Title
CRC-Aided List Decoding of Short Convolutional and Polar Codes for Binary and Non-binary Signaling

Author
King, Jacob

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

CRC-Aided List Decoding of Short Convolutional and Polar Codes for Binary and Non-binary Signaling

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical and Computer Engineering

by

Jacob Aaron King

2022

ABSTRACT OF THE THESIS

CRC-Aided List Decoding
of Short Convolutional and Polar Codes
for Binary and Non-binary Signaling

by

Jacob Aaron King
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2022
Professor Richard D. Wesel, Chair

This thesis consists of two main sections investigating the performance of cyclic-redundancy-check-aided (CRC-aided) list decoding on short block codes. The first section analyzes the performance of tail biting convolutional codes with CRC (CRC-TBCCs) and polar codes with CRC (CRC-Polar) with an eye toward the 5G standard. The second section concerns designing optimal CRC-convolutional codes for nonbinary orthogonal noncoherent signaling.

The first section focuses on designing a code for the physical broadcast channel of the 5G standard. The 5G standard encodes a 32-bit message with a 24-bit CRC and a (512, 32+24) polar code, with bit repetition to arrive at a final blocklength of 864 bits. We design shorter CRCs for this polar code in order to improve its performance. We also design low rate CRC-TBCCs with 32 bit messages as an alternative to the CRC-Polar in the 5G PBCH. CRCs are designed to optimize the distance spectrum of the concatenated CRC-Polar or CRC-TBCC. We call these CRCs distance-spectrum-optimal (DSO). We consider both adaptive and nonadaptive list decoders for these codes and compare their performance and

complexity. Simulation results show that our CRC-TBCC and CRC-Polar designs significantly outperform the polar code in the 5G standard, with some CRC-TBCC designs closely approaching the random coding union (RCU) bound.

The second section presents designs for CRC-TBCCs and zero-terminated convolutional codes with CRC (CRC-ZTCCs) for communication with noncoherent orthogonal signaling. We design Q-ary convolutional codes to maximize the minimum distance, and then design Q-ary DSO CRCs for these convolutional codes, extending the work of Lou et. al. and Yang et. al. to nonbinary fields. The Q-ary code symbols are mapped to a Q-ary orthogonal signal set and sent over an AWGN channel with noncoherent reception. We consider cases where Q is a power of 2. We also derive a saddlepoint approximation for the calculation of the RCU bound for this channel. The RCU bound is a useful benchmark for the performance of CRC-convolutional codes, and we compare the performance of our codes to this bound.

The thesis of Jacob Aaron King is approved.

Gregory J. Pottie

Christina P. Fragouli

Richard D. Wesel, Committee Chair

University of California, Los Angeles

2022

To my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The completion of this thesis would not have been possible without the support from friends, family, advisors, mentors, and colleagues. First of all, I would like to thank Prof. Richard Wesel for his guidance through my Bachelor's and Master's degrees. In the nearly three years that we have known and worked with each other, he has been a wonderful teacher and advisor, and we have had many fascinating discussions on information theory and channel coding. His enthusiasm for the field has inspired my own, and I am grateful that I had such an amazing advisor.

I would like to thank Zeta Associates, Inc. for their financial support throughout my Master's program. In addition, I would like to thank the many people at Zeta who acted as mentors to me during my research. Prof. William Ryan has been an indispensable source of support and knowledge. He has acted as a second advisor to me throughout my Master's education, and his lessons and discussions on information theory, coding theory, and signal processing have helped me immensely in my research. I also thank Dr. Sheeyun Park and Dr. Brandon Erickson for their mentorship during the summers I interned at Zeta. I am very lucky to have such a wide range of mentors from which to draw knowledge from, and I look forward to working with them.

I would like to thank all of the faculty members of UCLA's Electrical and Computer Engineering Department for their undergraduate and graduate courses that formed the foundation of my research. I would like to extend a special thanks to Prof. Lara Dolecek, Prof. Suhas Diggavi, and Prof. Christina Fragouli for their classes in the fields of probability theory, information theory, and signal processing, all of which have been especially helpful. I would also like to thank the UCLA Mathematics Department, whose classes taught me the importance of mathematical rigor and have shaped how I think about and approach problems in my research. Finally, I would like to thank Deeona Columbia and the graduate academic staff for their help with the administrative side of the Master's program.

In addition to the faculty of UCLA, I would also like to thank Prof. Dariush Divsalar (JPL), Prof. Stephen Wilson (UVA), Prof. Alexander Vardy (UCSD), Dr. Peter Kazakov and Dr. Tsonka Baicheva for the discussions we shared on error correction coding. I was unfortunately only able to meet with Prof. Vardy once, and I am saddened by his untimely passing. I am grateful to have continued the collaboration with Hanwen Yao from UCSD.

I would like to thank my colleagues in the UCLA Communications Systems Laboratory: Dr. Hengjie Yang, Linfang Wang, Chester Hulse, Holden Grissett, and many more. I want to extend a special thanks to Dr. Hengjie Yang, who has been a wonderful mentor to me throughout my undergraduate and graduate education and whose doctorate work has inspired my own research.

I want to thank all of my friends who have supported me throughout my education at UCLA: Andrew Arrieta, Ethan Uetrecht, Ranger Woodland, Claire Sterling, Melvin Prasetyo, Noah Himed, Chris Ryu, and many others. To all of you, I am grateful for your friendship.

Finally, I extend my deepest gratitude to my family, whose unconditional love and support have made this thesis possible.

CHAPTER 1

Introduction

Design of low complexity short blocklength codes is instrumental for modern reliable communication. Many such codes have been presented throughout the years, including convolutional codes, turbo codes, low-density parity-check (LDPC) codes, and polar codes. In recent years, list decoding with cyclic redundancy check (CRC) concatenated codes has been shown to perform very well at short blocklengths [3] [4]. In this thesis, we present designs and performances of CRC-aided convolutional and polar codes for a variety of practical contexts.

Polar codes, introduced by Arıkan in [5], have received wide attention as a provably capacity achieving class of codes, even making their way into the 5G standard [1] [6]. However, polar codes under successive cancellation (SC) decoding are well known to be sub-optimal for short blocklengths. To solve this, a common approach is to precode with a CRC before polar encoding and to decode with a successive cancellation list (SCL) decoder. This CRC-aided SCL decoding of polar codes has been shown to significantly improve polar code performance over SC decoding [4] [7]. In this thesis, we call the concatenation of a CRC and polar code a CRC-Polar code.

In contrast, convolutional codes are a much older class of code, first introduced by Peter Elias in the 1950s [8]. In the decades since, two main classes of convolutional codes have emerged. These are zero terminated convolutional codes (ZTCCs) and tail biting convolutional codes (TBCCs) [9]. Recently, many publications have shown the effectiveness of CRC-aided list decoding of convolutional codes [10], [3]. Following the notation of CRC-Polar codes, we denote these concatenated codes by CRC-ZTCC or CRC-TBCC codes.

These works design CRCs for each specific convolutional code used in order to minimize the frame error rate (FER) union bound based on the distance spectrum of the concatenated CRC-convolutional code. CRCs with this property are known as *distance-spectrum-optimal* (DSO) [11].

We consider two different channels for the design of our codes. The first is a standard binary input additive white Gaussian noise (BI-AWGN) channel with BPSK modulation. The second is a noncoherent orthogonal Q-ary frequency shift keying channel (QFSK) with AWGN, where Q is a power of two. We aim to design good short blocklength codes for these channels. For our work, we focus on designs for polar codes and convolutional codes with CRC-aided list decoding.

In 2010, Polyanskiy, Poor, and Verdú presented tight bounds for the performance of finite blocklength codes [12]. For a code with a fixed blocklength n and fixed number of codewords M, the FER of the best (n, M) code is upper bounded by the *random-coding union bound* (RCU bound) and lower bounded by the *meta-converse bound* (MC bound). While the RCU bound acts as an achievability bound, in practice it is very difficult to design codes that can beat the RCU bound. As a result, the RCU bound acts as a good benchmark to compare against for the performance of short blocklength codes.

This thesis consists of two largely unrelated parts. In the first part, we compare CRC-TBCC and CRC-Polar codes on the BI-AWGN BPSK channel, with an eye toward the blocklengths used in the 5G standard. We design DSO CRCs for each code and compare the FER performance, undetected error rates, and decoding complexities of these two classes of codes. We also compare the distance spectra of these two classes of codes, analyzing the differences between the distance spectra and how these differences affect the error performance.

In the second part, we instead consider the Q-ary noncoherent orthogonal signaling channel and design CRC-convolutional codes for this channel. We build on work by Ryan and Wilson [2] and design Q-ary DSO CRCs for good Q-ary convolutional codes. We also extend the work of Font-Segura *et al.* [13], who derived a saddlepoint approximation for the

RCU bound, by modifying their equations for the Q-ary noncoherent orthogonal channel. We compare our CRC-ZTCC and CRC-TBCC designs to the RCU bound found by this saddlepoint approximation.

1.1 Summary of Contributions

Chapter 2 Contributions

In Chapter 2, we present two different ways to improve the FER performance of the CRC-Polar concatenated code used for the 5G PBCH channel.

The first way is to shorten the length of the CRC. A shorter CRC for a polar code results in more frozen bits, which have 100% reliability. As a trade off, a shorter CRC has less powerful detection capabilities, making undetected errors more likely. We show that shortening the CRC length from 24 bits to 11 bits significantly improves the overall error rate performance, at the cost of undetected error rate.

The second way is to replace the CRC-Polar concatenated code with a CRC-TBCC concatenated code. Following [14], we design distance-spectrum-optimal CRCs for a specific rate-1/5 TBCC from [10]. With this design, we show that we can further improve the error rate performance. Our best design approaches within 0.2 dB of the RCU bound.

We also analyze the decoding complexity of both the Polar and TBCC decoders through decoding run time. We find that the LVA decoder for CRC-TBCC codes is significantly faster than the SCL decoder for CRC-Polar codes, and is therefore able to support much higher list sizes with equivalent decoding run times.

Chapter 3 Contributions

Building on ideas from Chapter 2, in Chapter 3 we design CRC-TBCCs and CRC-Polar codes with equivalent rates, blocklengths, and CRC lengths to compare their performance

and distance properties.

We take the design idea of DSO CRCs for convolutional codes presented in [11] and develop efficient design procedures for the design of DSO CRCs for polar codes. We then design DSO 11-bit CRCs for the (512,43) polar code in 5G and a (516, 43) rate-1/12 TBCC, and we puncture four bits of the CRC-TBCC to rate match the CRC-Polar code.

The distance spectra of the CRC-Polar and punctured CRC-TBCC codes are then analyzed. At high SNR, the FER performance of block codes converges on the union bound, which is determined by the distance spectrum of the code. We show that the distance spectra of these two codes are qualitatively different and result in different effects on the union bounds.

Finally, we simulate our optimal CRC-TBCC and CRC-Polar codes using various list decoders and analyze the FER performance, undetected error rate, and decoding complexity. We show that our CRC-TBCC design has better error rate performance and a significantly faster decoder than the CRC-Polar. Similar to the CRC-TBCC in Chapter 2, our best CRC-TBCC design approaches within 0.2 dB of the RCU bound.

Chapter 4 Contributions

In Chapter 4, we are interested in CRC-aided list decoding of convolutional codes for non-coherent orthogonal signaling with non-binary alphabets. Once again building on [11], we generalize the design procedure to Q-ary alphabets to design Q-ary DSO CRCs for Q-ary zero-state terminated convolutional codes (ZTCCs), where Q is a power of two.

Once again, the RCU bound is used as a benchmark for the performance of our codes. The calculation of the RCU bound as presented in Polyanskiy *et al.* [12] is generally intractable for most practical codes, including the codes we use. Instead, we use a saddlepoint approximation of the RCU bound presented by Font-Segura *et al.* in [13]. Given the exotic nature of our channel, we derive a channel equation for this channel and a set of modifications

to the equations in [13] to work for our channel.

Finally, we present simulation results for a set of 4-ary CRC-ZTCCs. We use 4-ary ZTCCs given by Ryan and Wilson in [2] and design optimal 4-ary CRCs for these ZTCCs. We then compare the TFR performance of these CRC-ZTCCs against the RCU bound and normal approximation. We find that our best performing CRC-ZTCC has a gap to RCU bound of 0.6 dB at TFR $= 10^{-4}$.

Chapter 5 Contributions

In Chapter 5, we extend the work in Chapter 4 to Q-ary CRC-TBCCs and higher orders of Q. We generalize the DSO CRC search algorithm for TBCCs in [14] for non-binary CRC-TBCCs. We discuss the challenges in using this algorithm for large values of Q, and we present optimizations to reduce the complexity of the algorithm.

We design CRC-TBCCs for $Q \in \{4, 8, 16\}$ and present simulation results for these codes. We find that our 4-ary CRC-TBCCs gain about 0.2 dB over our 4-ary CRC-ZTCC designs in Chapter 4, resulting in a gap to RCU bound of roughly 0.45 dB at a TFR of 10^{-4}. Our best 8-ary CRC-TBCC design has a gap to normal approximation of about 0.6 dB, and our 16-ary CRC-TBCC designs have a gap to the normal approximation of around 0.8 dB.

Part I

Comparison of Convolutional and Polar Coding with CRC-Aided List Decoding on the BI-AWGN BPSK Channel

CHAPTER 2

CRC-Aided List Decoding of Convolutional and Polar Codes for Short Messages in 5G

This chapter was first presented at the IEEE International Conference on Communications (ICC) in May 2022. A written version of the presentation is available in [15].

Polar codes have seen wide interest since Arıkan first described the paradigm and showed that it could achieve channel capacity [5]. Polar codes have found application in the physical broadcast channel (PBCH) of the 5G standard [1], [6]. In particular, Fig. 2.1 shows how a polar code is used to transmit a 32-bit message over the 5G PBCH. First the 32-bit message is protected by a 24-bit CRC to provide a 56-bit input to the polar code. The polar code produces 512 bits, which are augmented by repetition to produce an 864-bit 5G PBCH codeword. This chapter explores ways to improve the frame error rate (FER) vs. E_b/N_0 performance of this 5G PBCH code and considers alternatives.

For example, FER vs E_b/N_0 improvement is achieved by reducing the length of the CRC. CRCs, as described in [16], are very powerful as error detecting outer codes. However, for the best FER vs E_b/N_0 performance, the error detection benefit provided by the CRC needs to be balanced with the corresponding overhead requirement. We show that replacing the 24-bit CRC with the smaller 11-bit or 12-bit CRC increases the number of frozen bits and reduces the FER for a given E_b/N_0.

This paper further improves the E_b/N_0 performance by replacing the polar code with a tail-biting convolutional code (TBCC). Specifically, a rate-1/5 TBCC is concatenated

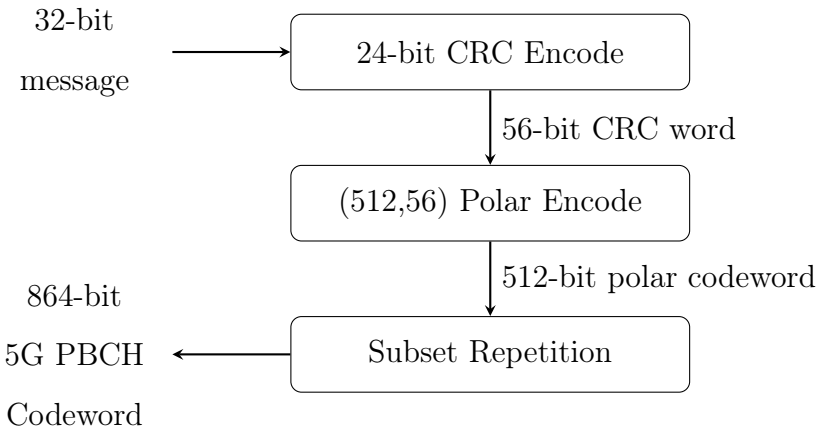

Figure 2.1: Block diagram of 5G PBCH polar encoding scheme. The PBCH uses a 24 bit CRC, then polar encodes to 512 bits before applying repetition to get to 864 bits.

with a CRC optimized for the specific TBCC. Lou *et al.* [11] introduced distance-spectrum-optimal (DSO) CRCs for zero-terminated convolutional codes. Recently, Yang *et al.* [3], [14] presented an algorithm for finding DSO CRCs for tail-biting convolutional codes, which this paper employs to find the optimized CRCs used in this paper.

As is the case for polar codes, CRC-TBCC performance is enhanced by optimizing the CRC length. Using DSO CRCs and the CRC length that minimizes the FER for a specific E_b/N_0 yields a CRC-TBCC concatenated code that has better performance than the CRC-Polar concatenation. Decoder complexity and performance depend on list size, but the list decoder for the CRC-TBCC code required less run time on our computer for better performance than the list decoder for the CRC-Polar concatenation.

2.1 Background

This section describes the polar and TBCC codes that we will consider, presents a list decoder for CRC-TBCC and CRC-Polar concatenated codes that employs parallel list decoding with exponentially increasing list sizes, and defines erasure failures and undetected errors.

2.1.1 Polar Codes

Polar codes were first introduced by Arıkan in [5] as a code suitable to take advantage of the channel polarization paradigm that he discovered. Polar codes compute the codeword by multiplying a message vector by a polar coding matrix. The message vector contains both actual message bits and "frozen" bits that are set to a fixed value and do not convey information. The polarization paradigm ensures that the actual message bits have very high reliability, while low reliability bits are frozen and convey no information. Polar codes have been shown to achieve channel capacity for asymptotically long blocklengths; however, they are less reliable with short messages.

Also presented in [5] is a proposed decoder for polar codes called a Successive Cancellation (SC) decoder. This decoding algorithm decodes the received codeword one bit at a time, using previously decoded bits to help decide the current bit. Frozen bits carry no information and are known to the decoder, so a decision only needs to be made on the message bits in the codeword. As noted in [17], the SC decoder is effective for decoding long messages, but is less effective for decoding the short messages used in 5G.

This is addressed in 5G by using Successive Cancellation List (SCL) decoding in conjunction with CRC precoding [1]. Instead of making a hard decision on each message bit of the received codeword, the SCL algorithm [4] instead implements parallel decoders, one for each of a set of possible decisions about the previous bits. When all the parallel decoders have each selected their distinct prospective codewords, these candidate codewords are then checked to see which pass the CRC check, and the most likely candidate that passes the check is selected. If no candidate passes the CRC, then a decoding failure is reported.

The performance of SCL improves as the number of parallel decoders, i.e. the list size, is increased. However, this improved performance comes with a significant complexity increase to support the parallel decoding [4]. For a specified list size, SCL retains the most likely codeword candidates at each step. In addition to the SCL algorithm, there have been many

other proposed improvements to Arıkan's initial SC decoder to increase decoding accuracy, decrease latency, and decrease complexity [18–20].

The polar code used in this paper is the PBCH polar code from the 5G standard [1]. This code has 32 message bits and is encoded with a 24-bit CRC. The 24-bit CRC has polynomial 0x1B2B117, with the most significant bit corresponding to the degree-24 term of the polynomial. This 56-bit message and CRC is then encoded with a (512,56) polar code, and then the first 352 bits are repeated to arrive at a final 864-bit codeword, as illustrated in Fig. 2.1. We omit the bit interleaving and CRC parity distribution used in the 5G standard, as these have no impact on the performance over an AWGN channel with no inter-symbol interference.

In addition to the 24-bit CRC that is used by the PBCH code, we also simulate this code with an 11-bit CRC provided in [1] that is used for the physical uplink control channel (PUCCH) code, followed by a (512,43) polar code and the same bit repetition. This 11-bit CRC has polynomial 0xE21. Recently, Baicheva and Kazakov [21], [22] performed an analysis on the 5G CRCs and presented alternative CRCs for polar codes than those in [1]. We also simulate the polar code with the 11-bit CRC 0xB5F and the 12-bit CRC 0x1395, provided by Peter Kazakov to achieve the best performance for this polar code.

A sequence of expected reliabilities is also provided in [1] to select the bits having the lowest expected reliabilities to be frozen bits. This paper uses adaptive SCL decoding for all polar codes we consider, as described in Section 2.1.3.

2.1.2 Tail-Biting Convolutional Codes

In contrast to polar codes, convolutional codes have been in use for decades [8]. Convolutional codes can be used for transmitting streams of data with continuous decoding [23], but they can also function as block codes. One form of block convolutional codes is the class of TBCCs [9], which avoid the overhead incurred by zero termination.

Our paper focuses on TBCCs because of their rate efficiency. The TBCC proposed in this paper as an alternative to polar coding for the 5G PBCH is taken from [10]. This code is a rate-1/5 TBCC with 32 message bits, concatenated with a CRC. The encoder has 8 memory elements with generator polynomials $(575, 623, 727, 561, 753)$ in octal. We do not use an interleaver for this code, and this is not a serially concatenated turbo code. The use of such a code is explored in [24].

Lou *et. al.* [11] show how low undetected error rate performance of a convolutional code is dominated by the minimum distance spectrum of the code. They present an algorithm to find DSO CRCs for a zero-terminated convolutional code by maximizing the minimum distance of the CRC-ZTCC concatenated code. This process is generalized to tail biting convolutional codes in [14].

In [10], CRCs were designed for zero-terminated convolutional codes even though some simulations in [10] involved TBCCs. In this paper, we deployed the algorithm described in [14] to identify optimal CRCs for the TBCC implementation of $(575, 623, 727, 561, 753)$ with CRC lengths varying from 8 to 16 bits. Table 2.1.2 provides the optimal CRC polynomials that resulted form our search.

CRC length	8	9	10	11	12	13	14	15	16
CRC poly (hex)	101	21F	4D5	A9D	123B	27C5	7CCF	8441	18077

Table 2.1: CRC polynomials for the TBCC at different CRC lengths. Each polynomial is given in hexadecimal with the most significant bit corresponding to the highest order term.

The Viterbi algorithm is a maximum-likelihood decoder for convolutional codes. The decoder traverses the trellis identifying the most likely path to each state in the trellis based on the received codeword. When multiple paths converge to the same state in the trellis, the decoder selects the most likely path, making an arbitrary choice to break ties. At the end of the trellis, the decoder selects the most likely surviving path. The Viterbi algorithm can

be augmented to support parallel list decoding [25], where every state stores a list of the L most likely paths instead of a single most likely path.

The TBCCs in this paper are decoded using an adaptive parallel list Viterbi algorithm (LVA) decoder based on the one described in [25]. The details of this decoder are also described in Section 2.1.3.

2.1.3 Adaptive List Decoding

This paper uses parallel list decoding with a doubling list size to explore the FER performance and decoding run time of both the 5G PBCH polar code and the proposed TBCC alternative. The parallel list decoder is implemented as an SCL decoder for polar codes and as an LVA [25] for convolutional codes.

This approach was proposed in [26] for polar codes, where it was called an "adaptive SCL" decoder. Each iteration of the algorithm acts like a parallel LVA or SCL decoder for the given list size. If a message candidate is not found that passes the CRC check, then the list size doubles until either a codeword is found that passes the CRC check or the maximum list size is reached. A block diagram of this list decoding algorithm is shown in Fig. 2.2.

There are two types of errors that can occur when using our list decoder. An *erasure* occurs when none of the decoded message candidates that the list decoder finds have a valid CRC when the decoder reaches the maximum list size. An *undetected error* occurs when one of the message candidates passes the CRC check, but it is not the same as the codeword sent. In this paper, we use the sum of the erasure rate and the undetected error rate (UER) as a primary metric for performance, which we refer to as the Total Failure Rate (TFR).

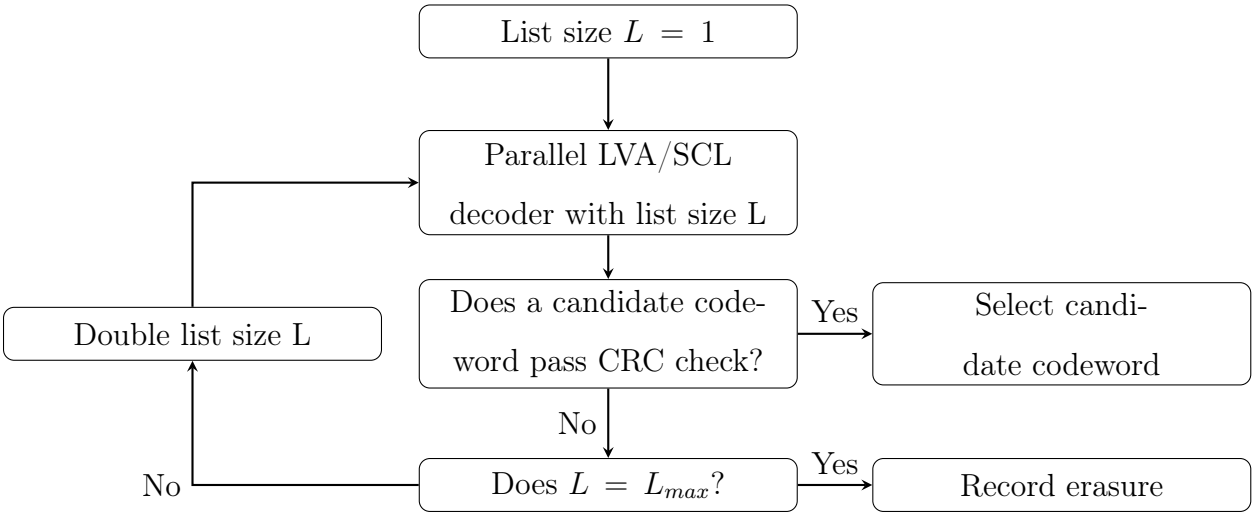

Figure 2.2: Block diagram of the adaptive parallel list decoder algorithm. It starts with a list size of 1 and runs the parallel LVA or SCL algorithm. This is repeated with the list size doubling every iteration until either a candidate codeword is found that passes the CRC check or the maximum list size is reached.

2.2 Optimal CRC Length for TBCC

This section shows how a specified TBCC has an optimal CRC length that minimizes the TFR. As an initial matter, a longer CRC should lead to an improved TFR at a fixed signal-to-noise ratio (SNR), but the longer CRC also reduces the code rate. To fairly compare the CRCs, we consider the TFR as a function of E_b/N_0, which accounts for the rate loss incurred by a longer CRC.

For two different fixed values of E_b/N_0, 2.5 dB and 3.5 dB, Fig. 2.3 shows the erasure rate, UER, and TFR for simulating the CRC-TBCC with different CRC lengths with a maximum list size of 2048. The CRCs used in these simulations are shown in Table 2.1.2, and for each length the CRC used is optimal according to the procedure from [14].

Fig. 2.3 shows that, for this example, as CRC length increases, erasure rate monotonically increases and UER monotonically decreases. Combining these two effects, TFR is convex or

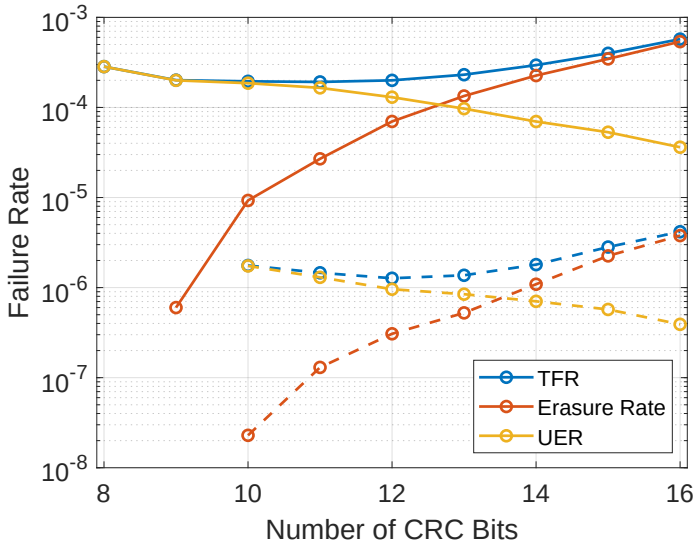

Figure 2.3: Plot of erasure failure rate, undetected error rate, and total failure rate vs. number of bits in the CRC for the TBCC. The solid curves correspond to a E_b/N_0 of 2.5 dB, and the dashed curves have an E_b/N_0 of 3.5 dB. A maximum list size of 2048 was used. The CRC length that minimizes TFR is 11 bits at 2.5 dB and 12 bits at 3.5 dB, but nearby CRC lengths have nearly equivalent TFRs.

quasi-convex with a single global minimum.

Thus, there is a CRC length that minimizes the TFR for a specified value of E_b/N_0 and a specified maximum list size. The optimal CRC length depends on the value of E_b/N_0. At $E_b/N_0 = 2.5$ dB, the CRC length that minimizes the TFR is 11 bits. At $E_b/N_0 = 3.5$ dB, the optimal length is 12 bits. However, the difference in FER between 11-bit and 12-bit CRCs at these values of E_b/N_0 is almost negligible, and can change when the maximum list size is changed.

2.3 Comparison of TBCCs and Polar Codes

Consider the problem of transmitting a 32-bit message. This section begins by comparing four CRC-Polar solutions to a rate-1/5 TBCC with an optimized CRC. For the TBCC, the

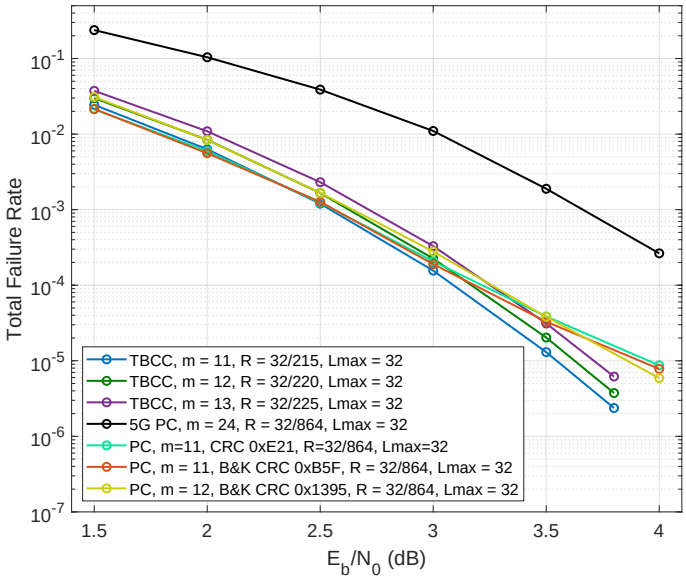

Figure 2.4: TFR vs. E_b/N_0 of CRC-TBCC and CRC-Polar codes with various CRCs. A maximum list size of 32 is used for all codes. The CRC-TBCCs and $m = 11$ and $m = 12$ CRC-Polar codes achieve similar performance, with the CRC-Polar codes exhibiting a floor at high E_b/N_0. These codes significantly outperform the $m = 24$ 5G CRC-Polar code.

optimal CRCs for each length, shown in Table 2.1.2, are designed according to [14]. The optimal CRC length, as discussed in Sec. 2.2, is also considered.

2.3.1 TFR vs. E_b/N_0 and TFR vs. Run Time

For a fixed maximum list size of 32, Fig. 2.4 shows TFR vs. E_b/N_0 for the 5G PC with 24-bit CRC solution, a rate-1/5 TBCC with CRCs of length $m = 11, 12$, and 13, and three additional PC/CRC solutions. The CRC-TBCC solutions all have similar performance, but the best performance is seen for CRC length $m = 11$ for all E_b/N_0. Note that the maximum list size $L_{max} = 32$ is significantly smaller than the maximum list size $L_{max} = 2048$ considered in Sec. 2.2 where the $m = 12$ CRC is optimal at $E_b/N_0 = 3.5$ dB.

In Fig. 2.4, the 5G PC with the $m = 24$ CRC specified in the 5G standard performs

significantly worse than the the CRC-TBCC solutions. To improve the PC performance, the CRC length was reduced to match the CRC length used for the convolutional code by using two $m = 11$ and one $m = 12$ CRC. The TFR vs. E_b/N_0 performance of the CRC-Polar codes with the shorter CRCs is similar to that of the TBCC/CC solutions, except for a TFR degradation seen at high E_b/N_0. This TFR degradation for adaptive list decoding of polar codes is further explored in Chapter 3.

While all the curves in Fig. 2.4 used the same maximum list size, the decoders do not have the same complexity or run time. To explore complexity vs. TFR performance, Fig. 2.5 shows TFR as a function of average simulation run time [1] at $E_b/N_0 = 3.5$dB for all the CRC-Polar codes and the best-performing CRC-TBCC from Fig. 2.4. All decoders used C implementations of the adaptive list decoding paradigm of [26]. We tried to make both the LVA and SCL implementations as efficient as possible, but of course other implementations may result in different run-time comparisons.

In general, the CRC-TBCC is able to support a higher maximum list size and achieve a lower TFR for a specified run time. For example, Fig. 2.5 shows that at an average decoding run time of 2.4 ms per decoded codeword, the CRC-TBCC achieves a TFR of 1.74×10^{-6} using a maximum list size of 1024, while the best polar code with a short CRC only achieves 2.23×10^{-5} with a list size of 64. The 5G polar code with a 24-bit CRC achieves a TFR of 2.05×10^{-3} with a list size of 32.

When the maximum list size is small in Fig. 2.5, increasing the list size can dramatically reduce TFR while having a negligible impact on decoding run time. Further increases in list size provide diminishing returns in TFR performance but carry significant run time penalties. Essentially, when the list size required to pass the CRC check is very large, the codeword that finally passes the CRC check will often be an undetected error failure so that TFR is not improved.

[1] All simulations were performed on a System76 Galaga Pro Ubuntu laptop with an Intel Core i7-8565U CPU @1.8GHz x 8 Processor.

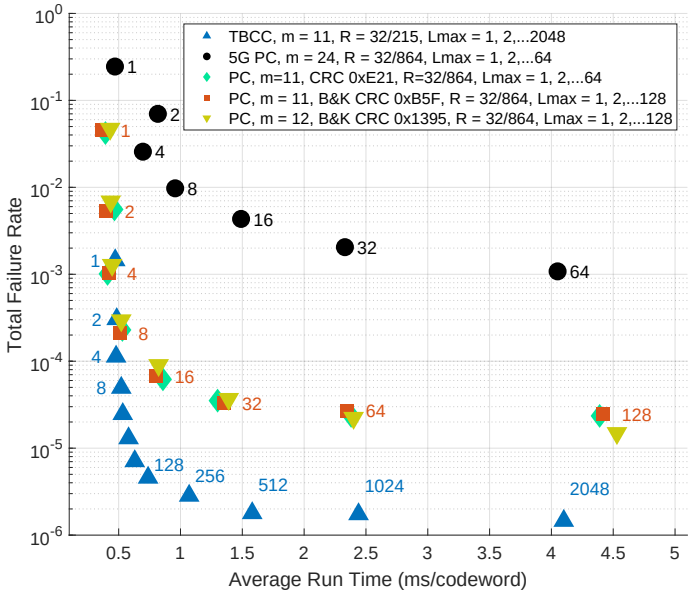

Figure 2.5: Average decoding run time in milliseconds of $m = 11$ TBCC and all PCs vs. TFR at $E_b/N_0 = 3.5$dB. Both $m = 11$ PCs and the $m = 12$ PC achieve far greater TFR performance at equivalent decoding run times than the 5G PC, and the $m = 11$ TBCC performs even better. There exists a trade-off between TFR and average decoding time when varying list size. Eventually, increasing list size further does not provide any benefit to reducing TFR.

We did not consider list sizes that required more than 5 ms of average run time. For the polar codes, this limited maximum list sizes to 64 or 128. However, Fig. 2.5 shows that negligible improvement in TFR would be expected for larger maximum list sizes for these codes.

For the list sizes that resulted in run times of about[2] 2.4 ms at $E_b/N_0 = 3.5$dB, Fig. 2.6 provides curves showing TFR vs. E_b/N_0 for the codes shown in Fig. 2.5.

[2]The run times are as follows: 2.44 ms per codeword for TBCC with $L_{max} = 1024$, 2.38 ms per codeword for 5G 0xE21 $m = 11$ PC with $L_{max} = 64$, 2.35 ms per codeword for B&K 0xB5F $m = 11$ PC, 2.4 ms per codeword for B&K 0x1395 $m = 12$ PC with $L_{max} = 64$, and 2.45 ms per codeword for 5G $m = 24$ PC with $L_{max} = 32$

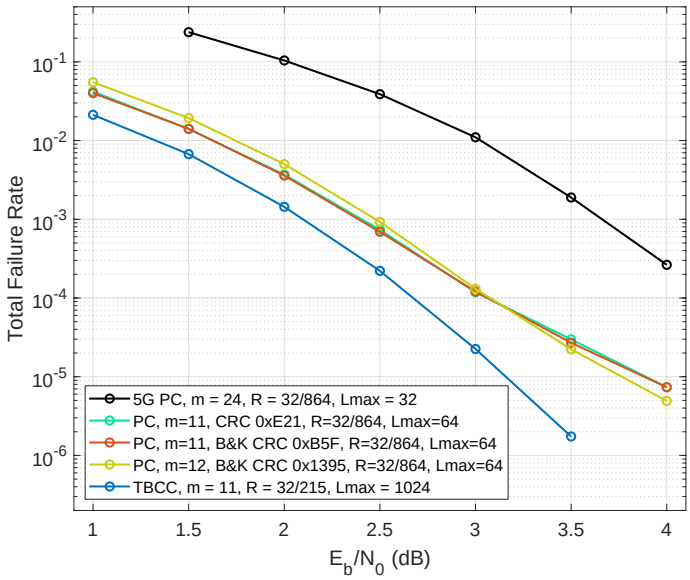

Figure 2.6: TFR vs. E_b/N_0 of $m = 11$ TBCC, 5G PC, and all $m = 11$ and $m = 12$ PCs. Each code is has a nearly equivalent average decoding runtime of around 2.4 ms per codeword, with the list sizes set according to Fig. 2.5 to achieve this.

2.3.2 Comparison to RCU and MC bounds

In 2010, Polyanskiy, Poor, and Verdú presented the RCU and MC bounds for finite blocklength codes [12]. These bounds act as an achievability and a converse bound on FER for codes of a given rate and blocklength, and serve as good benchmarks for the performance of finite blocklength codes. Fig. 2.7 shows TFR vs. E_b/N_0 for the CRC-TBCC with $m = 11$, as well as saddlepoint approximations [13] for the RCU bound and MC bound for this CRC-TBCC. At $L_{max} = 2048$, the CRC-TBCC TFR approaches the RCU bound.

Fig. 2.8 shows the E_b/N_0 gap from the RCU bound for all CRC-TBCC and CRC-Polar codes that were considered in the earlier figures. For TBCCs with $L_{max} = 2048$, the $m = 11$ CRC has the smallest gap across the entire TFR range. For example, at TFR of 1.46×10^{-6} the gap to the RCU bound is 0.19 dB.

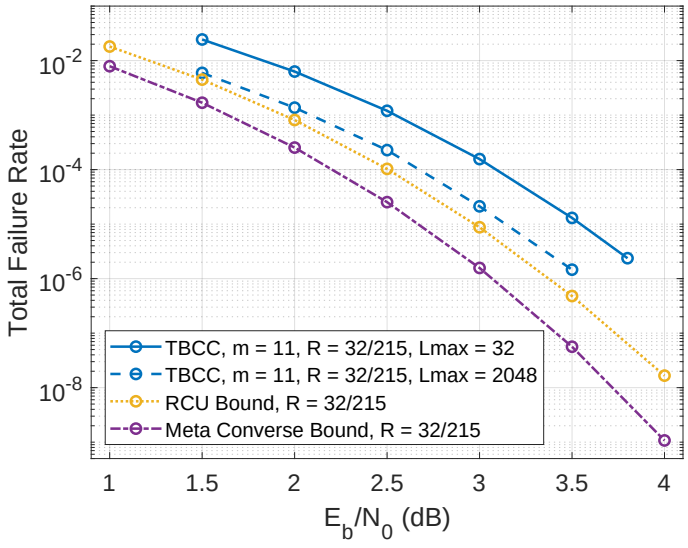

Figure 2.7: Plots of TFR, RCU Bound, and Meta Converse Bound of the $m = 11$, Rate 32/215 TBCC vs. E_b/N_0 dB. The $L_{max} = 2048$ curve approaches very close to the RCU bound. However, as shown in Fig. 2.5, increasing the list size further is unlikely to improve TFR further.

2.4 Exactly Matching Polar Rates via Repetition

The 5G CRC-Polar solution for the PBCH has rate 32/864, which is about 4 times lower than the CRC-TBCC solutions we propose. However, the number of transmitted bits can be increased through repetition to 864 bits so that our CRC-TBCC solutions can be deployed with exactly the same rate as the PC/CRC solution of 5G. Our proposed rate 1/5 TBCC with an 11-bit CRC has an overall rate of 32/215. Repeating 211 of the 215 code bits four times, and repeating the remaining 4 bits five times times produces an overall rate of 32/864, which exactly matches the CRC-Polar solution.

Consider a base code C. The code resulting from M times repetition of every code bit of C has exactly the same TFR performance as the original code at a fixed value of E_b/N_0. To see this, note that for fixed E_b/N_0 repeating each individual code bit M times increases the value of E_b by M, which requires a noise level M times higher in order to maintain

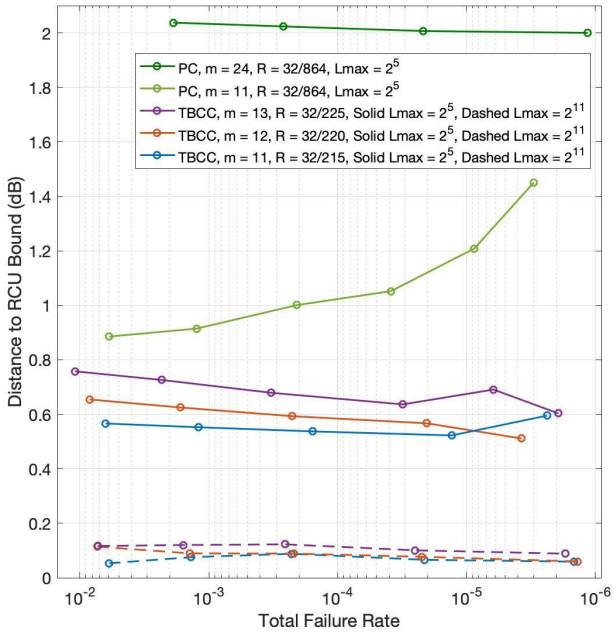

Figure 2.8: Gap to RCU bound vs TFR for all TBCC and polar codes simulated in this section. Solid lines have $L_{max} = 32$, and dashed lines have $L_{max} = 2048$. The TBCCs with $L_{max} = 2048$ get very close to RCU Bound. The TBCCs with $L_{max} = 32$ also get closer to the RCU bound than all polar codes with $L_{max} = 32$. All $m = 11$ and $m = 12$ PCs outperform the $m = 24$ 5G PC.

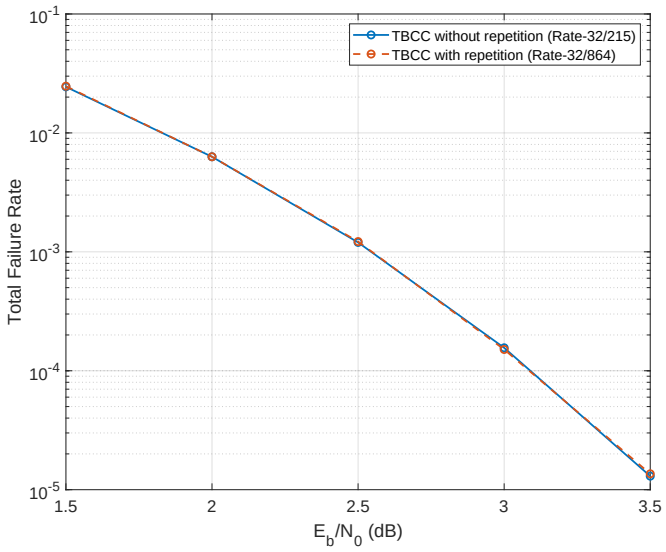

Figure 2.9: TFR vs. E_b/N_0 curves for $m = 11$ CRC-TBCC with and without bit repetition. These two codes have identical TFR performance.

an equivalent E_b/N_0. However, each received instance of the repeated code bits may be averaged together before decoding with no loss in performance, creating an effective symbol that reduces the noise level by a factor of M. These effects cancel each other out, so the symbols in the two codes will have the exact same level of effective noise, so that the decoder performance is equivalent.

We now analyze applying repetition to our rate-32/215 TBCC with an 11-bit CRC to get a new code that is rate-32/864. Importantly, not every bit is repeated the same number of times. 211 of the 215 code bits are repeated 4 times, but the remaining 4 bits are repeated 5 times. Thus, in the repetition code, the 4 bits repeated an extra time will have significantly lower noise than if every bit was repeated an equal number of times, and the remaining 211 bits will have slightly higher noise.

If we include the rate penalty but not the noise benefit of repeating those last four bits an extra time, we can bound the loss of the rate-32/864 to within 0.02 dB of the performance of the original rate-32/215 CRC-TBCC code or the equivalent rate-32/860 code. Even if we

assume the worst case of a 0.02 dB loss in TFR performance with the repetition code, the difference between the repetition code and the original code is negligible. So, we expect the repetition code to have nearly equivalent performance to the original.

We simulated the repetition code to confirm the tightness of this bound, which can be seen in Figure 2.9. When simulated, the curves for the original code and the repetition code lie directly on top of each other, being nearly impossible to distinguish between.

2.5 Conclusion

This chapter shows two ways to improve the TFR vs. E_b/N_0 performance of the current 5G PBCH CRC-Polar code. Reducing the CRC from 24 bits to 11 or 12 bits, which allows significantly more bits to be frozen, improves performance while still utilizing the paradigm of a polar code that uses CRC-aided list decoding. It is worth noting that the 24-bit CRC has a significantly lower undetected error rate than the polar codes with shorter CRCs, due to the extra bits allocated to the CRC for error checking. Even better TFR performance is achieved by replacing the polar code with a rate-1/5 TBCC with CRC-aided list decoding. Repetition coding can be used to exactly match the rate of the current 5G PBCH code, but provides no benefit to TFR performance.

For the TBCC, the length of the CRC was optimized and the CRC polynomial was designed to optimize the distance spectrum of the concatenated CRC-TBCC so as to minimize the TFR. The CRC-TBCC solution has TFR vs. E_b/N_0 performance very close to the RCU bound when the maximum list size is allowed to be large, so that the CRC-TBCC solution is approaching the best performance that is theoretically guaranteed to be possible. Notably, the decoding complexity for a given list size is lower for the TBCCs than for the polar codes, which allows the CRC-TBCC to benefit from larger maximum list sizes.

Our CRC-TBCC designs utilize [14], which shows how to optimize CRCs for use with TBCCs. We tried to find the best available CRC to use with the polar code. We considered

the 11-bit CRC specified in the 5G standard [1], although that CRC is not specified for this polar code. We also noted the work of Baicheva and Kazakov [21], [22] focused on designing CRCs to be used with polar codes and contacted them for assistance. We are grateful to Peter Kazakov for providing the 11-bit and 12-bit CRCs that provided the best performance that we observed for CRC-aided decoding of the polar code. These CRCs maximize the free distance of the CRC error detection code for the specific overall code lengths of 43 and 44 bits, respectively.

This chapter focused on ways to improve the performance of the PBCH code in the 5G standard, which resulted in an empirical comparison between polar and convolutional codes. The next chapter takes a more theoretical approach to these comparisons, where we analyze the distance spectra of these two classes of codes and study more in depth the effects of CRC-aided adaptive list decoding.

CHAPTER 3

Design, Performance, and Complexity of CRC-Aided List Decoding of Convolutional and Polar Codes for Short Messages

This chapter is an adaptation of a paper submitted to IEEE Transactions on Communications.

In the previous chapter, we showed that the design of this CRC-Polar concatenated code in the 5G standard is sub-optimal when attempting to minimize TFR. Specifically, it is shown that using a shorter CRC with the polar code significantly improves TFR performance of the code. It is also shown there that a TBCC [9] concatenated with a CRC and decoded with a LVA decoder can outperform the improved CRC-Polar code. CRC-aided list decoding of TBCCs has recently been shown [3], [10] to perform very well at short blocklengths, approaching and even surpassing the random coding union (RCU) bound.

The comparison in Chapter 2 suggests that CRC-TBCCs are a better solution than CRC-Polar codes for the 5G standard, but the analysis is incomplete. In this chapter, we expand on the work in Chapter 2 by designing the best CRC-TBCC and CRC-Polar codes we can, and we perform a direct comparison between our code designs. In addition to comparing simulation results, we also compare the distance spectra of the CRC-Polar and CRC-TBCC.

Lou *et al.* [11] show the importance of designing DSO CRC codes for specific convolutional codes. In this chapter, we design a DSO 11-bit CRC for a specific TBCC using the algorithm in [14]. We also design a DSO 11-bit CRC for the 5G Polar code by extending the ideas

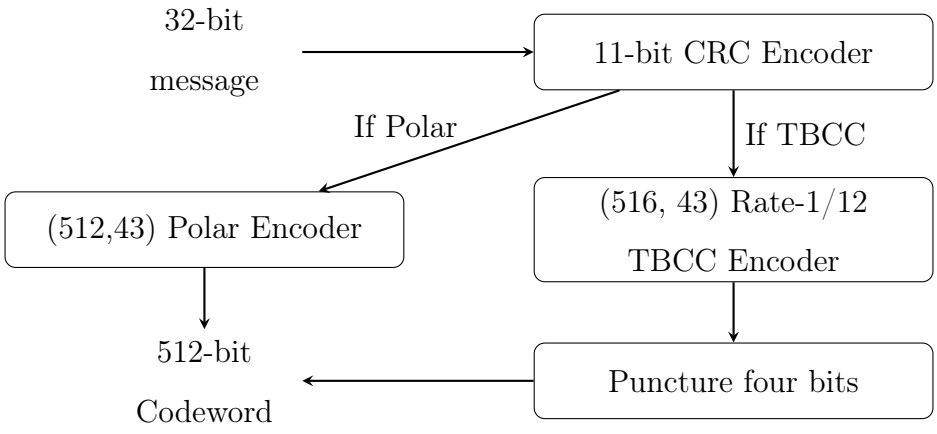

Figure 3.1: Encoding Scheme for CRC-Polar and punctured CRC-TBCC codes.

in [11] to CRC-Polar codes. This improves the design over the CRC-Polar in the previous chapter, where we did not design CRCs for the specific polar code used.

We wish to design CRC-Polar and CRC-TBCC codes with equivalent message lengths, CRC lengths, and blocklengths so as to facilitate a fair comparison. We also want to avoid rate matching through bit repetition or puncturing as much as possible. Given that polar encoding is limited to blocklengths that are powers of two, we aim for a target blocklength of 512 bits in our CRC-Polar and CRC-TBCC designs. The encoding schemes of the CRC-TBCC and CRC-Polar are shown in Figure 3.1.

3.1 Polar Code Design

In the 5G New Radio (5G NR) technical specification [1], six CRCs are proposed to concatenate with polar codes and LDPC codes for different message lengths. We list the six proposed CRCs in Table 3.1. The generator polynomials of the CRCs are denoted in hexadecimals, where the high-order coefficients correspond to the most significant bits. For example, the generator polynomial $x^{11} + x^{10} + x^9 + x^5 + 1$ for CRC11 in Table 3.1 is denoted as 0xE21.

For the PBCH polar code in 5G, the 32 message bits are first encoded with CRC24C listed

Table 3.1: CRCs proposed in the 5G NR technical specification [1]

Label	Generator Polynomial
CRC24A	0x1864CFB
CRC24B	0x1800063
CRC24C	0x1B2B117
CRC16	0x11021
CRC11	0xE21
CRC6	0x61

in Table 3.1. Then, both the 32 message bits and the appended 24 CRC bits are interleaved together and encoded using a (512,56) polar code. The frozen set of the polar code is determined using the reliability sequence in the 5G standard [1]. A detailed description of the encoding process is given in [27], and a simplified description is also shown in Figure 2.1. Because they do not impact performance on a binary-input AWGN channel, in our work we omitted the rate matching, interleaving, and CRC parity distribution described in [27]. As in Chapter 2, we decode our CRC-Polar codes with a successive cancellation list (SCL) decoder [4].

For a CRC-Polar code with 32 message bits and m CRC bits, $(32 + m)$ synthesized bit channels of the inner polar code need to be unfrozen. Regarding the length of the CRC, the following trade off can be observed. By using a longer CRC, the probability that a random decoding path passes the CRC check at the end of the list decoding process will be smaller, resulting in an expected lower undetected frame error rate. However, a longer CRC corresponds to more unfrozen bit channels for the inner polar code. In this way, the total failure rate of the code might be damaged by introducing too many low reliability bit channels.

This trade off is very similar to the trade off of CRC length m vs. TFR for CRC-TBCC

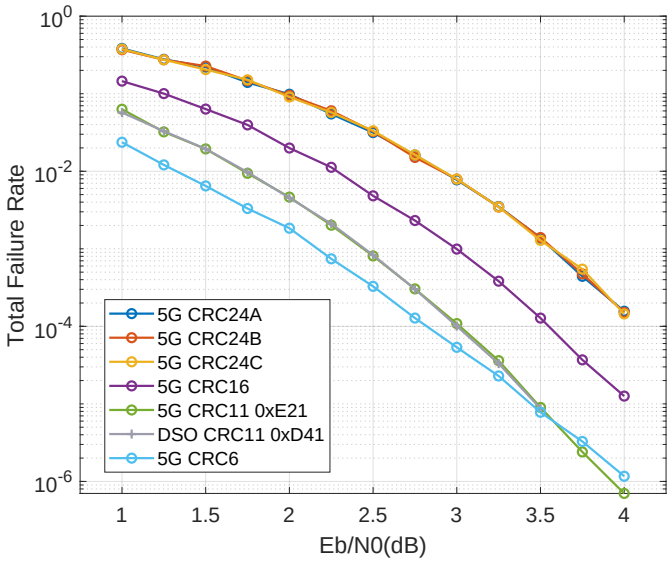

Figure 3.2: List decoding performance (list size $L = 32$) of length-512 5G polar codes with 32 message bits and different CRCs

codes in Figure 2.3, although for CRC-Polar codes the mechanism of the trade off is different. For CRC-Polar codes, a longer CRC results in fewer frozen bits, and thus fewer completely reliable bit channels. For CRC-TBCC codes, a longer CRC increases the blocklength of the codeword, resulting in less energy per code symbol at constant E_b/N_0. Figure 2.3 shows that there is a CRC length that minimizes total failure rate for a CRC-TBCC, and we can expect a similar behavior from CRC-Polar codes.

To find a good CRC length for our CRC-Polar code, we simulate the list decoding performance of all the CRCs (listed in Table 3.1) proposed in the 5G NR technical specification. The total frame error rates of those CRCs are shown in Figure 3.2. In this comparison, the list size is set to $L = 32$, and the channels are binary-input additive white Gaussian noise (AWGN) channels. Our result shows that among all the 5G CRCs, CRC6 has the best performance at low SNRs, while the best performance at high SNRs is obtained by CRC11. Following this result, if we limit our search space to CRC lengths specified in the 5G standard, CRCs with 11 bits achieve the best list decoding performance at high SNR.

This matches the results in Chapter 2, where an 11-bit CRC significantly outperformed a 24-bit CRC for the CRC-Polar code.

However, CRC11 in the 5G standard is not known to be specifically designed together with the inner polar code. Lou [11] and Yang [14] give algorithms for designing DSO CRCs for ZTCCs and TBCCs respectively. In the next subsection, we present analogous design procedures for designing DSO CRCs for polar codes.

3.1.1 Design of Distance Spectrum Optimal 11-bit CRC

It is known that on AWGN channels, the maximum-likelihood (ML) decoding performance of binary linear codes is governed by their weight distribution, and can be well approximated by the union bound. Here, we seek to design an 11-bit CRC that provides the best ML decoding performance at high SNRs. For this purpose, we design a DSO 11-bit CRC.

The union bound on FER based on the distance spectrum is given by

$$FER < \sum_{d=d_{min}}^{n} A(d)P_2(d).$$

Here, $A(d)$ is the number of codewords at weight d, $P_2(d)$ is the pairwise error probability of two codewords at distance d, and n is the blocklength of the code. We wish to find the 11-bit CRC that minimizes this union bound of the CRC-Polar concatenated code. As SNR grows large, this minimization problem can be well approximated by maximizing the minimum distance d_{min} and minimizing $A(d_{min})$. We design an 11-bit CRC according to these criteria.

In the first step of our design procedure, we use the algorithm in [28] to compute the entire weight distribution of the (512,43) inner polar code, whose frozen set is chosen according to the reliability sequence in the 5G standard [1]. The partial weight distribution of this polar code for codewords with weight up to 128 is shown in Table 3.1.1.

In the second step, we use the method in [29] to obtain all codewords of weight 64, and all codewords of weight 96 for this polar code. Consider the following experiment devised

Table 3.2: Partial weight distribution of the (512,43) 5G polar code

d	0	64	96	128
$A(d)$	1	536	9600	496988

in [29]. Transmit the all-zero codeword through AWGN channels in the extremely high SNR regime, and decode the channel output using a list decoder. It is reasonable to expect that in this experiment, the list decoder will produce codewords of low weight. As the list size L increases, since the decoder is forced to generate a list of size exactly L, more and more low-weight codewords emerge. For the (512,43) 5G polar code, by using list size $L = 32768$ in this experiment, we are able to obtain all 536 codewords of weight 64, and all 9600 codewords of weight 96. The list size required for us to obtain all codewords of weight 128 is too large. Hence we stop this experiment at $L = 32768$.

In the third step, we go over all CRCs with 11 bits, and check which one of them can eliminate most of those low weight codewords. We consider all the 11-bit CRCs with generator polynomials in the form $x^{11} + \cdots + 1$, such that the leading coefficient of x^{11} and the constant term at the end are both fixed to be 1. There are $2^{10} = 1024$ 11-bit CRCs in our search space, and we find out that 79 of them can eliminate all codewords of weight 64 and weight 96 for the (512,43) inner polar code.

In the fourth step, we compute the complete weight distribution of the concatenated CRC-Polar codes using all those 79 CRCs by the brute-force search, and find out that the distance spectrum optimal CRC has generator polynomial 0xD41. This CRC-Polar code has $d_{min} = 128$ and $A(d_{min}) = 219$. In Section 3.3, we present the distance spectra of the inner 5G polar code and the CRC-Polar code with CRC 0xD41.

3.1.2 Adaptive List Decoding

In this paper, we decode our CRC-Polar code using a successive cancellation list (SCL) [19] decoder. For large list sizes ($L > 64$), it is very computationally costly to always generate a list of L codewords for every message sent. However, a large list size does tend to significantly improve TFR performance compared to small list sizes.

We choose to implement an adaptive parallel list decoder to balance these effects, as in [29], [15], and Chapter 2 of this thesis. The adaptive list decoding algorithm is as follows. We begin by running a parallel list decoder with an initial list size $L = L_{min}$. If the decoder finds a codeword that passes the CRC check, it selects that codeword and terminates. However, if no codeword on the list passes the CRC check, we double L and run another parallel list decoder. This continues until a valid codeword is found or the maximum list size L_{max} is reached. We will call such a decoder an (L_{min}, L_{max}) adaptive list decoder.

Unfortunately, the SCL decoder is not maximum likelihood, so changing the list size can change whether a codeword appears in the list or not. As a result of this, the performance of the (L_{min}, L_{max}) adaptive SCL decoder performance is not identical to that of the non-adaptive SCL decoder with $L = L_{max}$. This is true when the CRC overhead is only 11 bits because a valid but incorrect codeword is more likely to be chosen. Care must be taken with selecting L_{min} and L_{max} of the adaptive SCL decoder.

3.2 TBCC Code Design

Recent results by Yang *et al.* [3] show that CRC-TBCCs with list decoding can approach and even surpass the RCU bound [12]. In this section we present this design procedure for a low rate CRC-TBCC to compare to the CRC-Polar designed in the previous section.

A comparison of CRC-TBCCs with list decoding to the 5G PBCH CRC-Polar code was performed in Chapter 2. However, the TBCC design used in Chapter 2 was not a fully proper

comparison to the 5G CRC-Polar code. The TBCC in Chapter 2 was a memory-8, rate-1/5 convolutional code, borrowed from [10]. With a 32-bit message and 11-bit CRC, the CRC-TBCC has a blocklength of 215 bits. However, the 5G CRC-Polar has a 512-bit blocklength before bit repetition, resulting in a significantly lower rate code than the CRC-TBCC. This significant difference in blocklength results in a comparison that is not completely fair. We solve this problem by designing a much lower rate TBCC to match the 512-bit blocklength of the CRC-Polar code.

For this paper, we designed a memory-8, rate-1/12 TBCC which we concatenate with an 11-bit CRC, resulting in a (516, 32+11) CRC-TBCC. We puncture four bits to arrive at a (512, 32+11) punctured CRC-TBCC, matching the rate of the 5G CRC-Polar code. We credit Dr. Dariush Divsalar with the suggestion to use such a low rate convolutional code design.

3.2.1 Design of TBCC and CRC Polynomials

Similar to the design criteria for DSO CRCs, we can define a distance-spectrum-optimal TBCC as the set of convolutional code polynomials that minimizes the union bound on the distance spectrum of the TBCC. We once again approximate this optimization by searching for the set of polynomials that maximize the minimum distance and minimize the number of codewords at the minimum distance.

A memory-8 binary convolutional code has $2^7 = 128$ possibilities for each polynomial. Finding the optimal rate-1/12 convolutional code via brute force search requires searching $\binom{128}{12} \approx 2.4 \times 10^{16}$ polynomial combinations. Even after eliminating equivalent polynomial combinations, the space of possible polynomial combinations is still far too large to exhaustively search through. Thus, a non-exhaustive search is performed. The search was done by examining the initial weight spectra of thousands of randomly generated memory-8, rate-1/12 convolutional codes and storing the one(s) with the largest d_{free} and the smallest $A(d_{free})$, $A(d_{free+1})$, $A(d_{free+2})$, and $A(d_{free+3})$. From there, we selected the best polynomial

combination found by this search.

Once the TBCC polynomials were selected, we then searched all $2^{10} = 1024$ possible 11-bit CRCs to find which CRC maximized d_{min} of the CRC-TBCC concatenated code and minimized $A(d_{min})$. We performed this search via an efficient CRC search algorithm for TBCCs described in [14].

3.2.2 Optimal CRC-TBCC Parameters

The best memory-8, rate-1/12, (516,43) TBCC that we found through our non-exhaustive search has generator polynomials {533, 727, 765, 445, 715, 635, 563, 555, 737, 557, 677, 511} in octal. This TBCC has a minimum distance of $d_{free} = 75$, with a total of 86 codewords at weight 75. Table 3.2.2 shows the weight distribution for the first few weights.

Table 3.3: Partial weight distribution of the (516,43) TBCC

d	0	75	76	79	80	84	87	88	91	92
$A(d)$	1	86	86	86	43	129	129	129	215	43

The 11-bit DSO CRC for this TBCC is 0xF69, where the most significant bit corresponds to the x^{11} term of the polynomial. The concatenated CRC-TBCC has a minimum distance of $d_{min} = 132$ and $A(d_{min}) = 37$.

We also must select four bits of the CRC-TBCC to puncture to match the length and rate of the CRC-Polar code. To select the optimal puncture pattern, we searched for the puncture positions that have the smallest effect on the minimum weight codewords of the CRC-TBCC; that is, we searched all weight-132 codewords for the positions that had the most 0's in that position. This way, by puncturing these positions, there would be as small an impact on the minimum weight codewords as possible, thus maximizing the d_{min} of the punctured codes.

Figure 3.3 shows the weight of each bit for the 37 weight-132 codewords of the CRC-

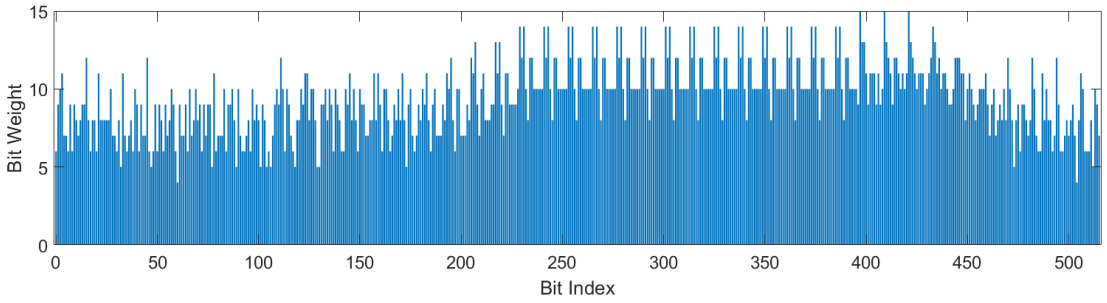

Figure 3.3: Bit weight for each bit index for all weight-132 codewords. These weights vary from 4 to 15 codewords, so it is very important to select low weight indices to puncture.

TBCC; that is, we show the number of weight-132 codewords for which the bit at each bit index is set to 1. This plot shows that there is a large variation in bit weights across different bit indices, so selecting which bits to puncture is very important to minimize the number of codewords we reduce in weight. We find that the bit indices 60 and 504 have the smallest bit weight at four codewords, and there are a total of 13 bit indices with a weight of five codewords. So, we select indices 60 and 504 as two of our punctured bits, and we also select two of the weight-5 indices to fill out the remaining two puncture positions.

Through exhaustive search of weight-5 bit indices, we found that puncturing bit positions {47, 60, 129, 504} results in the largest d_{min} and the smallest $A(d_{min})$. This resulted in a punctured (512,32) CRC-TBCC code with $d_{min} = 130$ and $A(d_{min}) = 1$. Unfortunately, there is no puncture pattern of the weight-4 and weight-5 indices that results in a d_{min} of 131, so a puncture pattern with only a single codeword at 130 is the best result.

3.2.3 Adaptive List Decoding

We use an adaptive list Viterbi decoder for our CRC-TBCCs. The algorithm is conceptually similar to that of the adaptive SCL decoder, but with a parallel LVA decoder [25] instead of a parallel SCL decoder. A critical difference is that the LVA decoder for list size L gathers the L trellis paths that have the L highest likelihoods for each possible end state. The

decoder then combines the trellis paths of each end state into one list and orders them in terms of likelihood, and then selects the most likely codeword that passes both the tail-biting condition and the CRC check. As the list size L tends to ∞, the LVA decoder becomes the maximum likelihood decoder for CRC-TBCCs.

For each beginning/ending state, a parallel LVA decoder always finds the L most likely codewords given the received noisy vector, ranked in order of most to least likely. For example, if the correct codeword is the kth most likely codeword for a given end state and the given received noisy vector, then the codeword will not appear on the list if $L < k$, and it will appear at position k if $L \geq k$. Similarly, if there is an incorrect codeword with the same beginning/ending state as the correct codeword, the incorrect codeword passes the CRC check, and the incorrect codeword is more likely than the correct codeword, then the incorrect codeword will always appear before the correct one on the list.

A common practice for decoding TBCCs is to use the wrap around Viterbi algorithm (WAVA), which tends to improve performance over standard Viterbi decoding of TBCCs [30]. For our CRC-TBCC, we use a WAVA inspired algorithm where we perform a single pass through the trellis to initialize metrics before transitioning to adaptive LVA decoding. Simulation results show that adaptive and nonadaptive LVA decoding with this single pass metric initialization have functionally identical TFR performance.

Results in [3] show that as SNR increases, the average list rank of the decoded codeword converges very quickly to one. For this reason, it makes sense to initialize our adaptive decoder with $L_{min} = 1$. This also implies that the adaptive decoder is significantly faster than the nonadaptive decoder. If the correct codeword is also the most likely, the adaptive decoder will terminate after its first iteration with $L_{min} = 1$, but the nonadaptive decoder will find all L most likely codewords before selecting the correct one. Simulation results in Section 3.4 confirm this fact.

We also include the TFR and throughput of the 24-bit CRC-polar code used in [15] and the previous chapter. Figure 3.9 suggests that under optimal decoding this should

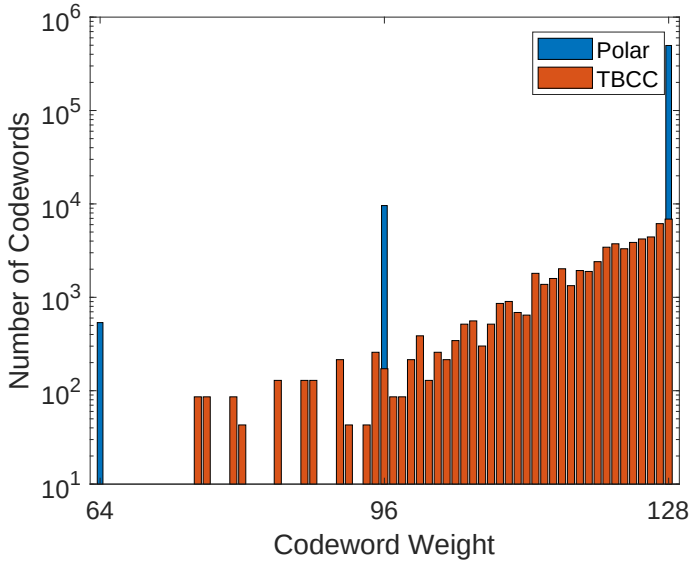

Figure 3.4: Partial weight distribution of (512,43) Polar and (516,43) TBCC. Polar codewords are concentrated at specific weights, while TBCC codewords are more spread out between weights.

outperform the 11-bit CRC-polar, but as discussed in Section V.A, the maximum list size of 1024 is not large enough for this result. Instead, for $L = (32, 1024)$, the 24-bit CRC-polar has similar decoding speed but significantly worse TFR performance compared to the 11-bit DSO CRC-polar.

3.3 Distance Spectra Analysis

In the previous sections, we have presented data of the partial distance spectra of the Polar code, CRC-Polar, TBCC, and punctured CRC-TBCC. In this section, we present an in depth comparison of the distance spectra between these codes.

Since the codes we are working with have a relatively small number of codewords ($2^{32} \approx$ 4.3 billion), we were able to compute the complete distance spectrum of the punctured CRC-TBCC and the CRC-Polar codes. Figure 3.4 shows the partial distance spectra of

the (512, 43) Polar code and the (516, 43) convolutional code up to weight 128, plotted on a log scale. This plot shows the distance spectra of both codes before the 11-bit CRC is applied to expurgate non-CRC-compliant codewords. We can see that these two distance spectra are very different qualitatively. The Polar code has a very sparse distance spectrum, with codewords only appearing at weights that are multiples of 32, with large numbers of codewords concentrated at these distances. By contrast, the TBCC distance spectrum is denser, with codewords appearing at every value of d, but with relatively low multiplicities at each individual distance.

Another property of the TBCC distance spectrum is that, at low weights, the number of codewords that appear at each weight is a multiple of the message length 43. This stems from the fact that if you cyclic shift the message word of a TBCC by any amount, the codeword after convolutional encoding will also be a cyclic shift of the original codeword, so it will also have the same weight. At low weights, codewords consist of a single error pattern surrounded by zeros, which means that low weight codewords are never cyclic. This results in every cyclic shift of a low weight codeword being a unique codeword, and since there are 43 bits in the word before encoding, there are 43 unique cyclic shifts. This breaks down at higher weights, where it is possible for a cyclic shift to produce a duplicate codeword.

Figure 3.5 shows the full distance spectra of the (512, 32) CRC-Polar and the (512, 32) punctured CRC-TBCC codes on a log scale. These codes have a very similar d_{min}, with the CRC-Polar code having $d_{min} = 128$, and the punctured CRC-TBCC having $d_{min} = 130$. Once again, the codewords of the CRC-Polar code are concentrated at discrete weights (multiples of 16), while the CRC-TBCC distance spectrum forms a more continuous shape.

Figure 3.5 also shows the cumulative codeword distance spectra of each code. That is, we plot the number of codewords with weight less than or equal to d as a function of d. We see that the cumulative codeword spectra of the two codes actually hug quite closely together. The CRC-TBCC has fewer cumulative codewords at the smallest weights, and visual inspection seems to show that it tends to have fewer cumulative codewords than the

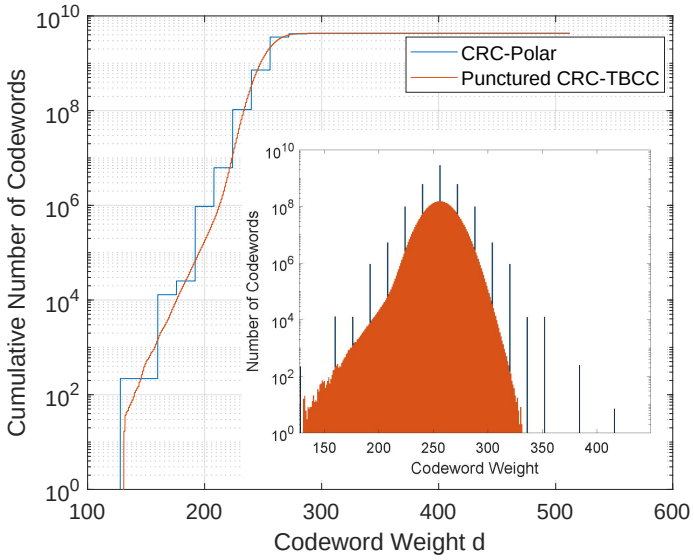

Figure 3.5: Distance spectra (inner plot) and cumulative distance spectra (outer plot) of CRC-Polar and punctured CRC-TBCC.

CRC-Polar at most weights. However, the CRC-TBCC is not strictly better than the CRC-Polar, as there are several points where the CRC-Polar has fewer cumulative codewords.

To better understand the effects of the distance spectra, Figure 3.6 plots the truncated union bound up to weight d vs. max weight d for the punctured CRC-TBCC, the DSO CRC-polar, and the 5G CRC-polar codes. We can observe a number of effects from Figure 3.6.

Firstly, when comparing the CRC-TBCC and the DSO CRC-polar, the CRC-TBCC is the clear winner from the perspective of truncated union bound at both values of E_b/N_0. The DSO CRC-polar code is unable to overcome the large union bound penalty that the 219 codewords at $d = 128$ incurs at the start, and the CRC-TBCC has a strictly better truncated union bound at every maximum distance. The same is true for the comparison between the DSO CRC-polar and the 5G CRC-polar code. The 5G CRC-polar has a much worse d_{min}, and this manifests as a large union bound penalty when compared to the DSO CRC-polar. From this comparison, we can see that replacing the 5G CRC with a DSO CRC for the polar

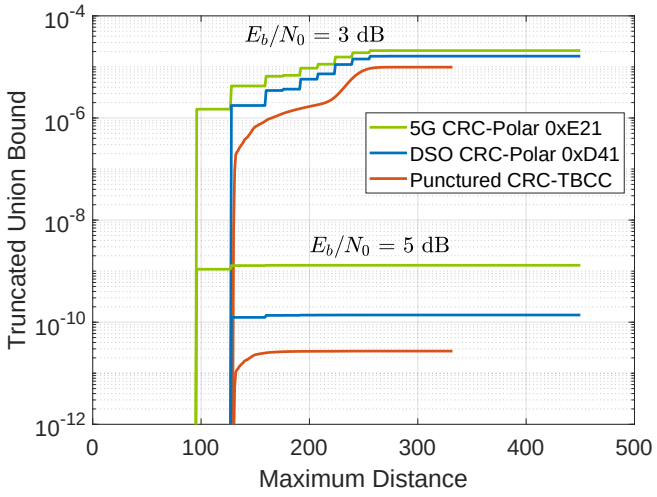

Figure 3.6: Truncated union bound of both codes vs. maximum distance. The punctured CRC-TBCC has a strictly better truncated union bound than the DSO CRC-polar, which itself is better than the 5G CRC-polar code. The CRC-TBCC has $d_{min} = 130$, the DSO CRC-polar has $d_{min} = 128$ and the 5G CRC-polar has $d_{min} = 96$. As E_b/N_0 increases, the codewords at d_{min} dominate the union bound.

code does noticeably improve the union bound.

Secondly, we notice a qualitative difference between the union bound curves for $E_b/N_0 = 3$ dB and $E_b/N_0 = 5$ dB. At $E_b/N_0 = 3$ dB, we can see that the union bound keeps increasing by significant amounts until codeword weights of around 250. This is roughly twice the d_{min} of the CRC-TBCC and DSO CRC-polar codes, and 2.5× the d_{min} of the 5G CRC-polar. For this value of E_b/N_0, while the d_{min} does impact the union bound, it is not the sole contributor to the final union bound value.

In contrast, when $E_b/N_0 = 5$ dB, the codewords at d_{min} play a much more significant role in the value of the union bound. For the CRC-TBCC, codewords stop mattering above a weight of around 150, and both CRC-polar codes have their union bounds determined almost entirely by their codewords at d_{min}.

As a consequence of the union bound becoming dominated by d_{min} as E_b/N_0 increases,

we notice that the gaps between the union bounds of the codes is larger at 5 dB than it is at 3 dB. The CRC-TBCC has a larger d_{min} and a $A(d_{min})$ than the other two codes, so its union bound performs significantly better as E_b/N_0 increases. Figure 3.6 shows that our design algorithm for finding DSO CRCs by maximizing d_{min} and minimizing $A(d_{min})$ becomes a better approximation as E_b/N_0 increases, and it also shows the importance of maximizing d_{min} for codes operating at high E_b/N_0.

3.4 Simulation Results

We now present simulation results for our CRC-TBCC and CRC-Polar designs. We analyze the total failure rate performance of these codes, the trade off between list size and undetected error rate, and the decoding speed of our decoders.

3.4.1 Total Failure Rate

Figure 3.7 shows the TFR performance of the CRC-TBCCs and CRC-Polar codes with various list decoding schemes, plotted together with a saddlepoint approximation of the RCU bound [13]. We use maximum list sizes of $L_{max} = 32$ and $L_{max} = 1024$. For the adaptive SCL decoder with $L_{max} = 1024$, we also vary the minimum list size, L_{min}, between 1, 4, and 32. Once again, we use the notation $L = (a, b)$ to refer to an adaptive list decoder with $L_{min} = a$ and $L_{max} = b$.

For $L_{max} = 32$, the CRC-Polar with nonadaptive SCL decoder has a slightly better TFR performance than its adaptive counterpart. Both of these codes perform worse than the CRC-TBCC.

For $L_{max} = 1024$, all codes have comparable TFR performances at low E_b/N_0. For the adaptive CRC-Polar curves, we see that changing L_{min} has a significant impact on TFR performance at high E_b/N_0. When $L_{min} = 1$, there is a significant error floor starting at $E_b/N_0 = 2.5$ dB. This error flooring effect is present in the comparisons in Chapter 2, where

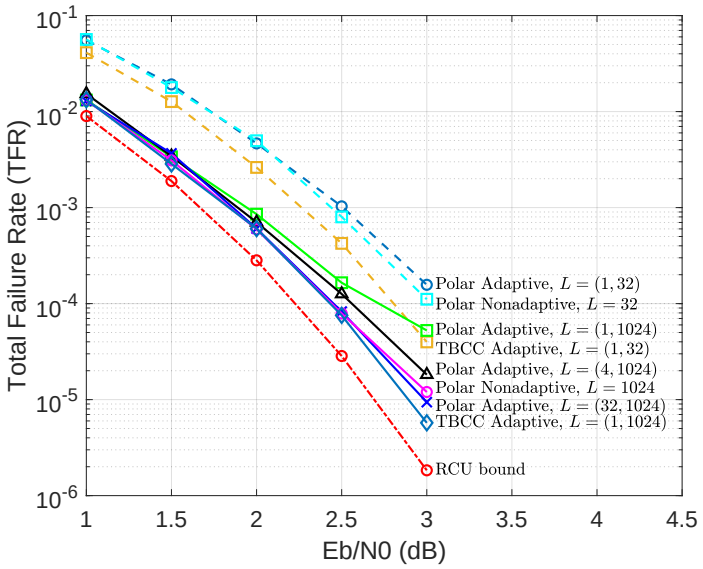

Figure 3.7: TFR vs. E_b/N_0 for all CRC-Polar and CRC-TBCC codes. The notation $L = (a, b)$ refers to an adaptive list decoder with $L_{min} = a$ and $L_{max} = b$. The CRC-TBCC has the best performance and is within 0.2 dB of the RCU bound.

an adaptive SCL decoder with $L_{min} = 1$ was also used. A SC decoder with list size 1 and 11-bit CRC turns out to have a high undetected error rate at high E_b/N_0, which explains this error floor.

As we increase L_{min}, we see that the performance of the adaptive CRC-Polar improves until it is about equal to the performance of the nonadaptive CRC-Polar when $L_{min} = 32$. Due to the extremely slow speed of the nonadaptive SCL decoder with $L = 1024$ (shown later) and the low target TFR, the 3 dB point of the nonadaptive CRC-Polar curve was computed with a relatively small sample size of error events (100). As such, this data point is not completely reliable, which explains why the adaptive CRC-Polar with $L_{min} = 32$ is slightly better.

Once again, the best performing code for $L_{max} = 1024$ is the CRC-TBCC, outperforming the best CRC-Polar by nearly a factor of two at $E_b/N_0 = 3$ dB. The CRC-TBCC TFR curve is about 0.2 dB away from the RCU bound, which matches the gap to RCU bound of the

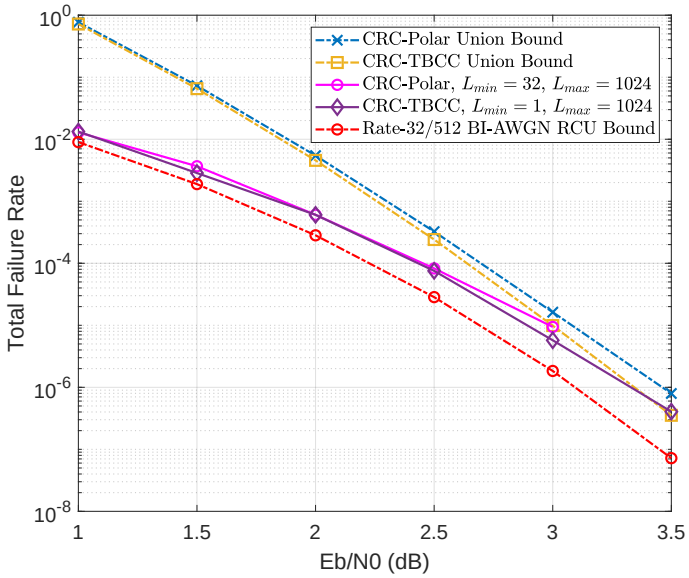

Figure 3.8: TFR curves of CRC-Polar and CRC-TBCC and Union Bound curves. The CRC-TBCC union bound separates from the CRC-Polar union bound at high E_b/N_0, performing a little better. This effect is also seen in the TFR curves of the codes as they converge on union bound.

CRC-TBCC in the previous chapter.

In the previous Section 3.3, we found the complete distance spectrum of the CRC-Polar and CRC-TBCC codes. With this information, we can plot the union bound of these codes against the simulation results. Figure 3.8 shows the union bound and the TFR curves for the best performing CRC-Polar and CRC-TBCC with $L_{max} = 1024$. We can see that at high E_b/N_0, both codes hug very closely to the union bound. Also, the CRC-TBCC has a slightly better union bound curve at high E_b/N_0, following the results from Figure 3.6. We expect the TFR curves to converge on the union bound as E_b/N_0 increases, meaning the CRC-TBCC will continue to outperform the CRC-Polar.

Given how tightly our CRC-TBCC and CRC-polar codes hug the union bounds, we can get an idea for how well a code can do by looking at the union bound. We decide to plot the union bounds of CRC-polar codes with every CRC from the 5G standard in Table I.

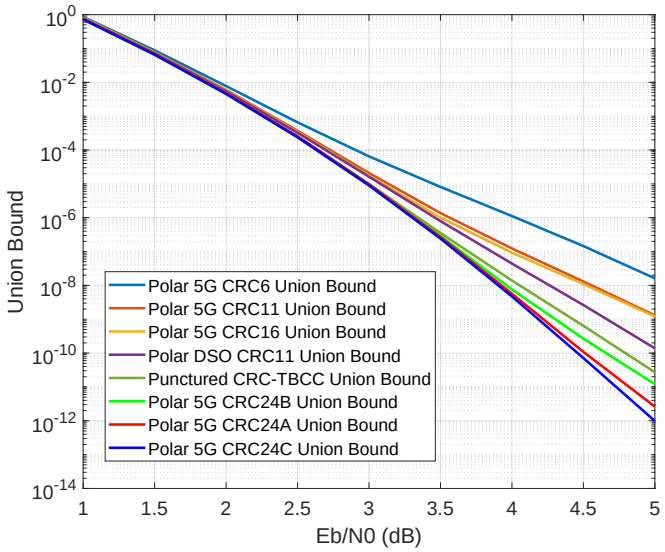

Figure 3.9: Union Bound curves for all 5G CRC-polar codes, as well as the 11-bit DSO CRC-polar and the CRC-TBCC. Union bound tends to improve as CRC length improves, but the DSO CRC-polar and CRC-TBCC union bounds perform significantly better than the 5G CRC11 polar code, showing the importance of designing DSO CRCs.

These union bounds are shown in Figure 3.9, along with the union bounds of the punctured CRC-TBCC and our 11-bit DSO CRC-polar code.

Firstly, we can see that at high SNR, larger CRCs result in better union bounds, with the CRC6 polar code performing the worst, and the CRC24C polar code performing the best. However, this conflicts with the simulation results in [15], which shows the the CRC24C polar code performs significantly worse than a CRC-polar with 11-bit CRC.

In practice, the problem with a CRC-polar code with a 24-bit CRC is the list size necessary to achieve good TFR performance. In all of our simulations of CRC-polar codes with a 24-bit CRC, up to a maximum list size of 1024, we have never found an undetected error. In other words, every single failure found has been a result of the maximum list size being too small. Our 11-bit DSO CRC-polar, in contrast, has a large proportion of its failures as undetected errors, upwards of 80% for high E_b/N_0. As such, its TFR curve converges closely

to its union bound curve.

From these results, in order for a code to achieve performance comparable to or better than the union bound, we expect that the maximum list size must be sufficiently large enough that most failures are undetected. This intuitively makes sense, as the calculation of the union bound assumes all failures are undetected. We conjecture that for a code with an m-bit CRC, a maximum list size of around 2^m is necessary for this union bound matching performance, motivated by simulation results in this paper and [15], and by results in [3] showing the expected list rank of CRC-convolutional codes converges to 2^m at low SNR.

Figure 3.9 also demonstrate the importance of designing DSO CRCs. We can see that the DSO CRC-polar union bound outperforms the 5G CRC11 union bound by roughly a decade at $E_b/N_0 = 5$ dB, and even outperforms the union bound for CRC16. In addition, the union bound for the CRC-TBCC performs nearly as well as one of the 24-bit CRC-polar codes. This demonstrates that DSO CRC design becomes very important in the high SNR/low TFR region.

3.4.2 Undetected Error Rate

So far we have focused on the TFR performance of these codes and how different list sizes affects the TFR. We will now explore the undetected error rate of these codes and its relation to TFR.

Figure 3.10 plots the TFR, undetected error rate, and erasure rate of the $L_{min} = 1$ CRC-TBCC and the $L_{min} = 32$ CRC-Polar codes at $E_b/N_0 = 2.5$ dB against maximum list size. When L_{max} is small, almost all errors are erasures, implying that L_{max} is not large enough for optimal performance. We confirm this when we increase L_{max} and see that TFR performance improves significantly. At large L_{max}, almost all errors are undetected, but even at $L_{max} = 1024$ that data imply that we could still improve TFR further by increasing L_{max}.

These data also suggest that an 11-bit CRC may be too short for applications where

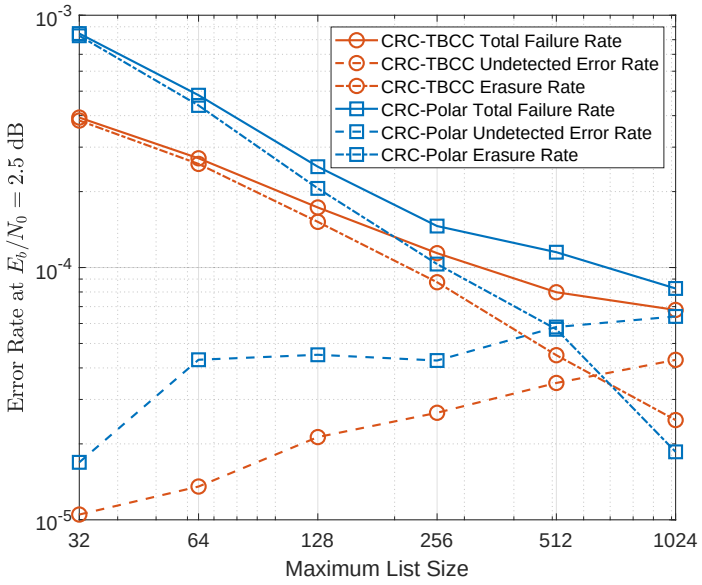

Figure 3.10: TFR, undetected error rate, and erasure rate curves as a function of maximum list size of adaptive decoder at $E_b/N_0 = 2.5$ dB. Blue curves with square markers are $L = (32, L_{max})$ CRC-Polar, and orange curves with circle markers are $L = (1, L_{max})$ CRC-TBCC.

minimizing undetected errors is very important. For these situations, a longer CRC will improve undetected error rate, but at the cost of TFR performance. For example, the 24-bit CRC in the 5G standard has significantly worse TFR performance than an 11-bit CRC [15], but its undetected error rate is substantially lower. Alternatively, concatenating a second CRC will also substantially improve undetected error performance.

3.4.3 Decoding Complexity

Finally, we present the decoding complexity for each decoder. Figure 3.11 shows TFR performance at $E_b/N_0 = 3$ dB plotted against decoding run time[1].

[1]All simulations were performed on a System76 Galaga Pro Ubuntu laptop with an Intel Core i7-8565U CPU @1.8GHz x 8 Processor.

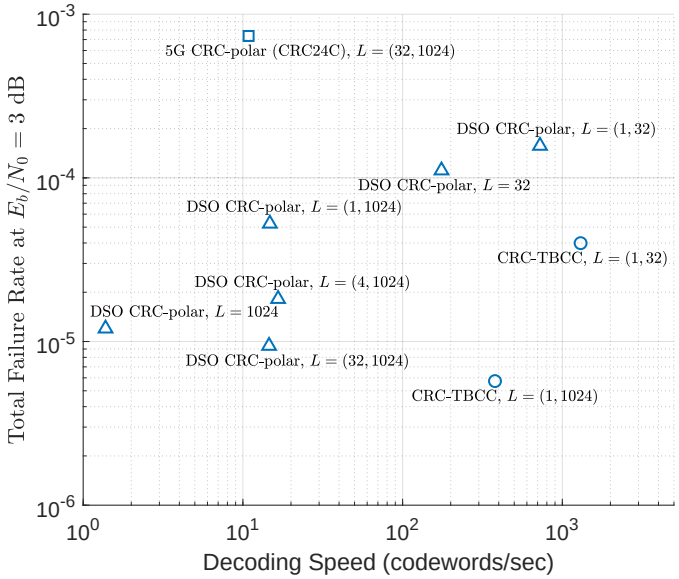

Figure 3.11: TFR at $E_b/N_0 = 3$ dB vs. decoded codewords per second for each decoder. The $L_{max} = 1024$ LVA decoder for CRC-TBCC has the best TFR performance and significantly larger codeword throughput than the SCL decoders for CRC-Polar.

For both $L_{max} = 32$ and $L_{max} = 1024$ decoders, the CRC-TBCC with LVA decoder has both the fastest decoding speed and the best TFR performance. In fact, for $L_{max} = 1024$, the LVA decoder is well over 10 times faster its SCL counterparts.

We also see that for the $L_{max} = 1024$ SCL decoders, increasing the minimum list size has negligible impact on decoding speed, but significant impact on TFR performance. This is because L_{min} is still small enough that the increase in complexity is small. If we were to increase L_{min} further, we would expect decoding complexity to converge toward the non-adaptive SCL decoder.

3.5 Conclusion

In this chapter we presented design methods for optimal low-rate CRC-Polar and CRC-TBCC codes compatible with the 5G PBCH coding standard. We designed these codes to

have equivalent rate, blocklength, and CRC length in order to make the comparison between them as fair as possible. We then did a direct comparison of these two codes, analyzing the properties of their distance spectra and comparing their error rates under simulation. We found that, compared to the CRC-Polar, the CRC-TBCC has a superior distance spectrum, a faster decoder, and better TFR performance.

We use a blocklength of 512 for our codes in this paper, which matches the blocklength of the polar encoder in the 5G PBCH code. However, the 5G PBCH polar code actually has a blocklength of 864, where the first 352 bits of the 512-bit polar code are repeated. We decided to ignore this repetition in this paper since repetition generates no improvement in TFR performance when compared against E_b/N_0 [15]. However, a rate-1/20 TBCC with a 32-bit message and 11-bit CRC has a blocklength of 860 bits, which is very close to the 864-bit blocklength of the PBCH without repetition. A well designed rate-1/20 CRC-TBCC could outperform the rate-1/12 CRC-TBCC we present in this paper and fit even better into the 5G standard. This is an area of future interest.

In 2019, Arıkan presented an improvement on his polar codes, named *polarization-adjusted-convolutional* (PAC) codes [31]. These codes have been shown to perform very well at short blocklengths, outperforming CRC-Polar codes [7]. We are interested in the performance of CRC-aided list decoding of PAC codes at these very low rates.

Part II

Design of CRC-convolutional codes for Noncoherent Q-ary Orthogonal Signaling

CHAPTER 4

CRC-Aided Short Convolutional Codes and RCU Bounds for Orthogonal Signaling

This chapter was first presented at the IEEE Global Communications Conference (GLOBECOM) in December 2022. A written version of this presentation is available in [32].

Phase coherency between transmitter and receiver is necessary for optimal reception. However, phase coherency can be difficult to achieve in certain contexts, so orthogonal signaling with noncoherent reception is often used. The most common examples of orthogonal signal sets are Q-ary Hadamard sequences and Q-ary frequency shift keying (QFSK) [33], the later of which we will focus on for this chapter. Non-coherent FSK signaling is of practical importance. It is currently used in Bluetooth [34]. More recently the LoRa standard has adopted noncoherent QFSK signaling [35] [36].

For values of Q greater than 8, noncoherent QFSK loss is small compared to coherent QFSK. In addition, for large values of Q, noncoherent QFSK performs nearly as well as BPSK signaling, at the expense of bandwidth. With these facts in mind, developing good codes for noncoherent QFSK is very important for contexts in which phase coherency is difficult or impossible. This occurs when there is a high relative velocity between the transmitter and the receiver or when the receiver must be very simple or inexpensive. A natural code choice for QFSK is a code based on a Q-ary alphabet so that code symbols are directly mapped to modulation symbols.

Binary convolutional codes concatenated with binary CRC codes have been shown to

perform very well on BPSK/QPSK channels earlier in this thesis, as well as in [10], [3], [11]. Following [11], we design Q-ary cyclic redundancy check (CRC) codes to be concatenated with optimal, Q-ary, zero-state-terminated convolutional codes (ZTCC). Consistent with the notation throughout this thesis, we denote this concatenated code by CRC-ZTCC.

The Q-ary CRC code design criterion is optimization of the distance spectrum of the concatenation of the CRC code represented by $g(x)$ and the convolutional code represented by $[g_1(x) \;\; g_2(x)]$, where each polynomial has Q-ary coefficients. With all operations over GF(Q), this concatenation is equivalent to a Q-ary convolutional code with polynomials $[g(x)g_1(x) \;\; g(x)g_2(x)]$, which is ostensibly a catastrophic convolutional code. However, rather than applying a Viterbi decoder to this resultant code, we employ the list Viterbi algorithm (LVA) [25]. For this chapter, we focus on 4-ary CRC-ZTCC code designs. Chapter 5 will explore larger values of Q.

Design of codes for noncoherent orthogonal signaling has been done for long messages in [37–40]. Here, we analyze Q-ary CRC-ZTCC codes for short messages. Optimal Q-ary convolutional codes for orthogonal signaling were described by Ryan and Wilson [2]. We design distance-spectrum optimal (DSO) CRCs for two of the codes in [2].

Since the pioneering work of Polyanskiy *et al.* [12], the random coding union (RCU) bound has been used as a measure of the performance quality of short-message binary codes. The RCU bound is very difficult to calculate, but Font-Segura *et al.* [13] derived a saddlepoint approximation for the RCU bound that is more practical to calculate. In this chapter we extend their work in [13] to the noncoherent QFSK channel. We also include in our comparisons the normal approximation [41] for its simplicity.

4.1 System Model

A block diagram of the full system model can be seen in Figure 4.1. We begin with a K symbol Q-ary message. This message is encoded with an m symbol CRC to create

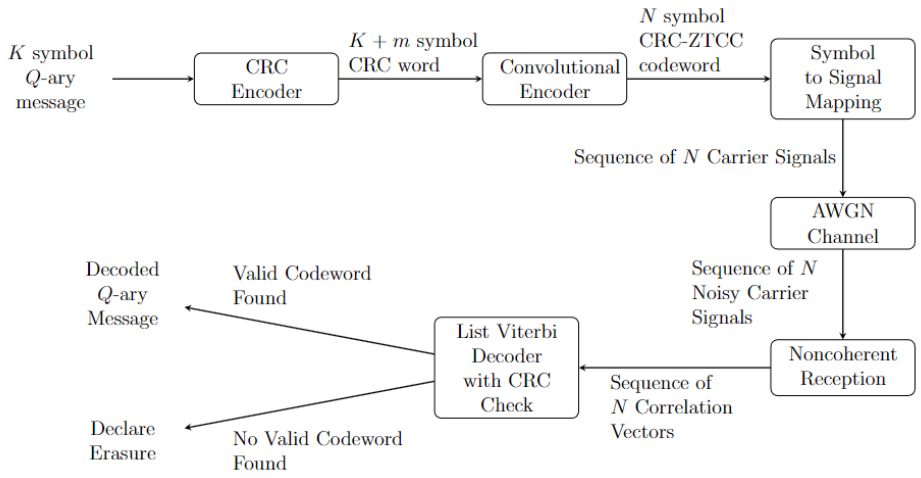

Figure 4.1: Block diagram of full system model.

a $(K+m)$ symbol Q-ary CRC word, and then this CRC word is passed through a zero terminated convolutional encoder to create an N symbol Q-ary CRC-ZTCC codeword. For this encoding, all arithmetic is done in the finite Galois field $GF(Q)$.

At this point, each symbol of the Q-ary CRC-ZTCC codeword is mapped to a corresponding frequency in our QFSK signal set. These signals are sent over an AWGN channel and detected by a noncoherent receiver. Rather than making a hard decision, the detector passes its information as metrics to an adaptive LVA decoder. This adaptive decoding algorithm is the same as the adaptive LVA decoders used in the previous chapters, adapted to work in $GF(Q)$ rather than binary. More details on the noncoherent receiver and LVA decoder are given in the following subsections.

4.1.1 Noncoherent QFSK Signaling

Our discussion here of the noncoherent QFSK channel follows [42]. For a codeword symbol $x \in \{1, 2, ..., Q\}$, the transmitter takes $x = i$ and transmits the corresponding duration-T signal $s_i(t, \phi) = A\cos(\omega_i t + \phi)$, where $A = \sqrt{2E_s/T}$ so that the energy of the signal is E_s, ϕ is uniform over $[0, 2\pi)$, and the frequencies $\omega_i/2\pi$ are separated by a multiple of the symbol

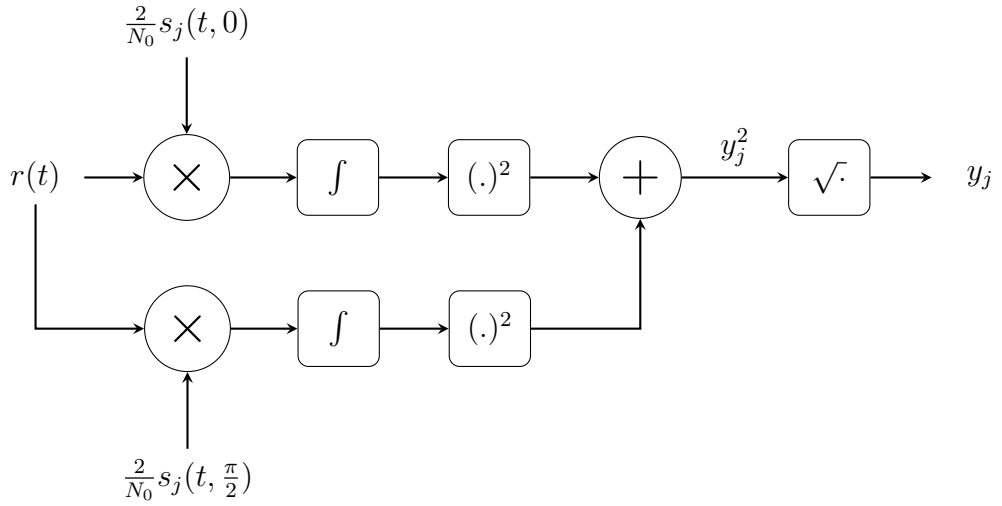

Figure 4.2: A single correlator bank for the noncoherent QFSK receiver

rate to ensure mutual orthogonality among the signals $s_i(t)$. The detector receives the signal $r(t) = s_i(t,\phi) + n(t)$, where $n(t)$ is zero mean AWGN with power spectral density $N_0/2$.

The noncoherent detector consists of Q pairs of correlators, with the jth pair correlating $r(t)$ against $\frac{2}{N_0}s_j(t,0)$ and $\frac{2}{N_0}s_j(t,\frac{\pi}{2})$. Figure 4.2 shows a correlator bank for the carrier frequency s_j [42]. The two correlator outputs are then squared and summed, and a square root is taken of the result. We denote this root-sum-square of the two values by y_j. The vector $\mathbf{y} = [y_1, ..., y_Q]^T$ is the soft decision output of the detector.

In the absence of coding, the magnitudes in $\mathbf{y}$ would be used to make a hard decision on the symbol sent. However, in our model we use CRC-ZTCC codes, so instead these magnitudes are passed as metrics to the LVA decoder.

If $i \neq j$, the correlation of $s_i(t,\phi)$ and $s_j(t,0)$ is 0 due to orthogonality, and the same is true for the correlation of $s_i(t,\phi)$ and $s_j(t,\frac{\pi}{2})$. As such, the value y_j will be the root-sum-square of two zero-mean Gaussian random variables with variance $\sigma^2 = 2E_s/N_0$. Thus, y_j will have a Rayleigh distribution with parameter $\sigma^2 = 2E_s/N_0$. If $i = j$, however, the Gaussian random variables that are root-sum-squared will not be zero mean. As a result, y_j will instead have a Rice distribution with parameters $\mu = 2E_s/N_0$ and $\sigma^2 = 2E_s/N_0$. The

Rayleigh and Rice distributions are as follows:

$$f_{\text{Rayleigh}}(y_j \mid x = i) = \frac{y_j}{\sigma^2} \exp\left[-\frac{y_j^2}{2\sigma^2}\right] \tag{1}$$

$$f_{\text{Rice}}(y_i \mid x = i) = \frac{y_i}{\sigma^2} I_0(y_i) \exp\left[-\frac{y_i^2 + \mu^2}{2\sigma^2}\right] \tag{2}$$

where $I_0(.)$ is the zeroth-order modified Bessel function of the first kind.

Given the message symbol $x = i$, the received vector $\mathbf{y}$ has a density function that is the product of one Rice density function, corresponding to y_i, and $Q - 1$ Rayleigh density functions, corresponding to all y_j for $j \neq i$. This yields the following channel equation for the noncoherent QFSK channel:

$$W(\mathbf{y} \mid x = i) = \frac{\prod_{k=1}^{Q} y_k}{\sigma^{2Q}} I_0(y_i) \exp\left[-\frac{\mu^2 + \sum_{k=1}^{Q} y_k^2}{2\sigma^2}\right] \tag{3}$$

4.1.2 Viterbi Algorithm Decoder and Metrics

For a BI-AWGN channel with BPSK modulation, the LVA decoder finds the trellis paths in the trellis that minimize the sum of euclidean distances between the received noisy BPSK values and the expected noiseless BPSK values for the path. This algorithm is a computationally efficient way of finding the path that maximizes sum of the optimal decoding metrics for the BI-AWGN channel with BPSK modulation.

For the noncoherent QFSK channel, the optimal decoding metrics (i.e. log likelihoods) involve maximizing the logarithm of the Bessel function $I_0(.)$ of the received values $\mathbf{y}$. However, the calculation of $\log I_0(.)$ is very complex and thus impractical when LVA decoding. Figure 4.3 shows the optimal metrics $\log I_0(y_j)$ and several approximations as y_j varies.

Van Trees [43] gives the approximation $\log I_0(y_j) \approx \frac{e^{y_j}}{\sqrt{2\pi y_j}}$, which is very accurate for moderate to large values of y_j, but diverges to infinity for very small values of y_j. In

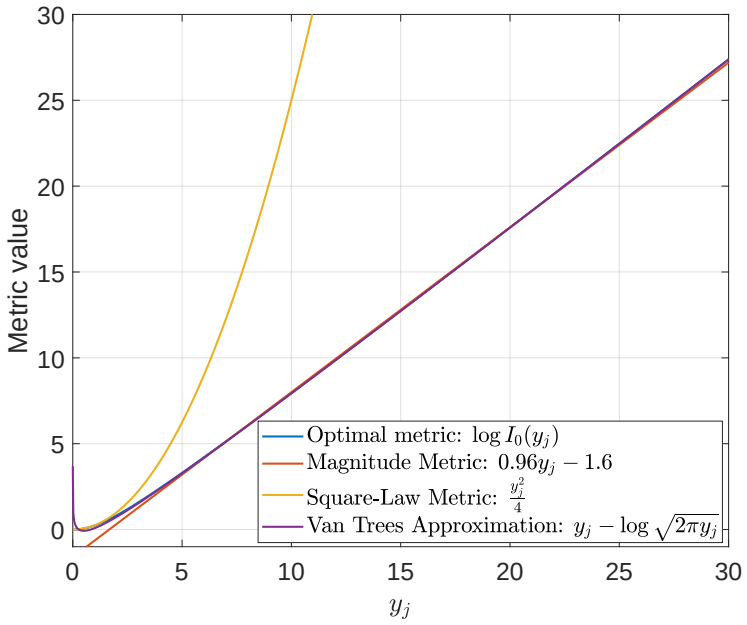

Figure 4.3: Comparison of several approximations for the optimal metrics $\log I_0(y_j)$

practice, the so called "square-law metric" y_j^2 (or $\frac{y_j^2}{4}$, as is in [43]) is often used instead of optimal metrics [2,37,44]. However, the square law metric does a poor job of approximating the optimal metric, and only works well as blocklength tends to infinity.

We see by visual inspection on Figure 4.3 that the optimal metric is roughly linear for most values of y_j, so we can also linearly approximate the optimal metrics by the so called "magnitude metric" $0.96y_j - 1.6$. This magnitude metric has the benefit of being very simple to calculate, and unlike the Van Trees approximation it doesn't explode to infinity at small values of y_j. This magnitude metric is the metric primarily used for simulations in Section 4.4.

4.2 CRC-ZTCC Code Design for QFSK

As in previous chapters, the frame error rate for a length-n block code with minimum distance d_{min} on the QFSK/AWGN channel is union upper bounded [2] as

$$FER < \sum_{d=d_{min}}^{n} A(d)P_2(d) \tag{4}$$

where $A(d)$ is the number of weight-d codewords in the code and $P_2(d)$ is the pairwise error probability for two codewords at distance d. A DSO CRC-ZTCC attempts to minimize this union bound, which can be well approximated for high SNR by maximizing the minimum distance d_{min} and minimizing $A(d_{min})$. In this section, we present the design procedure for the DSO CRC-ZTCCs we use in this chapter.

Ryan and Wilson [2] have found optimal non-binary convolutional codes for small memory. These codes are optimal in the sense of maximizing the free distance d_{free} and minimizing the information symbol weight at each weight $w \geq d_{free}$. We aim to design DSO CRCs for these convolutional codes by maximizing d_{min} and minimizing $A(d_{min})$ for the concatenated CRC-ZTCC block code.

In this section, we adapt the methods in [11] to find DSO CRCs for 4-ary convolutional codes. We consider a memory-2 ($\nu = 2$) and a memory-4 ($\nu = 4$) code from [2]. The convolutional code generator polynomials g_1 and g_2 can be found in Table 4.1 and the optimal CRC polynomials are in Table 4.2, with the x^0 coefficient appearing on the left. These polynomials are elements of $GF(4)[x]$, with $GF(4) = \{0, 1, \alpha, \beta\}$ where α is a primitive element of $GF(4)$ and $\beta = \alpha^2$.

The DSO CRC polynomials for each convolutional code and each CRC-ZTCC are found through an exhaustive search. We begin by initializing a list with every CRC polynomial of degree m and setting a max weight to search to $\tilde{d}$. For every weight from $w = d_{free}$ to $w = \tilde{d}$, we find the number of codewords of Hamming weight w for each CRC-ZTCC concatenated code with polynomials $[g(x)g_1(x) \;\; g(x)g_2(x)]$.

Table 4.1: Generator Polynomials for the Memory-2 and Memory-4 4-ary Convolutional Codes

ν	g_1	g_2	d_{free}	$N_t(d_{free})$	$N_c(d_{free})$
2	(1, 1, 1)	(1, α, 1)	6	6	381
4	(1, 1, 1, β, α)	(1, α, 1, α, β)	9	6	378

Codewords are found by the same process used in [11], adapted for CRC-ZTCC codes in $GF(4)$. This is done by traversing through the trellis of the CRC-ZTCC code for each CRC. We begin in the zero state. For 4-ary CRC-ZTCC codes, each state can transition into four possible new states, one for each element of $GF(4)$. We traverse through the trellis, allowing all possible state transitions, and we maintain a list of every codeword constructed this way. A trellis path is eliminated from contention if the corresponding codeword reaches a weight of $\tilde{d}$ before rejoining the zero state. If a path reaches the end of the trellis in the zero state with a codeword weight $w \leq \tilde{d}$, we increment the count of the number of codewords at weight w for this CRC-ZTCC.

After the distance spectra up to $\tilde{d}$ for every CRC-ZTCC is found, we find which CRC-ZTCC has the largest d_{min}. If multiple CRC-ZTCCs have the same d_{min}, we select whichever CRC-ZTCC has the least number of codewords at d_{min}. If there is a tie for the smallest number of codewords at d_{min}, we compare the number of codewords at $d_{min} + 1$, and we continue incriminating until the tie is broken. Table 4.1 and Table 4.2 show the minimum distances d_{free} and d_{min} for the convolutional codes and CRC-ZTCCs, respectively.

Often, the distance spectrum for a convolutional code is given in terms of of the number of error events at each weight w, as in [45]. This metric only cares about the number of paths on the trellis that diverge from the zero state and eventually rejoin, independent of codeword length. However, in this paper we analyze CRC-ZTCCs as a block code, so the more important metric is the number of codewords of weight w. Table 4.1 and Table 4.2 provide both the number of error events on the trellis at $w = d_{min}$, $N_t(d_{min})$, and the number

Table 4.2: DSO CRC Polynomials for the Memory-2 and Memory-4 4-ary CRC-ZTCCs

ν	m	g	d_{min}	$N_t(d_{min})$	$N_c(d_{min})$
2	3	$(1, \beta, 1, \alpha)$	11	21	1305
2	4	$(1, 0, 0, \beta, \alpha)$	12	18	612
2	5	$(1, 0, 0, \alpha, \beta, 1)$	13	6	273
2	6	$(1, \alpha, \beta, 0, 1, 1, \alpha)$	15	48	2442
2	7	$(1, 0, 1, \beta, \beta, \beta, 0, \alpha)$	16	21	1029
2	8	$(1, \alpha, \alpha, \alpha, 1, \alpha, \alpha, \alpha, \beta)$	17	9	345
4	3	$(1, \beta, \alpha, \beta)$	14	30	1839
4	4	$(1, 0, 0, \beta, \beta)$	15	15	921
4	5	$(1, 0, \beta, \beta, 0, 1)$	16	3	174
4	6	$(1, \beta, 1, \alpha, \alpha, 1, \beta)$	18	21	1266
4	7	$(1, 1, 1, \alpha, \beta, \beta, 1, \alpha)$	19	9	561

of codewords for the block code, $N_c(d_{min})$.

$N_t(d_{min})$ and $N_c(d_{min})$ are related, since low weight codewords will always consist of a single error event of the $[g_1(x)g(x) \;\; g_2(x)g(x)]$ concatenated trellis, and the rest of the codeword will remain in the zero state. In that sense, $N_c(d_{min})$ effectively counts the number of possible offsets each trellis error event can take in the codeword. For example, for a CRC-ZTCC block code with a length-70 trellis, a length-10 trellis error event can appear at 60 different offsets in the codeword. As a result, this error event is only counted once for $N_t(.)$, but is counted 60 times for $N_c(.)$.

4.3 RCU Bound Equations

The RCU bound is an achievability bound for codes of a given rate and finite blocklength, first described by Polyanskiy, Poor, and Verdú in 2010 [12]. The RCU bound is defined

in [12] as follows: let n and M be positive integers. Let $P^n(x)$ be a probability distribution for a random coding ensemble for codewords of length n, and let $W^n(y|x)$ be a length-n channel transition probability. The RCU bound for a length-n code with M codewords is given by

$$\mathrm{rcu}(n, M) = \mathbb{E}_{X,Y}\left[\min\left\{1, (M-1)\mathrm{pep}(X, Y)\right\}\right] \tag{5}$$

where $\mathbb{E}_{X,Y}$ is the expectation over X and Y, X is a random variable drawn from $P^n(x)$, Y is a random variable drawn from $W^n(y|X)$,

$$\mathrm{pep}(X, Y) = \mathbb{P}[i(\bar{X}; Y) \geq i(X; Y) \mid X, Y] \tag{6}$$

is the pairwise error probability with $\bar{X}$ drawn from $P^n(x)$, and $i(X;Y)$ is the mutual information density of X and Y.

Calculating the RCU bound using this definition is computationally hard for most practical situations. In 2018, Font-Segura *et. al.* [13] presented a saddlepoint approximation for the RCU bound to reduce computation complexity. In this section, we will present the equations for the saddlepoint approximation of the RCU bound, find expressions for the derivatives of necessary functions, and apply the noncoherent orthogonal signal channel model from Section 4.1 to the equations in [13].

We start with Gallager's E_0-function [46], which is a function of a distribution over the message symbol alphabet $P(x)$ and channel $W(y \,|\, x = i)$. We will write $W(y \,|\, i)$ for $W(y \,|\, x = i)$ for notational simplicity. The Gallager E_0 function is defined as

$$E_0(\rho) = -\log \int \left(\sum_{i=1}^{Q} P(x = i) W(y \,|\, i)^{\frac{1}{1+\rho}} \right)^{1+\rho} \mathrm{d}y \tag{7}$$

where log is the natural logarithm. For the saddlepoint approximation of the RCU bound, we must find the first and second derivatives of $E_0(\rho)$ with respect to ρ. We will assume a

uniform distribution $P(x) = 1/Q$, as this is optimal for symmetric channels as is the case for our channel.

These derivatives are notationally complex due to exponentiation in ρ, so to simplify we define the following functions:

$$f(y, \rho) = \sum_{i=1}^{Q} W(y \,|\, x = i)^{\frac{1}{1+\rho}} \tag{8}$$

$$g(y, \rho) = \frac{\partial}{\partial \rho} \left(f(y, \rho)^{1+\rho} \right) \tag{9}$$

We now find the derivatives of E_0 in terms of f and g. Note that we will use the notation $f'(y, \rho) = \frac{\partial}{\partial \rho} f(y, \rho)$ since all derivatives are with respect to ρ.

We can rewrite $E_0(\rho)$ as

$$E_0(\rho) = (1 + \rho) \log Q - \log \left(\int f(y, \rho)^{1+\rho} \mathrm{d}y \right) \tag{10}$$

This yields the following for the derivatives of $E_0(\rho)$:

$$E_0'(\rho) = \log Q - \frac{\int g(y, \rho) \mathrm{d}y}{\int f(y, \rho)^{1+\rho} \mathrm{d}y} \tag{11}$$

$$E_0''(\rho) = \frac{(\int g(y, \rho) \mathrm{d}y)^2}{(\int f(y, \rho)^{1+\rho} \mathrm{d}y)^2} - \frac{\int g'(y, \rho) \mathrm{d}y}{\int f(y, \rho)^{1+\rho} \mathrm{d}y} \tag{12}$$

The relevant derivatives of f and g are shown in equations (13)-(16).

$$f'(y,\rho) = -\sum_{i=1}^{Q} \frac{W(y\,|\,i)^{\frac{1}{1+\rho}} \log W(y\,|\,i)}{(1+\rho)^2} \tag{13}$$

$$f''(y,\rho) = \sum_{i=i}^{Q} \frac{W(y\,|\,i)^{\frac{1}{1+\rho}} \log W(y\,|\,i)}{(1+\rho)^3} \left(\frac{\log W(y\,|\,i)}{1+\rho} + 2 \right) \tag{14}$$

$$g(y,\rho) = f(y,\rho)^{1+\rho} \left(\log f(y,\rho) + \frac{(1+\rho) f'(y,\rho)}{f(y,\rho)} \right) \tag{15}$$

$$\begin{aligned} g'(y,\rho) = {} & g(y,\rho) \left(\log f(y,\rho) + \frac{(1+\rho) f'(y,\rho)}{f(y,\rho)} \right) \\ & + f(y,\rho)^{1+\rho} \left(\frac{2f'(y,\rho)}{f(y,\rho)} + (1+\rho) \left(\frac{f''(y,\rho)}{f(y,\rho)} - \left(\frac{f'(y,\rho)}{f(y,\rho)} \right)^2 \right) \right) \end{aligned} \tag{16}$$

With these derivatives, we finally give the saddlepoint approximation for the RCU bound as given in [13]. Let $R = \frac{1}{n} \log M$ be the code rate. We define $\hat{\rho}$ to be the unique solution to the equation $E_0'(\hat{\rho}) = R$. We also define the *channel dispersion* $V(\hat{\rho}) = -E_0''(\hat{\rho})$. The saddlepoint approximation of the RCU bound is given by

$$\mathrm{rcu}(n,M) \simeq \tilde{\xi}_n(\hat{\rho}) + \psi_n(\hat{\rho}) e^{-n(E_0(\hat{\rho}) - \hat{\rho}R)} \tag{17}$$

where the functions $\tilde{\xi}(.)$ and $\psi_n(.)$ are given by

$$\tilde{\xi}(\hat{\rho}) = \begin{cases} 1 & \hat{\rho} < 0 \\ 0 & 0 \le \hat{\rho} \le 1 \\ e^{-n(E_0(1,P)-R)} \theta_n(1) & \hat{\rho} > 1 \end{cases} \tag{18}$$

$$\psi_n(\hat{\rho}) = \theta_n(\hat{\rho}) \left(\Psi(\hat{\rho}\sqrt{nV(\hat{\rho})}) + \Psi((1-\hat{\rho})\sqrt{nV(\hat{\rho})} \right) \tag{19}$$

The function $\psi_n(.)$ is given in terms of the functions $\Psi(.)$ and $\theta_n(.)$ which are defined as

$$\Psi(z) = \frac{1}{2} \mathrm{erfc}\left(\frac{|z|}{\sqrt{2}} \right) \exp\left(\frac{z^2}{2} \right) \mathrm{sign}(z) \tag{20}$$

$$\theta_n(\hat{\rho}) \simeq \frac{1}{\sqrt{1+\hat{\rho}}} \left(\frac{1+\hat{\rho}}{\sqrt{2\pi n \bar{\omega}''(\hat{\rho})}} \right)^{\hat{\rho}} \tag{21}$$

where $\bar{\omega}''(\hat{\rho})$ is given as

$$\bar{\omega}''(\hat{\rho}) = \int \mathcal{Q}_{\hat{\rho}}(y) \left[\frac{\partial^2}{\partial \tau^2} \alpha(y,\tau) \Big|_{\tau=\hat{\tau}} \right] \mathrm{d}y \tag{22}$$

and the derivative in $\bar{\omega}''(\hat{\rho})$ is evaluated at $\hat{\tau} = 1/(1+\hat{\rho})$. The closed form expression for $\frac{\partial^2}{\partial \tau^2}\alpha(y,\tau)$ is given by equations (23)-(26).

$$\frac{\partial^2}{\partial \tau^2}\alpha(y,\tau) = \frac{\beta(y,\tau)\frac{\partial^2}{\partial \tau^2}\beta(y,\tau) - \left(\frac{\partial}{\partial \tau}\beta(y,\tau)\right)^2}{\beta(y,\tau)^2} \tag{23}$$

$$\beta(y,\tau) = \frac{1}{M}\sum_{i=1}^{M} W(y|i)^{\tau} \tag{24}$$

$$\frac{\partial}{\partial \tau}\beta(y,\tau) = \frac{1}{M}\sum_{i=1}^{M} W(y|i)^{\tau} \log W(y|i) \tag{25}$$

$$\frac{\partial^2}{\partial \tau^2}\beta(y,\tau) = \frac{1}{M}\sum_{i=1}^{M} W(y|i)^{\tau} \left(\log W(y|i)\right)^2 \tag{26}$$

Finally, the distribution $\mathcal{Q}_{\hat{\rho}}(.)$ is defined as

$$\mathcal{Q}_{\hat{\rho}}(y) = \frac{1}{\mu(\hat{\rho})} \left(\frac{1}{M}\sum_{i=1}^{M} W(y|i)^{\frac{1}{1+\hat{\rho}}} \right)^{1+\hat{\rho}} \tag{27}$$

where $\mu(\hat{\rho})$ is a normalization constant for the distribution $\mathcal{Q}_{\hat{\rho}}(y)$.

The noncoherent QFSK channel has the channel transition probability $W(y \,|\, x)$ given in Section 4.1. Noteworthy about these equations is that the y in the integrals is the vector $\mathbf{y}$ of correlator outputs from Section 4.1. This vector $\mathbf{y}$ consists of a total of Q elements (one for each signal in our signal set), and every y_j of $\mathbf{y}$ exists in the domain $[0, \infty)$. As such, the

integrals with respect to y in the saddlepoint approximation must be evaluated over the full domain of $\mathbf{y}$, $\mathbb{R}_+^Q$, or the space of length-Q real vectors with all positive elements.

While the calculation of the saddlepoint approximation of the RCU bound is less complex than the true calculation of the RCU bound, it is still very complex. Much of this complexity comes from solving $E_0'(\hat{\rho}) = R$, since we must do multi-dimensional numerical integration to solve for $\hat{\rho}$ in the integrand of the equation. The normal approximation [41], by contrast, only relies on simple numerical integration to find the channel capacity of the noncoherent QFSK channel [44] and the channel dispersion V, and is thus much less complex to calculate. We now present the normal approximation for the noncoherent QFSK/AWGN channel, following [41].

Over the ensemble of length-n, rate-R codes, according to the normal approximation the frame-error rate (FER) is approximately given by

$$FER = q\left(\frac{n(C-R) + 0.5\log_2(n) + O(1)}{\sqrt{nV}}\right) \tag{28}$$

where $q(x) = \int_x^\infty e^{-z^2/2}dz/\sqrt{2\pi}$ and C and R are both in units of bits per channel use. The channel capacity C in this expression is given [44] by the expectation of the channel information density, $i(X;Y)$,

$$C = \mathbb{E}\left[i(X;Y)\right] \tag{29}$$

which can be rearranged [44] into

$$C = \log_2(Q) - \mathbb{E}_{y|x=1}\left[\log_2\left(1 + \sum_{i=2}^{Q}\Lambda_i(y)\right)\right] \text{ (bits/use)} \tag{30}$$

where

$$\Lambda_i(y) = \frac{I_0\left(y_i\right)}{I_0\left(y_1\right)} \tag{31}$$

and y_1 is Rician and each y_i is Rayleigh. In the FER expression, V is the channel dispersion given by

$$V = \mathbb{E}\left[\left(\log_2(Q) - \log_2\left(1 + \sum_{i=2}^{Q} \Lambda_i(y)\right) - C\right)^2\right] \tag{32}$$

4.4 Results

4.4.1 TFR for Various CRC Lengths

We simulate CRC-ZTCC codes with the convolutional generator polynomials shown in Table 4.1 and the CRC polynomials in Table 4.2. We used a message length of $K = 64$ 4-ary symbols, i.e. 128 bits, for all CRC-ZTCCs. Our codes are rate-$K/(2 * (K + m + \nu))$, where m symbols are added for the CRC code and ν symbols are added for zero-state termination. Similar to the previous chapters, we plot the TFR of these codes, or the sum of the undetected error rate and the erasure rate.

The decoder we used was an adaptive list Viterbi algorithm (LVA) decoder with a maximum list size of 2048, as in [15]. The adaptive LVA uses the same algorithm as in the previous chapters, employing parallel list decoding with an initial list size of 1 and doubling the list size until either a message candidate is found that passes the CRC check or the maximum list size of 2048 is reached. Unless stated otherwise, all codes are decoded using the magnitude metric as described in Section 4.1.2.

Fig. 4.4 shows the FER vs. E_b/N_0 of the $\nu = 2$ rate-1/2 CRC-ZTCC codes. The values of m vary from $m = 3$ to $m = 8$. We also include the FER curve of the ZTCC without CRC concatenation ($m = 0$). We see that increasing the length of the CRC improves the performance of the CRC-ZTCC code. However, there is still a 1 dB gap between the best FER performance and the RCU bound.

Fig. 4.5 shows FER vs. E_b/N_0 for the $\nu = 4$ rate-1/2 CRC-ZTCC codes. As in Fig.

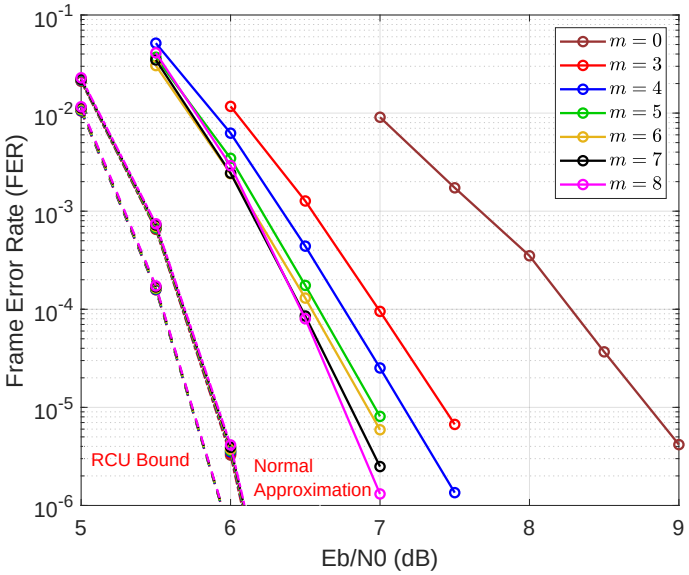

Figure 4.4: FER vs. E_b/N_0 for all $\nu = 2$ CRC-ZTCC codes. The solid lines are the data for codes, and the dashed lines are the corresponding RCU bounds. Each message had a length of $K = 64$ 4-ary symbols, and the decoder had a maximum list size of 2048. The best CRC-ZTCC code has a gap of about 0.9 dB to RCU bound at an FER of 10^{-4}.

4.4, the performance of the CRC-ZTCC improves as m increases. Fig. 4.5 shows $\nu = 4$ CRC-ZTCCs approach the RCU bound more closely than the $\nu = 2$ CRC-ZTCCs, reducing the gap down to 0.59 dB at FER $= 10^{-4}$.

Fig. 4.6 compares every code we have simulated to its respective RCU bound, to visualize the performance of all the codes, plotting the gap between each code's performance and the RCU bound as a function of the FER. Larger values of m achieve smaller gaps to the RCU bound. Generally, the $\nu = 4$ CRC-ZTCCs have smaller gaps to the RCU bound than $\nu = 2$ CRC-ZTCCs. The gap to RCU bound increases as the FER decreases.

Our best 4-ary CRC-ZTCC has a gap to RCU bound of around 0.59 dB at FER $= 10^{-4}$. This matches closely with the results in [3] for the analysis of binary CRC-ZTCCs. This motivates the search for optimal CRCs for 4-ary tail biting convolutional codes (CRC-TBCCs), since binary CRC-TBCCs in [3] approach the RCU bound closely. This will be

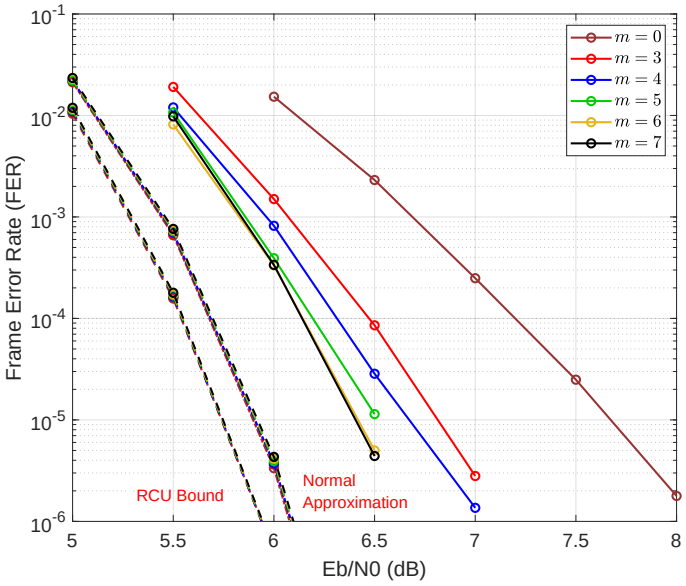

Figure 4.5: FER vs. E_b/N_0 for all $\nu = 4$ CRC-ZTCC codes. The solid lines are the data for codes, and the dashed lines are the corresponding RCU bounds. Each message had a length of $K = 64$ 4-ary symbols, and the decoder had a maximum list size of 2048. The best CRC-ZTCC code has a gap of about 0.59 dB to RCU bound at an FER of 10^{-4}.

explored in Chapter 5.

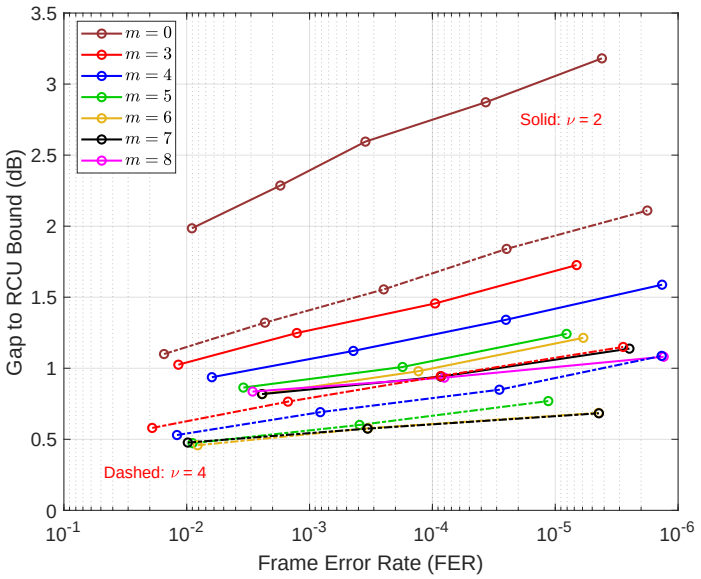

Figure 4.6: Gap to RCU bound vs. FER for every CRC-ZTCC code. Higher values for m have a smaller gap, and the $\nu = 4$ codes have a smaller gap than the $\nu = 2$ codes. The gap also increases as FER decreases.

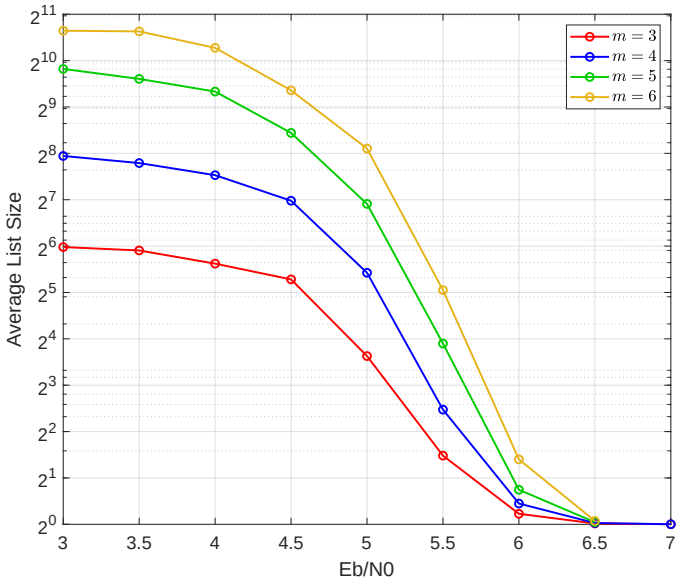

Figure 4.7: Average List Size vs. FER of $\nu = 4$ CRC-ZTCCs.

4.4.2 Expected List Rank

In [3], Yang analyzes the expected list rank when decoding binary CRC-ZTCCs. He finds that at very low SNR (or equivalently high FER), the expected list rank roughly converges on 2^m, where m is the length of the CRC. The probability that a randomly selected codeword passes the CRC check for an m-bit CRC is about 2^{-m}, and at very low SNR we expect each codeword to be effectively random, so this result makes sense. For our non-binary case, an m-symbol Q-ary CRC passes random codewords with probability Q^{-m}. So we would expect the average list rank of our LVA decoder to converge to around Q^m at very low SNR.

Also found in [3] is that expected list rank converges to one very quickly as SNR increases, since at high SNR the first codeword on the list is very likely to be the correct codeword. For the Q-ary case, we expect a similarly quick convergence to one.

Figure 4.7 shows simulations for expected list rank for the $v = 4$ 4-ary CRC-ZTCC with CRC lengths $m = \{3, 4, 5, 6\}$. These simulation results match exactly what we would expect from [3]. At $E_b/N_0 < 3.5$ dB, expected list rank is very close to $2^{2m} = 4^m$, and at

$E_b/N_0 > 6.5$ dB, the expected list rank is effectively one. For the $m = 6$, curve, we would expect the average list rank to converge to $4^6 = 2^{12}$, but we are limited by our maximum list size of $2048 = 2^{11}$.

Our CRC-ZTCCs achieve a target FER of 10^{-4} at E_b/N_0 between 6 and 6.5 dB according to Figure 4.5, depending on CRC length. Looking to Figure 4.7, we see that all of our CRC-ZTCCs have an expected list rank less than four in this range. So, even at moderate FER, the vast majority of codewords selected by the LVA decoder are very small. The adaptive LVA decoding algorithm takes advantage of this fact. A parallel list decoder with $L = 2048$ will always generate a list of 2048 codewords, even though most of the time the codeword it selects is within the first 10. The adaptive LVA decoder starts with small list sizes, and it only has to go up to large list sizes very rarely.

4.4.3 Decoding Metrics

In Section 4.1.2, we discussed the various decoding metrics that can be used to approximate the optimal log likelihood metrics for this channel. Figure 4.8 plots the $v = 4, m = 6$ CRC-ZTCC with the square-law, van trees approximation, and magnitude metrics. We see that the magnitude metric performs the best of these three, which we can attribute to the magnitude metric best approximating the optimal metrics over its entire domain.

We can approximate the optimal metrics even better by using a piecewise approximation. Van Trees [43] shows that $\log I_0(x)$ is well approximated by the square-law metric $\frac{x^2}{4}$ for small x and by $x - \log\sqrt{2\pi x}$ for large x. Through visual inspection, we choose $x = 1$ as the cutoff between these two metrics. Figure 4.8 also includes this piecewise metric, which marginally improves on the magnitude metric. This implies that decoding with the optimal metric does offer an improvement over the magnitude metric, but only a small one. Given the computational simplicity of the magnitude metric, it appears to be a good approximation for practical use.

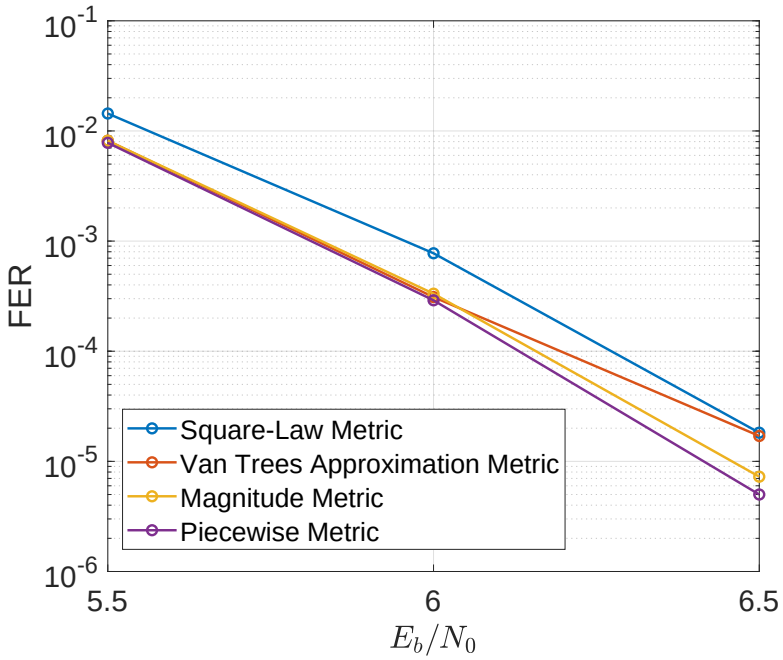

Figure 4.8: Performance of the $v = 4, m = 6$ CRC-ZTCC with various decoding metrics.

4.5 Conclusion

This chapter presents CRC-ZTCC concatenated codes for Q-ary orthogonal signaling, designing CRCs for specific ZTCCs to optimize the distance spectrum to achieve the best possible FER performance. To compare with our simulations, the paper also derives saddlepoint approximations of the RCU bound and presents the normal approximation for the noncoherent QFSK channel.

List decoding using a distance-spectrum optimal CRC significantly improves the minimum distance and the FER performance compared to the ZTCC decoded without the benefit of CRC-aided list decoding. At FER 10^{-4}, CRC-aided list decoding improves the $\nu = 2$ ZTCC by between 1.2 and 1.4 dB and the $\nu = 4$ ZTCC by between 0.7 and 0.8 dB. The performance improvement increases as the size of the CRC is increased. At low FER (or equivalently high SNR) the expected list rank quickly approaches 1, which we take advantage of with an adaptive LVA decoding algorithm so that the average complexity burden of such

list decoding is minimal. Our best CRC-ZTCC design is within 0.59 dB of the RCU bound at an FER of 10^{-4}.

This paper focuses on $Q = 4$ ZTCC-CRC code designs, but TBCC-CRCs have been shown to approach very close to the RCU bound at short blocklengths. In the next chapter, we present design procedures non-binary CRC-TBCCs for this noncoherent orthogonal QFSK channel, and we also explore values of Q larger then 4. List decoding with optimal log likelihood metrics is also an area of future interest.

CHAPTER 5

Design and Performance of CRC-Aided Tail-Biting Convolutional Codes for Orthogonal Signaling

The previous chapter presents design procedures for DSO Q-ary CRCs for Q-ary ZTCCs, and gives results for some specific 4-ary examples. In this chapter, we extend those ideas to design DSO Q-ary CRCs for Q-ary TBCCs. In [3], Yang shows that CRC-TBCCs tend to outperform CRC-ZTCCs on the BI-AWGN channel, even managing to surpass the RCU bound. This motivates the search for good CRC-TBCC codes for the noncoherent Q-ary orthogonal channel.

In [14], Yang *et al.* detail an efficient search algorithm to find DSO CRCs for TBCCs. This algorithm is a generalization of the algorithm in [11] for ZTCCs. We further generalize the algorithm in [14] for Q-ary TBCCs, much like the previous chapter where we generalized [11] to Q-ary ZTCCs. We also design DSO CRCs using this generalized algorithm, and we present simulation results for the performance of our designed Q-ary CRC-TBCC codes.

The previous chapter limits its scope to $Q = 4$ CRC-ZTCCs. In this chapter, we expand the scope to include CRC-TBCCs for $Q = 4$, $Q = 8$, and $Q = 16$. We also discuss the challenges presented by increasing Q.

5.1 System Model

The system model for this chapter is nearly identical to the system model in the previous chapter for CRC-ZTCCs, but with a few minor differences. The convolutional encoder is

now a rate-1/2 tail-biting convolutional encoder rather than a rate-1/2 zero terminated convolutional encoder. This eliminates the overhead symbols incurred by zero-termination, so the overall rate of the code is now $\frac{K}{2(K+m)}$.

We still use an adaptive LVA decoder, but now the decoder is a tail-biting decoder. A common TBCC decoder is the wrap-around Viterbi algorithm (WAVA) decoder [47] [30]. We use the same WAVA inspired adaptive decoding algorithm as in Chapters 2 and 3, where we run a single pass through the trellis to initialize metrics before transitioning to adaptive LVA decoding. At the end of decoding, we check if the codewords found meet the tail-biting condition, and ignore codewords that don't.

All other blocks in the system (CRC encoding, symbol to signal mapping, AWGN channel, and noncoherent reception) remain unchanged from the previous chapter.

5.2 DSO CRC Search

In this section, we will present a high level overview of the algorithm in [14] and discuss the challenges in adapting it to Q-ary codes. A full description of the algorithm is available in [14].

5.2.1 Search Algorithm Overview

The CRC search consists of three main phases: irreducible error event collection, tail biting path construction, and CRC search. Rather than finding all codewords of a TBCC, this algorithm is efficient by only finding the codewords with weight less than $\tilde{d}$, where $\tilde{d}$ is a hyperparameter set before the search begins. We will give high level description of each of these steps.

IEE Collection

An *irreducible error event* (IEE) is defined as a length-l path through the TBCC trellis such that the initial state s_0 and the final state s_l are the same, and every other state s_i is larger than the initial state s_0. Mathematically speaking, a sequence $\mathbf{s} = \{s_0, s_1, ..., s_{l-1}, s_l\}$ is a length-l IEE if $s_0 = s_l$ and $s_i > s_0$ for every $0 < i < l$. We also define the set

$$\text{IEE}(\sigma_i, l, w) = \{\mathbf{s} = \{s_0, ..., s_l\} \mid \mathbf{s} \text{ is an IEE}, s_0 = \sigma_i, \text{weight}(\mathbf{s}) = w\}$$

as the set of all length-l IEEs with initial state σ_i and Hamming weight w.

The IEE Collection algorithm constructs the sets $\text{IEE}(\sigma_i, l, w)$ for all states $0 < \sigma_i < s_{max}$, all lengths l less than the trellis length, and all weights $0 <= w <= \tilde{d}$. Here, s_{max} is the largest state on the trellis, and for a memory-v binary TBCC is defined by $s_{max} = 2^v - 1$.

Starting with state $\sigma_i = 0$, we traverse through the trellis to create a list of all paths starting at state σ_i. If a path ever reaches a weight greater than $\tilde{d}$ or passes through a state $\sigma_j < \sigma_i$, the path is dropped from the list. Once a path returns to state σ_i, the path is added to the set $\text{IEE}(\sigma_i, l, w)$. Once all IEEs have been found for $\sigma_i = 0$, we repeat the collection for $\sigma_i = 1$, and continue until all starting states have been searched. This constructs all IEEs for the TBCC.

TBP Construction

Similar to an IEE, we define a *tail-biting path* (TBP) for a length-L trellis to be a sequence of states $\{s_0, s_1, ..., s_{L-2}, s_{L-1}\}$ such that $s_0 = s_{L-1}$ and all other states $s_i > s_0$ for every $0 < i < L - 1$. In essence, a TBP is an IEE that is the length of the trellis.

Yang shows [14] that any TBP with initial state σ can be constructed by concatenating IEEs with initial state σ. The TBP construction step constructs all TBP with weight less than $\tilde{d}$ via dynamic programming by concatenating IEEs. If a TBP has a weight greater than $\tilde{d}$, it is thrown out. Otherwise, it is stored for the third phase.

CRC Search

Once we have a list of all TBPs for all starting states, [14] shows that every TBCC codeword path is a cyclic shift of one of the TBPs. So, we generate a list of all low weight codewords by storing all cyclic shifts of TBPs found, and we also store the corresponding message word for each codeword. We organize these codewords into lists based on Hamming weight.

Next, we initialize a list of all length-m CRCs. For binary codes, there are a total of 2^{m-1} m-bit CRCs. We aim to find the CRC with the largest d_{min} and the least $A(d_{min})$. To do this, we start by initializing $d = d_{min}$, and we traverse through the list of TBCC codewords with weight d and find the value of $A(d)$ for each CRC on the list. We then remove any CRC from the least whose value of $A(d)$ is greater than the smallest value of $A(d)$. If there are more than one CRC remaining on the list, we increment d by one and repeat until there is only one CRC left. This CRC is the DSO m-bit CRC for the TBCC.

5.2.2 Adapting to Non-binary CRC-TBCCs

This DSO CRC search algorithm is largely agnostic to binary vs. nonbinary CRC-TBCCs, but a few parameters need to be generalized from the binary to the Q-ary case. For a memory-v Q-ary TBCC, the total number of states is given by Q^v. An m-symbol Q-ary CRC consists of $m + 1$ coefficients, of which the first coefficient is always one, and the last coefficient is never zero. Thus, the total number of possible m-symbol Q-ary CRCs is given by $(Q - 1) * Q^{m-1}$. Notice that when $Q = 2$, these two parameters become 2^v and 2^{m-1}, matching the results for the binary case.

As we increase the value of Q, the memory requirements and complexity of the IEE Collection scale exponentially. For a given state Q in the trellis at time t, the number of connections to states in the trellis at time $t + 1$ is equal to Q, since there is one connection for every possible input in the field Q. As a result, when traversing through the trellis, the number of paths grows roughly by a factor of Q for every time step further in the trellis. This

can be a monumental memory burden for even moderate values of Q and $\tilde{d}$, easily requiring many gigabytes of memory to store all the paths during the collection. Luckily, we are able to reduce this complexity and memory usage through some clever optimizations.

First, so long as the trellis length is not too small in relation to $\tilde{d}$, every codeword with weight less than $\tilde{d}$ will mostly consist of remaining in the zero state, with only one or two error events that deviate from the zero state. This results from the fact that any sufficiently long path that stays away from the zero state will always accumulate a nonzero Hamming weight, so the only way to keep the weight low is to stay in the zero state for most of the trellis [1]. Because of this, we can assume that every codeword found by this algorithm will eventually touch the zero state, and as a result we only need to perform a search for the sets IEE$(0, l, w)$; that is, we only need to search for IEEs starting from state zero. This reduces complexity somewhat, but unfortunately a large proportion of IEEs start in state zero anyway, so the benefit is marginal.

We can further reduce the memory requirements by exploiting symmetries in $GF(Q)$, the field in which arithmetic is done for Q-ary CRC-TBCCs. A property of $GF(Q)$ is that the entire field can be generated by the zero element and a single generator element α. That is, we can write $GF(Q) = \{0, \alpha, \alpha^2, ..., \alpha^{Q-2}, \alpha^{Q-1} = 1\}$ [2]. Now, consider the IEE $\mathbf{s} = \{0, s_1, s_2, ..., s_{l-1}, 0\}$. Using the structure of $GF(Q)$, we can rewrite this IEE as $\mathbf{s} = \{0, \alpha^{p_1}, \alpha^{p_2}, ..., \alpha^{p_{l-1}}, 0\}$, where $s_i = \alpha^{p_i}$. We can now divide every element by α^{p_1}, to arrive at a new IEE $\bar{\mathbf{s}} = \{0, 1, \alpha^{p_2-p_1}, ..., \alpha^{p_{l-1}-p_1}, 0\}$. Importantly, the length and weight of $\bar{\mathbf{s}}$ is identical to those of $\mathbf{s}$.

This shows that, for a given IEE with $s_1 \neq 1$, we can find a related IEE with $s_1 = 1$. As a corollary, for an IEE $\mathbf{s} = \{0, 1, s_2, ..., s_{l-1}, 0\}$, we can generate a family of $(Q-1)$ IEEs $\{\mathbf{s}, \alpha\mathbf{s}, \alpha^2\mathbf{s}, ..., \alpha^{Q-2}\mathbf{s}\}$, where $\alpha^p\mathbf{s} = \{0, \alpha^p, \alpha^p * s_2, ..., \alpha^p * s_{l-1}, 0\}$, and every IEE in this

[1] This is true so long as the convolutional code is not *catastrophic*. Catastrophic convolutional codes are generally avoided in practical use, so this restriction is not an issue

[2] This structure can be proven with the fundamental theorem of finitely generated abelian groups

family will have the same length and weight as $\mathbf{s}$. The consequence for reducing memory usage and complexity in the collection phase is that we only have to search for IEEs with $s_1 = 1$. Once the collection phase is finished, we can then generate all of the families of IEEs from the IEEs found during the collection. This reduces memory and complexity by a factor of $(Q-1)$ for this phase.

5.3 Results

With the CRC search algorithm adapted for Q-ary CRC-TBCCs, we now find DSO CRCs for our Q-ary TBCCs and present the results. For this section, we will analyze each value of Q used in its own subsection.

5.3.1 4ary CRC-TBCC

Design and Parameters

We begin with the simplest non-binary case, where $Q = 4$. Given the small value of Q, we are able to design CRC-TBCCs with more memory elements and longer CRCs. We first design a good pair of 4-ary TBCC polynomials to use by finding the pair with the largest d_{min} and the smallest $A(d_{min})$, which is exactly the same criterion we use to design DSO CRCs. We will focus on $v = 4$ TBCCs for this subsection.

For rate-1/2, $v = 4$ 4-ary convolutional codes, there are $Q^v = 256$ possible CRC polynomials we could use, and $\binom{256}{2} = 32640$ possible pairs of polynomials. We search through all of these possible pairs to find which satisfy the DSO criterion. This search is done using the DSO CRC search algorithm presented in the previous section and setting the CRC length $m = 0$. Using the same notation from last chapter, we denote the number weight-d of trellis error events by $N_t(d)$ and the number of weight-d codewords as $N_c(d)$.

From this search, we find a total of 12 TBCC polynomial pair with $d_{free} = 9$ and

$g_1(x)$	$g_2(x)$	$N_t(9)$	$N_t(10)$	$N_t(11)$	$N_t(12)$
$(1,0,1,1,\alpha)$	$(1,\beta,\alpha,\beta,\beta)$	3	27	60	177
$(1,0,1,1,\beta)$	$(1,\alpha,\beta,\alpha,\alpha)$	3	27	60	177
$(1,0,\alpha,1,1)$	$(1,\alpha,\beta,\beta,\alpha)$	3	27	60	177
$(1,0,\alpha,1,\alpha)$	$(1,1,1,\alpha,1)$	3	27	60	177
$(1,0,\beta,1,1)$	$(1,\beta,\alpha,\alpha,\beta)$	3	27	60	177
$(1,0,\beta,1,\beta)$	$(1,1,1,\beta,1)$	3	27	60	177
$(1,1,\alpha,0,1)$	$(1,\alpha,\alpha,1,\beta)$	3	27	60	177
$(1,1,\alpha,1,\beta)$	$(1,\alpha,\alpha,0,\alpha)$	3	27	60	177
$(1,1,\beta,0,1)$	$(1,\beta,\beta,1,\alpha)$	3	27	60	177
$(1,1,\beta,1,\alpha)$	$(1,\beta,\beta,0,\beta)$	3	27	60	177
$(1,\alpha,1,0,\alpha)$	$(1,\beta,1,1,1)$	3	27	60	177
$(1,\alpha,1,1,1)$	$(1,\beta,1,0,\beta)$	3	27	60	177
$(1,1,1,\beta,\alpha)^*$	$(1,\alpha,1,\alpha,\beta)^*$	6	15	75	153

Table 5.1: DSO TBCC polynomials and partial distance spectra, up to $d_{min}+3$. We also show the partial distance spectrum of the polynomial pair from [2] (starred) that is used in the previous chapter.

$N_t(d_{free}) = 3$. [3] Table 5.1 shows all 12 DSO TBCC polynomials and their $N_t(d)$ for the first few weights. Just like Table 4.2, α is the generator of GF(4), and $\beta = \alpha^2$.

Table 5.1 shows that all 12 TBCC polynomials have equivalent distance spectra up to $d = 12$. We can even show an equivalence relation between several of these polynomial pairs. For instance, it can be shown that β is also a generator of GF(4), so replacing α with β and vice versa will create an equivalent TBCC. Thus, the first and second pairs of

[3] We did this search with a message length of 70 symbols. The search returned these polynomials with $N_c(d_{free}) = 210$, which is equal to $3*70$. Since every cyclic shift of the message of a tail-biting codeword also results in a tail-biting codeword (see Section 3.3), this tells us that these polynomials have three weight-9 error events, with each error event corresponding to 70 codewords. In order to be agnostic to codeword length, we list the number of trellis error events rather than number of codewords.

m	$g_{CRC}(x)$	d_{min}	$N_c(d_{min})$
3	$(1, \alpha, \beta, 1)$	12	12
4	$(1, 1, \beta, 1, 1)$	14	204
5	$(1, \alpha, 0, \beta, \beta, \alpha)$	15	30
6	$(1, \beta, \alpha, 1, \alpha, 1, \alpha)$	17	660

Table 5.2: DSO 4-ary CRCs for $m \in \{3, 4, 5, 6\}$.

polynomials are completely equivalent, as an example. We arbitrarily choose the first pair $\{(1, 0, 1, 1, \alpha), (1, \beta, \alpha, \beta, \beta)\}$ as the TBCC polynomial pair we will use for the rest of this subsection.

At the end of Table 5.1, we also show the partial distance spectrum of the polynomial pair from Ryan and Wilson in [2], which is also the polynomial pair we used for the CRC-ZTCC in Chapter 4. We see that this polynomial pair is competitive with the DSO TBCCs, having the same d_{min} but a larger $N_t(d_{min})$.

Now that the TBCC has been designed, we now design DSO CRCs for this TBCC. We use the algorithm described in Section 5.2 to find these CRCs. We use a message length of 64 symbols, so an m-symbol CRC has a $(64 + m)$-symbol CRC word. Table 5.2 shows the DSO CRC polynomials for $m \in \{3, 4, 5, 6\}$. In this case, $m = 6$ was the largest CRC we were able to design before running into significant complexity burdens.

Simulation Results

Now that we have found our DSO CRC-TBCCs, we will present some simulation results for these codes. We begin by comparing our $m = 6$ CRC-TBCC to the $m = 6$ CRC-ZTCC from the previous chapter. Figure 5.1 shows the results of these simulations. We see that the CRC-TBCC performance improves by around $0.2 - 0.25$ dB over the CRC-ZTCC. This also reduces the gap to RCU bound at FER $= 10^{-4}$ to around 0.45 dB. Note that these two

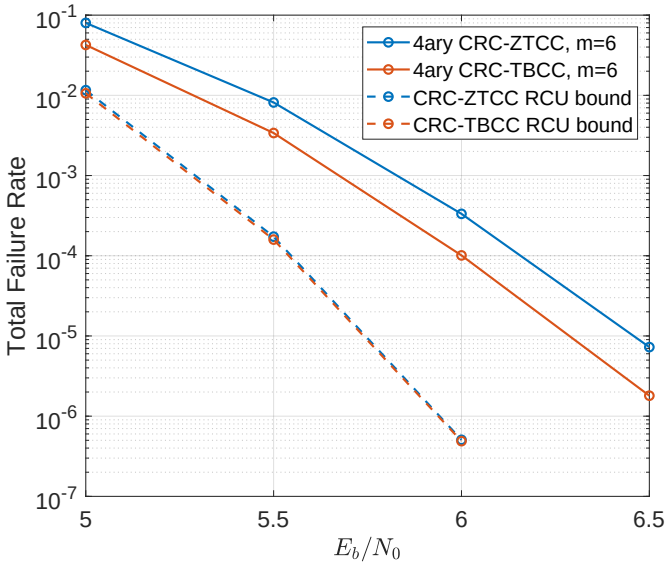

Figure 5.1: TFR curves of 4-ary CRC-ZTCC and CRC-TBCC with $m = 6$. The CRC-ZTCC improves over the performance of the CRC-ZTCC by about 0.2 dB.

codes have slightly different rates (since TBCC doesn't incur the rate penalty from zero-termination), so the RCU bounds for these codes are also slightly different; however, the difference is negligible for this case.

We also plot the CRC-TBCC performance for all m-symbol CRCs in Figure 5.2, as well as the RCU bound and normal approximation[4]. We see that TFR performance improves as m increases, approaching to a gap to RCU bound of around 0.45 dB at TFR $= 10^{-4}$. However, much like the results for CRC-ZTCCs, it seems that increasing the length of the CRC m has diminishing returns, as the improvement from $m = 3$ to $m = 4$ was much larger than the improvement from $m = 5$ to $m = 6$.

Figure 5.2 shows the TFR of the CRC-TBCC codes, ignoring the undetected error rates. Figure 5.3 shows the proportion of errors that are undetected. We can see that up to $m = 5$,

[4]Changing the value of m also changes the rate of the CRC-TBCC code, so each CRC-TBCC actually has slightly different RCU bound and normal approximation curves. This differences in these curves is negligible (see Figure 4.5), so we only plot the RCU bound and normal approximation of the $m = 5$ CRC-TBCC to avoid clutter.

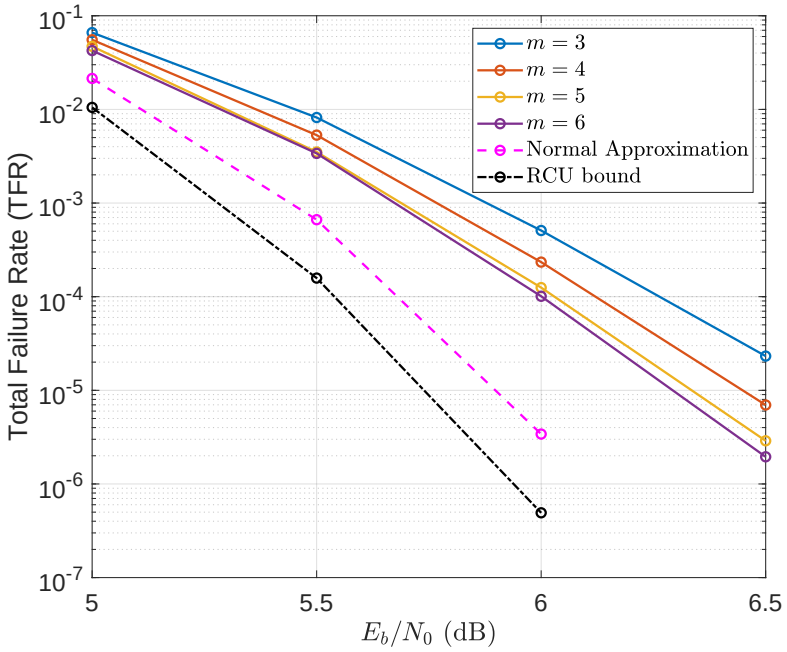

Figure 5.2: TFR curves of 4-ary CRC-TBCCs with $m \in \{3, 4, 5, 6\}$. As we increase m, TFR performance improves. The $m = 6$ CRC-TBCC has a gap to RCU bound of about 0.45 dB at TFR $= 10^{-4}$.

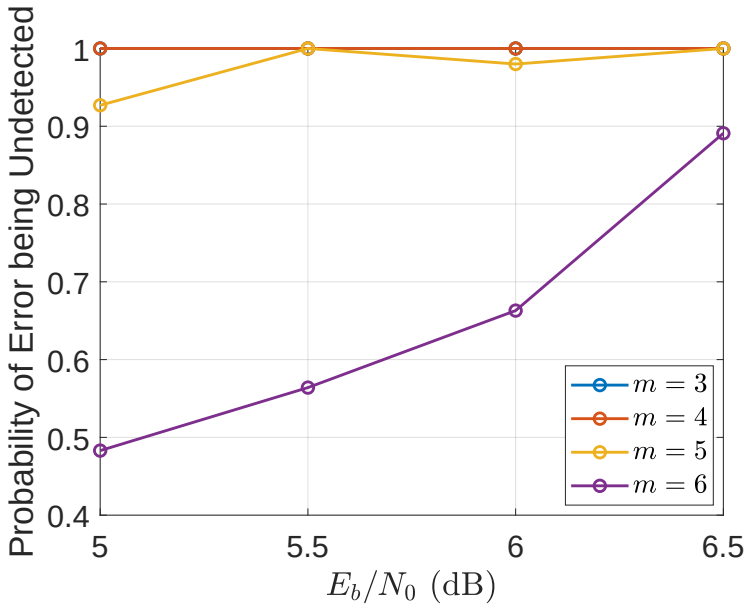

Figure 5.3: Proportion of undetected errors for each CRC-TBCC.

basically 100% of errors are undetected errors. Once we get to $m = 6$, the proportion of undetected errors falls below 100%; however, the proportion is still high, approaching 90% at high E_b/N_0. This graph implies that increasing m can still improve TFR performance, as this would convert the remaining undetected errors to either correct decisions or erasures.

Finally, we also plot the gap to RCU bound for our CRC-TBCCs, just like we did for CRC-ZTCCs in the previous chapter. [Do this stuff later].

5.3.2 8ary CRC-TBCC

Now, we design good 8-ary CRC-TBCC codes. For this subsection, we will borrow the $v = 2$ 8-ary rate-1/2 convolutional code from [2]. Since GF(8) is completely generated by a single generator α (and the zero element), we will give all code parameters in terms of powers of α. The 8-ary TBCC we use has polynomials $\{(1, 1, \alpha^4), (1, \alpha, \alpha^4)\}$. It has a free distance of $d_{free} = 6$ and $N_t(d_{free}) = 7$. We select a memory-2 code for this section since it is easier to design DSO CRCs for convolutional codes with small memory, due to their relatively low weight codewords.

m	$g_{CRC}(x)$	d_{min}	$N_c(d_{min})$
3	$(1, \alpha^6, \alpha^2, \alpha^3)$	11	35
4	$(1, \alpha^4, \alpha^3, \alpha, \alpha^2)$	13	1460
5	$(1, 0, \alpha^3, \alpha^5, \alpha^5, \alpha)$	14	26

Table 5.3: DSO 8-ary CRCs for $m \in \{3, 4, 5\}$.

We design three DSO CRCs for this TBCC, for values of $m \in \{3, 4, 5\}$. We again use a message length of $K = 64$ 8-ary symbols, so the CRC-TBCC code has a CRC word length of $(64 + m)$ symbols. The CRC polynomials are given in Table 5.3.

Figure 5.4 shows simulation results for these 8-ary CRC-TBCCs, as well as the normal approximation. The RCU bound computation becomes prohibitively complex for $Q > 4$, so

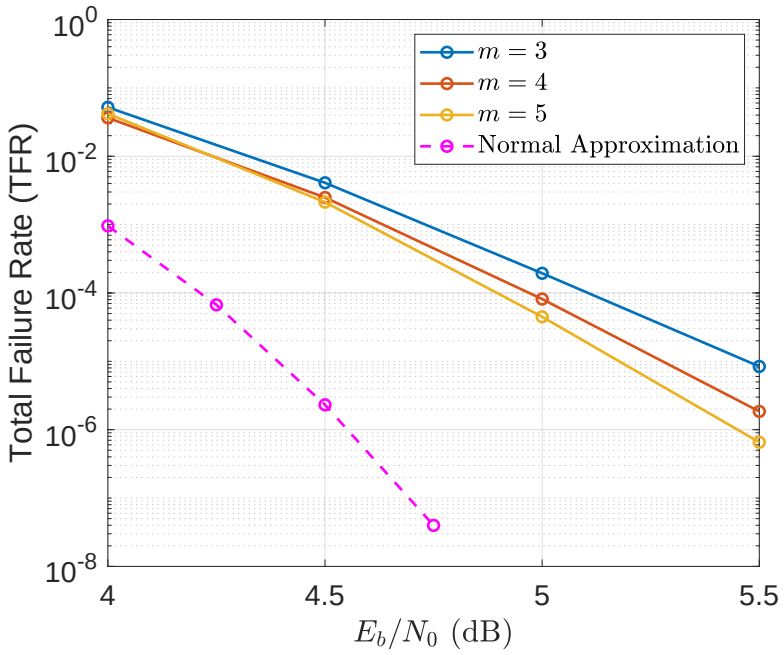

Figure 5.4: TFR curves for 8-ary CRC-TBCCs with $m \in \{3, 4, 5\}$.

we do not compare to the RCU bound. More work is needed to simplify the computation for larger values of Q. We expect the RCU bound to be a few tenths of a dB better than the normal approximation, similar to the $Q = 4$ case.

Just like for our $Q = 4$ codes, as we increase m, the TFR performance of the CRC-TBCCs improves. The $m = 5$ CRC-TBCC has a gap to the normal approximation of around 1.5 dB. This is a large gap, largely due to the fact that a memory-2 convolutional code is not a very powerful code. Chapter 4 shows how improving the memory of a 4-ary ZTCC significantly improves the TFR performance. Designing DSO CRCs for more powerful TBCCs is an area of future interest.

Just like the $Q = 4$ codes, we also plot the undetected error proportion of these CRC-TBCCs, shown in Figure 5.5. Again, we see that as we increase m, the proportion of undetected errors falls, as is expected. In this case, for the $m = 5$ CRC-TBCC, the proportion of undetected errors is below 0.4 across the range of E_b/N_0 we consider, which implies that

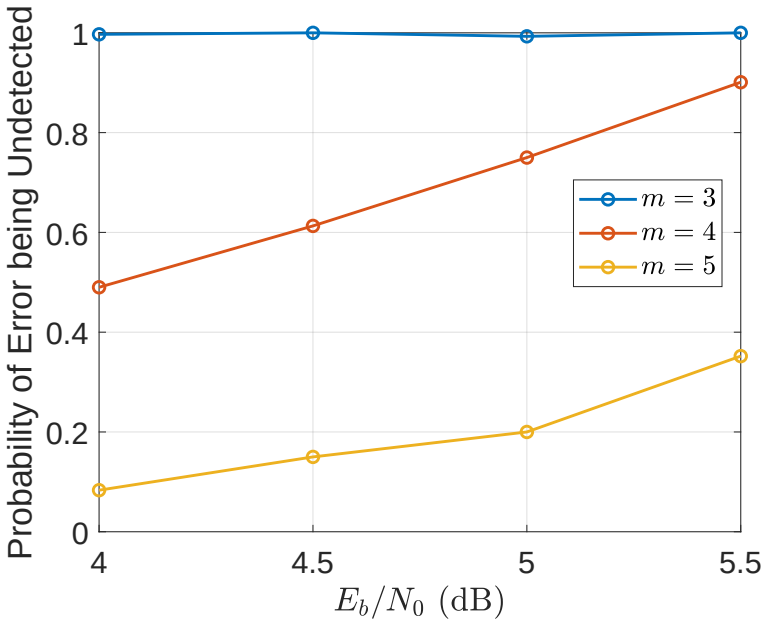

Figure 5.5: Proportion of undetected errors for all 8-ary CRC-TBCCs.

increasing m further will only have a small benefit to TFR performance.

16-ary CRC-TBCC

$g_{CRC}(x)$
$(1, \alpha^4, 1, 0, \alpha^{10}, \alpha^6, \alpha^{12})$
$(1, \alpha^{13}, \alpha^{11}, \alpha^4, \alpha^2, \alpha^8, \alpha^{13})$
$(1, \alpha^{12}, \alpha^{14}, \alpha^{13}, \alpha^8, \alpha^3, \alpha^4)$
$(1, \alpha^{12}, \alpha^{13}, \alpha^9, \alpha^4, \alpha^4, \alpha^7)$

Table 5.4: 16-ary CRCs for $m = 6$.

For 16-ary CRC-TBCCs, we once again borrow a $v = 2$ 16-ary convolutional code from [2]. This just like the 8-ary TBCC, this 16-ary TBCC has polynomials $\{(1, 1, \alpha^4), (1, \alpha, \alpha^4)\}$. Of course, in this case α is the generator of GF(16) rather than GF(8). This code has $d_{free} = 6$

and $N_t(d_{free}) = 15$.

Unfortunately, for 16-ary CRC-TBCCs, the search algorithm for DSO CRCs becomes very computationally complex. Instead, for this section we generate a list of $m = 6$ 16-ary CRCs at random and compare their performances. Table 5.4 has the CRC polynomials for all of the 16-ary CRCs used.

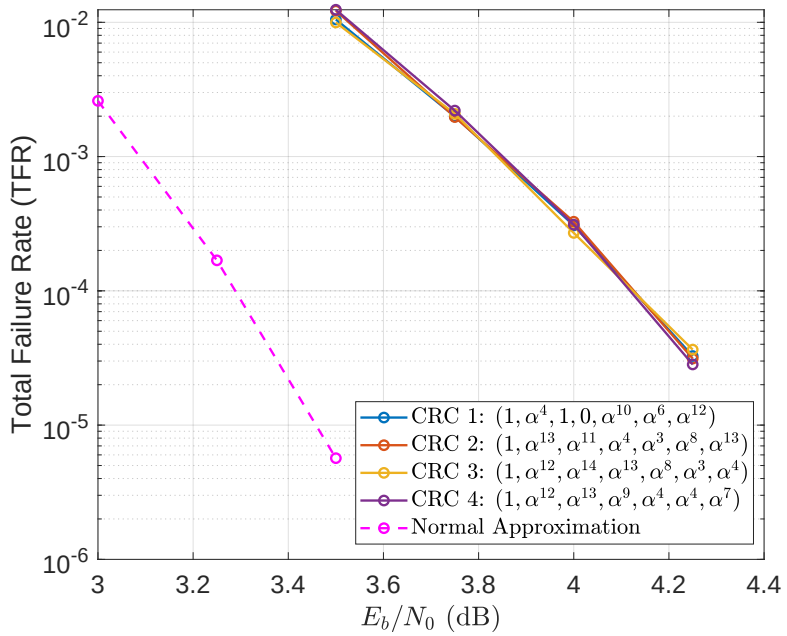

Figure 5.6: TFR curves and normal approximation for 16-ary CRC-TBCCs.

Figure 5.6 shows simulation results for these CRC-TBCCs. We see that all of the CRCs we choose have roughly equivalent TFR performance. This implies that most CRC polynomials will perform about as well as any other. Our curves lie about 0.8 dB away from the normal approximation at TFR $= 10^{-4}$.

As has been shown for binary CRC-TBCCs, we expect that a DSO CRC will outperform a randomly selected CRC. Thus, we could improve our TFR performance by designing and using a DSO CRC for this 16-ary CRC-TBCC. We can also improve TFR performance by using a TBCC with more memory elements, since a memory-2 convolutional code is not very

powerful. These are areas of future interest.

5.4 Conclusion

In this chapter, we presented a generalization of the algorithm in [14] to design DSO CRCs for Q-ary CRC-TBCCs for $Q = 2^q$. We discussed the structure of the GF(Q) fields that these CRC-TBCCs exist on top of. We also presented a way to exploit this structure to reduce the complexity of the search algorithm, reducing the computation and memory needed by a factor $(Q - 1)$.

Using this DSO CRC search algorithm, we designed DSO CRCs for $Q = 4$ and $Q = 8$ CRC-TBCCs, and we presented our simulation results. Our best 4-ary CRC-TBCC improves over our best 4-ary CRC-ZTCC by about 0.2 dB, resulting in a gap to RCU bound of about 0.45 dB and a gap to normal approximation of around 0.3 dB at TFR = 10^{-4}. Our 8-ary CRC-TBCCs are less powerful, resulting to a gap to normal approximation of around 0.6 dB. We could significantly reduce this gap with a more powerful convolutional code, but designing optimal CRCs becomes much harder as we increase the free distance of the convolutional code.

For our maximum list size of 2048, we find that our 4-ary CRC-TBCCs have a very high undetected error rate. In comparison, the 8-ary CRC-TBCCs have a significantly lower undetected error rate. This makes sense intuitively, as an $m = 5$ 4-ary CRC fails to detect one out of every 4^5 codewords, while an $m = 5$ 8-ary CRC passes one out of every 8^5 codewords. This shows that we can expect a 8-ary CRCs to be more powerful than 4-ary CRCs, which is also shown through simulation.

For $Q = 16$ CRC-TBCCs, the runtime needed to find optimal CRCs becomes prohibitively high, so instead we randomly select CRCs to use for our 16-ary CRC-TBCCs. We find that the CRCs we selected all have roughly equivalent TFR performance, with a gap to the normal approximation of around 0.8 dB. We expect an improvement in TFR with

a designed DSO CRC.

While we have shown some ways to optimize the DSO CRC search algorithm, the search is still prohibitively complex for even moderate values of Q and moderate CRC lengths m. We wish to design optimal CRC-TBCCs for up to $Q = 64$ and beyond, but more optimizations to the search algorithm are needed in order to reach this goal. This is an area we are very interested in for the future.

Part III

Conclusion

CHAPTER 6

Conclusion

In this thesis, we have shown design methods and performances of CRC-aided list decoding of zero-terminated convolutional codes, tail-biting convolutional codes, and polar codes. We consider designs for both a standard binary-input AWGN channel and a noncoherent orthogonal Q-ary FSK channel for $Q = 2^q$.

For the BI-AWGN channel, we began by presenting ways to improve the performance of the CRC-Polar code used in the 5G standard. By shortening the 24-bit CRC used in the standard to an 11-bit CRC, we can improve TFR performance significantly, but at the cost of undetected error rate. We can further improve performance by instead using a low rate TBCC with 11-bit DSO CRC, which also has a significantly faster decoder.

In this case of designing for the 5G standard, the CRC-Polar and CRC-TBCC codes we analyzed had different rates. For a fair comparison, we designed 11-bit DSO CRCs for a CRC-Polar and CRC-TBCC code such that both codes had the same message length and block length. By analyzing the qualitative and quantitative differences between the distance spectra of the two codes, we expected the CRC-TBCC to have slightly better performance at high SNR. This expectation was then backed up with simulation results, showing the CRC-TBCC has better TFR performance, better undetected error rate, and a significantly faster decoder.

We then moved on to designing CRC-convolutional codes for the noncoherent orthogonal QFSK channel. We generalized DSO CRC search techniques in [11] and [14] for the case of non-binary alphabets, and designed several CRC-ZTCC and CRC-TBCC codes. We also

modified the RCU bound equations in [13] to apply to our Q-ary orthogonal channel, and presented the difficulties with using this approximation for large values of Q. For 4-ary CRC-ZTCCs, we show that our best designs approach within 0.6 dB of the RCU bound at TFR $= 10^{-4}$. Our 4-ary CRC-TBCC designs improve on the CRC-ZTCC designs by roughly 0.2 dB. Finally, we also present simulation results for $Q = 8$ and $Q = 16$ CRC-TBCCs.

There is still much more work to be done in designing codes for the Q-ary orthogonal channel. We were unable to design DSO CRCs for 16-ary TBCCs in this thesis due to the complexity required in the search, but a more optimized algorithm and more computing power could solve this issue. In addition, we are interested in designing codes up to $Q = 64$ and beyond, so this is another area of future work.

REFERENCES

[1] "ETSI TS 138.212 V15.2.0 LTE; multiplexing and channel coding (3GPP TS 38.212 version 15.2.0 release 15)," July 2018.

[2] W. E. Ryan and S. G. Wilson, "Two classes of convolutional codes over GF(q) for q-ary orthogonal signaling," *IEEE Trans. Commun.*, vol. 39, no. 1, pp. 30 – 40, Jan 1991.

[3] H. Yang, E. Liang, M. Pan, and R. D. Wesel, "CRC-aided list decoding of convolutional codes in the short blocklength regime," vol. 68, no. 6, pp. 3744–3766, Jun 2022.

[4] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[5] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

[6] V. Bioglio, C. Condo, and I. Land, "Design of polar codes in 5G new radio," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 29–40, Firstquarter 2021.

[7] H. Yao, A. Fazeli, and A. Vardy, "List decoding of Arıkan's PAC codes," in *2020 IEEE Int. Sym. on Inf. Theory (ISIT)*, June 2020, pp. 1–6.

[8] P. Elias, "Coding for noisy channels," *Proc. IRE Conv. Rec. part 4*, pp. 37–46, 1955.

[9] H. Ma and J. Wolf, "On tail biting convolutional codes," *IEEE Trans. Commun.*, vol. 34, no. 2, pp. 104–111, February 1986.

[10] E. Liang, H. Yang, D. Divsalar, and R. D. Wesel, "List-decoded tail-biting convolutional codes with distance-spectrum optimal CRCs for 5G," in *2019 IEEE Global Commun. Conf. (GLOBECOM)*, Dec 2019, pp. 1–6.

[11] C.-Y. Lou, B. Daneshrad, and R. D. Wesel, "Convolutional-code-specific CRC code design," *IEEE Trans. Commun.*, vol. 63, no. 10, pp. 3459–3470, October 2015.

[12] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[13] J. Font-Segura, G. Vazquez-Vilar, A. Martinez, A. G. i Fàbregas, and A. Lancho, "Saddlepoint approximations of lower and upper bounds to the error probability in channel coding," in *2018 52nd Annu. Conf. on Inf. Sciences and Syst. (CISS)*, March 2018, pp. 1–6.

[14] H. Yang, L. Wang, V. Lao, and R. D. Wesel, "An efficient algorithm for designing optimal CRCs for tail-biting convolutional codes," in *2020 IEEE Int. Sym. Inf. Theory (ISIT)*, June 2020, pp. 1–6.

[15] J. King, A. Kwon, H. Yang, W. Ryan, and R. D. Wesel, "CRC-aided list decoding of convolutional and polar codes for short messages in 5G," in *2022 IEEE Int. Conf. on Commun. (ICC)*, May 2022, pp. 1–6.

[16] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[17] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *2009 IEEE Int. Sym. on Inf. Theory (ISIT)*, June 2009, pp. 1–5.

[18] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378 – 1380, December 2011.

[19] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes," *IEEE Trans. Commun.*, vol. 61, no. 8, pp. 3100–3107, August 2013.

[20] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *Proc. 48th Asilomar Conf. Signals, Syst. Comput.*, Nov 2014, pp. 1–5.

[21] T. Baicheva and P. Kazakov, "CRC selection for decoding of CRC-polar concatenated codes," in *Proceedings of the 9th Balkan Conference on Informatics*, Sep 2019, pp. 1–5.

[22] T. Baicheva, P. Kazakov, and M. Dimitrov, "Some comments about CRC selection for the 5G nr specification." [Online]. Available: https://arxiv.org/abs/2104.02639

[23] R. Wesel, "Convolutional codes," *Wiley Encyclopedia of Telecommunications*, pp. 1–8, 2003.

[24] R. Schiavone, "Channel coding for massive IoT satellite systems," Master's thesis, Polytechnic University of Turin, 2021.

[25] N. Seshadri and C. E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 313–323, Feb. 1994.

[26] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2044–2047, November 2012.

[27] C. Pillet, V. Bioglio, and C. Condo, "On list decoding of 5g-nr polar codes," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[28] H. Yao, A. Fazeli, and A. Vardy, "A deterministic algorithm for computing the weight distribution of polar codes," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 1218–1223.

[29] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE communications letters*, vol. 16, no. 12, pp. 2044–2047, 2012.

[30] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two decoding algorithms for tail-biting codes," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1658–1665, Oct 2003.

[31] E. Arıkan, "From sequential decoding to channel polarization and back again." [Online]. Available: https://arxiv.org/abs/1908.09594v3

[32] J. King, W. Ryan, and R. D. Wesel, "CRC-aided short convolutional codes and RCU bounds for orthogonal signaling," in *2022 IEEE Global Commun. Conf. (GLOBECOM)*, Dec 2022, pp. 1–6.

[33] J. Proakis, *Digital Communications*. McGraw-Hill, 1995.

[34] J. C. Haartsen, "The bluetooth radio system," *IEEE Personal Communications*, vol. 7, pp. 28–36, Feb 2000.

[35] "AN1200.22 LoRa modulation basics," May 2015.

[36] Z. Li and Y. Chen, "BLE2LoRa: Cross-technology communication from bluetooth to LoRa via chirp emulation," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun 2020, pp. 1–6.

[37] A. G. i Fàbregas and A. J. Grant, "Capacity approaching codes for non-coherent orthogonal modulation," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 4004 – 4013, Nov 2007.

[38] M. Valenti and S. Cheng, "Iterative demodulation and decoding of turbo-coded m-ary noncoherent orthogonal modulation," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 9, pp. 1739 – 1747, Sep 2005.

[39] M. C. Valenti, D. Torrieri, and T. Ferrett, "Noncoherent physical-layer network coding with FSK modulation: Relay receiver design issues," *IEEE Trans. Commun.*, vol. 59, pp. 2595–2604, Sep 2011.

[40] S. Cheng, M. C. Valenti, and D. Torrieri, "Robust iterative noncoherent reception of coded FSK over block fading channels," *IEEE Trans. Wireless Commun.*, vol. 6, pp. 2595–2604, Sep 2007.

[41] M. C. Coskune, G. Durisi, T. Jerkovits, G. Liva, W. Ryan, B. N. Stein, and F. Steiner, "Efficient error-correcting codes in the short blocklength regime," *Physical Communication*, vol. 34, pp. 66–79, Jun 2019.

[42] S. G. Wilson, *Digital Modulation and Coding*. Prentice Hall, 1996.

[43] H. L. Van Trees, *Detection, Estimation, and Modulation Theory (Part I)*. John Wiley Sons, 1968.

[44] W. E. Stark, "Capacity and cutoff rate of noncoherent FSK with nonselective rician fading," *IEEE Trans. Commun.*, vol. 33, no. 11, pp. 1153 – 1159, Nov 1985.

[45] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Pearson, 2004.

[46] R. G. Gallager, *Information Theory and Reliable Communication*. John Wiley & Sons, 1968.

[47] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Decoding of codes based on their tail biting trellises," in *IEEE International Symposium on Information Theory*, Jun 2000, p. 342.