threshold vs code rate pair (R_i, R_o).

cal BER of irregular codes increases faster than of regular codes. This leads to the question of whether using irregular outer codes can cause the optimal outer rate for Raptor codes to occur at lower rates. In order to answer this question, for each of the rate pairs $\{(0.5/0.99, 0.99), (0.5/0.98, 0.98), (0.5/0.97, 0.97), (0.5/0.95, 0.95), (0.5/0.9, 0.9), (0.5/0.8, 0.8), (0.5/0.75, 0.75)\}$ we determined the minimum E_b/N_o threshold at which an output degree distribution that can be used to design a functioning Raptor code exists as given in Figure 6.15.

As can be seen in Figure 6.15, even if a capacity achieving outer code is used to design Raptor codes, the optimal performance will still be provided by higher rates. Figure 6.15 shows that as the outer rate increases, the threshold of Raptor codes improves. The reason for this behavior can be explained using Figures 6.4, 6.5, 6.9, and 6.10, whereas the outer rate increases, the waterfall region shifts to the left and the prospect of a lower SNR threshold improves. LT codes are very sensitive to changes in the overhead, and this behavior explains why the optimal outer rate of Raptor codes cannot occur at lower rates.

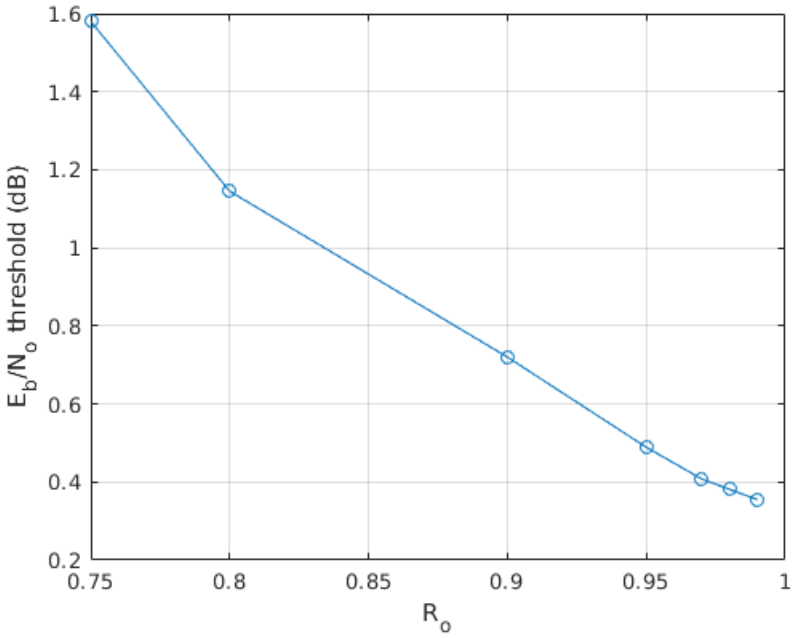

Figure 6.15. Threshold vs code rate pair (R_i, R_o).

6.5 Designing Optimized Raptor Codes

Given a set of outer code rates $R_o = \{R_{o1}, R_{o2}, \ldots, R_{o\kappa}\}$ and a set of inner code rates $R_i = \{R_{i1}, R_{i2}, \ldots, R_{i\kappa}\}$, where $R_i = R/R_o$. In an ideal case, if $R_{o1} > R_{o2} > \cdots > R_{o\kappa}$, and equivalently $R_{i1} < R_{i2} < \cdots < R_{i\kappa}$, the best (lowest) E_b/N_o threshold can be provided by the pair (R_{i1}, R_{o1}). This is because the BER curve with the lowest SNR will be provided by the inner decoder with a rate of R_{i1}. Therefore, aiming to design Raptor codes by using the highest outer rate possible increase the potential for achieving a lower (better) SNR threshold. Based on this observation, we propose an optimization method that considers the optimality of the inner and outer rates in designing the output distribution of Raptor codes. Other optimization and design methods we found in the literature fix the outer code rate to a constant value, usually 50/51, or ignore it in the process of searching the space of distributions for the optimized output distributions. The proposed optimization algorithm starts with the highest possible outer rate and hence the lowest inner rate, returns the optimized distribution and calculates its threshold. Next, the outer rate is decreased by a value of 0.01 and the output distribution is optimized for the new inner rate. This process continues as long the newfound distribution has a lower SNR threshold.

6.5.1 Formulating the Optimization

The objective is: design an output degree distribution such that $P_e^{(l)}$ defined in (4.13) goes to zero at the minimum E_b/N_o value possible for each given rate pair (R_i, R_o).

For a given instance of the overall code rate R, the outer code is fixed as a regular LDPC code with rate R_o and fulfilled by its equivalent code in Table 6.2. The output degree distribution of the inner LT code is optimized as described in the objective above. The constraints are: (a) $\Omega_i \geq 0 \forall i \in [1, k]$, (b) $\sum_{i=1}^{k} \Omega_i = 1$, (c) $\omega(1 - \lambda(1 - x)) > x \forall x \in (0, 1)$. The first and second constraints provide the conditions necessary for a valid distribution. The third condition ensures successful decoding on the LT part and recovery of all input symbols.

6.5.2 Optimized Distributions

Considering the outer rate $R_o = 0.99$ as the highest rate possible, we ran the optimization algorithm for three instances of the realized code rate $R = 1/2$, $5/7$, and $1/3$ and obtained the output distributions $\Omega_{o1}(x)$, $\Omega_{o2}(x)$, and $\Omega_{o3}(x)$, respectively.

$$\Omega_{o1}(x) = 0.010231x^1 + 0.44309x^2 + 0.23276x^3 + 0.085498x^5 + 0.064981x^6 + 0.047326x^7 + 0.001434x^8 + 0.05186x^{15} + 0.033444x^{19} + 0.029376x^{70}$$

$$\Omega_{o2}(x) = 0.0105240x + 0.46261x^2 + 0.23760x^3 + 0.017284x^5 + 0.163260x^6 + 0.048895x^{15} + 0.032601x^{19} + 0.02723x^{70}$$

$$\Omega_{o3}(x) = 0.009416x + 0.433130x^2 + 0.190730x^3 + 0.076959x^4 + 0.115320x^5 + 0.029014x^{10} + 0.095242x^{11} + 0.050189x^{40}$$

As can be seen in Figure 6.16, the optimal rate pair for the optimized distributions is $(R/0.99, 0.99)$.

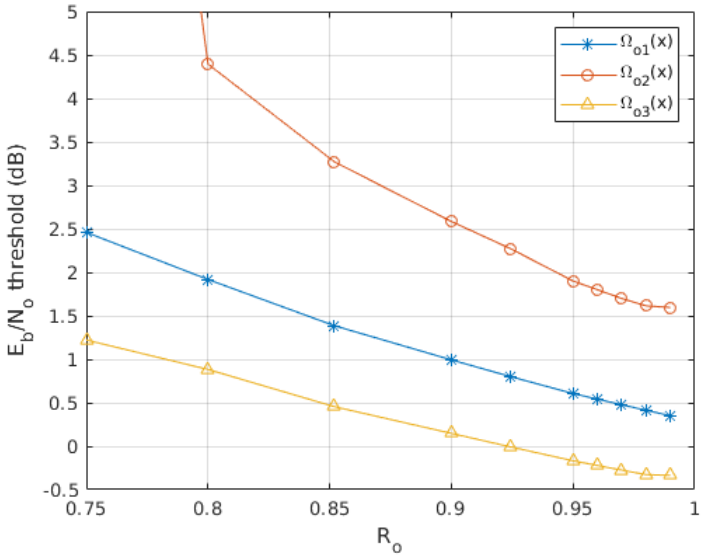

Figure 6.16. Threshold vs code rate pair (R_i, R_o).

6.5.3 Performance Comparison

Compared to the distributions designed in [12], which provide the lowest SNR thresholds we found in the literature for Raptor codes, the optimized distributions reduce (improve) the SNR decoding thresholds about 0.05 dB. Figure 6.17 shows the asymptotic BER performance of Raptor codes with our optimized output distribution compared with the output distributions $\Omega_2(x)$, $\Omega_5(x)$, and $\Omega_6(x)$ given in [12], and $\Omega_1(x)$ given in [2]. The optimized distributions can improve upon the performance of Raptor code compared to degree distributions that do not consider the inner and outer rate optimality in the design process.

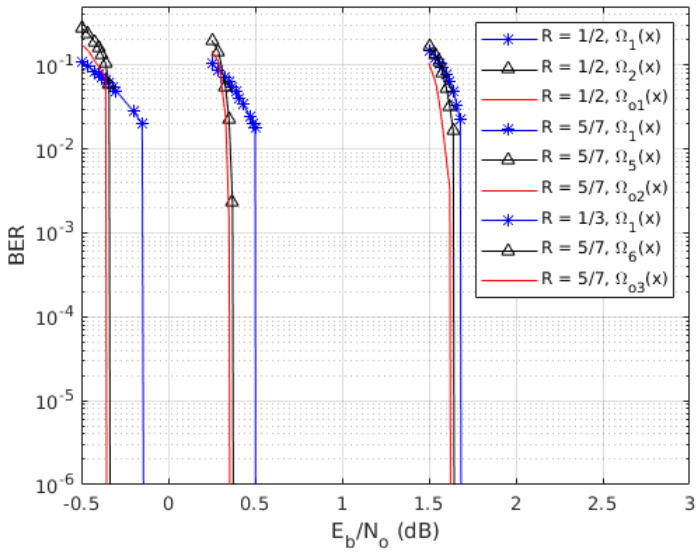

Figure 6.17. Comparing asymptotic BER performance.

$$\Omega_2(x) = 0.00967x + 0.45025x^2 + 0.20937x^3 + 0.02332x^4 + 0.14735x^5 + 0.11249x^{11} + 0.04755x^{40}$$

$$\Omega_5(x) = 0.00959x^1 + 0.47527x^2 + 0.19096x^3 + 0.04689x^4 + 0.12159x^5 + 0.02441x^{10} + 0.08605x^{11} + 0.05186x^{15} + 0.04523x^{40}$$

$$\Omega_6(x) = 0.00896x^1 + 0.44293x^2 + 0.13185x^3 + 0.21253x^4 + 0.00602x^5 + 0.14513x^{10} + 0.0525x^{40}$$

For overall realized code rate $R = 0.5$, Figure 6.18 show the decoded BERs of a Raptor code with the optimized output distribution $\Omega_{o1}(x)$ operating at its optimal rate pair $(0.5/0.99, 0.99)$ compared to that of a Raptor code with distribution $\Omega_2(x)$ operating at its optimal rate pair $(0.5/0.98, 0.98)$. The advantage of supplying the inner code with lower rate i.e. higher redundancy by using the rate pair $(R/0.99, 0.99)$ can be observed in the better performance $\Omega_{o1}(x)$ offers.

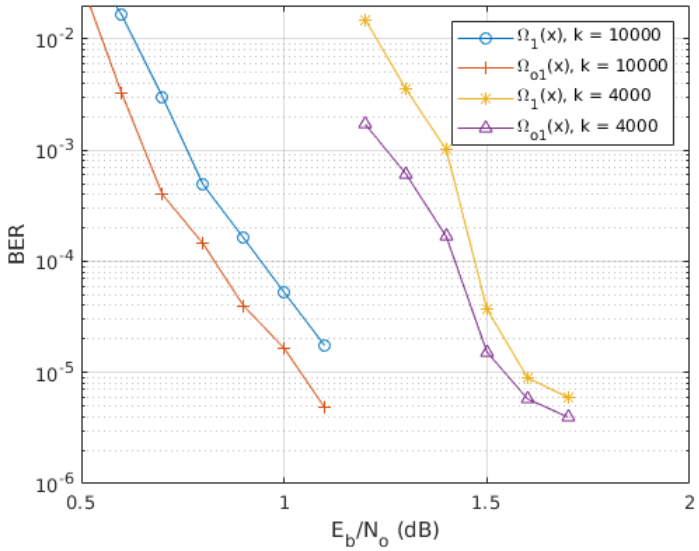

Figure 6.18. Comparing BER performance at $R = \frac{1}{2}$.

Using higher outer code rates can lead to an increase in the number of cycles in the decoding graph of the outer LDPC code. Figure 6.19 shows the BER performance of a Raptor code at realized code rate half and with the output distribution $\Omega_{o1}(x)$ and $(4, 400)$ LDPC outer code. The curves demonstrate the performance in two cases: (a) cycles are allowed to form freely in the decoding graph of the LDPC code, and (b) a cycle removing algorithm is used by the LDPC code that removes cycles while maintaining the rank of the parity check matrix. The curves indicate that the general impact of cycles on the decoding performance of the optimized codes is noticeable, though not significant.

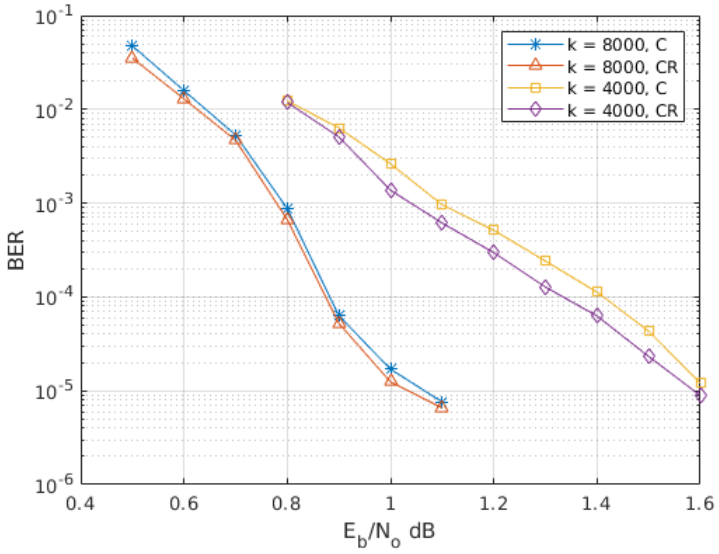

Figure 6.19. Effects of cycles on decoding, .

We used our proposed optimization approach in designing Raptor codes that are meant to improve the decoding threshold and performance on the BIAWGN channel, with no constraints on the output degree distributions and Poisson input degree distributions. However, the approach can be extended to other channels, and output degree distributions that are designed for specific applications such as unequal error protection, cooperative networks, etc.

CHAPTER 7

CONCLUSION

Channel codes that use belief propagation decoding on the BIAWGN channel such as Raptor codes need an accurate estimation of the channel SNR. When the estimated SNR is not equal to the true channel SNR, this condition is known as SNR mismatch and it can degrade the performance of the decoder or cause it to completely fail if the mismatch is high enough. We studied the effects of SNRM on Raptor codes. We started by studying the suitability of the DDE algorithm to simulate the behavior of Raptor codes under SNRM and applied the required modifications for the algorithm to correctly simulate the BP decoder of Raptor codes in the presence of SNRM. Using DDE, we determined the decoding threshold of Raptor codes for different values of SNR offset. We tested different output distributions and at multiple instances of the realized code rate in order to reach more accurate and general conclusions. Determining the threshold gave us a means to quantify the effects of SNRM on Raptor codes. We observed that SNR underestimation is slightly less detrimental to BP decoding compared to SNR overestimation for lower levels of mismatch, however, as the mismatch increases underestimation becomes more detrimental, a property that was previously observed in LDPC and Turbo codes. Determining the threshold can be used to estimate how much SNRM tolerance to expect for a given channel SNR value. Equivalently, it can help in estimating the SNR needed to ensure a certain level of tolerance to SNRM. Also, comparing the thresholds of different Raptor codes for a range of SNRM ratios, we can recognize which codes are comparably more tolerant to SNRM. By comparing the asymptotic performance of Raptor codes and irregular LDPC codes, we observed that they have a similar response to SNRM in the case of SNR underestimation, while, irregular LDPC codes offer

better performance in the case of SNR overestimation. Taking advantage of what we learned from the effects of SNRM on Raptor codes, we proposed an optimization method to design output degree distributions of the LT part that can be used to construct Raptor codes with more tolerance to high levels of SNRM. In channels with high levels of SNRM, our optimized distributions can outperform other distributions. However, the output distribution given in [2] offers, on average, the better performance among the distributions we tested and optimized.

Another aspect of Raptor codes that can be used to improve the performance is using optimal inner and outer rates. Using DDE based asymptotic analysis, we showed the effect of the rate pair choice on the decoding threshold of Raptor codes and how the optimal rate pair is decided. Testing Raptor codes with different output distributions, we showed that each code can have a different optimal rate pair. Using the optimal rate pair we can further improve the performance and avoid suboptimal use of Raptor codes in terms of inner and outer rates. Also, using asymptotic analysis we reached the conclusion that even by using a capacity achieving outer code, the optimal outer rate of Raptor codes cannot occur at lower values. Finally, we proposed an optimization method that considers the optimality of the code rate pair. The designed distributions show improvement in both the decoding threshold and performance compared to other code designs that do not consider the optimality of the inner and outer rates.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM, pages 56–67. ACM, 1998.

[2] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, June 2006.

[3] M. Luby. LT codes. In *Foundations of Computer Science. Proceedings. The 43rd Annual IEEE Symposium on*, pages 271–280, 2002.

[4] R. G. Gallager. Low density parity-check codes. *Cambridge, MA: MIT Press*, 1963.

[5] D. MacKay and R. Neal. Good codes based on very sparse matrices. *Proc. 5th IMA Conf., Cryptography and Coding (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag*, 1025:100–111, 1995.

[6] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.

[7] R. Palanki and J. S. Yedidia. Rateless codes on noisy channels. *in Proc. IEEE Int. Symp. Inform. Theory, June–July*, page 37, Jun. 2004.

[8] O. Etesami and A. Shokrollahi. Raptor codes on binary memoryless symmetric channels. *IEEE Transactions on Information Theory*, 52(5):2033–2051, May 2006.

[9] Z. Cheng, J. Castura, and Y. Mao. On the design of Raptor codes for binary-input Gaussian channels. *IEEE Transactions on Communications*, 57(11):3269–3277, Nov 2009.

[10] S. H. Kuo, Y. L. Guan, S. K. Lee, and M. C. Lin. A design of physical-layer raptor codes for wide SNR ranges. *IEEE Communications Letters*, 18(3):491–494, March 2014.

[11] A. Kharel and L. Cao. Improved fountain codes for BI-AWGN channels. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, March 2017.

[12] A. Kharel and L. Cao. Analysis and design of physical layer Raptor codes. *IEEE Communications Letters*, PP(99):1–1, 2017.

[13] R. M. Tanner. A recursive approach to low complexity codes. *EEE Trans. Information Theory*, 27(9):533–547, Sep. 1981.

[14] F. Zarkeshvari J. Zhao and A. Banihashemi. On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes. *IEEE Trans. Commun.*, 53(4):549–554, Apr. 2005.

[15] V. Savin. Self-corrected min-sum decoding of LDPC codes. In *2008 IEEE International Symposium on Information Theory*, pages 146–150, July 2008.

[16] and R. M. Tanner and C. Jones and. Improved min-sum decoding algorithms for irregular LDPC codes. In *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pages 449–453, Sep. 2005.

[17] L. Yuan. Performance of min-sum for decoding fountain codes over biawgn channels. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, Sep. 2012.

[18] H. Saeedi and A. H. Banihashemi. Design of irregular LDPC codes for BIAWGN channels with SNR mismatch. *IEEE Transactions on Communications*, 57(1):6–11, January 2009.

[19] T. Richardson A. Shokrollahi and R. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47:619–637, Feb. 2001.

[20] Sae-Young Chung, T. J. Richardson, and R. L. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation. *IEEE Transactions on Information Theory*, 47(2):657–670, Feb 2001.

[21] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, Oct 2001.

[22] H. Saeedi and A. H. Banihashemi. Performance of belief propagation for decoding LDPC codes in the presence of channel estimation error. *IEEE Transactions on Communications*, 55(1):83–89, Jan 2007.

[23] L. Yuan, J. Pan, and L. Yuan. Performance analysis for decoding lt codes over biawgn channels with snr mismatch. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6, Oct 2017.

[24] Sae-Young Chung, G. D. Forney, T. J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 db of the Shannon limit. *IEEE Communications Letters*, 5(2):58–60, Feb 2001.

[25] D. Mackay. Fountain codes. *IEEE Proc. Commnications*, 152(2), Dec. 2005.

[26] D. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inform. Theory*, 45:399–431, Mar. 1999.

[27] David J.C. MacKay and Radford M. Neal. Good codes based on very sparse matrices. *Cryptography and coding: 5th IMA conference*, pages 100–111, Dec. 1995.

[28] William E. Ryan and Shu Lin. *Channel Codes:Classical and Modern*. Cambridge University press, 2009.

[29] T. J. Richardson and R. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47:638–656, Feb. 2001.

[30] Y. Li M. Yang and W. Ryan. Design of efficiently encodable moderate-length high-rate irregular LDPC codes. *IEEE Trans. Commun.*, 52:564–571, Apr. 2004.

[31] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA, Morgan Kaufmann, 1988.

[32] M. Watson T. Stockhammer M. Luby, A. Shokrollahi and L. Minder. Raptorq forward error correction scheme for object delivery. In *Available at http://tools.ietf.org/html/draft-ietf-rmtbb-fec-raptorq-03*, Aug. 2010.

[33] M. Mitzenmacher M. Luby and A. Shokrollahi. Analysis of random processes via and-or tree evaluation. *in Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms, San Francisco,CA*, page 364–373, Jan. 1998.

[34] M. Luby. A note on the design of degree distributions. *unpublished*, 2001.

[35] T. A. Summers and S. G. Wilson. SNR mismatch and online estimation in turbo decoding. *IEEE Transactions on Communications*, 46(4):421–423, April 1998.

[36] M. A. Jordan and R. A. Nichols. The effects of channel characteristics on Turbo code performance. In *Military Communications Conference, 1996. MILCOM '96, Conference Proceedings, IEEE*, volume 1, pages 17–21 vol.1, Oct 1996.

[37] S. Asoodeh. New stopping criterion for Turbo code in the presence of SNR mismatch. In *International Congress on Ultra Modern Telecommunications and Control Systems*, pages 182–186, Oct 2010.

[38] Chun Ling Kei and Wai Ho Mow. A class of switching Turbo decoders against severe SNR mismatch. In *Proceedings IEEE 56th Vehicular Technology Conference*, volume 4, pages 2197–2200 vol.4, Sept 2002.

[39] M. El-Khamy, Jinhong Wu, Jungwon Lee, H. Roh, and Inyup Kang. Near-optimal Turbo decoding in presence of SNR estimation error. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 3737–3742, Dec 2012.

[40] S. Jayasooriya, M. Shirvanimoghaddam, L. Ong, and S. J. Johnson. Analysis and design of Raptor codes using a multi-edge framework. *IEEE Transactions on Communications*, 65(12):5123–5136, Dec 2017.

VITA

Hussein Fadhel graduated top of his class with B.Sc. degree in electronics and control engineering from the technical college, Kirkuk, Iraq in 2007 and he worked there after graduation as a teaching assistant at the labs of the electronics and control engineering department from 2008 to 2010. In 2010 he was awarded a Fulbright scholarship to pursue M.S. degree in electrical engineering and joined the University of Mississippi, USA in August 2010. He received M.S. in engineering science with the emphasis in telecommunications in May 2012. After that he returned to his job at the technical college, Kirkuk, Iraq and worked as an instructor from 2012 to 2104. Then, started his Ph.D. program at the University of Mississippi, USA in January 2015 and earned a Ph.D. in engineering science - electrical engineering in May 2019. During his Ph.D. program, he worked as a teaching assistant, research assistant, and lab instructor at the department of electrical engineering.