ON POLARIZATION ADJUSTED CONVOLUTIONAL CODES OVER FADING AND ADDITIVE WHITE GAUSSIAN NOISE CHANNELS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

By

Seyed Sadra Seyedmasoumian Charandabi

May 2022

On Polarization Adjusted Convolutional Codes over Fading and Additive White Gaussian Noise Channels
By Seyed Sadra Seyedmasoumian Charandabi
May 2022

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Tolga Mete Duman (Advisor)

Erdal Arıkan

Ayşe Melda Yüksel Turgut

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

ON POLARIZATION ADJUSTED CONVOLUTIONAL CODES OVER FADING AND ADDITIVE WHITE GAUSSIAN NOISE CHANNELS

Seyed Sadra Seyedmasoumian Charandabi
M.S. in Electrical and Electronics Engineering
Advisor: Tolga Mete Duman
May 2022

Ultra-reliable and low-latency communications (URLLC), which focuses on delay sensitive applications and services, is one of the three main pillars of 5G New Radio (NR) network architecture. URLLC's physical layer design is challenging since it must meet two contradictory requirements: ultra-low latency and ultra-high reliability. Short packets are used to minimize latency but at the cost of a significant loss of coding gain. Alternatively, system bandwidth can be increased, which is not always practical, particularly for some URLLC applications in industrial control that use unlicensed spectrum. In order to improve reliability, we must utilize robust channel codes in conjunction with retransmission techniques. Therefore, the construction of block codes with short blocklengths (e.g., a thousand or less information bits) is receiving significant attention with emerging wireless communications applications.

In this thesis, we review existing channel coding bounds with short blocklengths for both additive white Gaussian noise (AWGN) and block fading channels. Furthermore, we investigate the performances of tail-biting convolutional, polar, and polarization adjusted convolutional (PAC) codes. With the motivation of reducing the decoding complexity of PAC decoders, we implement an alternative sequential decoding algorithm, namely, creeper algorithm, and describe a simplified list decoding approach. We also conduct an investigation on the performance of PAC codes and channel coding limits for block fading channels. Furthermore, we derive a method for computing approximate weight distribution of PAC codes, which can be used for an accurate performance bound; and, employing this approximation, we design PAC codes utilizing simulated annealing for optimization of the rate profiles. The results show that the newly designed PAC code rate profiles offer superior performance.

Keywords: Short blocklength channel codes, polarization-adjusted convolutional codes, weight distribution, union bound, tail biting convolutional codes, polar codes, rate profiles.

ÖZET

SÖNÜMLÜ VE EKLEMELİ BEYAZ GAUSS GÜRÜLTÜ KANALLARI ÜZERİNDE POLARİZASYON AYARLI EVRİŞİMLİ KODLAR

Seyed Sadra Seyedmasoumian Charandabi
Elektrik Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Tolga Mete Duman
Mayıs 2022

Gecikmeye hassas uygulamalar ve hizmetlere odaklanan ultra güvenilir ve düşük gecikmeli iletişim (URLLC), 5G Yeni ağ mimarisinin üç ana ayağından biridir. URLLC'nin fiziksel katman tasarımı, birbiriyle çelişen iki gereksinimi karşılaması gerektiğinden zorludur: ultra düşük gecikme süresi ve ultra yüksek güvenilirlik. Kısa paketler gecikmeyi en aza indirmek için kullanılır, ancak bunun pahasına kodlama kazancında önemli bir kayıp olur. Alternatif olarak, sistem bant genişliği arttırılabilir, bu her zaman pratik değildir, özellikle endüstriyel kontrolde lisanssız spektrum kullanan bazı URLLC uygulamaları için. Güvenilirliği artırmak için, yeniden iletim teknikleriyle bağlantılı olarak güçlü kanal kodlarını kullanmalıyız. Bu nedenle, kısa blok uzunlukları (örneğin, bin veya daha az bilgi biti) olan blok kodlarının oluşturulması, yeni geliştirilen kablosuz iletişim uygulamaları ile büyük ilgi görmektedir.

Bu tezde, hem eklemeli beyaz Gauss gürültüsü (AWGN) hem de blok sönümlü kanallar için kısa blok uzunluklu mevcut kanal kodlama sınırlarını gözden geçiriyoruz. Ayrıca, kuyruk ısıran evrişimli, kutupsal ve polarizasyon ayarlı evrişimli (PAC) kodların performanslarını araştırıyoruz. PAC kod çözücülerinin kod çözme gecikmesini azaltma motivasyonuyla, alternatif bir sıralı kod çözme algoritması, yani Creeper algoritması uyguluyoruz ve basitleştirilmiş bir liste kod çözme yaklaşımı tanımlıyoruz. Ayrıca, PAC kodların blok sönümlü kanallardaki performanslarını ve kanal kodlama limitlerini inceliyoruz. Ayrıca, doğru bir performans sınırı için kullanılabilecek PAC kodlarının yaklaşık ağırlık dağılımını hesaplamak için bir yöntem türetiyoruz; ve bu yaklaşımı kullanarak, oran profillerinin optimizasyonu için simüle edilmiş tavlama algoritmasını kullanarak PAC kodları tasarlıyoruz. Sonuçlar, yeni tasarlanmış PAC kod oranı profillerinin üstün performans sunduğunu göstermektedir.

Anahtar sözcükler: Kısa blok uzunluklu kanal kodları, polarizasyon ayarlı evrişimli kodlar, ağırlık dağılımı, birliğe bağlı, kuyruk ısırma evrişimli kodlar, kutupsal kodlar, oran profilleri.

Acknowledgement

First and foremost, I would like to convey my heartfelt thanks to Prof. Tolga M. Duman, my M.S. adviser, for his unwavering support, vast knowledge, encouragement, and patience throughout my studies. I would like to express my gratitude to him for his contributions to and encouragement of my research through thoughtful discussions and ideas. It would not have been feasible to do this research without his invaluable assistance, and I consider myself quite fortunate because he is an outstanding advisor and supporter to me.

I would also like to thank my examining committee members Prof. Arıkan and Prof. Yüksel, for their helpful and insightful comments.

This thesis was funded in part by the Scientific and Technical Research Council of Turkey (TUBITAK) under grant 119E589, and in part by ASELSAN Inc. under the "5G Platform" project, which is partly funded by TUBITAK under the project number 1160206. I gratefully acknowledge these supports.

I would like to thank all of the CTAR members, Mert Ozateş, Büşra Tegin, Talha Akyıldız, Mahdi Shakiba Herfeh, Berke Eren, Mohammad Javad Ahmadi, Ozan Aygün, Dr. Javad Haghighat, Atif Ali, Mucahit Gümüş and Uras Kargı for a very productive and charming atmosphere. My special thank goes to Dr. Mohammad Kazemi for helping me in writing this thesis with his useful comments. Also, I would like to thank Mohsen Moradi and Amir Mozammel for informative conversations about polar codes.

Last but not least, I would like to express my gratitude to my beloved family for their precious support.

Contents

List of Figures

List of Tables

Abbreviations

3GPP 3rd Generation Partnership Project

AWGN Additive white Gaussian noise

B-DMC Binary-input discrete memoryless channel

BEC Binary erasure channel

BI-AWGN Binary-input additive white Gaussian noise

BICM Bit-interleaved coded modulation

BLER Block error rate

BPSK Binary Phase shift keying

CSI Channel state information

CSIR Channel state information at the receiver

DEGA Density evolution with Gaussian approximation

eMBB Enhanced Mobile Broadband

FA False alarm

FER Frame error rate

HWD Hamming weight distribution

i.i.d. Independent and identically distributed

LDPC Low-density parity-check code

LLR Log-likelihood ratio

MC Meta-converse

MD Missed detection

ML Maximum likelihood

mMTC Massive Machine-Type Communications

NA Normal approximation

NN Neural-network

NR New Radio

PAC Polarization-adjusted convolutional

PDF Probability density function

PPV Polyanskiy-Poor-Verdu

PWD Probabilistic weight distribution

QAM Quadrature amplitude modulation

QPSK Quadrature Phase shift keying

RBF Rayleigh block fading

RCB Random coding bound

RCU Random coding union

RM Reed-Muller

SA Simulated annealing

SC Successive cancellation

SCF Successive cancellation flip

SCL Successive cancellation list

SNR Signal to noise ratio

SSCL Simplified successive cancellation list

TBCC Tail biting convolutional code

URLLC Ultra-reliable low-latency communications

WAVA Wrap-around Viterbi algorithm

WEF Weight enumerating function

Chapter 1

Introduction

The strict requirements of ultra-reliable low-latency communications (URLLC) services are among the most challenging aspects of implementing 5G features over mobile networks. The problem is even more demanding when we need services beyond the 5G promise [1]. URLLC requires low block error rates (BLERs) yet with less overall network latency and less complexity, which are conflicting requirements. In order to address this issue, we need to utilize short blocklength codes (in the range of 50 to 1000 bits [2]) to reduce latency; however, the coding scheme should be powerful enough to meet the reliability demands. The crucial need for designing finite-length codes has aroused interest to find achievable rates in finite-block length regimes in the recent literature, see, e.g., [3, 4].

In this work, we focus on two promising coding schemes with short blocklengths that have received attention in the literature in recent years. The first approach in this thesis focuses on convolutional codes, specifically, tail-biting convolutional codes. Since zero tail termination imposes extra bits on the codeword that reduce the rate, which becomes highly significant, especially in short block lengths, tail-biting termination has been proposed in the literature to address this problem. The second approach that we focus on is polarization-adjusted convolutional (PAC) codes. PAC codes are designed to improve the performance of polar codes in the short blocklength regime. In this thesis, decoding algorithms that exist in

the literature for tail-biting convolutional codes and PAC codes are implemented, and simulation results are obtained. Furthermore, we implement and review two other decoding algorithms for PAC codes, namely, the stack and creeper algorithms. Moreover, we develop a method to approximate the weight distribution of PAC codes. The derived (approximate) weight enumerating function of a given PAC code allows us to compare the performance of different codes using the corresponding union bound for maximum-likelihood (ML) decoding without running lengthy decoding simulations. By exploiting the union bound and simulated annealing based discrete optimization, we provide an effective and simplified design of the PAC code rate profiles.

1.1 Overview

Three main service categories for 5G have been defined by the third generation partnership project (3GPP). The first service is enhanced mobile broadband (eMBB), designed for services with high bandwidth requirements such as virtual reality, augmented reality, and high-resolution video streaming. The second category is massive machine-type communication (mMTC), which supports the massive number of machine-type devices with ultra-low power consumption, while the third one is URLLC [1]. URLLC requires low BLERs yet with less overall network latency. The physical layer design of URLLC is very challenging since these two aspects, namely ultra-low latency and ultra-reliable communications, are conflicting. One could use short packets to reduce latency, which causes a severe loss in coding gain. With this motivation, design and performance evaluation of short blocklength codes have gained significant interest in recent years.

In this thesis, we revisit some fundamental results on the performance bounds of codes in the short blocklength regime. These results enable us to properly compare the performance of various codes and decoding techniques. We first investigate the performance of convolutional codes in the short blocklength regime. We discuss that termination is a possible solution; however, zero tail termination imposes extra bits on the codeword, which is wasteful. To address this problem,

tail-biting termination is reviewed and simulated. We implement various decoders for tail-biting convolutional codes, among which the less complex suboptimal decoder called the wrap-around Viterbi algorithm (WAVA) offers a performance near the theoretical bounds.

Polar codes, introduced in [5] are the first family of codes that provably achieve the symmetric channel capacity with efficient encoding and decoding for different channels. Since its invention, the interest and research effort on polar codes have increased in academia and industry, and they have found practical applications, particularly in wireless communications. Indeed, they have been adopted as part of the $5th$ generation wireless systems (5G) standard. While polar codes can achieve the capacity of binary-input memoryless channels for infinite blocklengths, their finite-length decoding performance can be improved by concatenating them with an outer CRC code. The standard 5G polar code includes a CRC-11 outer code for the uplink channels, and a CRC-24 for the downlink ones [6] with a successive cancellation list decoder and a list size 8 [7].

Polarization adjusted convolutional codes (PAC) are proposed in [8] to improve the performance of polar codes in the short blocklength regime. These codes rely on concatenation of a rate-1 convolutional code with a polar code. PAC codes have attracted significant attention, as they are very flexible and approach the dispersion approximation. In this thesis, the PAC code structure is described, and several decoding schemes such as sequential decoders (Fano, stack, and creeper) and list decoders (successive cancellation list [SCL] decoder and simplified successive cancellation list [SSCL] decoder) are analyzed in terms of the performance and complexity. The SSCL decoder is employed in the majority of our simulations in the rest of the thesis. This is because the SSCL decoder, with a list size of 256, has an excellent performance similar to that of the Fano algorithm while being less complex since it does not repeatedly go back and forth in the decoding tree.

We compare the performance of PAC codes with the corresponding theoretical bounds for different channel models such as binary input additive white Gaussian

noise (BI-AWGN) channels, binary erasure channels (BEC), and Rayleigh block-fading (RBF) channels. Also, the performance of PAC codes is compared with the standard 5G polar codes with high order modulations, such as 64-QAM. Puncturing of the PAC codes to obtain higher rates is also investigated.

We also present several available results on the weight distribution of polar codes. These results are important to assess the performance of the codes from a theoretical perspective. We particularly, focus on the recent achievements in this area and reproduce two of them, namely, the probabilistic approach, e.g. in [9] and [10], and the deterministic one in [11].

Finally, we develop a method to compute the full weight distribution of the PAC codes in an approximate form. Utilizing the weight distributions, we compute the corresponding union bound; and based on the value of the union bound at a fixed SNR, we compare the performance of a given PAC code design without running a lengthy decoder simulation. Using this simplified performance assessment approach, we employ a discrete optimization method, namely, simulated annealing, to design PAC code rate profiles.

1.2 Thesis Contributions

The main objective of this thesis is to review the channel coding approaches for practical use in URLLC, and provide suitable decoding and design algorithms that maintain low decoding complexity and offer superior frame error rate performance.

We implement the existing methods in the literature to determine the ultimate limits of communications for the additive white Gaussian noise (AWGN) and the binary input AWGN (BI-AWGN) channels for small blocklengths. In addition, we compute the dispersion approximation for block fading channels in the finite blocklength regime. We also study the performance of PAC codes over the block fading channels via simulations. We then reproduce the results available in the

literature for the most promising classical and modern channel codes for use in finite blocklengths. Specifically, we consider the tail-biting convolutional codes with WAVA decoder as a classical approach. We also review the improvements that make the polar codes suitable to be used in 5G NR systems. We also, implement successive cancellation combined with sequential decoding algorithms to decode the polarization adjusted convolutional codes. As for the sequential decoding, we employ stack, Fano and creeper algorithms. We also describe a simplified list decoder combined with a successive cancellation algorithm for PAC codes.

As a major contribution of the thesis, we develop a method of computing the weight distribution of PAC codes in an approximate form by employing a probabilistic technique. We demonstrate that the results well match the exact weight distributions for small codes which can be computed using a brute-force algorithm. We also present an approach to employ the results (along with the union bound on the code performance) to design specific PAC codes, more precisely, to determine suitable rate profiles via simulated annealing. Numerical examples illustrate that the PAC codes with the designed rate profiles offer superior performance.

Part of our results in this thesis is reported in a conference paper which is accepted for presentation:

- S. Seyedmasoumian and T.M. Duman, "Approximate weight distribution of polarization-adjusted convolutional (PAC) codes," *2022 IEEE International Symposium on Information Theory (ISIT) (ISIT 2022)* (accepted).

1.3 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, we provide the finite blocklength bounds for AWGN and BI-AWGN channels. Furthermore, we give

an overview of suitable channel codes with short blocklengths. Moreover, we reproduce the results for candidate coding schemes in the literature. In Chapter 3, we focus on the performance of PAC codes over different channels, and describe a fast list decoding algorithm for them. Furthermore, we extend finite blocklength bounds in Chapter 2 to the case of block fading channels and study the performance of PAC codes over such channels. In Chapter 4, we describe the newly proposed approximation method for computing the weight enumerating function (WEF) of PAC codes. Additionally, a PAC code design, more precisely, rate profile design, based on the derived approximate WEF is presented, along with simulations and numerical results. Finally, in Chapter 5 we conclude the thesis and provide several future research directions.

Chapter 2

Preliminaries and Literature Review

In this chapter, we give a short background for the subsequent chapters by explaining the fundamentals of short blocklength codes for URLLC. We first review the finite length information-theoretic results. Then, an overview of the channel coding schemes which are frequently used in the short blocklength regime is presented.

2.1 Finite Block-Length Performance Limits for BI-AWGN Channels

Consider a BI-AWGN channel with the input-output relationship:

$$Y = \sqrt{\Omega}X + Z, \tag{2.1}$$

where Ω is the signal-to-noise ratio (SNR), X is the binary phase shift keying (BPSK) modulated codeword $\in \{1, -1\}^n$, Y is the observation and Z is the additive white Gaussian noise with zero mean and variance σ^2, i.e., $Z \sim \mathcal{N}(0, \sigma^2)$.

A commonly used technique to identify the limits of communications for a code with block length of N, is the so-called normal approximation which can be written as follows [12].

$$R_{NA} = C - \sqrt{\frac{V}{N}}\log_2(e)Q^{-1}(P_e) + O\left(\frac{\log_2(N)}{N}\right), \tag{2.2}$$

where, $O(.)$ is the standard big-O notation, and $Q^{-1}(.)$ is the inverse of Q-function defined as

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty \exp\left(-t^2/2\right)dt. \tag{2.3}$$

Furthermore, R_{NA} is the rate, C is the channel capacity, V is the channel dispersion, which in comparison to a deterministic channel with the same capacity, quantifies the channel's stochastic variability. P_e is the block error rate and N is codeword length.

Let us go through the calculation of the normal approximation of the BI-AWGN in finite blocklength regime based on [13]. In a BI-AWGN channel scenario the false alarm (FA) and Neyman-Pearson missed detection (MD) can be expressed as

$$P_{FA}(\lambda) = P\left[\sum_{i=1}^{n} h(u_i) \leq n\lambda\right], \tag{2.4}$$

$$P_{MD}(\lambda) = P\left[\sum_{i=1}^{n} h(v_i) > n\lambda\right], \tag{2.5}$$

where

$$h(x) = \ln\left(1 + e^{-2x}\right), \tag{2.6}$$

$$u_i \sim \mathcal{N}(d_i\Omega, \Omega),$$
$$v_i \sim \mathcal{N}(\Omega, \Omega).$$

These probabilities can be calculated using the Laplace transform method (see

[13]). The Laplace transform which converges for any s is given by

$$H(s) = \frac{1}{\sqrt{2\pi\Omega}} \int_{-\infty}^{\infty} e^{\frac{-1}{2\Omega}(x-\Omega)^2} e^{-s.h(x)} dx. \tag{2.7}$$

The l-th derivative of $H(s)$ can be written as

$$H^{(\ell)}(s) = \frac{1}{\sqrt{2\pi\Omega}} \int_{-\infty}^{\infty} e^{\frac{-1}{2\Omega}(x-\Omega)^2} e^{-s.h(x)} (-h(x))^{\ell} dx. \tag{2.8}$$

The channel capacity is given by

$$C = 1 + \frac{H'(0)}{\ln 2}, \tag{2.9}$$

and, the dispersion is given by

$$V = H''(0) - (H'(0))^2, \tag{2.10}$$

where

$$H'(s=0) = \frac{1}{\sqrt{2\pi\Omega}} \int_{-\infty}^{\infty} e^{\frac{-1}{2\Omega}(x-\Omega)^2} (-\ln(1+e^{-2x})) dx, \tag{2.11}$$

and

$$H''(s=0) = \frac{1}{\sqrt{2\pi\Omega}} \int_{-\infty}^{\infty} e^{\frac{-1}{2\Omega}(x-\Omega)^2} (\ln(1+e^{-2x}))^2 dx. \tag{2.12}$$

Finally, the achievable rate is obtained as [12]

$$R_{NA} = C - \sqrt{\frac{V}{N}} \log_2(e) Q^{-1}(P_e) + o\left(\frac{1}{\sqrt{N}}\right), \tag{2.13}$$

where $o(.)$ is the standard little-o notation.

Normal approximation of an achievable rate is obtained attained by omitting the $o(.)$ term; however, to find a more accurate approximation we take $o(\frac{1}{\sqrt{n}}) =$

$\frac{\log_2(N)}{2N}$. Finally, normal approximation for BI-AWGN channels can be obtained by

$$Pe = Q\left(\left((C - R_{NA}) - \frac{log_2(N)}{2N}\right)\sqrt{\frac{N}{V}\frac{1}{log_2(e)}}\right). \tag{2.14}$$

2.2 Channel Coding for the Short Block-length Regime

In the following subsections we review powerful channel coding constructions for short blocklengths. Particularly, we focus on tail-biting convolutional codes with a near ML decoding algorithm based on multiple uses of Viterbi decoding on the code trellis. Furthermore, we investigate polar-based codes with different decoding schemes and review the construction and performance of PAC codes.

2.2.1 Convolutional Codes

Convolutional codes were invented by Elias in 1955. They have been widely used for wireless, space, and broadcast communications since about 1970. A convolutional code is generated by passing the information sequence to be transmitted through a linear finite-state shift register. One method for describing a convolutional code is to give its generator matrix. Another method to describe a convolutional code is specifying a set of n vectors, one vector for each n modulo-2 adders. Each vector contains Kk dimensions and has the connections of the encoder to that modulo-2 adder, where K is the number of stages in the convolutional encoder and, k is the message length.

There exist different methods for decoding convolutional codes, including Viterbi, stack, or Fano algorithms. Viterbi algorithm outputs the codeword that

maximizes the probability of the received sequence conditioned on the information sequence; that is, it outputs the maximum-likelihood decision. The sequential decoding schemes such as Fano and stack algorithms are suboptimal decoding approaches with reduced complexity.

2.2.1.1 Viterbi Decoding Algorithm

In the Viterbi decoding algorithm, the codewords are paths through a trellis diagram. For ML decoding, we need to find the best path, more precisely, the path that has the lowest Hamming or Euclidean distance to received sequence. This distance is denoted as path metric.

We can summarize the Viterbi algorithm as follows.

1. Assuming that the convolutional encoder is at the zero state initially, assign the path metric zero to the initial node; set $t = 0$.

2. For each node at depth $t + 1$, find for each of the predecessors at depth t, the sum of the path metric of the predecessor and the branch metric of the connecting branch. Determine the maximum of these sums and assign it to this node and label the node with the shortest path to it.

3. If the end of the trellis is reached, stop and choose as the decoded codeword a path to the terminating node with the optimal metric; otherwise, increment t by 1 and go to step 2.

As an example, Let $r = (100110010100)$ be the received sequence. The trellis diagram of the code is given in Fig. 2.1, and a step-by-step Viterbi decoding flow is depicted in Fig. 2.2.

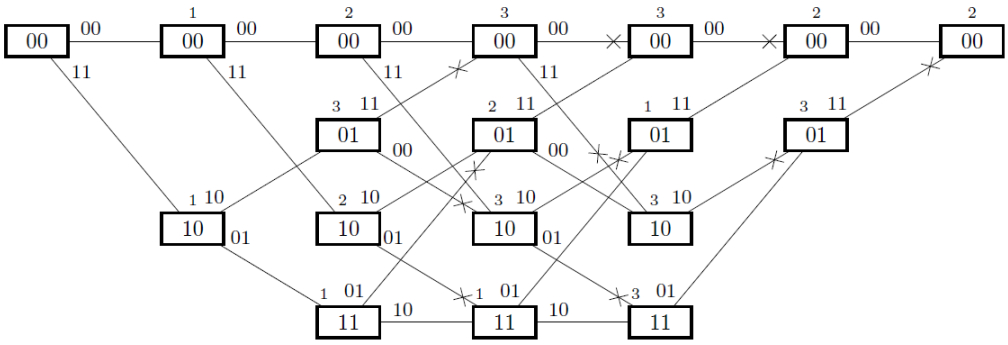

Figure 2.1: An example of Viterbi decoding—hard decisions.

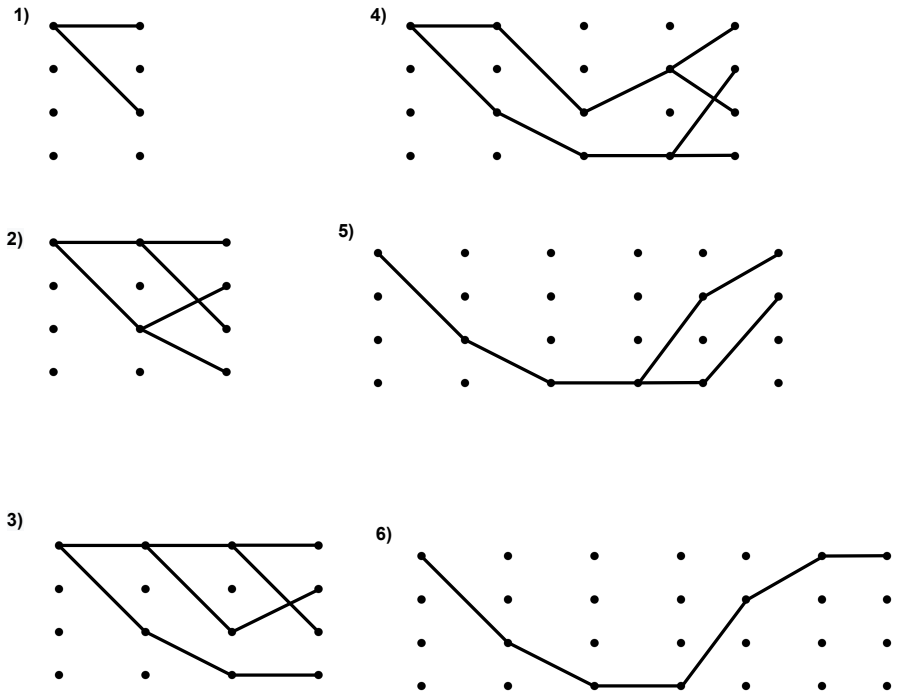

Figure 2.2: Development of subpaths through the trellis.

2.2.2 Tail-Biting Convolutional Codes

In zero-tail termination of convolutional codes, we use a binary sequence of length L with m dummy zeros as the input. The resulting code is an $((L+m)n, L)$ linear block code. The code rate is $\frac{1}{n}\frac{L}{L+m}$ where the term $\frac{m}{L+m}$ is called the rate loss [14]. The rate loss due to dummy zeros in short blocklength codes may be unacceptably high. A solution to this reduction in rate is to use tail-biting trellis termination. In tail-biting convolutional codes (TBCCs), we first initialize the encoder by inputting the last m information bits into the encoder and ignore the output. With $l = L-m$, we input all $l+m$ information bits into the encoder and take the resultant $(l+m)n$ output bits as our codeword. Tail-biting can be used to construct compelling regular trellis representations of block codes. Assuming a trellis of L sections ($L \geq m$), the tail-biting condition is the restriction that the convolutional encoder state at time $t = 0$ is equal to the encoder state at time L. The Fig. 2.3 summarizes the tail-biting structure.

We note that it is also possible to use a recursive tail-biting encoder as discussed in [15]. In the recursive case, the encoding state depends on the entire information vector; thus, we must calculate the initial state for a given information vector that leads to the same state after $l+m$ cycles. In [15], authors has calculated a relation between the initial state and zero-output solution as

$$(Z^{l+m} + \mathbf{I}_m)x_0 = x_{l+m}^{[zs]}, \tag{2.15}$$

where, $\mathbf{I}_m$ is $m \times m$ identity matrix. Z can be obtained from inspecting signal space representation of generator matrix. Therefore, using (2.15) we can obtain a recursive relation. The resulting block code is $((l+m)n, (l+m))$ so the rate is $\frac{1}{n}$ and there is no rate loss.

Suppose that we have $R = \frac{b}{c}$, as the rate of convolutional code with memory size m. A tail-biting trellis of length L corresponds to a total of $K = bL$ information symbols, c symbols per branch, block length $N = cL$ and 2^b branches per

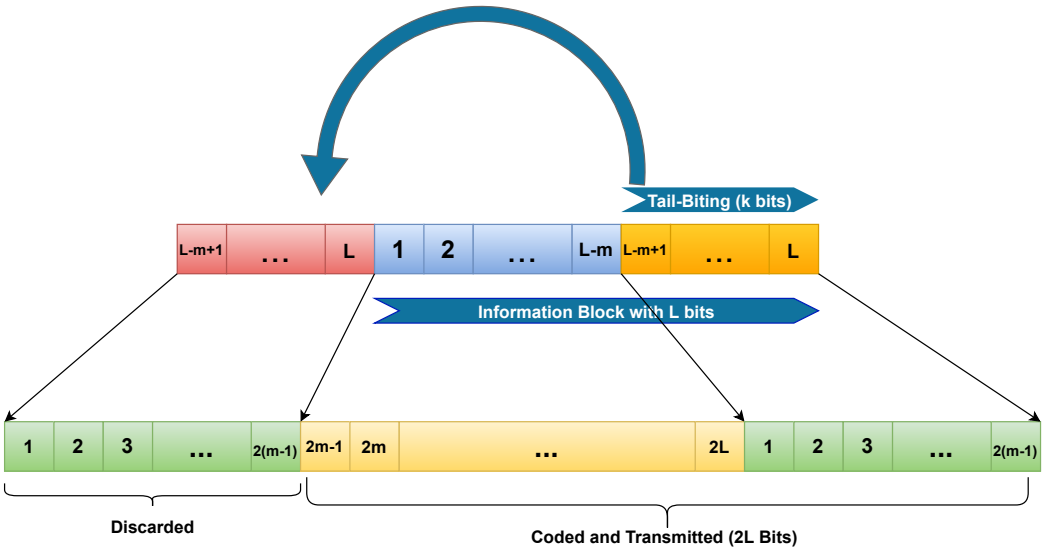

Figure 2.3: Tail-biting convolutional code.

trellis node. The number of codewords is

$$M = 2^K = 2^{bL}. \tag{2.16}$$

Let $\boldsymbol{u}_{[0,L)} = \boldsymbol{u}_0\boldsymbol{u}_1...\boldsymbol{u}_{L-1}$ denote the information sequence (input) and $\boldsymbol{v}_{[0,L)} = \boldsymbol{v}_0\boldsymbol{v}_1...\boldsymbol{v}_{L-1}$ indicate the output sequence. Then, the polynomial generator matrix for memory size m is

$$\boldsymbol{T}(D) = T_0 + T_1D + ... + T_mD^m, \tag{2.17}$$

where T_i 's are $b \times c$ matrices. Assuming that the encoder state is all zero at time $t = 0$, we have for $0 \leq t < L$

$$v_t = u_tT_0 + u_{t-1}T_1 + ... + u_{t-m}T_m. \tag{2.18}$$

2.2.2.1 Encoding of TBCCs

We assume in a tail-biting encoder that we have the encoder state at time $t = 0$ equal to the encoder state at time $t = L$, and that the input is the all zero

sequence. The output sequence corresponding to this assumption is

$$\boldsymbol{v}_{[0,L)}^{zi} = \boldsymbol{u}_{[0,L)}^{zi} \boldsymbol{T}_L^{zi}, \tag{2.19}$$

where $\boldsymbol{u}_{[0,L)}^{zi} = (0, 0, ..., 0, u_{L-m}, u_{L-m+1}, ..., u_{L-1})$ and

$$\boldsymbol{T}_L^{zi} = \begin{pmatrix} 0 & \cdots & & 0 \\ \vdots & & \ddots & \\ 0 & & \cdots & 0 \\ T_m & \cdots & & 0 \\ T_{m-1} & T_m & & \\ \vdots & & \ddots & \\ T_1 & T_2 & \cdots & T_m \end{pmatrix}.$$

We can compactly write (2.19) as

$$\boldsymbol{v}_{[0,L)} = \boldsymbol{u}_{[0,L)} \boldsymbol{T}_L^{tb}, \tag{2.20}$$

where $\boldsymbol{T}_L^{tb} = \boldsymbol{T}_L + \boldsymbol{T}_L^{zi} = \begin{bmatrix} T_0 & T_1 & \cdots & & T_m & 0 & \cdots & 0 \\ 0 & T_0 & T_1 & \cdots & & T_m & & \vdots \\ \vdots & & \ddots & \ddots & & & \ddots & 0 \\ 0 & \cdots & 0 & T_0 & T_1 & \cdots & & T_m \\ T_m & & & & T_0 & T_1 & \cdots & T_{m-1} \\ T_{m-1} & T_m & & \ddots & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & & & & \ddots & T_1 \\ T_1 & T_2 & \cdots & T_m & 0 & \cdots & 0 & T_0 \end{bmatrix}$.

2.2.2.2 Performance of TBCCs

The maximum likelihood tail-biting decoder involves determining the best path in the trellis with identical initial and final states, i.e., under the constraint that the starts and ends are in the same states (the ending state being at a depth of $(l+m)$ in the trellis). However, we do not have any information of starting (ending) state,

so any of the states are equally probable to be the starting (ending) state prior to the decoding procedure. To derive the best path, we take advantage of the trellis representation of codes and the Viterbi algorithm. We can run M parallel Viterbi algorithms where M is the number of states in the trellis and select the decoded bits based on the Viterbi algorithm, which gives the best metric [2]. Each of the Viterbi decoders with a given starting (ending) state operates a sub-trellis with equal initial/final state for all possible states. Each Viterbi decoder outputs the most likely codewords for the given initial (final) state. Among the M obtained codewords, a further ML search is performed; in other words, we continue the Viterbi decoding process until we get a valid codeword (satisfying same initial and final states), providing the final decision. With the squared Euclidean distance as the metric, the best path is the one with the least accumulated metric.

The ML decoding algorithm's complexity is proportional to M, i.e., we need to perform 2^{mK} full rounds of Viterbi algorithm for an (N, K, m) code. This decoding approach becomes highly impractical with increased code memory. However, if one can predict the starting state, only one Viterbi trial would be sufficient.

Wrap-around Viterbi algorithm (WAVA) introduced in [14] is a circular decoding algorithm used to process tail-biting trellises iteratively. It runs the Viterbi algorithm successively for multiple iterations, and at the beginning of each iteration, improves the decision. In WAVA, the state metric of $s \in M$ at iteration i, denoted as $M_t^i(s)$ accumulates branch metric continuously starting from the very first iteration. At the end of iteration i, survivors of length k (equal to the size of transmitted bit sequence) can be output by tracing back. The best path and the best tail-biting path (a path with the same initial and final states) up to iteration i are denoted as $\rho_{best}^{(i)}$ and $\rho_{TB,best}^{(i)}$, respectively.

We can summarize WAVA as follows:

1. Start from all the states in S_0 with state metrics set to zero.
2. Process the trellis up to the end (length K) and record the $\rho_{best}^{(i)}$ and $\rho_{TB,best}^{(i)}$ if they exist.

3. At the end of first iteration if $\rho_{best}^{(1)} = \rho_{TB,best}^{(1)}$, stop decoding and report $\rho_{TB,best}^{(1)}$ as the output sequence. Otherwise continue.

4. At iteration i, for $i > 1$ the state metrics in the initial states are set to the metrics at the end previous iteration. In other words: $M_0^i(s) = M_K^{i-1}(s)$. At the end of iteration, if $\rho_{TB,best}^{(i)}$ is being recognized as most likely TB path, report it as the output sequence. Otherwise go to step 5.

5. Repeat until the maximum number of iterations is reached.

6. Output $\rho_{TB,best}^{(max(i))}$ if it exists, otherwise output $\rho_{best}^{max(i)}$.

Table 2.1 summarizes the codes selected for the numerical results. The generator polynomials are selected from those that lead to large minimum distances [16] with different m values $\{8, 11, 14\}$ selected from [2].

Figs. 2.4 to 2.6 depict the structure of encoders employed.

Table 2.1: Selected encoders for the examples.

Generators (in octal notation)	m	(N, K)
[515,677]	8	(128,64)
[5537,6131]	11	(128,64)
[75063,56711]	14	(128,64)

In Fig. 2.7, we compare the performance of (128,64) TBCC decoded with ML and WAVA with 4 iterations, where we can see that WAVA has a negligible degradation in the performance.

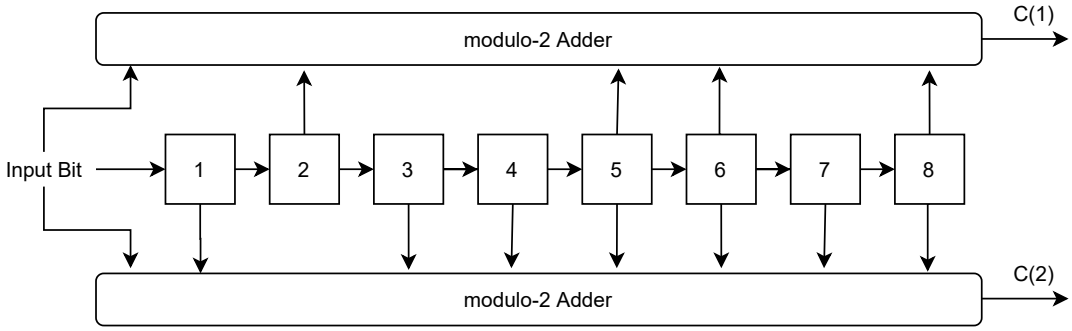

Figure 2.4: [515,677] encoder corresponding to m=8.

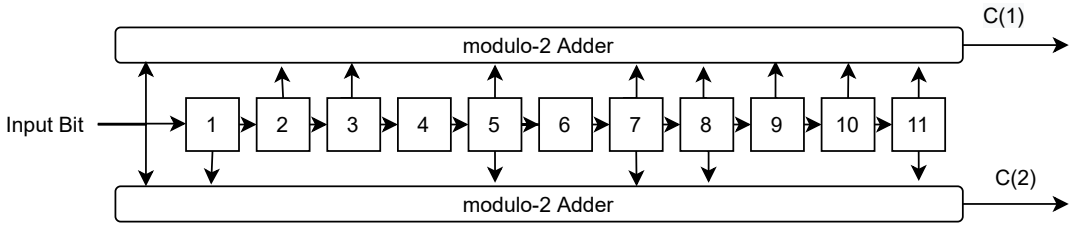

Figure 2.5: [5537,6131] encoder corresponding to m=11.

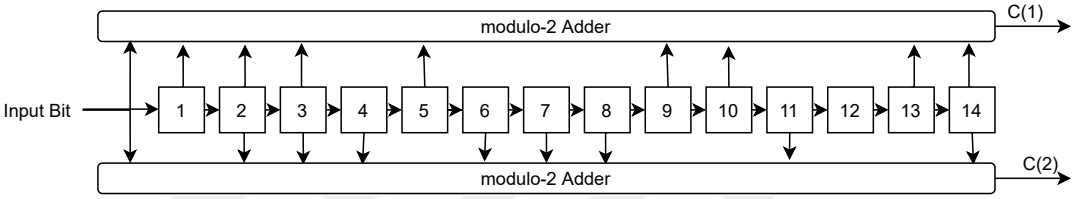

Figure 2.6: [75063,56711] encoder corresponding to m=14.

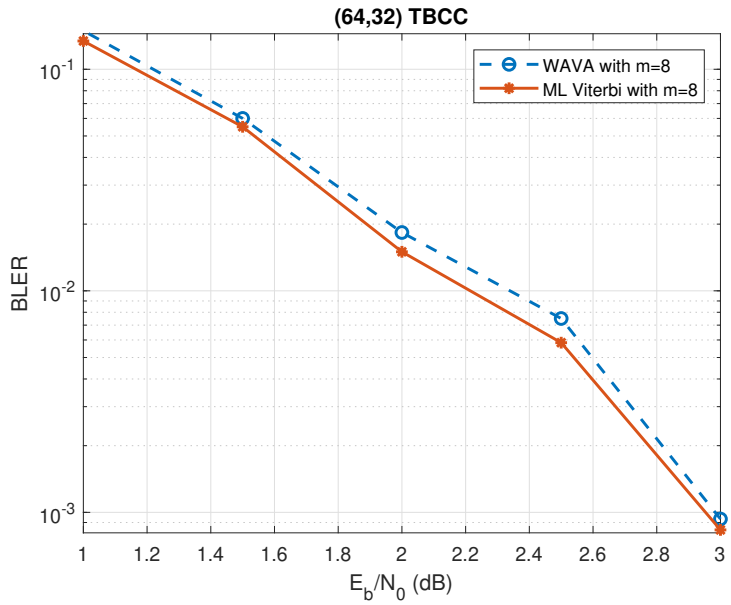

Figure 2.7: WAVA and ML decoding of the TBCC, example.

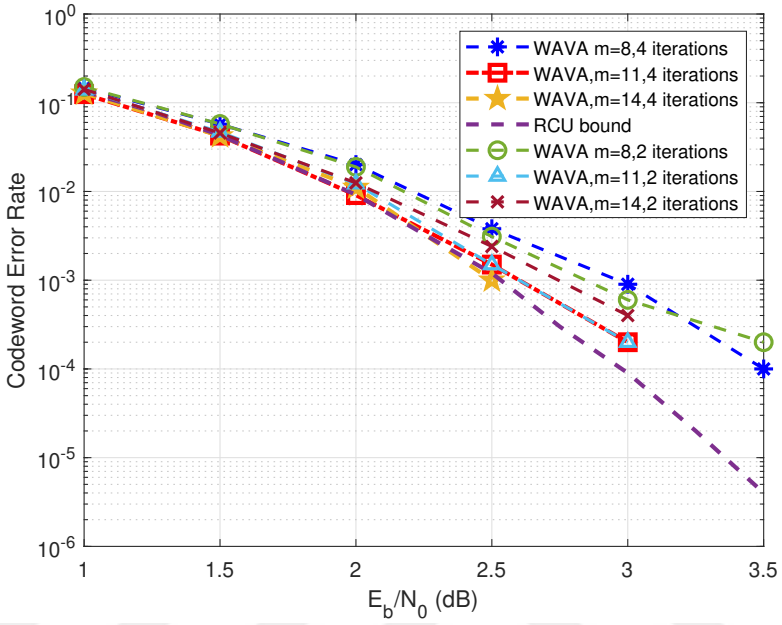

Figure 2.8: WAVA vs RCU bound with different iterations.

The results show that the performance of WAVA with $m = 14$ and even $m = 11$ is very close to that of ML decoding. The ML decoding complexity increases linearly with M (the number of states) because we need to perform 2^{mK} Viterbi decoding algorithms, while in WAVA, we get feedback after each Viterbi search is ended; and hence, we need very few Viterbi trials. As a specific example, we perform a simulation using the code with $m = 8$ with Matlab, and find out that the average running time for each block in the ML decoder is 7.119 seconds, while this number for WAVA is 0.086 seconds. These results show that WAVA is about 80 times less complex than ML Viterbi decoder for this specific example. Also in Fig. 2.8, we depict the performance of WAVA for different iterations for a (64,32) code. We also observe that the code is near optimal with a very close performance to the random coding union (RCU) bound, as number of iterations or size of m increases.

Figs. 2.9 and 2.10, depict the results for the WAVA decoding of TBCC for $N = 128$ and 256, respectively. Both of these simulations are for rate 1/2 codes, and the number of iterations is selected as 4. The results are also compared with

the corresponding normal approximation.

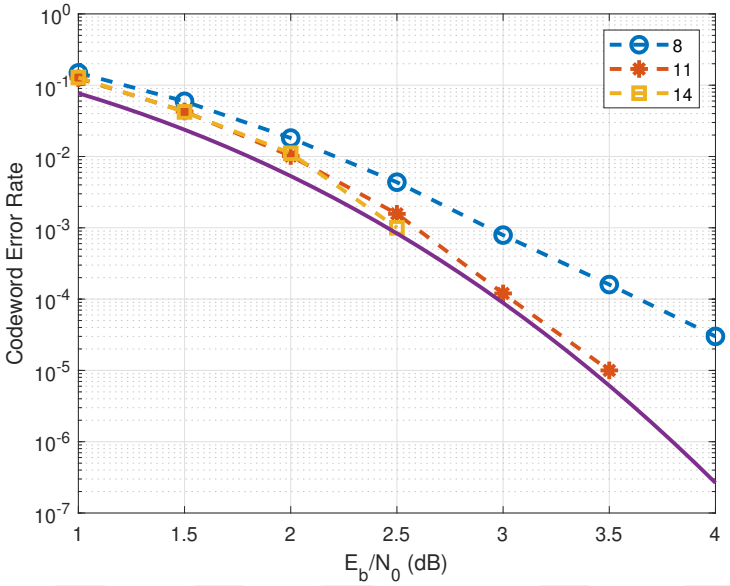

Figure 2.9: Codeword error rate vs. E_b/N_0 for (128,64) binary TBCCs with different memories $m = 8$, $m = 11$ and $m = 14$ compared to normal approximation.

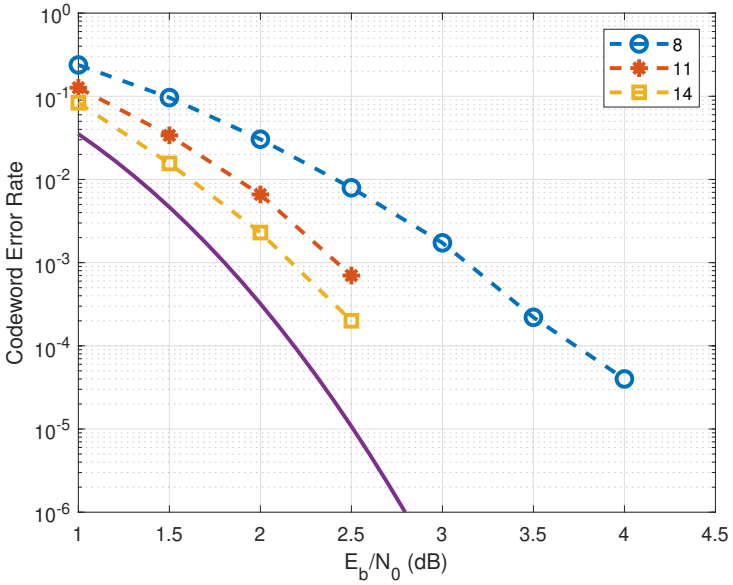

Figure 2.10: Codeword error rate vs. E_b/N_0 for (256,128) binary TBCCs with different memories $m = 8$, $m = 11$ and $m = 14$ compared to normal approximation.

2.2.3 Polar Codes

Polar codes are the first class of codes that can provably attain the capacity of a symmetric binary-input discrete memoryless channel (B-DMC) using efficient encoding and decoding methods introduced in [5]. The basic concept of polar code construction, also known as channel polarization, is to transform the independent copies of a binary symmetric memoryless channel into noiseless and useless synthetic channels by applying a recursive transformation to the input bits, then, transmitting information bits over noiseless channels while the inputs to noisy channels are predetermined values (frozen bits). In other words, the synthesized channels become either noiseless or extremely noisy when the code length $N = 2^n$ approaches infinity. The fraction of noiseless channels approaches the channel capacity. Let $W : \{0,1\} \to \mathcal{Y}$ be a binary memoryless symmetric (BMS) channel, characterized in terms of its transition probabilities $W(y|x)$ for all $x \in \{0,1\}$ and $y \in \mathcal{Y}$, where $\boldsymbol{y}$ denotes the received codeword. Let $W^N : \{0,1\}^N \to \mathcal{Y}^N$ be the channel corresponding to N independent uses of W, the ith bit channel $W_N^{(i)} :\to \{0,1\} \to \{0,1\}^{i-1} \times \mathcal{Y}^N$ for $0 \leq i \leq N-1$ can be defined as [11]

$$W_N^i(\boldsymbol{y}, \boldsymbol{u}_{i-1}|u_i) = \frac{1}{2^{N-1}} \sum_{u' \in \{0,1\}^{N-1}} W^N(\boldsymbol{y}|(\boldsymbol{u}_{i-1}, u_i, \boldsymbol{u'})G_N), \tag{2.21}$$

where $(\boldsymbol{u}_{i-1}, u_i, \boldsymbol{u'})$ means concatenation of $\boldsymbol{u}_{i-1}$, u_i, and $\boldsymbol{u'}$. We say that a pair of binary-input channels can be obtained by a single-step transformation of two independent copies of a binary-input channel:

$$\left(W_N^{(i)}, W_N^{(i)}\right) \to \left(W_{2N}^{(2i-1)}, W_{2N}^{(2i)}\right).$$

The positions of $N - K$ least reliable synthetic channels, or equivalently frozen bit positions differ based on channel characteristics.

An efficient polar code design is density evolution based Gassian approximation (DEGA) [17]. Assume polar code with $N = 2^n$, an (N, K) polar code is a binary linear block code generated by K rows of the polar transformation matrix

$\boldsymbol{G}_N = \boldsymbol{B}_N \boldsymbol{P}_2^{\otimes n}$, where

$$\boldsymbol{P}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

where $\boldsymbol{P}_2^{\otimes n}$ is the n-th Kronecker power of $\boldsymbol{P}_2$, and $\boldsymbol{B}_N$ is an $N \times N$ bit-reversal permutation matrix. We denote information plus frozen vector of length N, by $\boldsymbol{u}$, and $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}_N$ denotes the obtained codeword.

Arıkan in [5], introduced a successive cancellation (SC) decoder and shown that the block-error probability of polar codes under SC decoding goes to zero as block length goes to infinity if the code rate is below the channel capacity. In SC the non-frozen bits are estimated consecutively thanks to channel polarization, based on their evolving log-likelihood ratio (LLR) with complexity of $O(NlogN)$.

Although the performance of polar codes under the SC in specific designs is very good, its performance in the short blocklength regime needs to be improved due to inferior minimum distance characteristics. To compensate for the suboptimality of the SC decoder, [18] introduced the successive cancellation list (SCL) decoder, whose computational complexity is proven to grow in the same way as that of the SC decoder in terms of blocklength. One way to improve the performance is using an outer code to increase their minimum distance thanks to SCL. To address this issue, getting aid from an outer higher rate code such as CRC is subject of study in [19] and [20]. Adding the CRC increases neither the encoder's nor the decoder's computational complexity considerably, while significantly reducing the block-error probability, making the error-rate performance of the modified polar codes under SCL decoding comparable to that of state-of-the-art low-density parity check (LDPC) codes.

In Fig. 2.11, performance of CRC-aided polar codes and standard polar codes are compared with the theoretical bounds with a blocklength of 128. RCU and random coding bound (RCB) are upper achievability bounds, while MC (metaconverse) and SPB59 are converse bounds. NA represents the normal approximation [21]. The results show that CRC-aided polar codes have excellent performance. As a consequence, the CRC-aided polar codes are used in 5G-NR systems.

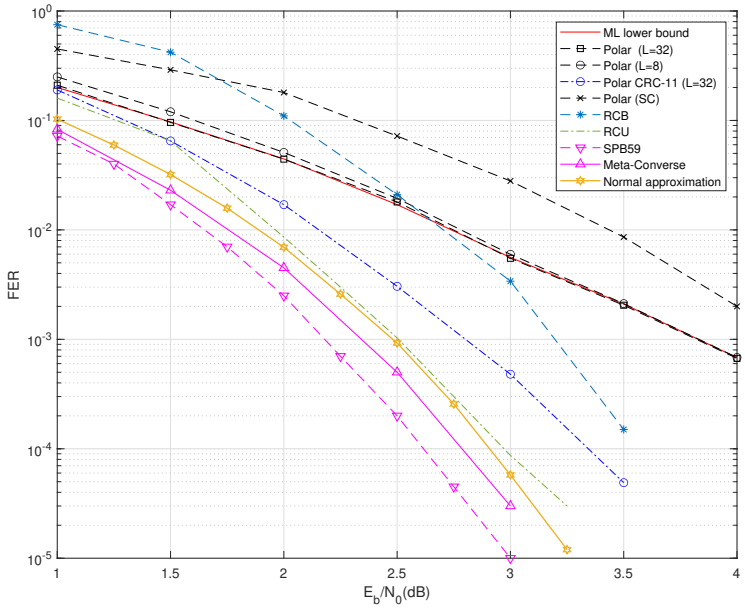

Figure 2.11: Codeword error rate vs. E_b/N_0 in dB of polar codes and CRC-aided polar codes for short blocks (128,64), BI-AWGN channel.

2.2.4 PAC Codes

PAC codes are constructed by a concatenation of a polar code and a convolutional code which the convolutional code acts as an outer precoding block. This scheme shifts the load of correcting an error to the convolutional coding side (outer coder). Fig. 2.12 summarizes the structure of the PAC encoding and decoding schemes. The codeword length is $N = 2^n$ where $n + 1$ is the number of layers in the polar tree (see Fig. 2.14). The received LLR from the channel are decoded and at the end, data is extracted with the knowledge of positions of frozen bits.

Consider a PAC code represented by $(N, K, \mathcal{A}, \boldsymbol{T}_N)$ where N and K are codeword and message lengths, respectively, $\mathcal{A}$ is the set of information (non-frozen) bit-positions and $\boldsymbol{T}_N \in \mathbb{R}^{N\times N}$ is an upper-triangular Toeplitz matrix constructed with the convolutional code generator $\boldsymbol{g} = [g_1, g_2, ..., g_\ell]$, where it is assumed that

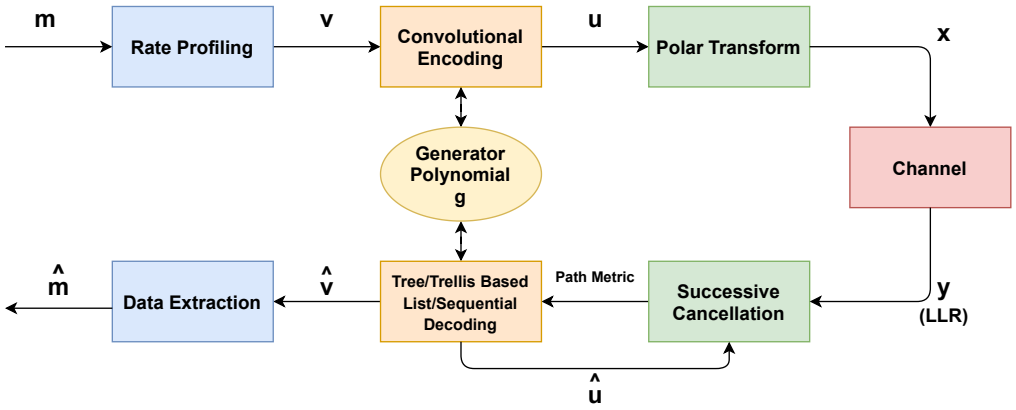

Figure 2.12: Block diagram of PAC code structure.

$g_0,\, g_\ell \neq 0$ (see [8]), i.e.,

$$\mathbf{T}_N = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_m & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_m & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & & & & \vdots \\ \vdots & & & & & & & \vdots \\ \vdots & & & & & & & \vdots \\ \vdots & & & & 0 & 0 & g_0 & g_1 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & g_0 \\ & & & & & & & \end{bmatrix}. \tag{2.22}$$

The encoding and decoding steps of PAC codes are depicted in Fig. 2.12. The encoder consists of three phases. In the first phase, K information bits $\boldsymbol{m} = [m_0, m_1, ..., m_{K-1}]$ are mapped to K information bit-locations in the input sequence $\boldsymbol{v}$ of length N using the information index set $\mathcal{A}$ where $\boldsymbol{v}_\mathcal{A} = \boldsymbol{m}$, and the cardinality of $\mathcal{A}$ equals K ($|\mathcal{A}| = K$), where $\boldsymbol{v}_\mathcal{A} = \{v_i | i \in \mathcal{A}\}$. The remaining $N - K$ bit-locations ($\mathcal{A}^c$) are called the frozen bit-locations, and $\boldsymbol{v}_{\mathcal{A}^c} = 0$ is selected. This phase is the *rate-profiling* step. The rate profile of a PAC code has a big impact on its error-correcting performance. As an effective approach, Reed-Muller (RM) rate profiling is employed in [8], where the bit-locations with higher RM scores are used as the information set. We note that for some parameters, we need to pick only a portion of the bit-locations with identical RM scores,

which makes identification of the optimal frozen (or, information) bit-locations challenging, even under the RM score function construction.

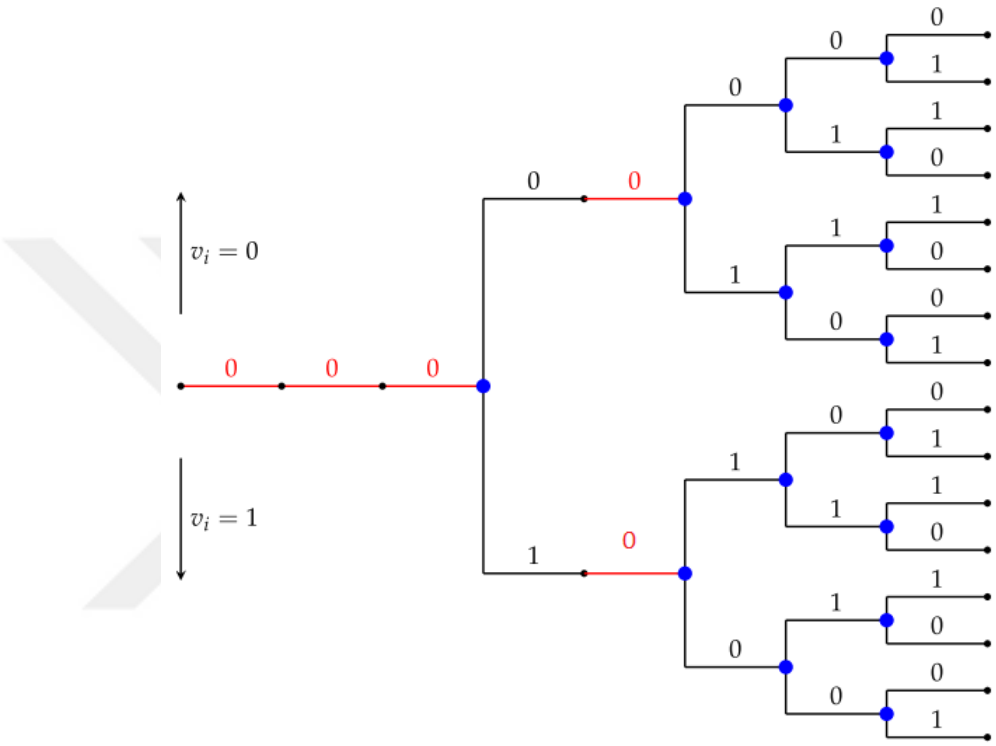

Figure 2.13: Irregular tree of PAC codes in convolutional decoding side.

The second phase in the encoding process is the one-to-one convolutional encoding step. We denote the output of the convolutional encoder by $\boldsymbol{u}$. The ith bit of the convolutional encoder is obtained by $u_i = \sum_{j=0}^{\ell} g_j v_{i-j}$, where $v_{i-j} = 0$ for $i < j$, i.e., $\boldsymbol{u} = \boldsymbol{v}\boldsymbol{T}_N$. The known useless bits lead the tree of convolutional code to be an irregular one as Fig. 2.13. In the PAC code structure, the convolutional encoder is one-to-one. The convolutional code can be described with a generator polynomial $\mathbf{g}$, where in the PAC code it is assumed that g_0 and $g_m \neq 0$. If we assume that the output of the convolutional encoder is $\mathbf{u}$, then we can say $\mathbf{u}$ is obtained as

$$u_i = \sum_{j=0}^{m} g_j v_{i-j}. \tag{2.23}$$

where, $v_{i-j} = 0$ for $i \leq j$.

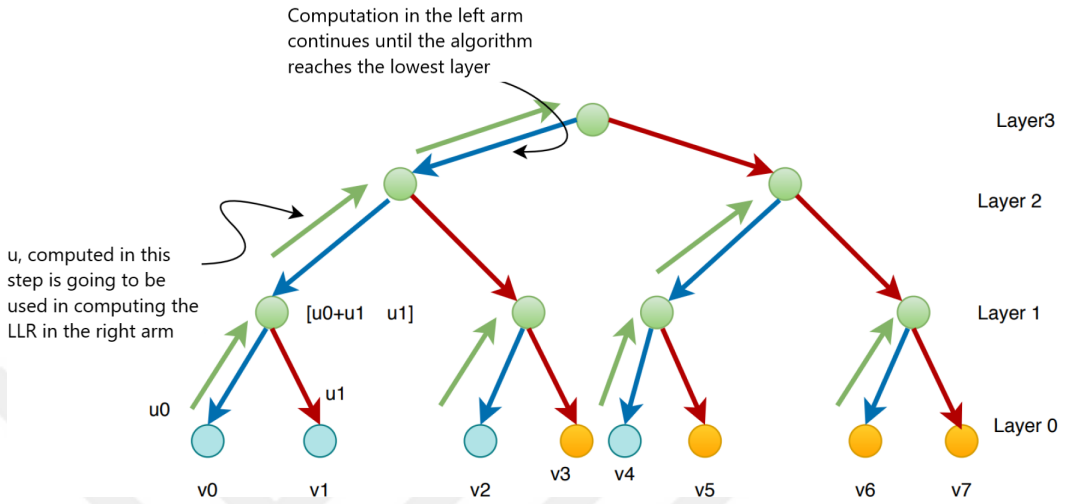

Figure 2.14: Successive cancellation tree of PAC code.

In the third phase of the PAC encoder, the output of the convolutional encoder is fed to a polar encoder. That is, the codeword is obtained as $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}_N$, where

$$\boldsymbol{G}_N = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{n}, \tag{2.24}$$

where $\otimes$ represents the Kronecker product.

In the decoding phase, the conventional or improved successive cancellation algorithms can be employed. Fig. 2.14 depicts an example with a blocklength $N = 8$, i.e., $n = 3$ corresponding to $n + 1 = 4$ layers indexed from 0 to 3 in the polar tree. Since the rate here is $\frac{1}{2}$, we have message length $K = 4$, and message sequence is denoted as $\mathbf{m} = [m_1, m_2, m_3, m_4]$. For $N = 8$, the frozen bit locations based on the RM score (which will be elaborated in the next subsection) are $\{0, 1, 2, 4\}$ that are shown by blue circles. This means that $\mathcal{A} = \{3, 5, 6, 7\}$, and $\boldsymbol{v} = [0, 0, 0, m_1, 0, m_2, m_3, m_4]$.

2.2.4.1 Rate Profiling

This step is the analogous to the frozen bit allocations in simple polar codes. Two adopted rate profiling methods introduced in [8] are polar and RM rate profiling.

Polar Rate-Profiling:
Let W_0, W_1,...,W_{N-1} be the N bit channels. $\mathcal{A}$ is chosen with respect to $\{W_i : i \in \mathcal{A}\}$, which are the K highest values in terms of their capacity $I(W_0), I(W_1), ..., I(W_N)$. For this rate profiling, DEGA method can be used.

Reed-Muller Rate Profiling:
Consider Reed-Muller score function $s(i) = w(i-1)$, where the $w(i-1)$ is the sum of ones in the binary representation of $i-1$, $i-1 \leq N-1$. For example binary representation of 20 is 10100, so $w(20) = 2$. K data bits are located at the positions with the higher RM score.

In very short blocks (e.g., $N = 64$ or 128), the RM score based rate profiling performs better while with larger blocklengths the rate profile based on Bhattacharyya parameters has better block error rate performance [22].

2.2.4.2 LLR Calculation over BI-AWGN channel

The channel input-output relationship is given by

$$y = x + \sigma.z \tag{2.25}$$

where x is the modulated version of the coded bit, y is the received signal, and z is standard Gaussian distributed, i.e., $z \sim \mathcal{N}(0,1)$. The LLR in BI-AWGN channel is computed as

$$LLR(x|y) = \ln\left(\frac{P(y|x=+1)P(x=+1)}{P(y|x=-1)P(x=-1)}\right) \tag{2.26}$$

$$= \ln\left(\frac{\exp\left(-\frac{(y-1)^2}{2\sigma^2}\right)}{\exp\left(-\frac{(y+1)^2}{2\sigma^2}\right)}\right), \tag{2.27}$$

since both $+1$ and -1 are equiprobable. Hence,

$$LLR = \ln\left(\exp\left(\frac{-(y^2+1-2y)+(y^2+1+2y)}{2\sigma^2}\right)\right) = \frac{4.y}{2\sigma^2} = \frac{4.y}{N_0}. \quad (2.28)$$

This LLR is input to the PAC decoder.

2.2.4.3 Fano Decoding of PAC Codes

In this subsection, we review a sequential decoding algorithm used for PAC codes, namely, the Fano algorithm. The Fano algorithm is an extremely clever algorithm that searches for the best path over code tree by moving from the current path to either its immediate predecessor path or the selected successor path. Fano decoder is an algorithm that does not require any storage, unlike some other sequential decoding algorithms (e.g., stack decoder), but it may be slower since it visits more nodes. Since it requires essentially no memory it is more suitable for hardware implementations. Fano algorithm starts at the starting node of the tree and examines a series of branches until it reaches the end node. The metric of the location after each movement between branches is calculated. The metric of the next node to be examined is the metric of the connected branch and is calculated using the metric of the current node. This process eliminates the need to store metrics for previously visited branches. However, some nodes can be visited multiple times [23].

If the decoder looks backward from the root, we assume that the "predecessor" of the root has a metric value of $-\infty$. The decoder can visit a node only if its Fano metric μ_F is larger than or equal to the current value of a certain threshold T which takes on only discrete values $...-2\Delta, -1\Delta, 0, \Delta, 2\Delta, ...$, where Δ is the step size. If the successor's metric falls below the threshold, then the Fano decoder first moves backward and then tries to move forward to its next best successor. If this fails, another move backward is necessary, and so on. Eventually, all paths with metrics above the threshold T have been systematically visited. When the Fano decoder reaches the node where the threshold T was increased the previous time, the Fano decoder lowers T by Δ and tries to move forward again, now with a lower threshold. This means that when a node is revisited, $T + \Delta \leq \mu_{cur}$ holds.

Eventually, the Fano decoder will reach the end of the tree and completes the decoding procedure.

An Important issue here is the calculation of the Fano metric for PAC codes. Let us say we are in the ith branch of the irregular tree. Branch metric is calculated using $y^N \in \mathbf{R}^N$ information passing through the channel as

$$\begin{aligned}
\Gamma(u^i; y^N) &= \log_2 P(u^i|y^N) && (2.29)\\
&= \log_2 \left(\frac{P(y^N|u^i)P(u^i)}{P(y^N)} \right) && (2.30)\\
&= \log_2 \frac{P(y^N|u^i)}{P(y^N)} + \log_2 P(u^i) && (2.31)\\
&= \log_2 \frac{P(y^N|u^i)}{P(y^N)} - iR(i) && (2.32)\\
&= \log_2 \frac{P(y^N|u^i)}{P(y^N)} - \sum_{j=1}^{i} b_j, && (2.33)
\end{aligned}$$

where, $R(i)$ represents the code rate up to the ith bit. The offset, $iR(i)$ is taken as a constant for reciprocal transactions. We do not need to assign a same value for $iR(i)$ for every location. In other words, $b_j = \rho$ for $j \in \mathcal{A}$ and $b_j = 0$ for $j \notin A$ for some constant ρ. The path metric is calculated incrementally as a sum of branch metrics of the form

$$\begin{aligned}
\gamma(u_j; y^N|u^{j-1}) &= \Gamma(u^j; y^N) - \Gamma(u^{j-1}; y^N) && (2.34)\\
&= \log_2 \frac{P(y^N|u^j)}{P(y^N|u^{j-1})} && (2.35)\\
&= \log_2 \frac{P(y^N|u^j)}{P(u_j = 0)P(y^N|u^{j-1}, u_j = 0) + P(u_j = 1)P(y^N|u^{j-1}, u_j = 1)}, && (2.36)
\end{aligned}$$

Therefore, from (2.33) and (2.34),

$$\gamma(u_j; y|u^{j-1}) = \log_2 \frac{P(y^N|u^i)}{P(y^N)} - b_j. \quad (2.37)$$

Selection of appropriate values for ρ and Δ are critical for performance-complexity tradeoff. A recent work, [24] investigates the near-optimum parameters for Fano

decoder of PAC codes. Table 2.2 summarizes suitable $\boldsymbol{g}$ and ρ choices that minimize the Fano algorithm's complexity for achieving a good frame error rate (FER).

Table 2.2: $\boldsymbol{g}$ and ρ table

N	K	$\boldsymbol{g}$(in octal notation)	ρ	Δ
128	29	3211	1.4	2
128	64	133	1.33	2
128	99	133	1.14	2

At this point, we need to calculate the LLR for the SC decoder to establish a connection with the probability ratio used in SC in the polar codes. We can write

$$LLR(u_j) = \frac{P(y^N, u^{j-1}|u_j = 0)}{P(y^N, u^{j-1}|u_j = 1)} \tag{2.38}$$

$$= \frac{P(y^N|u^{j-1}, u_j = 0)}{P(y^N|u^{j-1}, u_j = 1)}. \tag{2.39}$$

Using (2.34) and (2.38), we obtain
for $u_j = 0$:

$$\gamma(u_j = 0; y^N|u^{j-1}) = \frac{LLR(u_j)}{\frac{1}{2}\left(1 + LLR(u_j)\right)}, \tag{2.40}$$

and for $u_j = 1$:

$$\gamma(u_j = 1; y^N|u^{j-1}) = \frac{1}{\frac{1}{2}\left(1 + LLR(u_j)\right)}. \tag{2.41}$$

These metrics are being used as path metrics in our code for Fano decoding. In a summary, we take the LLR as $(n+1) \times N$ matrix which $n = \log_2(N)$. We feed $4y/N_0$, where y is the received signal, to the first row of this LLR matrix. Our aim in the polar code tree is to calculate the bottom row of the LLR matrix by successive cancellation algorithm. $LLR(u_j)$ here represents this value, depending on u_j and previous u's. After finding $LLR(u_j)$ by (2.40) and (2.41), we calculate the path metrics, which are used in the convolutional decoding tree.

2.2.4.4 Stack Decoding of PAC Codes

The Fano decoder runs slowly because the it frequently moves back and forth on consecutive nodes. The stack algorithm can also be used as a PAC decoder. The stack algorithm works faster than the Fano decoder because it visits fewer nodes, but unlike the Fano algorithm, significant memory usage is required. Considering the system requirements, the most suitable algorithm can be obtained by evaluating the relationship between speed and memory. The adjective "stack" is used to indicate that the paths in the partially explored tree are ordered by the Fano metrics. The best path is placed at the top of the stack; then we have the second best path and so on. In the stack algorithm, we follow the steps below:

1. Load the stack with the root and the metric zero. The metric calculation of this single node in the start is done using the Fano metric as described previously.
2. Branches are sorted in memory in descending order of their Fano metrics. If the number of paths in memory is higher than the memory size, the last ranked path is deleted.
3. Remove the top node and place its successors in the stack according to their metrics.
4. If the top path leads to the end of the tree, then stop and choose the top path as the decoded codeword; otherwise, go to Step 3.

As expected, the performance of the stack algorithm is dependent on the memory size. While an inferior performance is obtained in very low memory sizes compared to Fano decoder, a performance close to that of Fano decoder can be obtained when the memory size is increased enough. However, by increasing the memory size, first of all, we lose memory resources, and also the complexity of decoder increases.

The BLER performance we get with different memory sizes when transmitted via a BI-AWGN channel for a (128,64) PAC code using RM rate profiling is given in

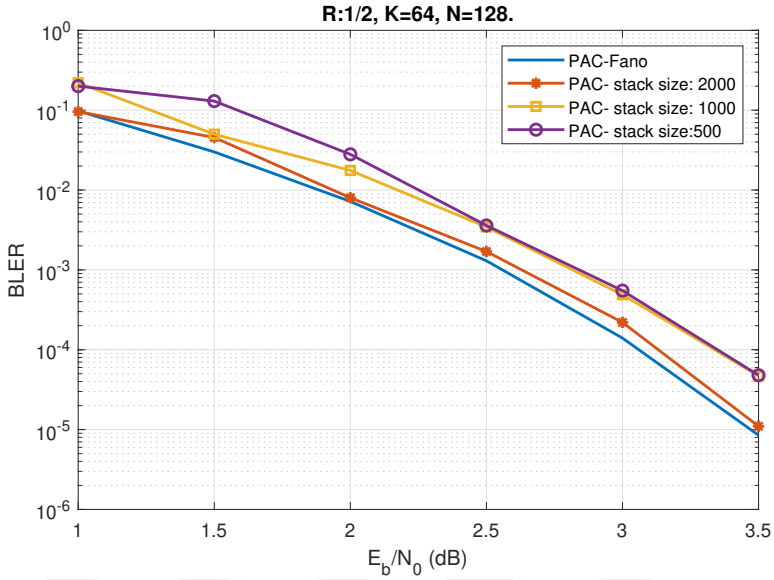

Figure 2.15: Performance of stack decoding with different stack sizes.

Fig. 2.15, compared to the Fano decoding results. The impulse response of the convolutional code used in this example is $\boldsymbol{g} = (1, 0, 1, 1, 0, 1, 1)$.

2.3 Chapter Summary

In this chapter, we have reviewed normal approximation for BI-AWGN channels to be used as a benchmark for the channel coding algorithms. We also studied structures of some channel codes that have good block error rate performances. The performance of tail-biting codes is close to theoretical bounds with large shift register sizes utilizing the WAVA decoder with relatively large number of Viterbi iterations. We have also investigated the performance of CRC-aided polar codes which find applications in 5G NR systems. Furthermore, we studied the structure of PAC codes which have better performance compared to CRC-aided polar codes over AWGN channels for short blocklengths.

Chapter 3

Performance of PAC Codes over AWGN and Rayleigh Block Fading Channels

In this chapter, we focus on the performance of PAC codes over different channels. In the previous chapter, we briefly reviewed several decoding algorithms for PAC codes. As discussed, the Fano decoder can be complex since, in some scenarios, the algorithm may be stuck going forward and backward without reaching the end of the tree. In these cases, a node visiting threshold can be utilized, which in turn may result in an inferior BLER. As an alternative, the stack decoder can be costly for hardware implementation.

With the motivation of reducing the complexity of PAC decoders, we first implement the creeper algorithm for decoding PAC codes, and then we describe a simplified list decoder. We provide extensive numerical results on the performance of PAC codes for different scenarios. Furthermore, we obtain the normal approximation based on finite length information theoretic results for block fading channels, and compare the performance of PAC codes with the approximation to confirm that they can be good candidates for use over block fading channels.

3.1 Creeper Decoding Algorithm for PAC Codes

The creeper algorithm [23] is designed as a compromise between the stack and Fano algorithms. It is a sequential decoder based on tree decoding representation. It can be implemented without the need for external memory, which reduces the clock frequency while it visits more nodes than the stack method but fewer than the Fano algorithm. If we denote the tree length as N, at most $2N$ nodes need to be stored. The overall algorithm can be explained as follows: move forward in the code tree as far as possible, and when leaving the current sub-tree, store the best node metrics that have been examined but not visited.

Let n_{cur} denote the node which is currently being visited by the decoder. Successors of this node are denoted as n_x and n_y with cumulative metrics μ_x and μ_y, respectively. Without loss of generality, we assume that $\mu_x \geq \mu_y$. We limit the algorithm's return, to only those nodes that are sufficiently promising, that is, those with metrics above the threshold T. These promising nodes are called buds and are stored on a node stack. There are two stacks in this algorithm, the node stack, that stores nodes and the threshold stack, which stores the metric values of selected nodes in the node stack. If both nodes n_x and n_y are considered "good" ($\mu_x \geq T$ and $\mu_y \geq T$), both nodes are stored in the node stack. If only n_x is good, then the decoder moves to this node but the node is not stacked. If both of the nodes are "bad", the decoder moves to the node at the top of the node stack. The nodes on the node stack are referred by the node stack pointer NP. The top node is denoted as $n(NP)$, the second top node as $n(NP-1)$, and so on. In other words, NP is considered as stack pointer where the stack node is denoted by $n(NP-k)$ where k is the location of the node relative to the top of the stack. The threshold stack stores a value $\mu_T(TP-k)$ for each node, although not every node's metric is entered. A T-node is a node on the node stack that has an entry in the μ_T stack. Non-T-nodes, are nodes that do not have an item on the threshold stack. Along with each node in the node stack, a flag is stored indicating whether it is a T-node or not. The flag corresponding to

$n(NP-k)$ is denoted by $F(NP-k)$. The threshold $T = Q(\mu_T(TP-1))$, where $Q(x) = \lfloor \frac{x}{\Delta} \rfloor.\Delta$, is used to determine which nodes are stored. Those nodes with metrics above T, are placed on the node stack.

When a node n_x is found with metric exceeding μ_{max} (μ_{max} denoting the maximum metric among all nodes previously visited), n_x and n_y are stored on the node stack as T-nodes. Otherwise if $\mu_x < \mu_{max}$, n_x and n_y are stored as non-T-nodes on the node stack. We can summarize the creeper algorithm as follows.

1. Initialization : $\mu_{max} \leftarrow -\infty$; $\mu_T(-1) \leftarrow -\infty$; $n(0) \leftarrow root$;
   $TP = NP = 0$; $n_{cur} \leftarrow root$.

2. Next, the successor metrics u_x and u_y and the threshold T are computed and the rules of the creeper algorithm (which are provided in Table 3.1) are applied (see [23] for more details).

3. After the use of Table 3.1, the new successor metrics and the new threshold are computed in the polar code side and the decoder applies one of the rules again.

4. The algorithm terminates when n_{cur} is at the end of the code tree.

We set a threshold of 1300000 node visits in both creeper and Fano algorithm to avoid the time-consuming loops. If the number of visited nodes gets larger than 1300000, we declare an error for that block.

In Fig. 3.1, we compare the performance of PAC codes with creeper and Fano decoders over an AWGN channel, with the code parameters $N = 64$ and $K = 32$. For these results, we take $\Delta = 3$ for both algorithms. Also, ρ (bias) is taken as 1. The result shows that the creeper algorithm has a small degradation in performance compared to the Fano decoder.

Figs. 3.2 and 3.3 illustrate the complexity of the creeper algorithm versus that of the Fano decoder. The results show that complexity of the decoder is decreased

around $\frac{1}{3}$ for higher SNRs with the creeper algorithm. In the lower SNR regime, we may encounter some blocks that are stuck in the forward and the backward loops, so the complexity reduction is not that high. We also note that if Δ is increased, a better complexity performance trade-off is obtained with respect to the Fano decoder. This is because in higher SNRs, in most of the cases, for a wide range of Δ values, the creeper decoder works well with minimum node visits. In fact, with the creeper algorithm, in higher SNRs, the algorithm may not even backtrack to the previously visited nodes, at all.

Table 3.1: Creeper algorithm's rules for each condition.

Condition	Decoder's action
$T \leq \mu_y$ and $\mu_x > \mu_{max}$	$n_{cur} \leftarrow n_x$ $n(NP+1) \leftarrow n_y$ $n(NP+2) \leftarrow n_x$ $\mu_T(TP+1) \leftarrow \mu_y$ $\mu_T(TP+2) \leftarrow -\infty$ $F(NP+1) \leftarrow 1$ $F(NP+2) \leftarrow 1$ $NP \leftarrow NP+2$ $TP \leftarrow TP+2$ $\mu_{max} \leftarrow \mu_x$
$T \leq \mu_y$ and $\mu_x \leq \mu_{max}$	$n_{cur} \leftarrow n_x$ $n(NP+1) \leftarrow n_y$ $n(NP+2) \leftarrow n_x$ $F(NP+1) \leftarrow 0$ $F(NP+2) \leftarrow 0$ $NP \leftarrow NP+2$
$\mu_y < T \leq \mu_x$	$n_{cur} \leftarrow n_x$ $\mu_T(TP) \leftarrow \max(\mu_y, \mu_T(TP))$ $\mu_{max} \leftarrow \max(\mu_x, \mu_{max})$
$\mu_x < T$ and $F(NP-1)=1$ and $max(\mu_x, \mu_T(TP)) \geq Q(\mu_T(TP-3))$	$n_{cur} \leftarrow n(NP-1)$ $\mu_T(TP) \leftarrow \max(\mu_x, \mu_T(TP))$ exchange$(\mu_T(TP), \mu_T(TP-1))$ exchange$(n(NP), n(NP-1))$ exchange$(F(NP), F(NP-1))$ $\mu_T(TP) \leftarrow -\infty$
$\mu_x < T$ and $F(NP-1)=1$ and $max(\mu_x, \mu_T(TP)) < Q(\mu_T(TP-3))$	$n_{cur} \leftarrow n(NP-1)$ $\mu_T(TP-2) \leftarrow \max(\mu_x, \mu_T(TP), \mu_T(TP-2))$ $NP \leftarrow NP-2$ $TP \leftarrow TP-2$
$\mu_x < T$ and $F(NP-1)=0$	$n_{cur} \leftarrow n(NP-1)$ $\mu_T(TP) \leftarrow \max(\mu_x, \mu_T(TP))$ $NP \leftarrow NP-2$

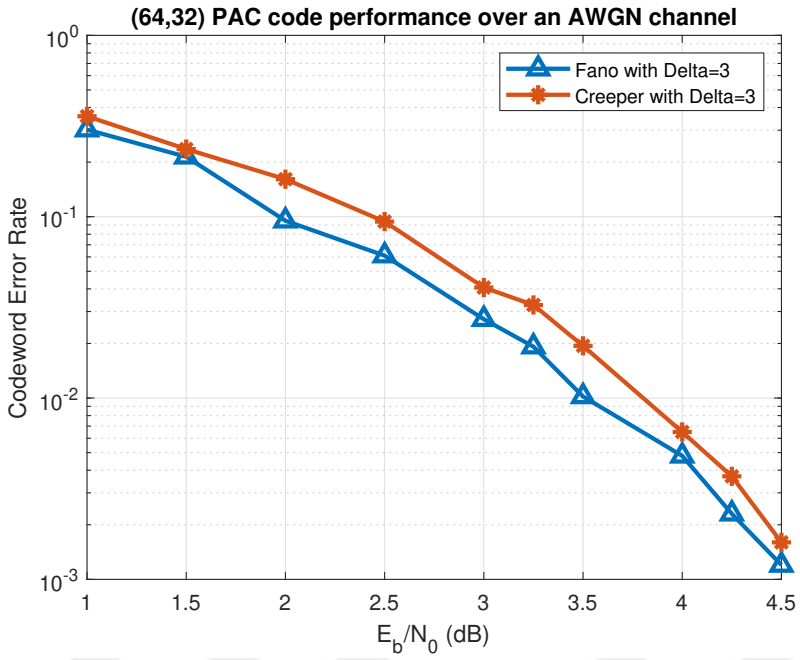

Figure 3.1: BLER performance comparison of the Fano and the creeper decoding over an AWGN channel.

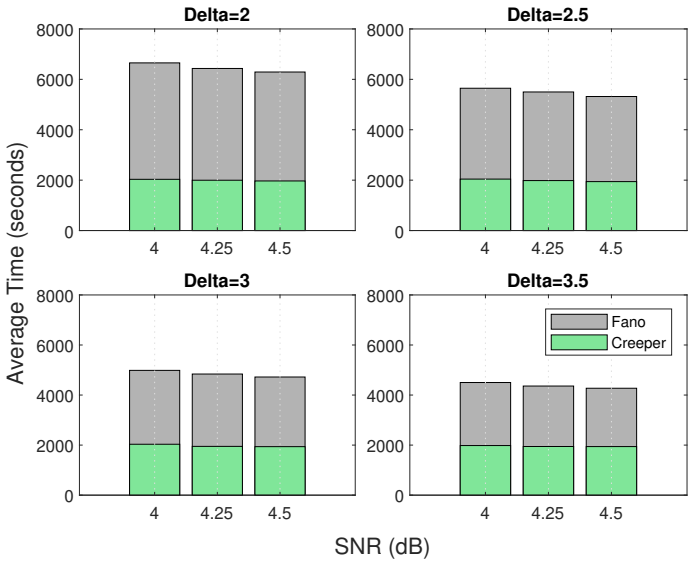

Figure 3.2: Complexity comparison in terms of average running time for decoding a single block in Matlab.

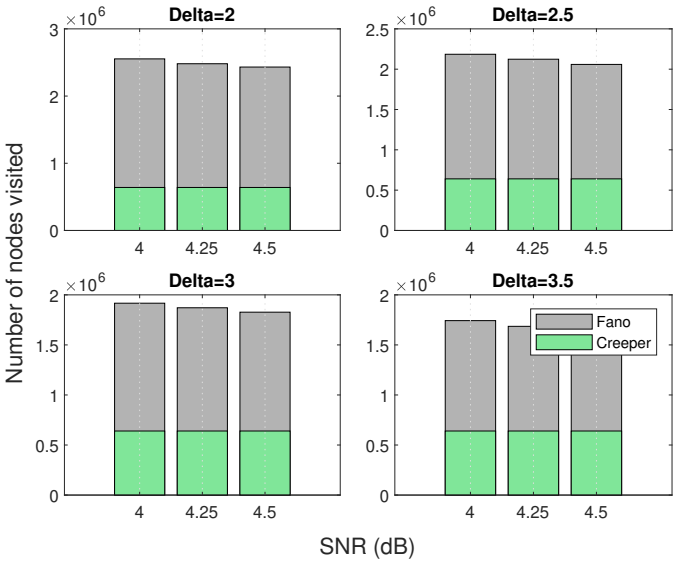

Figure 3.3: Complexity comparison in term of the number of nodes visited.

3.2 A Simplified Successive Cancellation List Decoder for PAC Codes

In this section, we go over a simplified successive cancellation algorithm based on [25] to reduce the decoding complexity of the PAC codes. We summarize this decoding algorithm in four different actions, and an example of those steps are shown by different colors in Fig. 3.4 for a PAC code with $N = 8$. The numbers denotes the number of steps. The SSCL decoder receives the LLR vector and performs the following steps.

- Action Left (L- depicted in red in Fig. 3.4): This is the very beginning step in each newly visited node in successive cancellation tree. Each node receives the LLR vector from its above node. Then the LLR of the current node is obtained by

$$LLR_{left}(LLR_1, LLR_2) = sgn(LLR_1).sgn(LLR_2).min(|LLR_1|, |LLR_2|), \tag{3.1}$$

where LLR_1 denotes the first half of LLR vector of the upper node, i.e.,

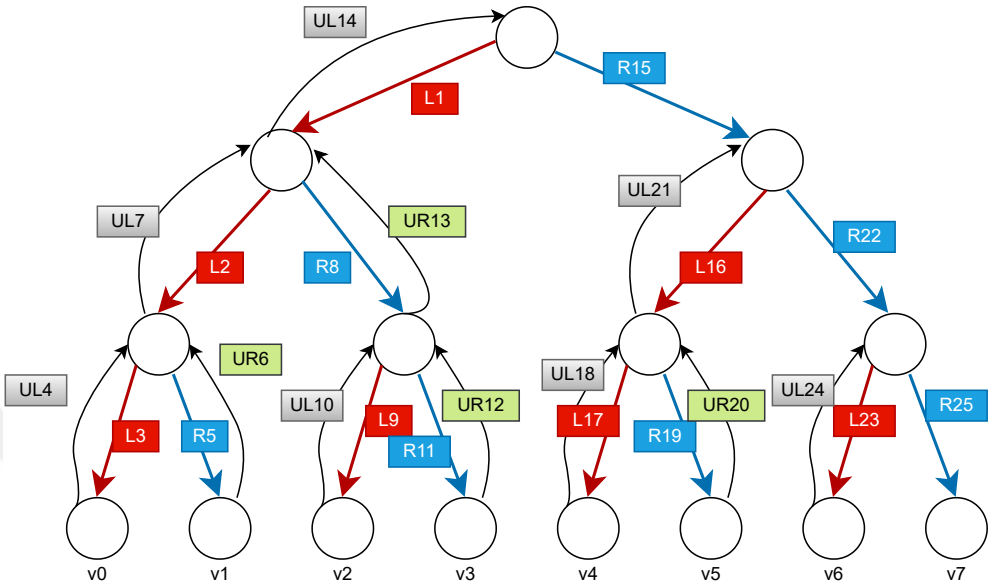

Figure 3.4: Simplified successive cancellation scheme on tree with N=8.

$LLR_1 = LLR(1, 2, ..., 2^{n/2})$, where n is the number of a layer that the upper node belongs to in the successive cancellation tree (see Fig. 2.14). Similarly, $LLR_2 = LLR(2^{n/2}+1, 2^{n/2}+2, ..., 2^n)$ denotes the second half of LLR vector of the upper node. This action continues on the left hand side, and computes LLR of each node until we reach the bottom of the tree. As we reach to the lowest layer in the SC tree, we have a scalar value for the LLR. The node j in the lowest layer either belongs to a bit that is from the frozen set, which we already know the value for v_j and using convolutional transform we can find u_j, or in a non-frozen set which two possible scenarios for u_j are available. After finding u_j, we do the following: If u_j's sign is in line with the corresponding scalar LLR, we are on the right path in the convolutional tree (see Fig. 2.13), otherwise, we should add a penalty to the corresponding path metric. The value of the metric is equal to the absolute value of the scalar LLR, i.e., $|LLR_{left}|$.

- Action Up Left (UL-depicted by gray in Fig. 3.4): In order to compute LLR in the adjacent right hand side node, the decoder goes to the upper node and creates vector $[LLR_1, LLR_2, \boldsymbol{u}_L]$, where LLR_1 and LLR_2 denote the first and second half of the current node's LLR vector and $\boldsymbol{u}_L$ denotes

the $\boldsymbol{u}$ vector computed in the lower node on the left hand side. Note that if the lower node is from the lowest layer, then $\boldsymbol{u}_L$ is an scalar, i.e., $\boldsymbol{u}_L = u_j$.

- Action Right (R- depicted by blue in Fig. 3.4): The LLR on the right hand side is obtained by

$$LLR_{right}(LLR_1, LLR_2, \boldsymbol{u}_L) = LLR_2 + (1 - 2\boldsymbol{u}_L).LLR_1, \tag{3.2}$$

  When we reach the lowest layer of the tree on the right hand side nodes, we either know the value of v_j or we have two possible values for that. We find corresponding u_j for each scenario and compare it with the value of LLR_{right} in the lowest layer. If they are in line with each other, we add no penalty, otherwise we add $|LLR_{right}|$ to the corresponding path metric.

- Action Up Right (UR-depicted in green in Fig. 3.4): When the $\boldsymbol{u}$ vector in the right hand side node is computed, then the $\boldsymbol{u}$ vector of the upper node is obtained by

$$\boldsymbol{u}_{upper} = [\boldsymbol{u}_R + \boldsymbol{u}_L, \boldsymbol{u}_R]. \tag{3.3}$$

In summary, at the beginning of the decoding, we continue to compute the LLR in the left arm of the binary tree until we reach the lowest layer (e.g., layer 3 in the example of Fig. 3.4). Then, if the node in the lowest layer is frozen, we already know the value of v_j, which is 0. With knowledge of v_j, we can find corresponding u_j. If the value of u_j is in line with the sign of the LLR, we add no penalty to the path metric; otherwise, we add the value of the LLR as a penalty. In other words, $PM = |LLR|$ for the cases that u_j does not have the same sign as the LLR; otherwise, we impose no penalty (i.e., $PM = 0$), where PM denotes the path metric. In the beginning, there is a single path in the list. For each specific path in the list, we keep lists of previously calculated u_j values. This simplifies the decoder since, otherwise, the decoder needs to calculate u_j values for each new path. When the number of passes is higher than the size of the list, we prune the paths with a higher path metrics. When we reach the last node of the tree, we pick the path with the lowest path metric in the list.

In Fig. 3.5, the performance of the SSCL decoder with different list sizes

compared to the Fano decoder over an AWGN channel is depicted. The results show that with a list size of 128, we can reach the performance of the Fano decoder. We highlight that since we use the simplified successive cancellation, the average time complexity can be reduced by around 75 % in comparison with the scenario that computes LLRs in the lowest layer for each branch from the beginning.

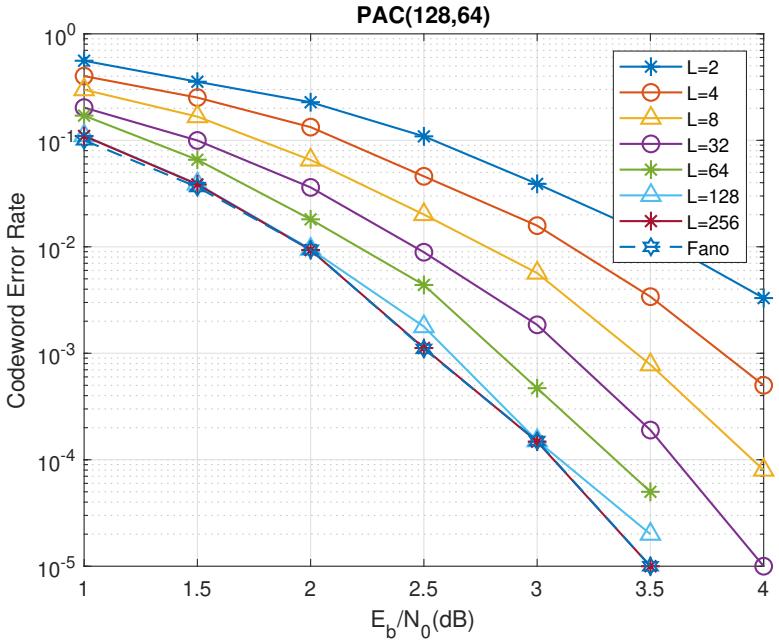

Figure 3.5: Performance of a PAC(128,64) code with the SSCL decoder compared to the Fano decoder with $\Delta = 3$.

3.3 Performance of PAC Codes over BI-AWGN Channels

This section compares the results of PAC and 5G-polar codes with available finite length coding bounds in the literature.

As discussed in the previous part, RM score function can be used at the rate-profiling step of the PAC encoder. When the codeword length in the PAC code

is $128 = 2^7$, we have $\binom{7}{0} = 1$ bit location with Reed-Muller score 0, $\binom{7}{1} = 7$ bit locations with RM score 1, $\binom{7}{2} = 21$ with 2, ... $\binom{7}{7} = 1$ with score 7. When the message length is 29, we can use the one location with a score of 7, seven locations with a score of 6, and 21 locations with a score of 5 as information bit locations. The remaining bit locations have scores less than 5, and they can be assigned as frozen bits. This type of allocation is named as **full rank** allocation. The (128,64) case can also have full rank allocation of bit locations. However, if we have the (128,50) case as an example, we need to pick 21 locations out of $\binom{7}{4} = 35$ bit locations with a score of 3. Random selection or selecting higher indexed bit locations with score 3 may spoil the overall performance of the code.

In Fig. 3.6, we demonstrate the performance of PAC codes compared with those of the standard 5G polar codes over an AWGN channel and the BI-AWGN dispersion approximation. We pick the message lengths as 29, 64 and 99 to allow for full rank bit allocation in the RM score function. For the decoding, we use the SSCL decoder with a list size of 256, which has a performance close to the Fano decoder. For convolutional encoding, the code polynomial 133 in octal notation is chosen. The simulated polar code in this result is the uplink 5G standard polar code with CRC-11, and it is decoded by a list decoder with list size 128.

We observe that, for some of specific cases, the performance of PAC codes are superior than that of the 5G-standardized polar codes, and they are very close to the dispersion approximation. Also, as the rate increases, the distance between the dispersion approximation and polar and PAC code performance gets larger. These results are very similar to those in [26] which uses the Fano decoder; however, with list size 128, the complexity of the decoder is lower than that of the Fano decoder.

In Fig. 3.7, we demonstrate the performance of PAC codes with different polynomials and rate-profiling schemes decoded by SSCL with list size of 256 compared with 5G polar code with CRC-11 and decoded with SCL with list size of 64. The channel is an AWGN channel and BPSK modulation is used. The code rate is $\frac{1}{2}$ with codeword length of 64. In the (64,32) scenario, the codeword length is $64 = 2^6$, and we need to assign 32 bits of the message to 32 unfrozen

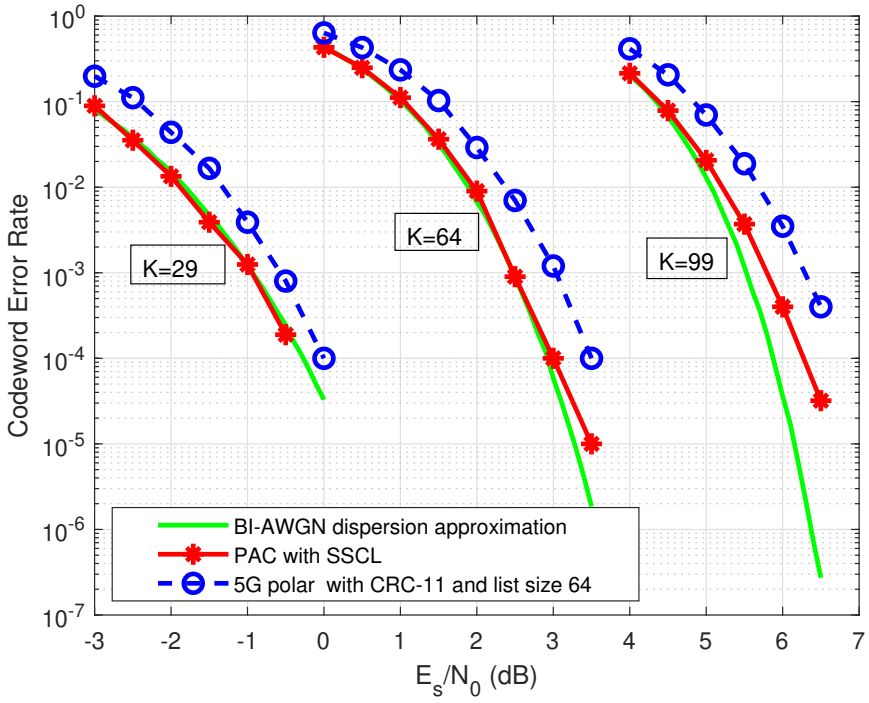

Figure 3.6: Performance of full rank PAC codes and standard 5G polar codes.

bit channel locations. For 32 unfrozen bit locations, we can pick $\binom{6}{6} = 1$ bit with RM score 6, $\binom{6}{5} = 6$ bits with RM score 5, and $\binom{6}{4} = 15$ bits with RM score 4, and we need to pick 10-bit locations with score 3 out of $\binom{6}{3} = 20$ bit locations. So the (64,32) scenario is not full rank. If we pick higher indexed 20-bit locations with this RM score (as prescribed in [22]), we get the result depicted as purple in Fig. 3.7 where it cannot even surpass the performance of 5G polar with CRC-11 in this scenario. So clearly, we need to change the positions of the information (unfrozen bits) positions. With this in mind, we pick the bit locations with higher RM scores first, then we pick bit locations with score 3 with lower indices. With this approach, the performance of the PAC code improves as shown in red in Fig. 3.7. We name this scheme as new Reed-Muller score function in the figure. However, with polynomial 133 and the new RM score, there is a 1 dB difference with the BI-AWGN dispersion approximation. To improve the performance, we try some other popular polynomials such as 177, 1566 and 3211 [26], and we find that the 3211 polynomial works better in this scenario as shown in yellow in Fig. 3.7.

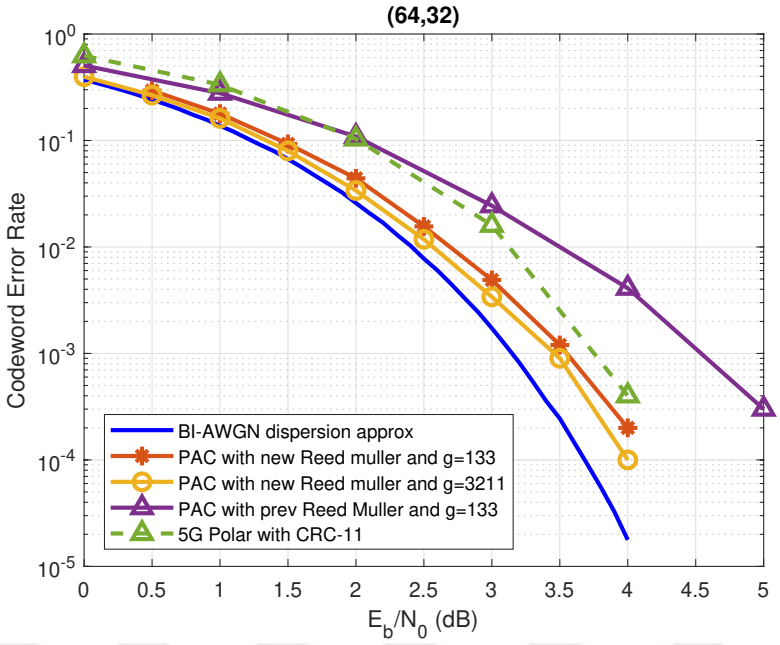

Figure 3.7: Performance of PAC(64,32) codes.

In Fig. 3.8, the performance of a PAC code with two popular convolutional code polynomials in the literature is simulated for the (128,64) PAC code by the SSCL decoder with list size 256. We use the full-rank RM score in design of the rate-profile. This result shows that the 133 polynomial, which has been used in the original paper [8], performs very closely to the BI-AWGN dispersion approximation. So, we confirm that for the (128,64) scenario, the selection of RM score rate-profiling and polynomial of 133 is near optimal.

In Figs. 3.9,and 3.10, we depict the performances of rate $\frac{1}{2}$ PAC codes with codeword lengths 128 and 64, respectively, decoded with a SSCL decoder with list size 256, and 5G polar code decoded with a SCL decoder with list size 128 over an AWGN channel. Meta-converse bound results [3] generated with the saddle point approximation (see [13], [27] and [28] for details) are also depicted. The meta-converse $O(n^{-2})$ asymptotic expansion is also given in [13], which is depicted in both figures, while the normal approximation computation is described in Sub-section 2.1. Random Coding Union (RCU) bound is also given in the figures for a more thorough comparison. In conclusion, we have a near optimal performance from the PAC codes in the (128,64) scenario (Fig. 3.9), while there is room for

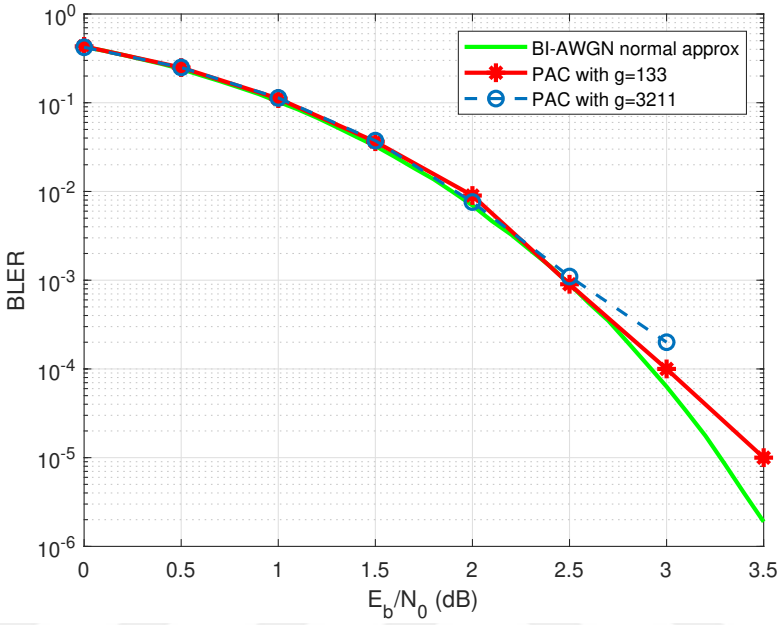

Figure 3.8: Performance of PAC(128,64) with different polynomials.

improvement for the (64,32) case.

3.4 Performance of Punctured PAC Codes

In this section, we present several performance results with higher rate PAC codes obtained via puncturing. Puncturing is the process of removing some of the bits after encoding with an error-correction code. This has the same effect as encoding with an error-correction code with a higher rate or less redundancy. However, with puncturing, the same decoder can be used regardless of how many bits have been punctured; therefore, it considerably increases the flexibility of the system without significantly increasing its complexity.

Just like polar codes, PAC codes also have the native length of power of two, i.e., $N = 2^n$. This enables us to benefit from high rates with a fixed message length. Also in many applications, it is necessary to adjust the code length to some desired value other than 2^n.

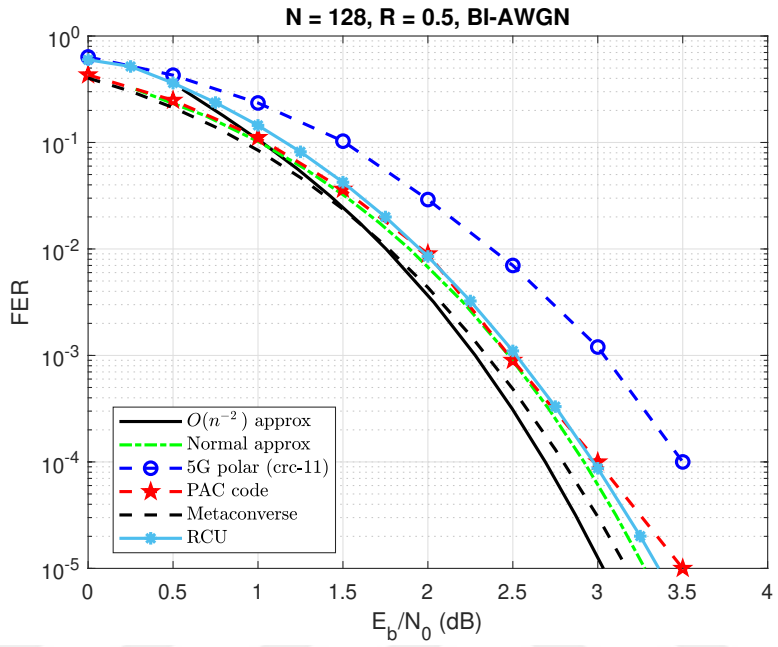

Figure 3.9: Performance of PAC(128,64) compared to the theoretical bounds.

For puncturing the PAC code, we first encode the message bits, then we remove k random bits out of N encoded bits, and send a sequence with a length of $N-k$. At the receiver side, we take the observations of punctured bits as 0 and employ the same decoder as before.

Figs. 3.11-3.14 show the performance of punctured PAC codes with different blocklengths. Here, we puncture 8, 18, and 28 bits out of 128 coded bits. In Fig. 3.11, we compare the different punctured PAC codes in terms of frame error rates as a function of E_b/N_0. Figs. 3.12 to 3.14 depict the performance of punctured PAC codes compared to the corresponding BI-AWGN dispersion approximation. All of the message lengths are 64. It can be seen that as the number of punctured bits increases, the difference between the PAC code performance and the dispersion approximation increases as well.

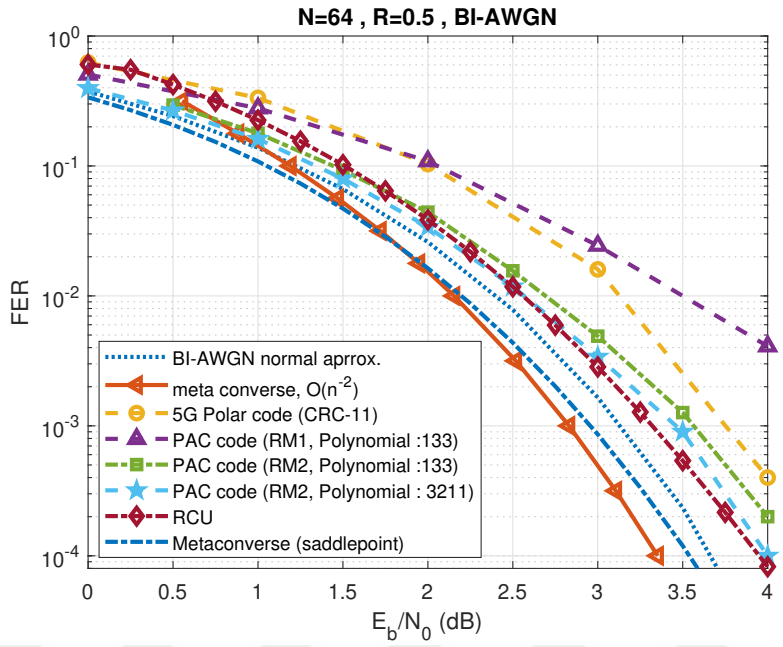

Figure 3.10: Performance of PAC(64,32) compared to the theoretical bounds.

3.5 PAC Code Performance over Rayleigh Block Fading Channels

This section investigates the performance of PAC codes over RBF channels in which the channel coefficients remain constant for a block of T consecutive symbols and change to an independent realization in the next block. The parameter T can be considered as the channel coherence time, or more generally, the number of time-frequency slots over which the channel stays constant. So a codeword of length $N = BT$ spans B independent channel realizations. Before presenting the simulation results, we obtain a normal approximation for the known CSI case.

3.5.1 Normal Approximation for Block Fading Channels

The system model is given by

$$y_b = h_b x_b + z_b, \tag{3.4}$$

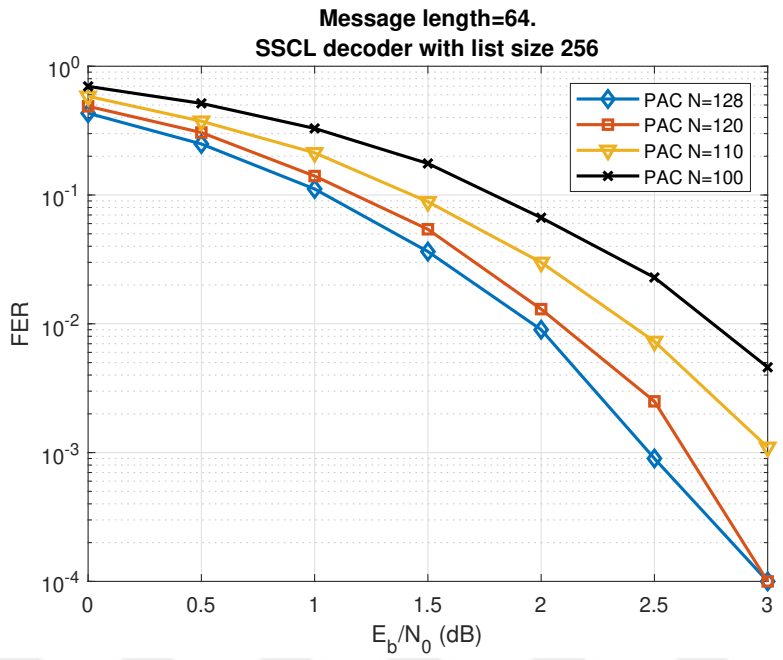

Figure 3.11: Performance of punctured PAC codes.

where h_k is the channel gain for the kth block ($k = 1, 2, ..., B$), x_k is the modulated (possibly bit interleaved) codeword and z_k is the Gaussian noise with a zero mean and variance σ^2. We first consider the case where h_k is perfectly known at the receiver side. To achieve more reasonable results, we modulate the coded data by $\frac{\pi}{4}$-QPSK modulation as depicted in Fig. 3.15. We note the following:

- $E_b = \dfrac{E(|x|^2)}{R_c} = \dfrac{2}{R_c}$ for $x = \{1 + j, 1 - j, -1 + j, -1 - j\}$.
- $\rho_k = \dfrac{E_b}{N_0} = \dfrac{2}{R_c N_0} = \dfrac{1}{R_c \sigma^2}$, is the instantaneous SNR over kth block.
- $z_t = \sigma(z_r + j z_i)$ where $z_r, z_i \sim \mathcal{N}(0, 1)$.
- $h_t = \dfrac{1}{\sqrt{2}}(h_r + h_i)$ where $h_r, h_i \sim \mathcal{N}(0, 1)$.

where R_c is the effective rate.

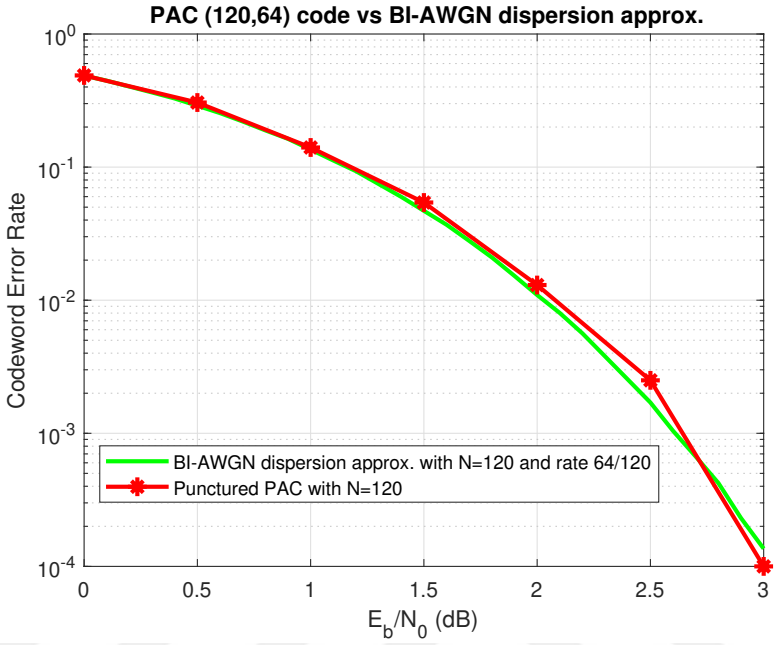

Figure 3.12: Punctured PAC code with blocklength 120 in BI-AWGN channel.

The AWGN channel can be modeled by (3.4) with $h_t = 1$ and the normal approximation for a AWGN channel is

$$R = C - \sqrt{\frac{V}{N}} Q^{-1}(P_e),$$

in nats, where the P_e is the BLER.

For the RBF channels, there are B channel coefficients $h_1, h_2, ..., h_B$, so we determine the average of the normal approximation over the block gains to obtain dispersion approximation of the RBF channel, i.e.,

$$P_{e,avg} = E_{[h_1,h_2,...,h_B]} \left[Q \left(\frac{C - R_c}{\sqrt{\frac{V}{N}}} \right) \right]. \tag{3.5}$$

Note that, we can regard B blocks of Rayleigh fading channels, as B parallel channels. Channel dispersion and channel capacity of B parallel channels are

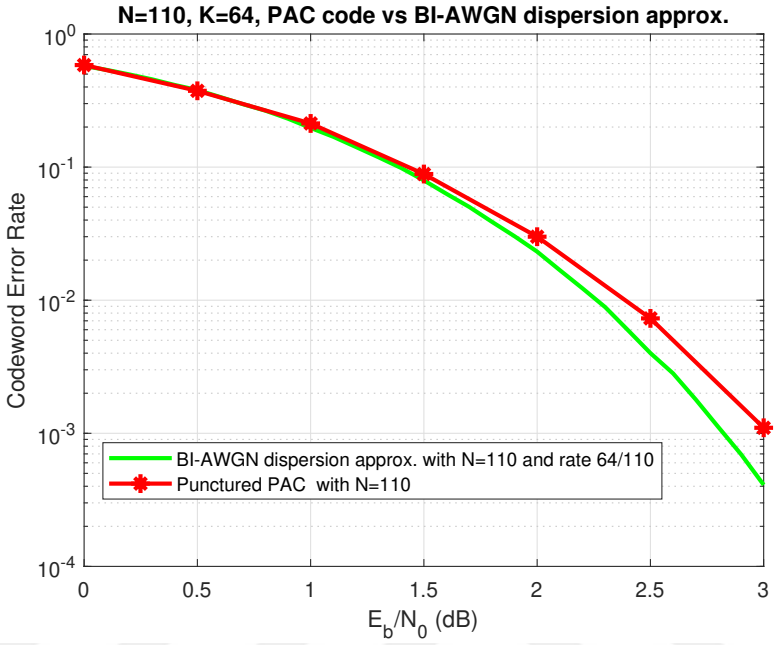

Figure 3.13: Punctured PAC code with blocklength 110 in BI-AWGN channel.

given in [13] as

$$C = \frac{1}{B}\sum_{k=1}^{B} log(1+\rho_k). \tag{3.6}$$

$$V = \frac{1}{B}\sum_{k=1}^{B} \frac{\rho_k(\rho_k+2)}{(\rho_k+1)^2}\log^2(e)\,. \tag{3.7}$$

So using (3.5-3.7), we can calculate the normal approximation of the Rayleigh block fading channel for any given B.

3.5.2 Performance of PAC Codes over Rayleigh Block Fading Channels with CSIR

In this subsection, we investigate the performance of PAC codes over Rayleigh block fading channels with channel state information known to the receiver.

In order to decode a PAC code modulated by QPSK and passed through Rayleigh block fading channel, we need to calculate the LLR, which is the input

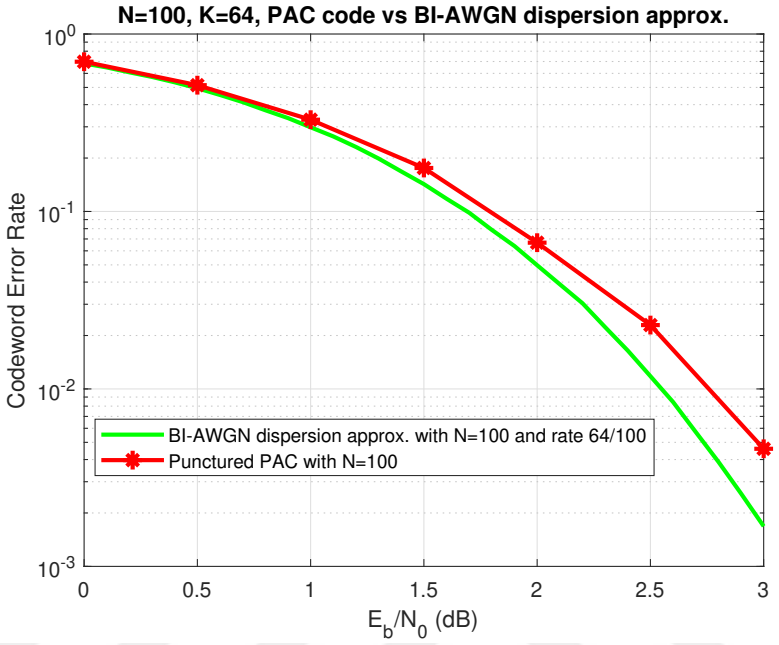

Figure 3.14: Punctured PAC code with blocklength 100 in BI-AWGN channel.

to the PAC decoder (SSCL here). Here, the channel information is assumed known at the receiver side. Let us assume $x = b_r + j.b_i$ where $b_r, b_i = \{\pm 1\}$ and h can be written as $h_t = h_r + jh_i$. Also let us assume $y = y_r + j.y_i$. So the LLR for b_r, ℓ_{b_r}, and b_i, ℓ_{b_i} are written as

$$\ell_{b_r} = \ln \frac{P(y|b_r = 1, b_i, h_t)}{P(y|b_r = -1, b_i, h_t)} \tag{3.8}$$

$$= \ln \frac{\sum_{t=0}^{1} P(y|b_r = 1, b_i = 2t - 1, h_t)}{\sum_{t=0}^{1} P(y|b_r = -1, b_i = 2t - 1, h_t)}. \tag{3.9}$$

The conditional probability density function (PDF) of y given x and h_t is:

$$P_{Y|X,h_t}(y|x, h_t) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|y - h_t x|^2}{2\sigma^2}\right), \tag{3.10}$$

where

$$|y - h_t x| = |y_r - h_r b_r + h_i b_i + j(y_i - h_i b_r - h_r b_i)|. \tag{3.11}$$

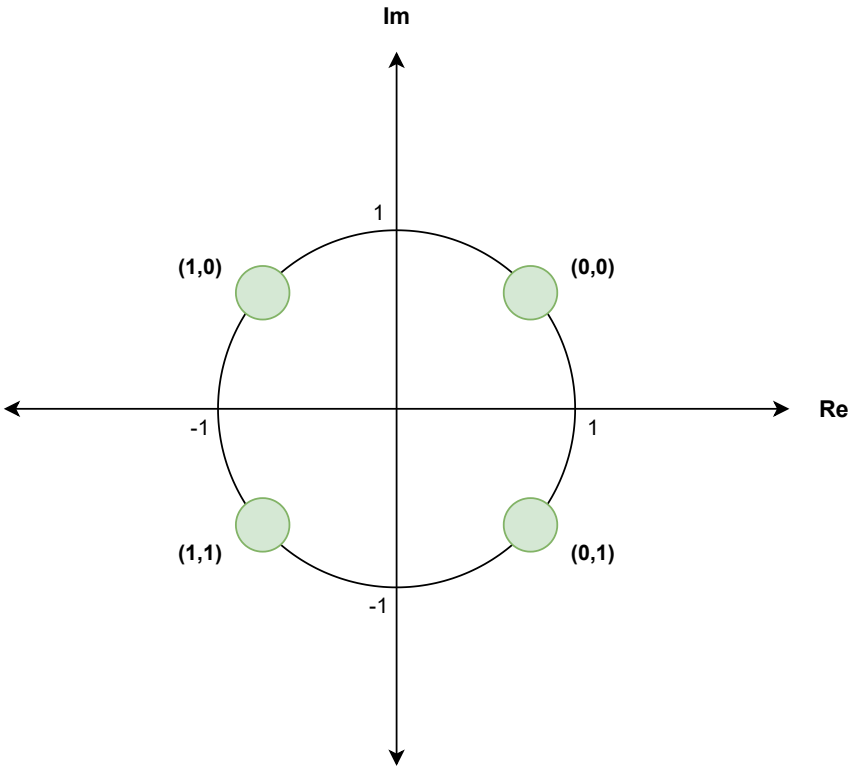

Figure 3.15: QPSK modulation set up used in our results.

Therefore, by putting corresponding values in (3.9), we calculate LLR for b_r as

$$\ell_{b_r} = \ln \frac{\exp(A) + \exp(B)}{\exp(C) + \exp(D)}, \tag{3.12}$$

where

$$A = -\frac{|y_r - H_r + H_i + j(y_i - H_i - H_r)|^2}{2\sigma^2}, \tag{3.13}$$

$$B = -\frac{|y_r - H_r - H_i + j(y_i - H_i + H_r)|^2}{2\sigma^2}, \tag{3.14}$$

$$C = -\frac{|y_r + H_r + H_i + j(y_i + H_i - H_r)|^2}{2\sigma^2}, \tag{3.15}$$

$$D = -\frac{|y_r + H_r - H_i + j(y_i + H_i + H_r)|^2}{2\sigma^2}. \tag{3.16}$$

Similarly,

$$\ell_{b_i} = \ln \frac{\exp(A) + \exp(C)}{\exp(B) + \exp(D)}. \tag{3.17}$$

Figs. 3.16-3.19 depict the FER of PAC code in RBF channel with the SSCL decoder and list size 256 and the performance of the TBCC decoded by WAVA with two memory sizes $m = 8$ and $m = 11$ for $N = 128$ and $K = 64$ with different block sizes, using QPSK modulation. For the rest of the figures, since 5G polar code performance is close to that of TBCC with $m = 8$, we just show the performance of the latter code.

In Fig. 3.16, we depict the performance of PAC codes compared with 5G-standardized polar with CRC-11, and TBCC with WAVA decoder over quasi-static Rayleigh fading channel ($B = 1$). The results show that over this channel, the performance of all of the channel coding schemes under study are very close to each other, and also to the dispersion approximation.

For fading block sizes higher than 1 ($B > 1$) in Figs. 3.17-3.19, we also depict the FER of a PAC code with bit interleaved coded modulation (BICM) where bits are interleaved before modulation; therefore, consecutive bits will experience different fading blocks. As shown, the BICM with PAC codes modulated with QPSK and with an SSCL decoder (list size 256) has a better performance compared to the other coding schemes. Also we can figure out that as B increases the FER of the codes deviate from the corresponding dispersion approximation.

In Fig. 3.19, the performance of various channel coding schemes is compared over the independent and identically distributed (i.i.d.) Rayleigh fading channel ($B = 64$ with QPSK modulation). PAC code with BICM and RM rate profiling gives around $0.5dB$ of improvement compared to TBCC with $m = 11$.

Figs. 3.20 and 3.21 depict the frame error rate of PAC and BICM PAC with SSCL with list size 128 compared to that of the TBCC decoded with WAVA with memories $m = 8$ and $m = 11$ using QPSK modulation over RBF channel with $B = 1$ and $B = 2$, respectively. For PAC rate-profiling we utilize the RM rate

profiling where we pick lower indices of those bit-locations that have same RM scores. This figure shows that PAC code with BICM has the best performance among the other codes and is close to the dispersion approximation. These results show that the utilized rate profiling performs well over RBF channels.

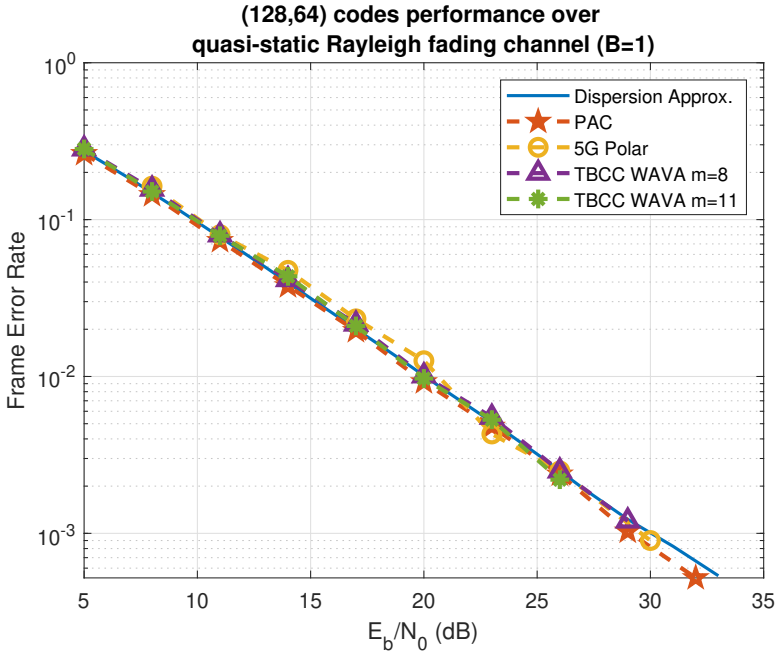

Figure 3.16: PAC, 5G polar and tail-biting code performance over a quasi-static Rayleigh fading channel.

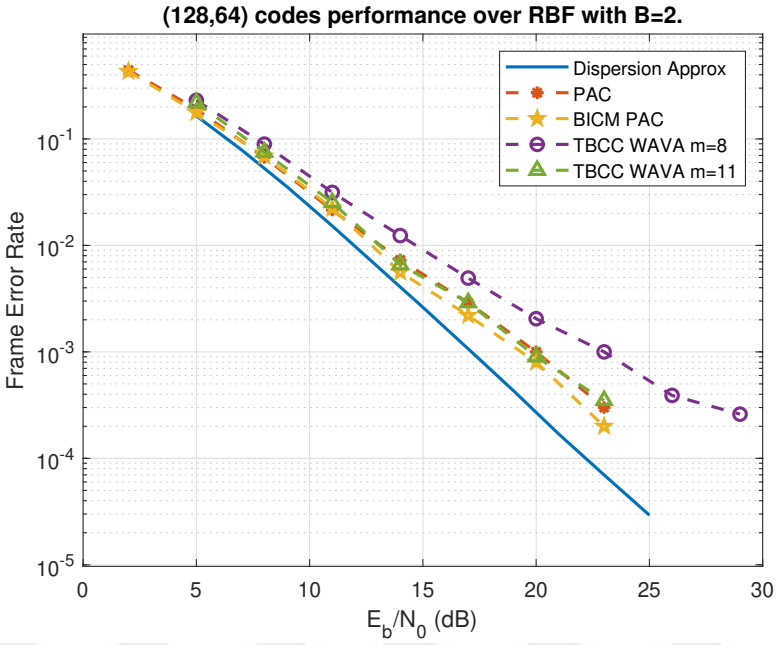

Figure 3.17: PAC code and TBCC performances over a Rayleigh block fading channel with $B = 2$.

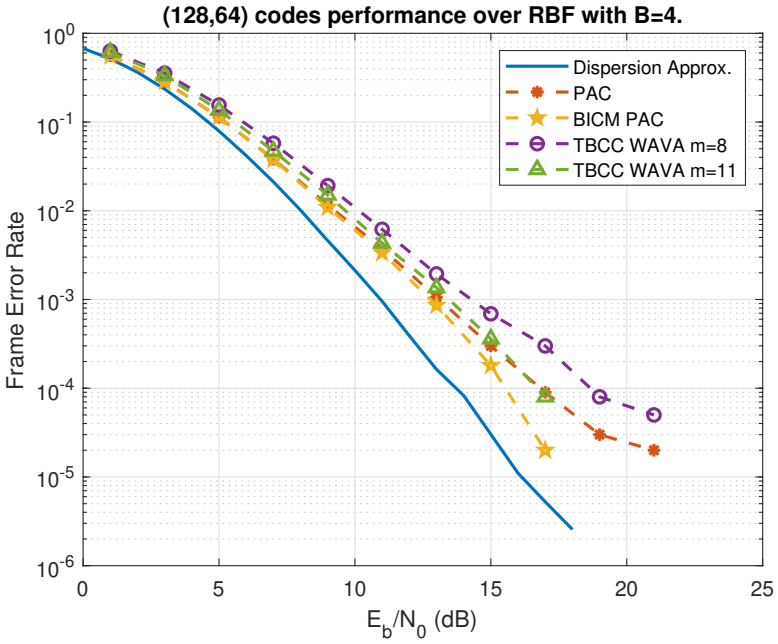

Figure 3.18: PAC code and TBCC performance over a RBF channel with $B = 4$.

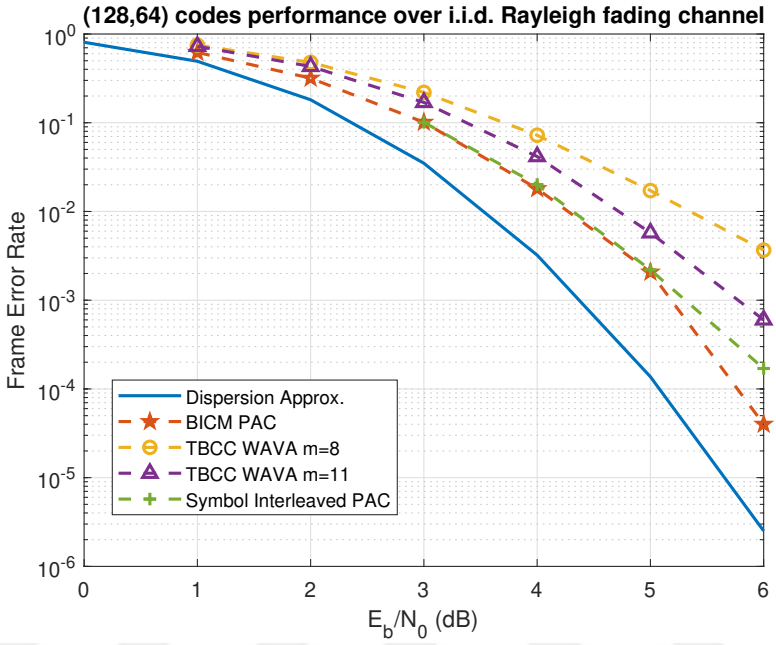

Figure 3.19: PAC code performance over an i.i.d. Rayleigh fading channel.

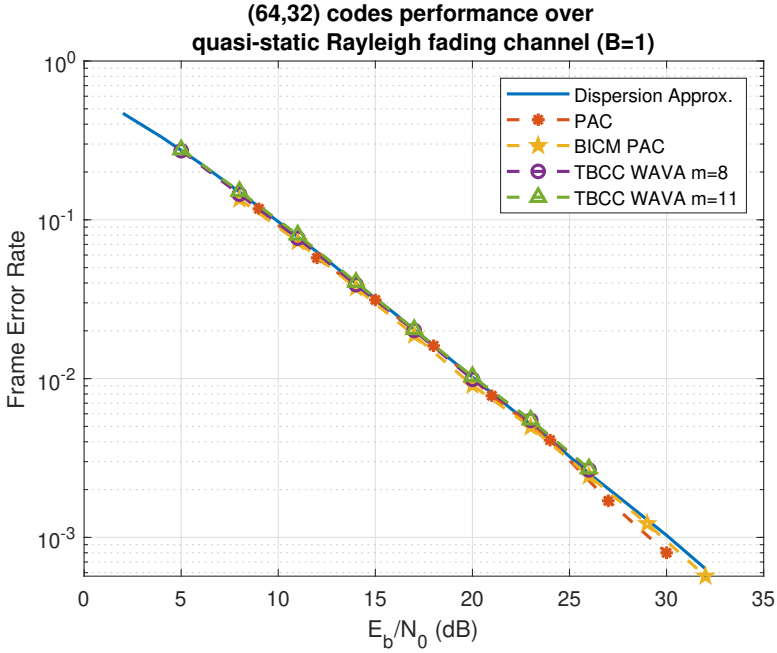

Figure 3.20: (64,32) PAC performance over a quasi-static Rayleigh fading channel.

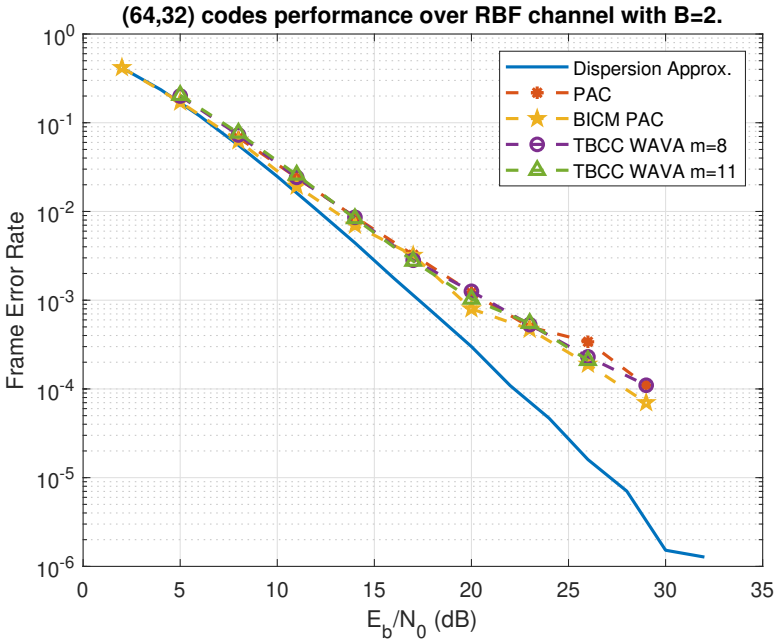

Figure 3.21: (64,32) PAC performance over a RBF channel with $B = 2$.

3.5.3 Performance of PAC Codes for Rayleigh Block Fading Channel with no CSIR

In this section, we investigate the performance of the PAC codes over Rayleigh block fading channels without CSIR, this time, that is the channel gain $h_t \geq 0$ is unknown to the receiver in this scenario. However, the receiver knows that the channel fading is a Rayleigh distributed. We calculate the LLR for this scenario for QPSK modulation based on [29] and [30] as follows:

$$\ell^{(i)} = \ln \frac{p(y|b^i(x) = 0)}{p(y|b^i(x) = 1)}, \tag{3.18}$$

$$= \ln \frac{\sum_{x \in X_0^i} p(y|x)}{\sum_{x \in X_1^i} p(y|x)}, \tag{3.19}$$

where

$$p(y|x) = \int_0^\infty \frac{1}{2\pi\sigma^2} e^{-\frac{|y - h_t x|^2}{2\sigma^2}} p_{h_t}(h_t) dh_t. \tag{3.20}$$

For an M-ary QAM constellation, assuming a normalized Rayleigh fading channel with no CSI at the receiver, we have the following conditional PDF of the received complex signal $y = y_r + jy_i$ given that the symbol $x = x_r + jx_i$ is transmitted [29]:

$$p(y|x) = \frac{1}{\sigma\alpha^3}\exp\left(-\frac{\gamma^2}{2\sigma^2\alpha^2}\right)\left(\frac{\beta}{\sqrt{2\pi}}\exp\left(\frac{b_r b_i y_r y_i}{\sigma^2\alpha^2}\right)\operatorname{erfc}\left(-\frac{\sqrt{2}\beta}{2\alpha\sigma}\right) + \frac{\alpha\sigma}{\pi}\exp\left(-\frac{b_r^2 y_r^2 + b_i^2 y_i^2}{2\sigma^2\alpha^2}\right)\right), \quad (3.21)$$

where

$$\gamma^2 = b_r^2 y_i^2 + b_i^2 y_r^2 + 2\sigma^2(y_r^2 + y_i^2), \quad (3.22)$$

$$\alpha = \sqrt{b_r^2 + b_i^2 + 2\sigma^2}, \quad (3.23)$$

$$\beta = b_r y_r + b_i y_i. \quad (3.24)$$

By substituting (3.21) in (3.18) for QPSK (4-QAM) values, we get the desired LLR calculations.

Figs. 3.22 to 3.25 provide comparison of PAC codes with and without CSI at the receiver with different number of fading blocks. We simulate (128,64) PAC code on $B = 1$, quasi-static case, $B = 2$, $B = 4$ and $B = 64$, which is the i.i.d. case. In all of the mentioned simulations we use QPSK modulation. Whenever the B is small, the performance of the no-CSI case is very close to that of CSIR case. As B is increased, the performance of the no-CSI case deteriorates significantly compared to the CSIR case.

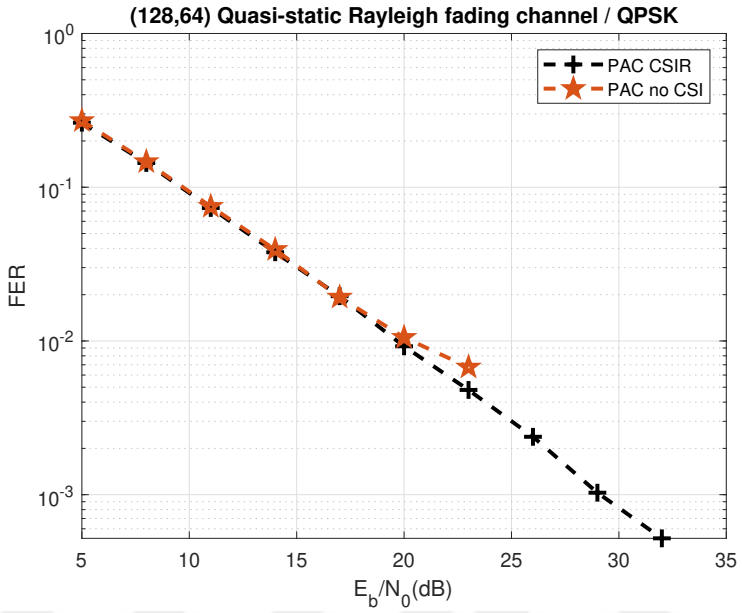

Figure 3.22: PAC code performance with CSI and no-CSI at the receiver over quasi-static Rayleigh fading.

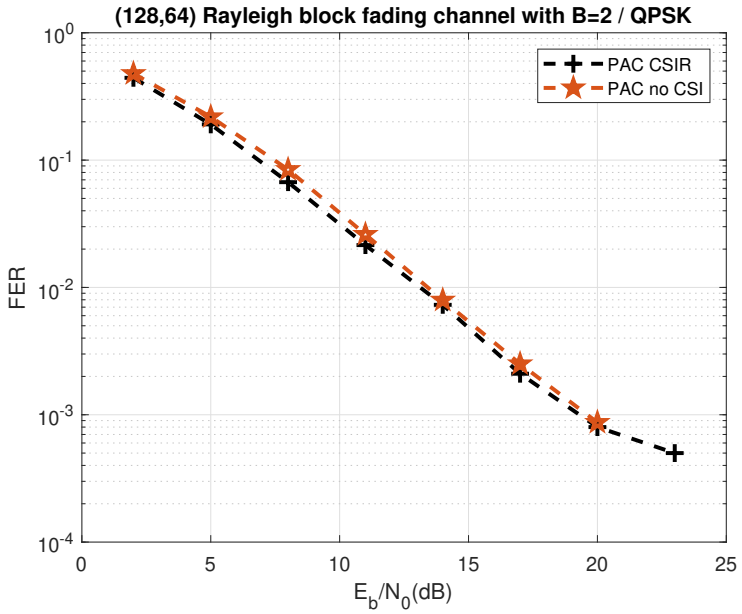

Figure 3.23: PAC code performance with CSI and no-CSI at the receiver over an RBF channel with $B = 2$.

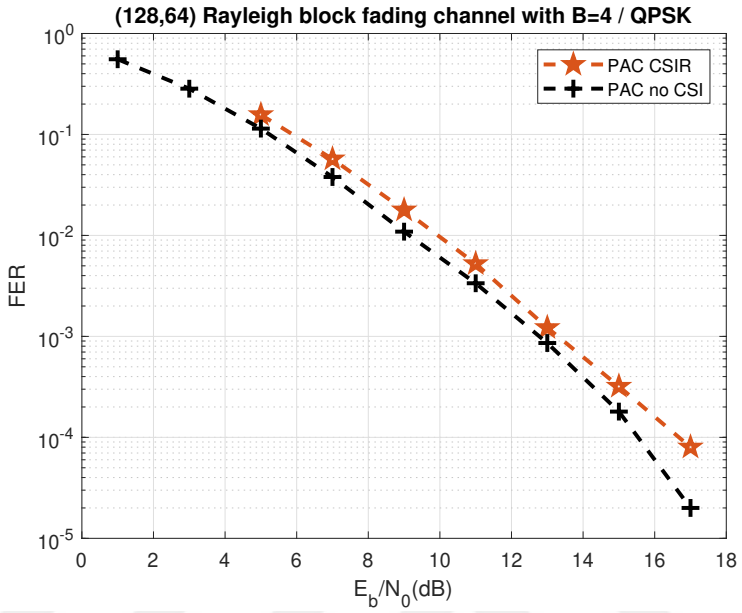

Figure 3.24: PAC code performance with CSI and no-CSI at the receiver over an RBF channel with $B = 4$.

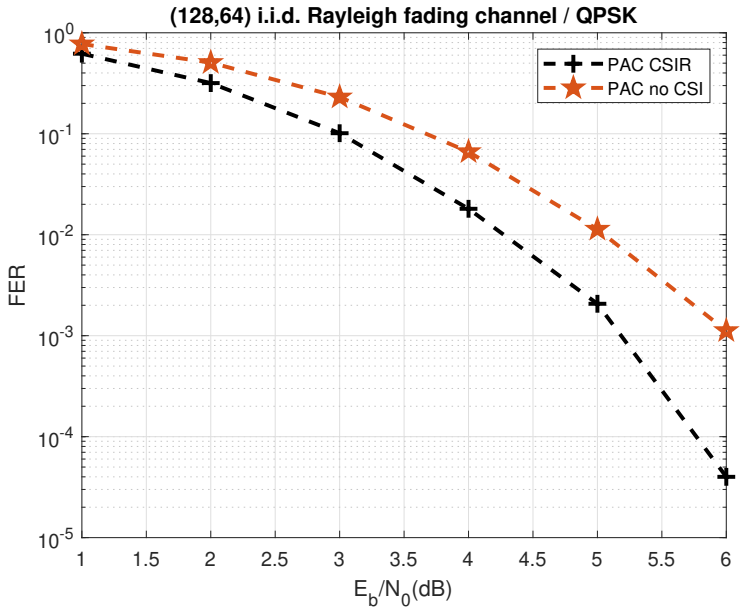

Figure 3.25: PAC code performance with CSI and no-CSI at the receiver over i.i.d. Rayleigh fading channel

3.6 Chapter Summary

In this chapter, we have considered the performance of PAC codes for different scenarios. In the first part of this chapter, we described decoding of PAC codes with creeper and SSCL decoding algorithms. Furthermore, we compared the codeword error rate performance of PAC codes over AWGN channels with the finite length approximation results. We observed that the PAC codes offer better performance than the 5G-standardized polar codes in the scenarios in which full-rank RM rate profiling is applicable. Moreover, we presented several results for punctured PAC codes, which show that we can obtain higher code rates without a significant degradation in the performance. We also investigated the performance of PAC codes over block fading channels with known CSI at receiver and without it. To summarize, the simulation results demonstrate that the PAC codes perform better than the other channel coding methods in short blocklength regime.

Chapter 4

Approximate Weight Distribution of PAC Codes and Rate Profile Design

The weight distribution of an error-correcting code plays a significant role in analyzing the error correction capabilities of the code. As polar codes are of great practical importance, it is of high value to develop their weight distribution. With this motivation, in [31], the authors conduct an algebraic study of polar codes, and present an explicit formula for the number of codewords of minimal weight. A probabilistic computation method is proposed in [9], and an improved version of this method is given in [10]. In [11] a deterministic solution to find weight enumerating function of polar codes which is effective with codes of length less than or equal to 128 is proposed.

As a main contribution of this thesis, in this chapter, we develop a method for approximating the weight distribution of PAC codes in a probabilistic fashion. We demonstrate that the results well match the exact weight distributions for small codes that can be computed using a brute-force algorithm. We also present a way employing the results (along with the union bound on the code performance) to design specific PAC codes, more precisely, to determine suitable rate profiles via

simulated annealing. Numerical examples illustrate that the PAC codes with the designed rate profiles offer superior performance.

4.1 Probabilistic Weight Distribution of Polar Codes

In [9], an enhanced probabilistic computation method for calculating weight distribution of polar codes has been introduced, which utilizes both recursive calculations and exhaustive search. This method is built on the idea that probabilistic weight distribution (PWD) of a polar code with length N can be obtained using the PWD of its two sub-codes with length $N/2$, and this computation, the estimation can be done in a recursive manner.

For a polar code with length $N = 2^n$, we denote the information and frozen bit vector by $u_1^N = \{u_1, ...u_N\}$, and the information and frozen sub-channels with $\mathcal{A}$ and $\mathcal{A}^c$, respectively. The polar codeword is obtained as

$$x_1^N = u_1^N \times G_N = \mathbf{u}_{\mathcal{A}}\mathbf{G}_{\mathcal{A}}, \tag{4.1}$$

where $G_N = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n}$ where $\otimes n$ is nth Kronecker multiplication and $\mathbf{G}_{\mathcal{A}}$ is composed of rows in G_N with indexes in $\mathcal{A}$. Based on the special characteristics of polar generator matrix, the codeword can be rewritten as:

$$x_1^N = \left\{x_1^{N/2}, x_{N/2+1}^N\right\} = \left\{\mathbf{x^0} \oplus \mathbf{x^1}, \mathbf{x^1}\right\}, \tag{4.2}$$

where $\mathbf{x^0}$ and $\mathbf{x^1}$ are equal to $u_1^{N/2}G_{N/2}$ and $u_{N/2+1}^N G_{N/2}$, respectively. Therefore, $\mathbf{x^0}$ and $\mathbf{x^1}$ are the polar codewords with length $N/2$ and the information index set $\mathcal{A}_0 = \left\{i \in \mathcal{A} | i \leq \frac{N}{2}\right\}$, $\mathcal{A}_1 = \left\{i \in \mathcal{A} | i > \frac{N}{2}\right\} - \frac{N}{2}$, respectively. Hence, the Hamming weight of x_1^N is calculated as

$$w\left(x_1^N\right) = w\left(\mathbf{x^0} \oplus \mathbf{x^1}\right) + w\left(x^1\right), \tag{4.3}$$

where $w(\boldsymbol{x})$, denotes the Hamming weight of polar codeword $\boldsymbol{x}$.

Define the weight distribution of a code as

$$A(Z) = \sum_{d} A_d Z^d, \tag{4.4}$$

where A_d is is the number of codewords with weight d. When the number of information bits is K, the total number of codewords is 2^K. Let $P^{(d)}$ be the the probability that a randomly selected codeword has weight d. Clearly, $P^{(d)} = A_d/2^K$, i.e.,

$$A(Z) = \sum_{d} P^{(d)} 2^K Z^d. \tag{4.5}$$

Let $P^{(d)}[\mathcal{A}, n]$ denotes the probability that polar codeword with length 2^n and information set $\mathcal{A}$ has weight d. This quantity can be obtained based on (4.3), by

$$P^{(d)}[\mathcal{A}, n] = \sum_{d_1} \{P(w(x^1) = d_1 | \mathcal{A}_1) \times P(w(x^0 \oplus x^1) = d - d_1 | \mathcal{A}_0, w(x^1) = d_1)\} \tag{4.6}$$

$$= \sum_{d_1} \sum_{d_0} p^{(d_1)}[\mathcal{A}_1, n-1] \times p^{(d_1)}[\mathcal{A}_0, n-1] \times f_n(t, d_0, d_1) \tag{4.7}$$

with $t = d - d_1$ and

$$f_n(t, d_0, d_1) = P\left(w\left(x^0 \oplus x^1\right) = t | w(x^0) = d_0, w(x^1) = d_1\right) \tag{4.8}$$

That is to say, the HWD of polar code with length N can be recursively obtained through that of polar code with length $N/2$ using (4.5).

The only obstacle to complete (4.19) is the computation of $f_n(t, d_0, d_1)$. If we assume that elements in the random binary vectors $x^0, x^1 \in \{0,1\}^{N/2}$ are independently and uniformly distributed, the above probability can be calculated

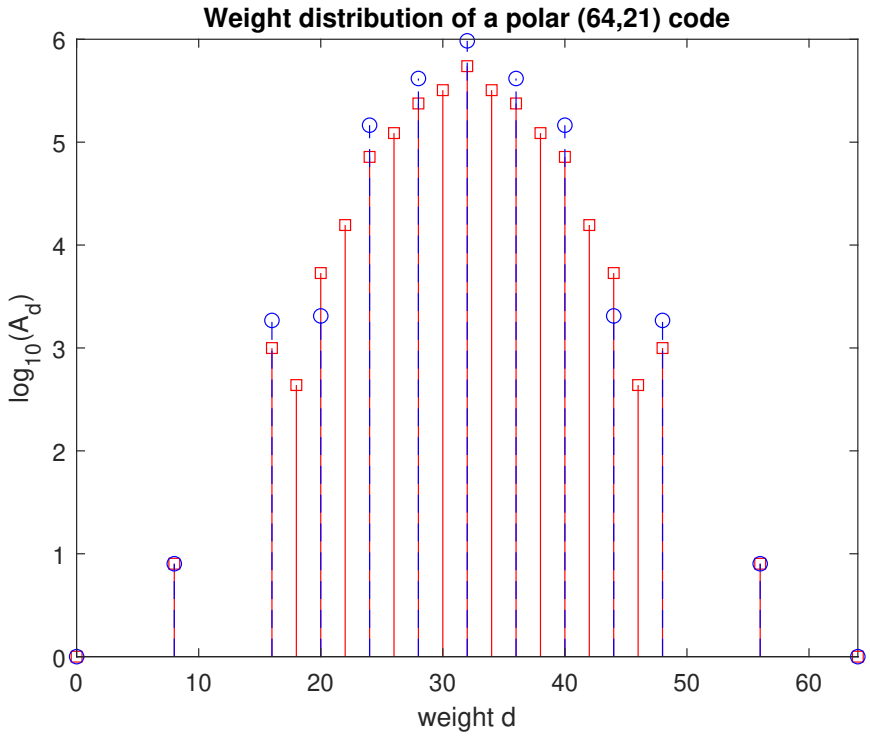

Figure 4.1: Exact (generated by exhaustive search) and approximate weight distribution.

as:

$$f_n(t,d_0,d_1) = \begin{cases} 0 & t < |d_0 - d_1| \ \ or \ \ t > min\{d_0 + d_1, N - d_0 - d_1\} \\ \binom{d_0}{m}\binom{N/2-d_0}{d_1-m}/\binom{N/2}{d_1}, & otherwise \end{cases} \tag{4.9}$$

where m is the number of positions at which the elements in x^0 and x^1 are both equal to 1. m can be calculated as $m = (d_0 + d_1 - t)/2$.

As an example, in Fig. 4.1, we depict the approximate weight distribution of a (64,21) polar code with the proposed scheme along with the exact one obtained via a brute-force search. The result shows that the approximation results are close to the exact weight distribution.

4.2 Deterministic Weight Distribution of Polar Codes

In this section, we review a recent paper that presents a deterministic algorithm to produce the weight enumerating function (WEF) of polar codes, and also gives some clues about finding WEF of polar codes with dynamically frozen bits (which may be used to find WEF of the PAC codes) [11]. The algorithm is based on a recursive procedure to compute the WEF of certain polar co-sets in successive cancellation decoding. To compute the full weight distribution of a polar code, the algorithm first presents the code as a disjoint union of certain polar co-sets, then sums the WEFs of all these co-sets.

In this part, bold letters, e.g., $\boldsymbol{u}$ denote vectors, and non-bold letters, e.g., u_i indicate symbols within that vector. $\boldsymbol{u}_i$ represents $(u_0, u_1, ..., u_i)$. Also, we use $\boldsymbol{u}_{even}$ and $\boldsymbol{u}_{odd}$ to indicate subvectors of $\boldsymbol{u}$ with only even and odd indices, respectively. Also, we use $\boldsymbol{u}_{i,even}$ and $\boldsymbol{u}_{i,odd}$ to denote the sub-vectors of $\boldsymbol{u}_i$ with only even indices and only odd indices, respectively. We use ψ to denote the empty vector.

For a polar code $\boldsymbol{C}$ with at least one frozen bit, let us denote its last frozen bit by u_s, and denote the unfrozen bits before u_s as **red bits**. Then $\boldsymbol{C}$ can be represented as a disjoint union of the following polar cosets:

$$\boldsymbol{C} = \bigcup_{\substack{u_i\, red : u_i \in \{0,1\} \\ u_i\, frozen : u_i = 0}} C_n^{(s)}(\boldsymbol{u}_{s-1}, u_s = 0).$$

Denote by $A_{\boldsymbol{C}}(X)$ the WEF of the entire code $\boldsymbol{C}$, then $A_{\boldsymbol{C}}(X)$ is the sum of the WEF for those cosets, namely,

$$A_{\boldsymbol{C}}(X) = \bigcup_{\substack{u_i\, red : u_i \in \{0,1\} \\ u_i\, frozen : u_i = 0}} A_n^{(s)}(\boldsymbol{u}_{s-1}, 0)(X).$$

As an example, lets take $n = 32$ and $k = 16$ so the code

rate is $R = 1/2$. The standard frozen bit indices are $F = \{0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 16, 17, 18, 20, 24\}$ and information indices are $I = \{7, 11, 13, 14, 15, 19, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31\}$; therefore, the information sequence is: $\boldsymbol{u} = \{u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}, u_{11}, , u_{12}, u_{13}, u_{14}, u_{15}, , u_{16}, u_{17}, u_{18}, u_{19}, u_{20}, u_{21}, u_{22}, u_{23}, , u_{24}, u_{25}, u_{26}, u_{27}, u_{28}, u_{29}, u_{30}, u_{31}\}$. The last frozen bit index is $s = 24$.

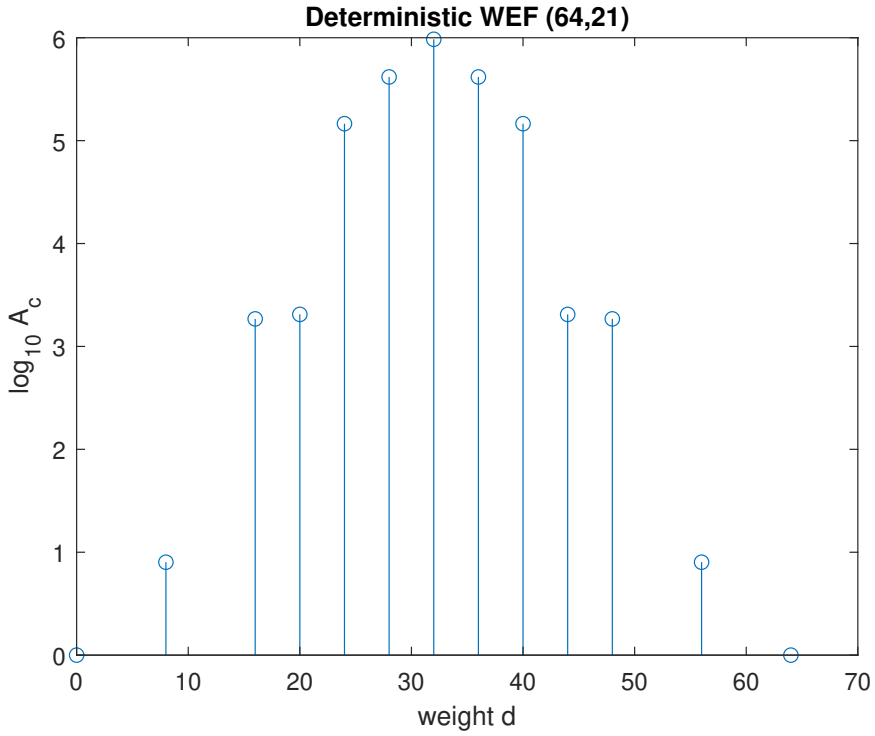

Figure 4.2: Deterministic polar WEF computation.

In this example, the information bits whose index is less than 24 are called **red bits**. We have nine red bits in this example. Hence "mixing factor" which refers to the number of red bits is 9. This means that we have 2^9 disjoint polar cosets. Therefore, $A_{\boldsymbol{C}}(X) = \sum_{\text{All possible } u_0^{23} \text{ permutations}} A_{32}^{(9)}(u_0^{23}, 0)(X)$.

As an example, we provide the WEF results based on [11] for a (64,21) polar code in Fig. 4.2. The results are verified by those of the exhaustive search as well.

While the above scheme gives us the weight distribution of polar codes; it does not look feasible for code lengths higher than 128 due to complexity as the number of iterations grows. Also, using this scheme for PAC code WEF computation appears very difficult (in particular, for practical code lengths).

4.3 Weight Distribution of PAC codes

The weight distribution of the PAC code is essential in calculating the union bound, which is the upper bound for the codeword error rate. The precise computation of weight distribution for most channel codes is highly complex as it is needed to calculate the Hamming distance of the codeword for each possible input message exhaustively. Therefore, an approximate scheme or an accurate scheme with a feasible computation complexity should be utilized. In the literature, many works address this issue in polar code, such as [32], [33] and [11]. In [9], a probabilistic method for computing the weight distribution of polar codes is introduced, and [10] which is enhanced. Here, we extend this method for the PAC codes.

Consider a PAC code represented by $(N, K, \mathcal{A}, \boldsymbol{T}_N)$ where N and K are codeword and message lengths, respectively, $\mathcal{A}$ is the set of information (non-frozen) bit-positions and $\boldsymbol{T}_N \in \mathbb{R}^{N\times N}$ is an upper-triangular Toeplitz matrix constructed with the convolutional code generator $\boldsymbol{g}$. We can split the codeword $\boldsymbol{x}$ into two parts, i.e., $\boldsymbol{x} = \{x_1^{N/2}, x_{N/2+1}^N\}$ where[1] $x_{N/2+1}^N = u_{N/2+1}^N \boldsymbol{G}_{N/2}$, and $x_1^{N/2} = (u_{N/2+1}^N \oplus u_1^{N/2})\boldsymbol{G}_{N/2}$. Furthermore, we can write $\boldsymbol{T}_N$ as

$$\boldsymbol{T}_N = \begin{bmatrix} \boldsymbol{T}_{N/2} & \boldsymbol{S}_{N/2} \\ \boldsymbol{0} & \boldsymbol{T}_{N/2} \end{bmatrix} \tag{4.10}$$

where $\boldsymbol{T}_{N/2}$ is the convolutional encoding matrix of a PAC code with length $N/2$,

[1] Throughout this chapter, we use x_a^b to represent $(x_a, x_{a+1}, ..., x_b)$, a subvector of $\boldsymbol{x}$.

and

$$\boldsymbol{S}_{N/2} = \begin{pmatrix} 0 & \cdots & & & \cdots & & 0 \\ \vdots & & \ddots & & & \ddots & \\ 0 & \cdots & & \cdots & & \cdots & 0 \\ g_\ell & 0 & \cdots & & \cdots & & 0 \\ g_{\ell-1} & g_\ell & 0 & \cdots & & \cdots & 0 \\ \vdots & & \ddots & \ddots & & & \vdots \\ g_1 & g_2 & \cdots & g_\ell & 0 & \cdots & 0 \end{pmatrix}. \tag{4.11}$$

The polar transformation matrix also can be wrriten as

$$\boldsymbol{G}_N = \begin{bmatrix} \boldsymbol{G}_{N/2} & \boldsymbol{0} \\ \boldsymbol{G}_{N/2} & \boldsymbol{G}_{N/2} \end{bmatrix}. \tag{4.12}$$

Therefore,

$$\boldsymbol{x} = \boldsymbol{v} \begin{bmatrix} \boldsymbol{T}_{N/2} & \boldsymbol{S}_{N/2} \\ \boldsymbol{0} & \boldsymbol{T}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{G}_{N/2} & \boldsymbol{0} \\ \boldsymbol{G}_{N/2} & \boldsymbol{G}_{N/2} \end{bmatrix} \tag{4.13}$$

Thus,

$$x_1^{N/2} = v_1^{N/2}\boldsymbol{T}_{N/2}\boldsymbol{G}_{N/2} \oplus v_{N/2+1}^{N}\boldsymbol{T}_{N/2}\boldsymbol{G}_{N/2} \oplus v_1^{N/2}\boldsymbol{S}_{N/2}\boldsymbol{G}_{N/2}, \tag{4.14}$$

and

$$x_{N/2+1}^{N} = v_1^{N/2}\boldsymbol{S}_{N/2}\boldsymbol{G}_{N/2} \oplus v_{N/2+1}^{N}\boldsymbol{T}_{N/2}\boldsymbol{G}_{N/2}. \tag{4.15}$$

We note that, $v_{N/2+1}^{N}\boldsymbol{T}_{N/2}\boldsymbol{G}_{N/2}$, and $v_1^{N/2}\boldsymbol{T}_{N/2}\boldsymbol{G}_{N/2}$ are PAC codes with $(N/2, |\mathcal{A}_2|, \mathcal{A}_2, \boldsymbol{T}_{N/2})$ and $(N/2, |\mathcal{A}_1|, \mathcal{A}_1, \boldsymbol{T}_{N/2})$, corresponding to the input sequences $v_{N/2+1}^{N}$ and $v_1^{N/2}$, respectively, and $\mathcal{A}_1 = \left\{i \in \mathcal{A} | i \leq \frac{N}{2}\right\}$, and $\mathcal{A}_2 = \left\{i \in \mathcal{A} | i > \frac{N}{2}\right\} - \frac{N}{2}$.

If we represent $v_1^{N/2}\boldsymbol{T}_{N/2}\boldsymbol{G}_{N/2}$, $v_{N/2+1}^{N}\boldsymbol{T}_{N/2}\boldsymbol{G}_{N/2}$ and $v_1^{N/2}\boldsymbol{S}_{N/2}\boldsymbol{G}_{N/2}$ as $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$ and $\boldsymbol{s}$, respectively, the Hamming weight of x_1^N is given by

$$h\left(x_1^N\right) = h\left(\boldsymbol{x}^{(1)} \oplus \boldsymbol{x}^{(2)} \oplus \boldsymbol{s}\right) + h\left(\boldsymbol{x}^{(2)} \oplus \boldsymbol{s}\right), \tag{4.16}$$

where $h(\boldsymbol{x})$ denotes the Hamming weight of sequence $\boldsymbol{x}$.

If we ignore the effect of the vector $\boldsymbol{s}$ (i.e., assume $\boldsymbol{s} = \mathbf{0}$), Eq. (4.16) is reduced to

$$h\left(x_1^N\right) = h\left(\boldsymbol{x}^{(1)} \oplus \boldsymbol{x}^{(2)}\right) + h\left(\boldsymbol{x}^{(2)}\right) \tag{4.17}$$

That is, in each step, we effectively split the PAC code into two PAC codes of half the length with information sets of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. For a given $\boldsymbol{x}^{(2)}$ vector, we model the $\boldsymbol{x}^{(1)}$ vector randomly, and write an approximate weight distribution in a probabilistic fashion. Let $P^{(d)}[\mathcal{A}, N, T_N]$ denotes the probability that PAC codeword with length N and information set $\mathcal{A}$ and polynomial $\boldsymbol{g}$ that generates T_N has weight d. We compute the probability that a codeword has a weight d as

$$\begin{aligned} P^{(d)}[\mathcal{A}, N, \boldsymbol{T}_N] &= \sum_{d_1} P\left(h(\boldsymbol{x}^{(2)}) = d_2 | \mathcal{A}_2\right) \\ \times P\left(h(\boldsymbol{x}^{(1)} \oplus \boldsymbol{x}^{(2)}) = d - d_2 | \mathcal{A}_1, h(\boldsymbol{x}^{(2)}) = d_2\right) &= \sum_{d_2} \sum_{d_1} P^{(d_2)}[\mathcal{A}_2, N/2, \boldsymbol{T}_{N/2}] \\ &\times P^{(d_1)}[\mathcal{A}_1, N/2, \boldsymbol{T}_{N/2}] \times f_N(t, d_1, d_2). \end{aligned} \tag{4.18}$$

with $t = d - d_2$, and

$$f_N(t, d_1, d_2) \;=\; P\Big(h\left(\boldsymbol{x}^{(1)} \oplus \boldsymbol{x}^{(2)}\right) \;=\; t | h(\boldsymbol{x}^{(1)}) \;=\; d_1, h(\boldsymbol{x}^{(2)}) \;=\; d_2\Big). \tag{4.19}$$

We are using a similar function to f_N defined in (4.9), given by $f_N(t, d_1, d_2) = \binom{d_1}{r}\binom{N/2-d_1}{d_2-r}/\binom{N/2}{d_2}$ for $t \geq |d_1 - d_2|$ or $t \leq min\{d_1 + d_2, N - d_1 - d_2\}$, and 0 otherwise, where $r = (d_1 + d_2 - t)/2$ is the number of positions at which the elements in $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are both equal to 1.

In other words, the weight distribution of a PAC code with length N can be recursively obtained through that of a PAC code with length $N/2$ until a threshold value of N_{th}. We can then evaluate the remaining weight distributions using exhaustive search to enhance the accuracy of the approximation.

To increase the accuracy of the approximation, we can consider the actual

value of the $\boldsymbol{s}$. We divide the whole sequence into two parts with information index sets of $\mathcal{A}_1$ and $\mathcal{A}_2$ until we reach a threshold codeword length N_{th}. After that we compute the weight distribution of each part in an exact manner by taking the matrix $\boldsymbol{S}_{N_{th}/2}$ into account, exhaustively. As depicted in Eq. (4.11), $\boldsymbol{S}_{N/2}$ is a sparse matrix and has an effect only on ℓ bits, i.e., on $v_{\frac{N}{2}-\ell+1}^{\frac{N}{2}}$. For the rest of the section, we represent $v_{\frac{N_{th}}{2}-\ell+1}^{\frac{N_{th}}{2}}$ by $\boldsymbol{v}^{(\boldsymbol{s})}$. For calculating $h(\boldsymbol{x}^{(2)} \oplus \boldsymbol{s})$, we fix $\boldsymbol{v}^{(\boldsymbol{s})}$, and find the weight distribution of $\boldsymbol{x}^{(2)}$ with codeword length $N_{th}/2$ and the set of information indices $\mathcal{A}_2$. We will have a list of $2^{|\mathcal{A}_s|}$ weight distributions for $\boldsymbol{x}^{(2)} \oplus \boldsymbol{s}$, where $\mathcal{A}_s = \{i \in \mathcal{A}_1 | i > N_{th}/2 - \ell\}$, which are the information bit locations related to $\boldsymbol{v}^{(\boldsymbol{s})}$. On the other hand, with fixed $\boldsymbol{v}^{(\boldsymbol{s})}$ we estimate the weight distribution of $\boldsymbol{x}^{(0)}$, which is the set of indices that are not produced by $\boldsymbol{v}^{(\boldsymbol{s})}$ and has information bit sets of $\mathcal{A}_0 = \{i \in \mathcal{A}_1 | i \leq N_{th}/2 - \ell\}$.

Let $P^{(d)}[\mathcal{A}, N_{th}, \boldsymbol{T}_{N_{th}}]$ denote the probability that a PAC codeword with length N_{th}, information set $\mathcal{A}$ and polynomial $\boldsymbol{g}$ that generates $\boldsymbol{T}_{N_{th}}$ has weight d. We can calculate $P^{(d)}[\mathcal{A}, N_{th}, \boldsymbol{T}_{N_{th}}]$ with fixed $\boldsymbol{v}^{(\boldsymbol{s})}$ by

$$P^{(d)}[\mathcal{A}, N_{th}, \boldsymbol{T}_{N_{th}}] = \sum_{d_1} \sum_{d_0} P\left(h(\boldsymbol{x}^{(0)}) = d_0 | \mathcal{A}_0, \mathcal{A}_s\right) \times P\left(h(\boldsymbol{x}^{(2)} \oplus \boldsymbol{s}) = d_1 | \mathcal{A}_2, \mathcal{A}_s\right) f_{N_{th}}(t, d_1, d_0), \quad (4.20)$$

with $t = d - d_1$, since $P\Big(h\left(\boldsymbol{x}^{(1)} \oplus \boldsymbol{x}^{(2)} + \boldsymbol{s}\right) = t | h\left(\boldsymbol{x}^{(2)} + \boldsymbol{s}\right) = d_1, h\left(\boldsymbol{x}^{(0)}\right) = d_0\Big)$ is simply $f_{N_{th}}(t, d_1, d_0)$.

After finding the $P^{(d)}$ for each part by (4.20), we use (4.18) to find the $P^{(d)}$ of PAC code with length N, in a recursive fashion. The corresponding weight distribution is found by (4.5) averaging over all possible $\boldsymbol{v}^{(\boldsymbol{s})}$'s.

4.3.1 Design of PAC Codes using Approximate Weight Distributions

In this part, we employ approximate weight distributions to design PAC codes using a discrete optimization method. The cost function for each design is assigned as union bound value at a fixed SNR.

4.3.1.1 Union Bound for AWGN

Using the weight enumerating function that we get we can calculate the union bound of the PAC code in the AWGN with BPSK modulation as

$$P_e \leq \sum_{d=1}^{N-1} A(d)P(d), \tag{4.21}$$

where $A(d)$ is the number of codeword with hamming weight of d and $P(d)$ is the probability of decoding codeword $\hat{c}$ instead of c with weight difference of d i.e. $P(d) = P(\boldsymbol{c} \to \hat{\boldsymbol{c}})$.

$$P(\boldsymbol{c} \to \hat{\boldsymbol{c}}) = Q\left(\sqrt{\frac{\gamma_s d_E^2(\boldsymbol{c}, \hat{\boldsymbol{c}})}{2}}\right) \tag{4.22}$$

where γ_s is the signal to noise ration (i.e. $\gamma_s = \frac{Es}{N_0}$) and $d_E^2(\boldsymbol{c}, \hat{\boldsymbol{c}})$ is calculated as following

$$d_E^2(\boldsymbol{c}, \hat{\boldsymbol{c}}) = \sum_{i=0}^{N-1} |x_i - \hat{x}_i|^2 = 4d. \tag{4.23}$$

Assuming unit symbol energy

$$P_e \leq \sum_{d=1}^{N-1} A(d)Q\left(\sqrt{\frac{2d}{N_0}}\right). \tag{4.24}$$

A well-designed rate profile can improve the error correction performance of the PAC code. The possible numbers of rate-profile information sequences for a code

with N codeword length and rate R is calculated as $\binom{N}{RN}$ which is a large number, and going through every possible outcome is infeasible. Having a fast method for calculating a weight distribution of PAC and therefore union bound of it, we can use the upper-bounded probability of codeword error at a fixed SNR as a cost function or a metric to evaluate the error-correcting performance of a PAC code given a specific rate-profile. Therefore, we need a discrete optimization technique to search possible rate-profiles and finds the one with the least cost or better metric where we use Simulated Annealing (SA) here.

Simulated Annealing is a probabilistic method inspired by thermodynamics for approximating the global minima of the given function. It is categorized as a discrete optimization scheme and can be used in problems where finding the approximate global minimum in a search area with an exponential or factorial number of elements is essential. The idea comes from the fact that by decreasing the temperature of the metal extremely slow, the desired shape of the metal can be obtained. For our problem we need to go over and evaluate different rate-profiles; however, brute force is extremely complex, therefore with simulated annealing we go over a portion of possible rate-profiling sets to find one with a minimum probability of the error at fixed SNR.

Algorithm 1, shows the psudo-code of the simulated annealing scheme used in our problem.

In this technique we first start with a specific rate-profile and then evaluate its probability of error using the union bound formula in 4.24 at fixed SNR, then we choose a new rate-profiling by perturbing a bit. The perturbing is done by choosing a bit-location randomly from the frozen set and replace it with a random bit location at the information set. After perturbing is done, we evaluate the metric of the newly generate rate-profile. If the cost of the later rate-profile is lower than the cost of the previous rate-profile we continue with the new rate-profile otherwise we preserve the older rate-profile except the probability of the acceptance is less than the random number between 0 and 1. If we assume that E_k and E_{k+1} are the cost of the older and newer rate-profiles respectively the

Algorithm 1 Simulated Annealing

Input N, K, $\boldsymbol{g}$, Starting $\mathcal{A}$, T_{max}, T_{min}, a
Output best_inf

best_E=1; i=1; T=Tmax;
while $T > T_{min}$ **do**
  $E_c \leftarrow cost(N, K, g, \mathcal{A})$
  $\mathcal{A}\_next \leftarrow$ Perturb a single bit location
  $E_n \leftarrow cost(N, K, g, \mathcal{A}\_next)$; $\Delta = E_n - E_c$
  if $\Delta < 0$ **then**
    $i \leftarrow i + 1$; Info $\leftarrow \mathcal{A}\_next$;
    if $En < best\_E$ **then**
      best_inf $\leftarrow$ Info; best_E $\leftarrow E_n$
    end if
    T $\leftarrow T_{max} \times a^i$
  else if $e^{(-\Delta/T)} > rand(0, 1)$ **then**
    $i \leftarrow i + 1$; Info $\leftarrow \mathcal{A}\_next$;
    T $\leftarrow T_{max} \times a^i$
  end if
end while

probability of the acceptance in temperature T is given as

$$p(\Delta) = \frac{e^{(\frac{-E_k}{T})}}{e^{(\frac{-E_k}{T})} + e^{(\frac{-E_{k+1}}{T})}} \tag{4.25}$$

$$= \frac{1}{1 + \exp\left(\frac{\Delta}{T}\right)} \approx \exp\left(\frac{-\Delta}{T}\right) \tag{4.26}$$

The acceptance probability is near to 1 at the beginning of the algorithm when the temperature is high. This property enables us to accept some of the rate-profiles that are not lower in the metric and continue with them which this step prevents the algorithm to stuck at the local minimums. This procedure continue until the T_{min} is reached. Parameter a in algorithm 1 is a numeric between 0 and 1, and the controlling factor for the speed of the annealing. When a is closer to 1 the pace of the temperature decrease is slow and when it is near to 0 it is very fast, i.e., in each step, when the algorithm accepts the ith new rate profile, temperature is set to $T_{max} \times a^i$, where a is a parameter between 0 and 1, controlling the speed

Table 4.1: Low-weight codewords of PAC (128,64) with polar rate profile

	A_8	A_{12}	A_{16}	A_{18}	A_{20}
Polar profile in [34]	48	0	11032	6024	$> 10^5$
Polar profile approx	48	0	11274	2241	311486

Table 4.2: Low-weight codewords of PAC (128,64) with RM rate profile

	A_8	A_{12}	A_{16}	A_{18}	A_{20}	A_{22}
RM profile in [34]	0	0	3120	2696	95828	$> 10^5$
RM profile approx	0	0	3285	1285	89563	362869

of the annealing. The acceptance probability is near one at the beginning of the algorithm when the temperature is high. This property enables the algorithm to accept some of the rate profiles that are not lower in the metric and continue with them. This step prevents the algorithm from being stuck at the local minima. This procedure continues until the T_{min} is reached. We note that the search can be done either over all possible rate profiles or over the bit locations with the same RM score in non-full rank RM rate profiles to determine which of those bit locations should be added to the information set. It is worth mentioning that when the search is over the bit-locations with the same RM score, perturbation is done only among those that have the specific RM score in the frozen and the information sets.

Let us now present several simulations and numerical results. In Fig. 4.3, the weight distribution of a (64,22) PAC code with polynomial 133 in octal notation and the RM rate profile is approximated with two methods: 1) assuming $\boldsymbol{s} = 0$ (labeled as "Randomized"), and 2) assuming $\boldsymbol{s} \neq 0$ (labeled as "Improved Randomized"). The results are compared with the exact weight distribution obtained by an exhaustive search. The results show that both approximations give close results to the exact weight distribution in this scenario. In Tables 4.1 and 4.2, we compare the approximate weight distributions of PAC codes with codeword length 128 and rate $\frac{1}{2}$ with the low weight results obtained in [34], using the code polynomial 133 in octal notation with RM and polar rate-profiles, respectively. We observe that the approximate WEF results are very close to those obtained via exhaustive search.

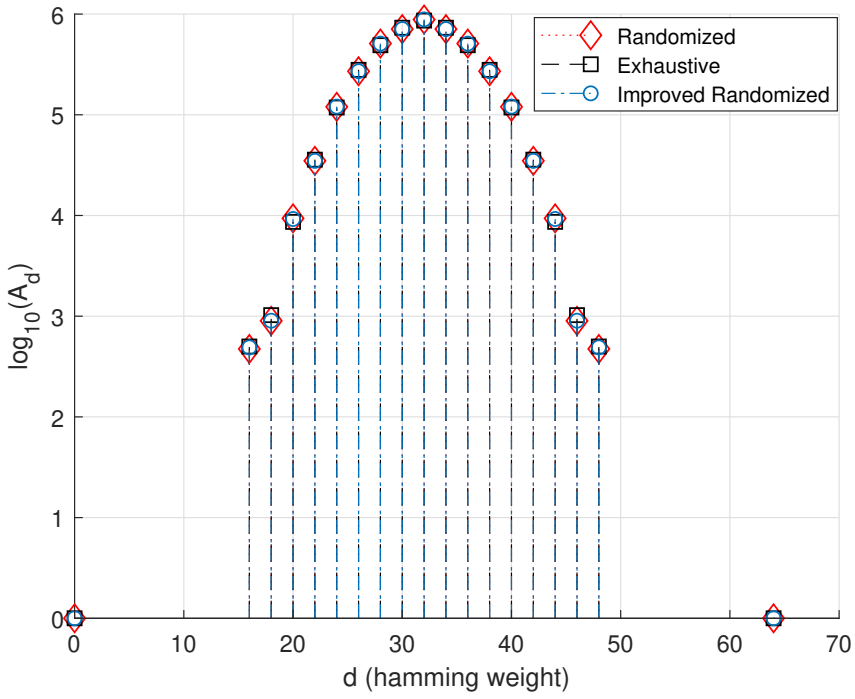

Figure 4.3: Comparison of approximated weight distribution of PAC (64,22) with discrete weight distribution generated by exhaustive search.

In Fig. 4.4, we design two rate profiles for a PAC (64,32) code, decoded with a list decoder and a list size of 32. One of the rate profiles is obtained by an exhaustive search over all possible combinations of bit-locations with an RM score of 3, along with bit-locations that have higher RM scores as information (non-frozen) part. The other proposed PAC design, is the output of the SA algorithm that searches among all possible rate profiles regardless of their RM score, obtained by setting $a = 0.99$, $T_{max} = 0.001$ and $T_{min} = 10^{-4}$ with the SNR fixed at $3dB$. The results show that the performance of the first approach is similar to those of Monte-Carlo-based [35] and Q-Learning [36] designs. We also compare the results with performance of 5G-standardized polar code with CRC-11 and DEGA design [17]. In this figure, RM_1 represents an RM rate profile that picks higher indexed bit-positions with the same RM score, while RM_2 picks lower indexed ones.

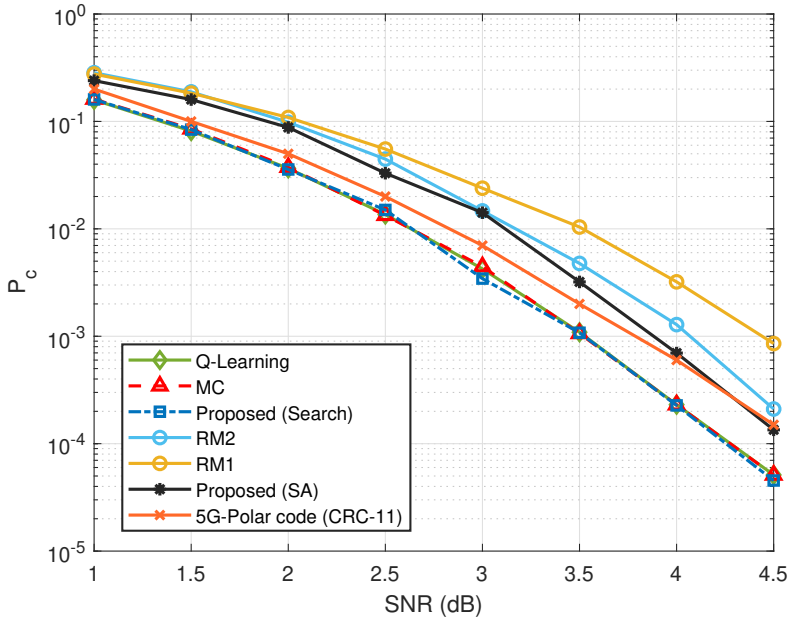

Figure 4.4: Performance of PAC (64,32) code with polynomial 133 using different rate-profiles and list decoder (with a list size of 32).

In Fig. 4.5, PAC (64,32) codes with the same rate profiles as in Fig. 4.4 are decoded with a list decoder of size 128. The results show that their decoding performances match the corresponding union bounds (which assumes ML decoding). The SA-based design has an inferior performance with the list decoder with a list size of 32, but its performance is superior to the other rate profiles when the list size is 128.

In Fig. 4.6, two rate profiles for a PAC (64,16) are given. The codeword error rates with both ML decoder and List decoder with a list size of 128 are presented. As seen in Fig. 4.6, one of the newly designed rate profiles (Proposed 2) has inferior performance with the list decoder even with a large list size, while its performance with ML decoder matches the union bound derived from the weight distribution of PAC code. This shows that some rate profiles may exist that are optimal under ML decoding; however, their performance may be inferior to the other proposed rate profiles with suboptimal decoders. We attribute this rather poor performance with a suboptimal decoder to the high mean computational

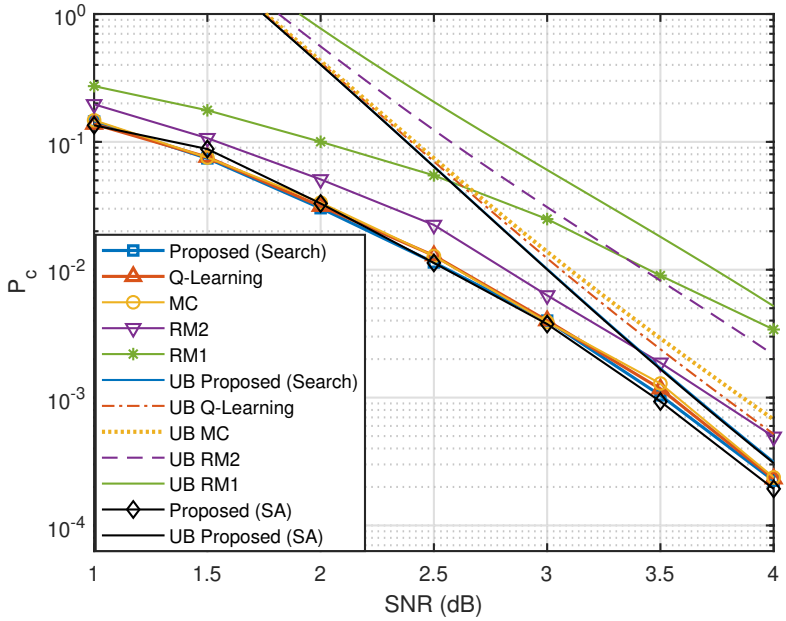

Figure 4.5: Performance of PAC (64,32) code with different rate profiles with list decoder (with a list size 128) and polynomial 133 with their corresponding approximate union bounds.

complexity of the sequential decoding. Since in PAC codes, the convolutional encoded sequence sees polarized channels, the mean complexity of the sequential decoding is finite when $\ell R_\ell < \sum_{i=1}^{\ell} R_0\left(1, W_N^{(i)}\right)$, for all $1 < \ell < N$ where N is the codeword length, $R_0\left(1, W_N^{(i)}\right)$ is the cut-off rate of the ith polarized channel i.e. $W_N^{(i)}$ and $R_\ell = \frac{\alpha}{\ell}$ with α being the number of non-frozen bit locations in the first ℓ bits of the rate profiled sequence (see [24] for details).

In Fig. 4.7, the ℓR_ℓ for the first and the second proposed rate profiles are compared with cutoff rates at 1 and 2 dB. It can be seen that the Proposed 1 rate profile is under the cutoff rate at 2 dB for most of the bit locations while the Proposed 2 rate profile is over. Fig. 4.7 justifies that for PAC code with the Proposed 2 rate profile, the sequential decoder visits the same nodes for extremely large number of times, or equivalently, list decoder needs an extremely large list size to match with the union bound results.

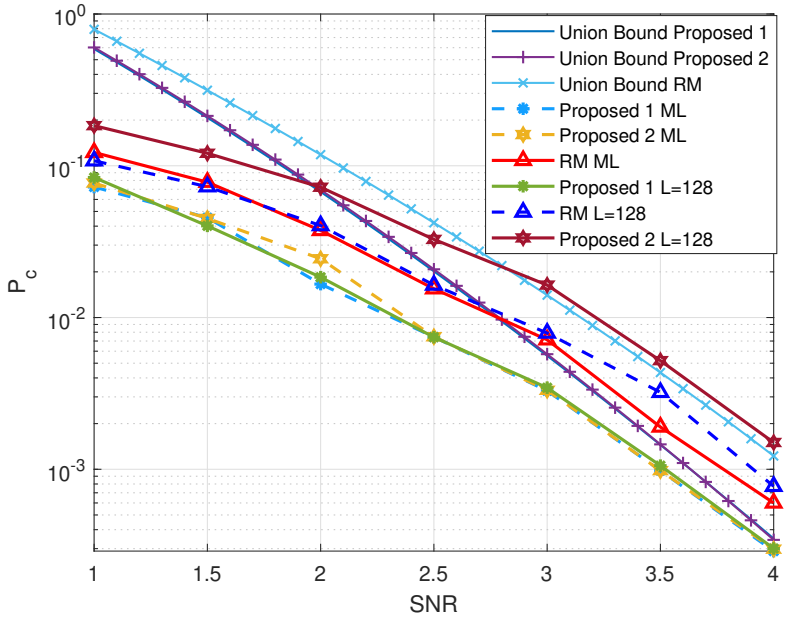

Figure 4.6: Performance of PAC(64,16) using polynomial 133 and different rate-profiles with ML decoder and list decoder (list size 128) and corresponding exact union bounds.

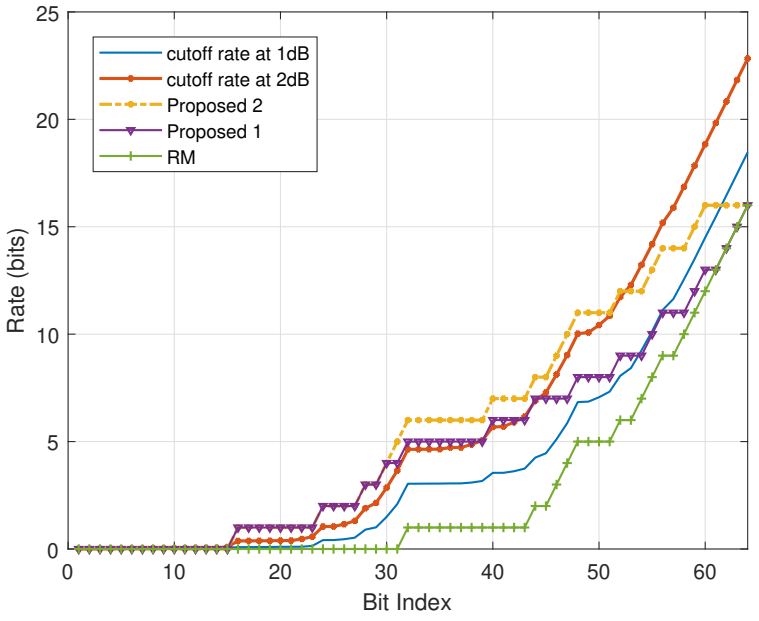

Figure 4.7: Comparison of rate-profiles in Fig. 4.6 with cutoff rates.

4.4 Chapter Summary

In this chapter, we reviewed probabilistic methods that approximate weight enumerating functions for polar codes. A low complexity way to find the weight distribution of polar codes is also reviewed. Furthermore, we develop a method to approximate weight distribution of PAC codes. We also provide a method of exploiting the derived approximate weight distribution along with a union bound to design specific PAC codes, particularly, to select the code rate profiles using a discrete optimization method based on simulated annealing. The results of the newly developed algorithm are presented in a recent paper [37].

Chapter 5

Conclusions and Future Work

In this thesis, we investigate short block length codes over AWGN and fading channel models. The first group of codes that we consider are convolutional codes specifically tail-biting convolutional codes. Different decoding algorithms are reviewed while the main focus is on the WAVA algorithm. As the second class of codes, PAC codes, introduced recently as a way of near optimal coding for short block lengths, are studied. Different sequential decoding algorithms are compared, and a simplified successive cancellation list decoder is proposed to overcome the complexity issue of the standard successive cancellation list decoder. The results demonstrate that the performance is close to that of the Fano decoder with a list decoder and list size of 128 or higher.

The reported results show that the performance of PAC codes for the BI-AWGN channel is better than that of the 5G polar codes with various rates and block-lengths. PAC codes, in BI-AWGN channels as well as Rayleigh block fading channels, perform very close to the dispersion approximation. It is also reported that they outperform the 5G standard polar codes when used within a BICM scheme with higher-order modulations. We also find that puncturing PAC codes, which is a useful tool for obtaining different code rates, is effective. That is, the performance of the punctured PAC codes with the proposed decoding scheme is near to the corresponding normal approximation results. We modify the existing

normal approximations on the codeword error rates to the case of Rayleigh block fading channels in the finite-length regime, and compare the performance of the PAC and 5G polar codes using these information theoretic results as well. The results show that with a proper PAC code design, the performance can be superior to those of the 5G standard polar codes and the tail-biting convolutional codes, and that the decoding complexity can be smaller than that of the WAVA decoder.

As a major contribution of the thesis, we develop a method to calculate the weight distribution of PAC codes. The approach is probabilistic, and results in an approximate weight distribution of PAC codes. The results have excellent match with the exact weight distributions that can be computed in a brute-force manner for small codes. We also provide a method of using the computed weight distributions along with a union bound to design specific PAC codes, particularly, to select the code rate profiles using a discrete optimization method based on simulated annealing. Numerical examples demonstrate that the PAC codes with the designed rate profiles via the approximate union bounds offer superior performance.

As future work, there are several directions that can be pursued with the main goal of improving the performance of short length channel codes. For instance, low complexity decoding algorithms for PAC codes and universal (near-optimal) designs for PAC code rate profiles can be pursued. As another potential research topic, implementation of genetic algorithm-based designs instead of simulated annealing, in order to find better rate profiles for PAC codes, can be given.

Another interesting direction is the study of pre-transformed polar codes. It is proven in [38] that pre-transformation with an upper-triangular matrix improves the weight spectrum of polar codes; however, the design of an optimal pre-transformation matrix is still open.

We note that SC-Flip (SCF) decoder, which uses additional SC decoding attempts when the initial SC decoding fails due to a single channel-induced erroneous bit, has recently received attention. It is possible that this decoder may

improve the FER performance of polar and PAC codes as well. Therefore, a detailed investigation may be interesting. Finally, we highlight that decoding PAC codes with deep neural networks may be a useful approach for an effective and low-complexity solution.

Bibliography

[1] M. Shirvanimoghaddam, M. S. Mohammadi, R. Abbas, A. Minja, C. Yue, B. Matuz, G. Han, Z. Lin, W. Liu, Y. Li, S. Johnson, and B. Vucetic, "Short block-length codes for ultra-reliable low latency communications," *IEEE Communications Magazine*, vol. 57, no. 2, pp. 130–137, 2019.

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

[3] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[4] W. Yang, G. Durisi, T. Koch, and Y. Polyanskiy, "Block-fading channels at finite blocklength," in *ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems*. VDE, 2013, pp. 1–4.

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

[6] Z. B. K. Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, "The development, operation and performance of the 5G polar codes," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 96–122, 2019.

[7] V. Bioglio, C. Condo, and I. Land, "Design of polar codes in 5G new radio," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 29–40, 2020.

[8] E. Arıkan, "From sequential decoding to channel polarization and back again," *arXiv preprint arXiv:1908.09594*, 2019.

[9] M. Valipour and S. Yousefi, "On probabilistic weight distribution of polar codes," *IEEE Communications Letters*, vol. 17, no. 11, pp. 2120–2123, 2013.

[10] Q. Zhang, A. Liu, and X. Pan, "An enhanced probabilistic computation method for the weight distribution of polar codes," *IEEE Communications Letters*, vol. 21, no. 12, pp. 2562–2565, 2017.

[11] 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)*, 2021, pp. 1218–1223.

[12] Y. Polyanskiy and S. Verdú, "Scalar coherent fading channel: dispersion analysis," in *2011 IEEE International Symposium on Information Theory Proceedings*, 2011, pp. 2959–2963.

[13] T. Erseghe, "Coding in the finite-blocklength regime: bounds based on Laplace integrals and their asymptotic approximations," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 6854–6883, 2016.

[14] R. Shao, S. Lin, and M. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Transactions on Communications*, vol. 51, no. 10, pp. 1658–1665, 2003.

[15] C. Weiss, C. Bettstetter, S. Riedel, and D. Costello, "Turbo decoding with tail-biting trellises," in *1998 URSI International Symposium on Signals, Systems, and Electronics. Conference Proceedings (Cat. No.98EX167)*, 1998, pp. 343–348.

[16] L. Gaudio, T. Ninacs, T. Jerkovits, and G. Liva, "On the performance of short tail-biting convolutional codes for ultra-reliable communications," in *SCC 2017; 11th International ITG Conference on Systems, Communications and Coding*, 2017, pp. 1–6.

[17] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications*, vol. 60, no. 11, pp. 3221–3227, 2012.

[18] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[19] H. Zhou, C. Zhang, W. Song, S. Xu, and X. You, "Segmented crc-aided sc list polar decoding," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–5.

[20] L. Xiang, Z. B. Kaykac Egilmez, R. G. Maunder, and L. Hanzo, "CRC-aided logarithmic stack decoding of polar codes for ultra reliable low latency communication in 3GPP new radio," *IEEE Access*, vol. 7, pp. 28 559–28 573, 2019.

[21] M. C. Coskun, G. Liva, J. Oestman, and G. Durisi, "Low-complexity joint channel estimation and list decoding of short codes," in *SCC 2019; 12th International ITG Conference on Systems, Communications and Coding*, 2019, pp. 1–5.

[22] M. Rowshan, A. Burg, and E. Viterbo, "Polarization-adjusted convolutional (PAC) codes: Sequential decoding vs list decoding," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1434–1447, 2021.

[23] R. Johannesson and K. S. Zigangirov, *Fundamentals of convolutional coding*. John Wiley & Sons, 2015.

[24] M. Moradi, "On sequential decoding metric function of polarization-adjusted convolutional (PAC) codes," *IEEE Transactions on Communications*, vol. 69, no. 12, pp. 7913–7922, 2021.

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

[26] M. Moradi, A. Mozammel, K. Qin, and E. Arikan, "Performance and complexity of sequential decoding of pac codes," *arXiv preprint arXiv:2012.04990*, 2020.

[27] J. Font-Segura, G. Vazquez-Vilar, A. Martinez, A. Guillén i Fàbregas, and A. Lancho, "Saddlepoint approximations of lower and upper bounds to the

error probability in channel coding," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, 2018, pp. 1–6.

[28] G. Vazquez-Vilar, A. G. i Fabregas, T. Koch, and A. Lancho, "Saddlepoint approximation of the error probability of binary hypothesis testing," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 2306–2310.

[29] R. Asvadi, A. H. Banihashemi, M. Ahmadian-Attari, and H. Saeedi, "LLR approximation for wireless channels based on Taylor series and its application to BICM with LDPC codes," *IEEE Transactions on Communications*, vol. 60, no. 5, pp. 1226–1236, 2012.

[30] R. Yazdani and M. Ardakani, "Linear LLR approximation for iterative decoding on wireless channels," *IEEE Transactions on Communications*, vol. 57, no. 11, pp. 3278–3287, 2009.

[31] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 230–234.

[32] Z. Liu, K. Chen, K. Niu, and Z. He, "Distance spectrum analysis of polar codes," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, 2014, pp. 490–495.

[33] R. Polyanskaya, M. Davletshin, and N. Polyanskii, "Weight distributions for successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 68, no. 12, pp. 7328–7336, 2020.

[34] H. Yao, A. Fazeli, and A. Vardy, "List decoding of Arıkan's PAC codes," *Entropy*, vol. 23, no. 7, p. 841, 2021.

[35] M. Moradi and A. Mozammel, "A monte-carlo based construction of polarization-adjusted convolutional (PAC) codes," *arXiv preprint arXiv:2106.08118*, 2021.

[36] S. K. Mishra, D. Katyal, and S. A. Ganapathi, "A modified Q-learning algorithm for rate-profiling of polarization adjusted convolutional (PAC)

codes," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 2363–2368.

[37] S. Seyedmasoumian and T. M. Duman, "Approximate weight distribution of polarization-adjusted convolutional (PAC) codes," in *2022 IEEE International Symposium on Information Theory (ISIT) (ISIT 2022)*, Espoo, Finland, Jun. 2022.

[38] B. Li, H. Zhang, and J. Gu, "On pre-transformed polar codes," *arXiv preprint arXiv:1912.06359*, 2019.