Guessing Random Additive Noise Decoding (GRAND), from Performance to Implementation

by

Wei An

B.Eng., Shanghai Jiao Tong University (1993)

M.Eng., University of Delaware (2000)

E.E., Massachusetts Institute of Technology (2010)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....

Department of Electrical Engineering and Computer Science

March 11, 2022

Certified by .....

Muriel Médard

Cecil H. Green Professor of Electrical Engineering and Computer Science

Thesis Supervisor

Certified by .....

Ken R. Duffy

Director of the Hamilton Institute, Maynooth University

Thesis Supervisor

Accepted by .....

Leslie A. Kolodziejski

Professor of Electrical Engineering and Computer Science

Chair, Department Committee on Graduate Students

Guessing Random Additive Noise Decoding (GRAND), from Performance to Implementation

by

Wei An

Submitted to the Department of Electrical Engineering and Computer Science
on March 11, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

To meet high reliability and low latency requirements in many new applications, such as those in Ultra Reliable Low Latency Communications (URLLC), a universal optimal decoder is desired that can not only be used to select the best short code candidate, but also can adapt itself to channel memory avoiding performance degradation. The Guessing Random Additive Noise Decoding (GRAND) algorithm makes such a decoder possible. Its prevailing research is outlined in Chapter 1.

The well-known Markovian channels are selected for investigation in Chapter 2, leading to the Markov ordered GRAND (GRAND-MO) decoder. By exploring channel statistical properties in its pattern generation, GRAND-MO achieves significant decoding gains with increasing channel memory, eliminating the need of interleavers. The algorithm is extended to high-order modulations by guessing symbol noises, voiding de-mappers and achieving additional decoding gains, especially with the augmented constellation.

Chapter 3 explains the rationale behind the basic version of Ordered Reliability Bits GRAND (ORGRAND), and extends the algorithm to its full version, overcoming the performance limitation in high SNR regions. A number of complexity control techniques ensure the robustness and feasibility of ORBGRAND for practical implementations. Its extension to high-order modulations is justified with additional decoding gain as well as the elimination of complex de-mapping operations.

Armed with both hard and soft detection variants of GRAND, Cyclic Redundancy Check (CRC) codes are evaluated and recognized with excellent performance, beating state-of-art CA-Polar codes. Random Linear Codes (RLCs) are also enabled to be good candidates for their security features. Owing to the advent of GRAND, the two codes, having long been neglected for error correction, become good candidates to URLLC applications, as presented in Chapter 4.

With decoding performance investigated for GRAND variants, their implementations are also studied. Computational complexity analysis is performed for each GRAND variant as well as CRC decoding. Moreover, a number of practical issues are addressed in Chapter 5 to facilitate hardware implementations of GRAND de-

coders. The investigation of GRAND from performance to implementation demonstrates GRAND's potentiality as a practical solution to URLLC applications.

Thesis Supervisor: Muriel Médard
Title: Cecil H. Green Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Ken R. Duffy
Title: Director of the Hamilton Institute, Maynooth University

Acknowledgments

It has been a long journey since I took the first engineering class in MIT. My original goal of taking classes in the school was to enhance my understanding of some engineering topics. However, lectures from several prestigious professors inspired my desire for deeper understandings, which can only be obtained by research. The journey continued, on and off, until I felt compressed to reach an achievement. After years of struggles in studying, research, programming, paper and grocery shopping, now I can see the light of the long expected achievement.

In the first place, I want to sincerely thank my research and thesis supervisor, Professor Muriel Médard, for admitting me to her excellent research team and exposing to me the fantastic research project. As an experienced engineer, I immediately realized the potentiality of the research direction and couldn't wait to be one of its contributors. It turned out to be an odyssey of hard working, but the return is fruitful. Her advice always opened a door for me when there seemed to be a dead end in research. Her kindness, patience and smiles are always encouraging and make it a happy experience to work in her team. On the other hand, her prudent attitude to academic research and high standard on paper writing have provided model examples for me to follow. My research experience under her supervision will be a life-time benefit.

Prof. Ken R. Duffy, as my co-supervisor in research, has taught me rich knowledge on technical subjects, paper writing, research tricks, and many others. His mathematician way of treating technical problems has a substantial influence on my working attitude, which is not only beneficial to my academic research but also to my industrial projects. His explanation of technical problems is always clear and down to the basic principles. It was a great experience to learn from him as a student.

I'm very grateful to Prof. Anantha Chandrakasan, the dean of engineering, for being my thesis reader. I have known him since the start of my journey in MIT many years ago. It was always a great benefit to attend his lectures in MIT or in companies. I wish to have more opportunities to learn from him, inside or outside MIT.

It was a great experience to work with Prof. Rabia Yazicigil Kirby and her hardware design team from Boston University. In working with her team on chip design projects, I had a lot of design feedback that was essential to practical implementations of the algorithms I was working on. Although I have some hardware design experience myself, the co-operation with her team brought to me the latest VLSI process information and cutting-edge design techniques. All the information has provided important reference to my research.

I wish my parents can share my happiness, though they are in heaven now. My aspiration of pursuing PhD originated from my father. He didn't have a chance to go to a graduate school in his time of China, but he ultimately became a renowned professor in aerodynamics area and supervised graduate students himself. From the many technical books and notes that he left, I can envision the passion and enthusiasm he had in his academic research in spite of the hardship imposed on him from outside. Compared to his achievement, mine might be mediocre, but still I wish he is proud of me. In my memory, my mother had forever been supportive to me. I knew that she would support me, as always, when I decided to complete my PhD program. I always felt that she was by my side when I was in difficulty. And I am certain that she is proud of me at this moment. I want to dedicate this thesis to them, my dearest parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Channel coding is an essential component of modern communication systems, which significantly enhances the communication efficiency with given bandwidth, power and noise level. Over years researchers have been seeking good codes with low-complexity decoders and their efforts result in a number of long codes with capacity-approaching performance and practical decoders. For short codes, which are gaining more attention recently for emerging new applications, many conclusions and techniques for channel codes with long lengths no longer hold, including their decoding performance and their decoders' effectiveness. Designing practical universal decoders and using them to evaluate achievable performance of short codes are the focus of this thesis, in the framework of a recently proposed innovative decoding approach.

1.1 Background

Since Shannon's 1948 opus [94] it has been known that channel capacity, the highest rate that an error correcting code can operate at while guaranteeing error-free communication over a noisy channel, is governed by the Shannon entropy of the channel's noise. By considering structureless random codes, his mathematical results proved that channel capacity is only achievable in the limit as the length of the error correcting code becomes large. By 1968, it was confirmed that his core theorems hold if structureless random codes are replaced with Random Linear Codes

(RLCs) [49], which offer a more efficiently stored code-book description. In 1978, however, Berlekamp, McEliece, and Van Tilborg reported that maximum likelihood (ML) decoding of linear codes is an NP-complete problem [20], establishing that there exists a sequence of linear codes for which the decoding complexity is exponential as a function of block length. This feature, which underpins the McEliece cryptosystem [72], effectively halted practical consideration of universal decoding algorithms, with a couple of notable exceptions recounted in the Related Work.

The focus on long codes led to a working paradigm of pairing structured codes and code-specific decoders. Examples of such pairings are Reed-Muller (RM) codes [75, 88] with Majority Logic decoding, BCH [23, 55] and Reed-Solomon codes [89] with Berlekamp-Massey (BM) decoding [21, 63, 71], Turbo codes [22] with soft detection iterative BCJR decoding [15], Low Density Parity Check Codes (LDPCs) [50] with belief propagation decoding [48, 66], and, most recently, CRC-Assisted Polar (CA-Polar) codes, used in control channel communications in 5G New Radio (NR), with CRC-Assisted Successive Cancellation List (CA-SCL) decoding [16, 78, 100]. The structured nature of these codes leads to restrictions on lengths and rates. They are usually constructed based on the assumption of independent and identically distributed noise.

Raw communication channels are seldom memoryless. Channel fading, inter-symbol interference, multi-user interference, and external noise sources all have inherent time-scales that result in time-dependent correlations in instantaneous Signal to Interference plus Noise Ratio (SINR). Essentially all forward error correction decoders assume, however, that channels are memoryless [53, 63] and, as we shall demonstrate, their performance degrades significantly if they are not.

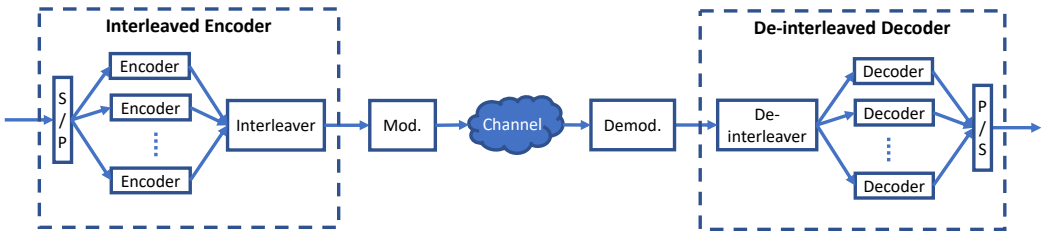


Figure 1-1: Interleavers in communication systems introduce significant latency.

As illustrated in Fig. 1-1, the engineering solution to this mismatch is to employ interleaving. In the transmitter, the interleaver permutes the location of bits across the code-words prior to their transmission. At the receiver, the de-interleaver recovers the original bit order, and the resulting signals are passed to a decoder for error correction. In this way, clumped errors are separated and distributed across multiple code-words, giving noise the appearance of being uncorrelated. Indeed, interleaving can be viewed as another layer of encoding over a larger scale of information bits. Some state-of-art long codes, such as Turbo codes, LDPC codes, and product codes [64], incorporate interleavers as part of their encoding procedure in order to achieve decoding performance that is close to capacity for a memoryless channel. The capacity of that memoryless channel is, however, lower than that of the original channel bursty channel as correlated noise has a lower entropy than independent, identically distributed noise that leads to the same average number of bit flips. Thus, interleaving maps a bursty channel to a memoryless channel with lower capacity.

In low-noise channel conditions, high-order modulations are often used to enhance spectral efficiency in wireless communications. Optical networks, linking those wireless endpoints, have also been migrating from conventional on-off-keying (OOK) formats to coherent systems with high-order modulation [74], and forward-error correction (FEC) to meet high-throughput, high-reliability criteria [104, 105]. Heavily correlated, localized errors have been found to significantly degrade the error correction performance of conventional decoders [25, 33]. High-order modulated symbols are, in particular, vulnerable to direct current (DC) offset due to the translation of received symbols in the constellation plane, which is a problem that exists in either single carrier or multi-carrier systems such as orthogonal frequency division multiplexing (OFDM) [77,111]. This type of contamination can be viewed as narrow-band noise centered at the subcarrier frequencies of signals, for which estimation of DC by averaging over a long duration of signals [68] is no longer valid. Again, coding over large data packets combined with interleavers is the conventional solution to restore memoryless-like channels.

Many emerging communication systems require low-latency operation, where small

bursts of data need efficient transmission [29, 43, 73, 80, 95]. Examples include machine communications in IoT systems [1,65,113], telemetry, tracking, and command in satellite communications [60, 69], control channels in mobile communication systems, Vehicle-to-Vehicle communications, and Ultra-Reliable Low Latency Communications (URLLC) as proposed in 5G-NR [2, 73, 80, 97, 99].

Short codes, such as CA-Polar codes adopted for 5G-NR control communications [13,78,100], are employed to meet the latency requirements of those new applications. However, since most potential short code candidates are designed to be fully functional in memoryless channels, interleavers are often inevitably needed in channels with memory, resulting in undesirable delays of the order of thousands of bits. The conflict between decoding performance and latency arouses the motivation to seek a new decoding approach that, unlike conventional decoders, has no assumption of channel noise properties or even modulation schemes.

Soft detection decoders offer a non-trivial decoding performance gain over hard decoders [63], which will be especially necessary for short, high-rate codes. However, many traditional codes do not have corresponding soft decoders. Some state-of-art codes with dedicated soft decoders, such as Turbo and LDPC codes, can reach near Shannon-capacity performance with long codes, but their performance degrades when used with short, high-rate codes. Even the celebrated CA-SCL algorithm for CA-Polar codes underperforms for short packets because of the inadequate utilization of CRC bits [10]. As a result, some conventional codes have received renewed attention [18, 82, 101], including Reed-Solomon Codes, BCH codes, Random Linear Codes (RLC) and Cyclic Redundancy Check (CRC) codes. However, these codes either are lack of efficient soft decoders, or have no effective decoders at all. Therefore, The development of practical universal decoders would open up a massively larger palette of potential code-books that can be decoded with a single algorithmic instantiation, greatly reducing hardware footprint, future-proofing devices against the introduction of new codes, and enabling the flexibility for each application to select the most suitable code-book.

Therefore, delivering URLLC necessitates efficient decoding of short, high-rate

codes, in channels with or without memory, and in binary or high-order modulation systems, motivating revisiting the possibility of high-accuracy universal decoders, which, for practical applications, have manageable complexity and power consumption.

1.2 Related Work

The challenge of channels with memory may possibly be ameliorated through the use of bespoke codes and decoders to match dedicated channel conditions. Among error correcting codes, cyclic codes, in particular, have a reputation for being capable of correcting burst errors [90], with Fire codes capable of correcting single burst error [46]. Burst-correcting convolutional codes are designed for a given burst range with a suitable guard space [70]. For storage media, single burst or multiple-dimensional error correcting codes are designed to handle burst errors, with restricted burst lengths or locations [31, 45]. These codes are specifically designed to handle certain patterns of burst errors or channel conditions, and thus cannot be applied in general URLLC applications.

Notably, Polar codes, which were the first non-random codes that were mathematically established to be capacity-achieving [13], have received significant attention for low-latency applications. Owing to their poor performance at practical block-lengths [14, 83, 93], however, they have not been adopted on their own. Instead, a concatenated design has been proposed where a CRC is first added to the data, which is then Polar coded, resulting in CA-Polar codes. These codes are usually decoded with a list decoding approach where a collection of candidate Polar code-words is first determined, and then a code-word that satisfies the CRC is selected [16, 61, 78, 100]. As they can be constructed at short block-lengths and have an efficient soft detection decoder, CA-Polar codes have been adopted for use for all control channel communications in the 5G New Radio standard [3]. Considered as a single code, a CA-Polar code is itself a linear code, albeit one that has no dedicated decoder. As a result, with an optimal decoder there is additional performance left to be squeezed of out of

them [10].

An alternate approach to designing code-specific decoders is to instead develop a universal decoder. One class of soft detection decoders that can decode any binary linear code, which works on a list-decoding principle, has been substantially investigated [17,34,47,51,106,109,112]. In Ordered Statistics Decoding (OSD), rather than compute the conditional likelihood of the received signal for all members of the codebook, instead the computation is done for a restricted list of candidate code-words that is hoped to contain the transmitted one. The algorithm permutes the columns of the parity check matrix in a manner that depends on the received signal reliability and Gaussian elimination is then performed to rewrite the generator matrix in systematic format, subject to checks that ensure a basis is identified, so that the systematic element of the code is based on the most reliable bits. Treating the code as a hash, a candidate list of code-words is determined by placing a ball of fixed Hamming distance around the reliable bits, and completing them with the hash. Transforming elements of this list back into the original basis, maximum likelihood decoding is performed on the restricted list. To achieve approximate-ML decoding performance, multiple stages of reprocessing are required, making it a challenge to implement the algorithm efficiently in hardware, especially for high throughput designs requiring large scale parallelism.

Neither CA-Polar code with its CA-SCL decoder nor the universal OSD algorithm has the potentiality of being adapted to channels with unexpected properties such as memory, indicating system failure due to performance loss or significant latency if interleavers are introduced for performance recovery. Bespoke burst error correcting decoders, on the other hand, cannot be applied to general applications, even without considering their lack of soft decoding capability.

1.3 The GRAND Algorithm

Guessing Random Additive Noise Decoding (GRAND) [35,37], first proposed in 2018 for hard detection channels, is a class of decoding algorithms that can decode any

block code. GRAND's practical promise as a single efficient mechanism for any moderate redundancy code is such that circuit-based implementations have already been investigated [4, 5, 91] that avail of the inherent high level of parallelizability of the algorithm. That work demonstrates GRAND's performance credentials in hard detection channels, such as data storage system applications or communication systems with hard detection demodulation.

GRAND's universal premise is that for a communication to be decodable the received signal must faithfully contain information regarding the transmitted code-word *and* the error effect of the noise experienced on the channel. While most decoding algorithms utilize the code-book's structure to identify the transmitted code-word, GRAND endeavors to find the effect of the noise and so recover the transmitted code-word. To do this, it requires two devices: a method by which to query if a string is an element of the code-book; and a mechanism to sequentially create putative noise-effect sequences in decreasing order of their likelihood of occurrence on the channel. Armed with these, GRAND aims to produce an error corrected decoding for *any* block code, without restriction to binary or, indeed, linear codes.

Unlike most deployed decoders, GRAND focuses on identifying the noise that impacted a communication rather than the code-word itself. Consider a transmitted binary code-word $c^n \in \mathcal{C}$ draw from an arbitrary rate R code-book $\mathcal{C}$, which is a set of 2^{nR} strings in $\{0,1\}^n$. Assume independent channel noise, N^n, which also takes values from $\{0,1\}^n$, additively alters c^n between transmission and reception. The resulting sequence is $y^n = c^n \oplus N^n$, where $\oplus$ represents addition modulo 2.

From y^n, GRAND attempts to determine c^n indirectly by identifying N^n through sequentially taking putative noise sequences, z^n, which we sometimes term patterns, subtracting them from the received signal and querying if what remains, $y^n \oplus z^n$, is in the code-book $\mathcal{C}$. If transmitted code-words are all equally likely and z^n are queried in order from most likely to least likely based on the true channel statistics, the first instance where a code-book element is found is an optimal maximum likelihood decoding [36]. GRAND's premise is that for moderate redundancy codes, there are much fewer potential noise sequences than code-words. In addition, it is established

in [36] that one need not query all possible noise patterns for the decoder to be capacity-achieving, and instead can determine a threshold number of queries, termed abandonment threshold, at which a failure to decode can be reported without unduly impacting error correction performance.

Algorithm 1 Guessing Random Additive Noise Decoding. Inputs: a demodulated channel output y^n; a code-book membership function such that $C(y^n) = 1$ if and only if y^n is in the code-book; and optional statistical noise characteristics or soft information, Φ. Output: decoded element $c^{n,*}$; and the number of code-book queries made, D, a measure of confidence in the decoding.

Inputs: Code-book membership function $C : \{0,1\}^n \mapsto \{0,1\}$; demodulated bits y^n; optional information Φ.

Output: Decoding $c^{n,*}$, soft output D

$d \leftarrow 0$, $D \leftarrow 0$.

while $d = 0$ **do**

  $z^n \leftarrow$ next most likely binary noise effect sequence (which may depend on Φ)

  $D \leftarrow D + 1$

  if $C(y^n \ominus z^n) = 1$ **then**

    $c^{n,*} \leftarrow y^n \ominus z^n$

    $d \leftarrow 1$

  end if

end while

return $c^{n,*}$, D

Pseudo-code for GRAND can be found in Algorithm 1, where the key step is "$z^n \leftarrow$ next most likely noise effect sequence". In the work that introduced GRAND the decoder only had access to a statistical description of the channel and hard-detection information. In that setting, GRAND provides ML decoding so long as the ordering of the putative noise effects matches the statistical description of the channel. For BSC channels, the statistical description is simply Hamming weights of noise sequences, which correspond to their likelihood and determine their query orders. The simplicity of GRAND's operation and the evident parallelizability of its code-

book queries have already resulted in the proposal [5] and realization [91] of efficient circuit implementations. The VLSI design in [5] focuses on maximizing throughput and minimizing worst-case latency by parallelization. The taped-out realization [91] provides a universal 128-bit hard decoder chip with class-leading measurements of precision, latency and energy per bit.

Incorporating soft detection information into decoding decisions is known to significantly improve accuracy [54, 56, 57]. Doing so requires that additional quantized reliability information be passed from the receiver to the decoder and, for GRAND, the development of an appropriate noise-effect pattern generator that can accurately and efficiently create noise-effect sequences in order of decreasing likelihood in light of that soft information.

Symbol Reliability GRAND (SRGRAND) [38, 40] is a variant that avails of the most limited quantized soft information where one additional bit tags each demodulated symbol as being reliably or unreliably received. SRGRAND retains the desirable parallelizability of the original algorithm, is readily implementable in hardware, and provides a $0.5 - 0.75$ dB gain over hard-detection GRAND [40]. At the far extreme, Soft GRAND (SGRAND) [39, 98] is a variant that uses real-valued soft information per demodulated bit to build a dedicated noise-effect query order for each received signal. Using dynamic max-heap data structures, it is possible to create a semi-parallelizable implementation in software and, being a true soft-ML decoder, it provides a benchmark for optimal decoding accuracy performance. However, its execution is algorithmically involved and does not lend itself to hardware implementation. Ordered Reliability Bits GRAND (ORBGRAND), as another soft variant of GRAND, has its basic version proposed in [41], where it only requires the rank order information of received bits by their reliability. The basic version shows good decoding performance in low-SNR scenarios but manifest performance degradation as SNR increases. Moreover, the integer partition algorithm required for pattern generation is not provided in the paper. An VLSI implementation of the basic ORBGRAND is published in [6] with noise patterns pre-generated and saved in memory, achieving desired throughput at cost of high power consumption.

While GRAND has had its ML performance and practical complexity demonstrated for BSC channels, it is not the borderline that GRAND can achieve in hard detection decoding regime. Besides memoryless channels, channels with memory are more frequently encountered in practice. Moreover, high-order modulations are widely adopted in high-throughput communications. It is of high research interest to explore GRAND's hard decoding capability in light of these extra channel and modulation information.

For soft-detection decoding, though SGRAND achieves the ML decoding performance and basic ORBGRAND manifests its complexity advantage, they represent two extreme accomplishments of soft detection GRAND. There is a need of a soft-detection variant of GRAND that keeps the complexity advantage of basic ORBGRAND and largely maintains the optimal performance of SGRAND. Moreover, the key issue of pattern generator implementation need be addressed for ORBGRAND to meet the practical requirement of power consumption.

The coverage of these topics in the thesis results in new hard and soft detection variants of GRAND, armed with which short code candidates are evaluated for URLLC applications. Combined with complexity analysis and solutions to hardware implementation issues, the thesis sufficiently illustrates the excellent performance of GRAND variants and their practical potentiality of implementations, providing solid supports to the promising prospects of GRAND in applications requiring high reliability and low latency.

1.4 Contributions and Publications

The contributions of the thesis and related publications are list below:

- Using the well-known Markov model for channels with memory, we investigate statistical properties of channels and propose the Markov-ordered GRAND (GRAND-MO) algorithm that not only ditches the need of interleavers, but also achieves significantly increasing decoding gains along with the depth of channel memory. Moreover, code-word properties that enable the outstanding perfor-

mance are discovered and can be used to guide the selection of code-books. The material is published in [9] and is presented in Chapter 2.

- The GRAND-MO algorithm is extended to high-order modulation systems and is demonstrated to preserve its excellent performance in Markovian Nearest Neighbor Error (NNE) channels, a class of Markovian channels extended to high-order modulations. In addition, by applying the decoding algorithm directly on high-order symbols, extra decoding gain is obtained due to the exploration of the modulation information that is not available in de-mapped binary bits. Computational complexity analysis is performed to both binary and high-order versions of GRAND-MO, demonstrating the algorithm's practicality. These results are included in the submitted journal paper [11], and are presented in Chapter 2.

- Similar to SRGRAND, where a single bit of reliability information brings significant decoding gain with trivial increase of decoder complexity, augmented constellation is introduced so that, with simple extra information from demodulator, not only is the excellent performance of high-order GRAND-MO extended to lower modulation orders, but also extra decoding gains are added to the already outstanding decoding performance. The result is presented in Chapter 2 and the SRGRAND result is published in [40].

- An integer partition algorithm is proposed that enables the basic ORBGRAND to be efficiently implemented in VLSI circuits. Based on it, and combined with a piece-wise linear approximation technique, we further propose the full version of ORBGRAND that overcomes the performance issue of the basic version in high SNR scenarios. Numerous algorithm complexity control techniques are shown to result in trivial performance loss, indicating the robustness of ORBGRAND. The algorithm is also demonstrated to be practical via computational complexity analysis. The soft decoder is extended to high-order modulations and presents attractive decoding gains. These results are presented in Chapter 3 and partial results are submitted in [8].

- With universal near-optimal hard and soft detection decoders obtained from GRAND, we are able to investigate existing channel codes and pick up the best one for URLLC applications. We find that the CRC code, conventionally used for error detection, turn out to be a good candidate and beat the state-of-the-art CA-Polar codes for which it serves as an assistant code. We then compare RLC with CRC code, which is also a short code candidate with security feature. The material is published in [10] and is covered in Chapter 4.

- Several hardware implementation issues are addressed for GRAND variants and CRC decoding. These techniques have either been used in simulations for published papers, or adopted in ongoing VLSI design projects. Partial results are published in [91] and the material is presented in Chapter 5.

Chapter 2

Hard Detection Decoding with Additional Information

2.1 Introduction

For URLLC applications in channels with memory, the conundrum of performance and latency may possibly be solved with bespoke codes and decoders to match channel conditions, but at the cost of significant design overhead and an unwieldy array of codes. An alternate solution, which is proposed in this chapter, is to develop a decoder that can provide effective error correction in correlated noise channels without resorting to interleaving and without altering the code to match noise structure.

The goal is accomplished by further development of the recently introduced Guessing Random Additive Noise Decoding (GRAND) algorithm to avail of the correlation in noise, which is can be used with any code structures and modulation orders. GRAND's universality stems from its effort to identify the effect of noise, from which the code-word is deduced. GRAND's operation lends itself to a development as the first decoding algorithm that is able to use existing codes but operate directly on high-order modulated symbols that are directly impacted by channel noise. In contrast, all practically applied channel decoders work on de-mapped bits and, in the core of their design, assume that it is the received bit stream that has been contaminated by noise, leveraging no modulation information.

The original GRAND design established abstract, mathematical properties of a class of Maximum Likelihood hard-detection decoders [36], but omitted physical layer and algorithmic considerations that we complete here for Markovian noise channels. In this chapter, we resolve the algorithmic details required to adapt GRAND, in a hard-detection setting, to make it suitable for channels that induce Markov correlated error bursts to either bits or higher-order modulated symbols, establishing that significant gains in Block Error Rate (BLER) performance are possible for moderate redundancy codes, while obviating the need to use interleaving and thus enabling URLLC. Central to GRAND's theoretical performance is the querying of putative noise sequences with a likelihood order that matches the channel statistics. Core to its practical realization is the ability to do so efficiently. For Markov channels and higher-order modulations, that is what GRAND-MO achieves.

While GRAND-MO is a universal decoder, in general, we observe that it works best in bursty channels with code-words that have limited structure in code-word format. For example, when a BCH[127,106] code is decoded with GRAND-MO in a binary channel subject to Markov noise, its performance improves with increasing noise correlation time-scales, while, in contrast, the performance of the standard BCH Berlekamp-Massey (B-M) decoder [21, 23, 71] quickly degrades. For highly correlated bursts, at a target BLER of 10^{-3} GRAND-MO sees a greater than 3dB gain when compared with the B-M decoder. This dramatic difference stems from the fact that GRAND-MO exploits the statistical structure of error bursts, instead of disregarding them, as standard decoders require. Notably, even with GRAND-MO, the BCH[127,106] code eventually loses its robustness when typical burst lengths are comparable to the length of the code-word. This arises because the structured pattern of BCH code-words clashes with common noise burst patterns. Using GRAND-MO's universality we find that Random Linear Codes (RLCs) do not possess this shortcoming.

Additional features are observed when GRAND-MO is extended to higher-order modulation systems. Using Quadrature Amplitude Modulation (QAM) and complex Gaussian noise, we define a hard detection channel when the receiver assumes

incorrectly that detected symbols land on nearest neighbors. We call this the nearest-neighbor-error (NNE) channel. Again, using the BCH[127,106] code as an example, even in a memoryless channel, the higher-order GRAND-MO achieves a decoding gain that is not available to conventional decoders that are designed to operate on de-mapped bits rather than symbols. This modulation-aware memoryless decoding gain increases with the modulation order, reaching ≈ 0.7dB with 256-QAM at a BLER of 10^{-3}. As Markov noise bursts are introduced into NNE channels, dramatic decoding gains are observed when the high-order GRAND-MO is adapted to the statistics of the channel, in a manner similar to results for binary systems. Moreover, the gains increase not only with channel memory but also modulation order. Therefore GRAND-MO holds particular promise for latency reduction in systems that adopt high modulation orders to meet large throughput requirements.

Computational complexity of GRAND-MO is analyzed for both binary and high-order versions, showing the feasibility of GRAND-MO for high-throughput and low-power implementations. In particular, it is noticed that GRAND-MO's complexity is bound to its BLER performance. For a given SNR, the algorithm enhancement results in improvements in both decoding performance and computational complexity, making GRAND-MO an economic solution with substantial improvements.

Like SRGRAND, in which a single bit reliability information from demodulator can enhance the decoding performance by $0.5 - 0.75$ dB, a concept of augmented constellation is proposed. With simple additional information provided by the demodulator at trivial cost, the enhancement barrier to lower-order modulation systems is removed, and further improvement is gained in higher-order modulation systems. This advantage is only available with GRAND, owing to its capability of operating directly on constellation symbols.

The rest of the chapter is organized as follows. Section 2.2 introduces the Markov channel model for binary channels as well as the Markov NNE channel for higher-order modulation. Section 2.3 provides an algorithmic description of the GRAND-MO putative error pattern generator, which is core to its performance and is matched to Markov channel statistics, and its extension to QAM schemes. Section 2.4 provides

simulated performance evaluation for Reed-Muller, BCH and RLC codes, and assesses the impact of interleavers on performance. Section 2.5 presents the performance of higher-order GRAND-MO in Markov NNE channels over various orders of QAM. The computational complexity is analyzed in Section 2.6, which is critical in demonstrating the practicality of GRAND-MO. Augmented constellation is introduced in Section 2.7 along with its performance evaluation. Section 2.8 summarizes the chapter.

2.2 Markov Channels and High-order Modulation Schemes

2.2.1 Binary Markov Channel and GRAND-MO

We consider a classic two-state Markov chain to model a binary channel with bursty errors [52]. When in the good state, G, the channel is error-free and the corresponding entry in N^n is a 0, and whenever the channel is in the bad state, B, there is an error and the corresponding entry in N^n is a 1. The transition probability from G to B is b and from B to G is g. The average error burst, where a burst is a set of consecutive 1s in N^n, is of length $1/g$ and the average length of an error-free run, a set of consecutive 0s in N^n, is $1/b$. The stationary bit-flip probability of the Markov channel is $p = b/(b+g)$ and its correlation is $1-b-g$. Note that if $b = 1-g$, the Markov channel is a memoryless BSC with $p = b$. Both b and g are assumed known and can be estimated in practice.

We seek to understand the likelihood of error sequences so that in Section 2.3 we can identify an algorithm that efficiently produces the z^n patterns sequentially from most likely to least likely for use with GRAND. Consider that $z^n_{m,l}$ has m bursts of errors, with l total flipped bits. Three cases arise:

- Case 0: $z^n_{m,l}$ begins and ends with 0s. The probability of this case is

$$P_0(m,l) = \frac{g}{1-b}\frac{(1-b)^n}{b+g}\left(\frac{bg}{(1-b)(1-g)}\right)^m\left(\frac{1-g}{1-b}\right)^l. \tag{2.1}$$

- Case 1: $z_{m,l}^n$ either begins or ends with a 1. The probability of this event is

$$P_1(m,l) = \frac{1-b}{g} P_0(m,l) \tag{2.2}$$

- Case 2: $z_{m,l}^n$ begins and ends with 1s, which occurs with probability

$$P_2(m,l) = \left(\frac{1-b}{g}\right)^2 P_0(m,l). \tag{2.3}$$

In all operating regimes, $p = b/(b+g) < 1/2$ and we therefore assume that $b < g$. If the occurrence of errors is positively correlated, $0 < 1-b-g$, so that $1 < (1-b)/g$ and hence

$$P_0(m,l) < P_1(m,l) < P_2(m,l), \tag{2.4}$$

while the reverse is true if the chain is negatively correlated. For fixed m, the probability decreases as l increases as $b < g$. If there is no memory so that the channel is a BSC, i.e. $0 = 1-b-g$, it can be directly confirmed that $P_0(m,l) = P_1(m,l) = P_2(m,l) = (1-p)^{n-l}p^l$ and that m, the number of bursts, has no relevance.

2.2.2 Higher-order Modulation and Markov Burst Channel

Among various types of higher-order modulation schemes [85], we consider QAM, owing to its high spectral efficiency and wide applications in high-throughput communication systems. Fig. 2-1 illustrates M-QAM with $M = 16$ constellation symbols and the minimum distance between symbols set to 2. Following common practice, a Gray code of length $\log_2(M)$ is used to map bit sequences to constellation symbols to ensure a single bit difference between adjacent symbols. An example Gray code is given in Fig. 2-1, where each decimal number represents a 4-bit code and serves as the index of the mapped symbol.

When transmitted, QAM symbols are disrupted by complex Gaussian noise, thus the received signals can be anywhere on the constellation plane. Hard detection decision regions are defined by central lines between symbols, as illustrated by the

dashed line square box surrounding symbol "1" in Fig. 2-1. If a received signal is within the hard detection region of its transmitted symbol, then the correct hard detection is made and, otherwise, an error occurs. As modulation schemes can be adaptively chosen so that hard detection errors are rare, in the Nearest Neighbor Error (NNE) channel we assume that hard detection errors only occur with immediate neighbors. For example, the symbols "0", "3", "5" and "9" are the nearest neighbors of the symbol "1" in Fig. 2-1. As SNR increases, the NNE channel accurately reflects the real hard detection channel, and our investigation can be extended to the full hard detection channel by including farther error symbols, albeit at the cost of additional complexity.

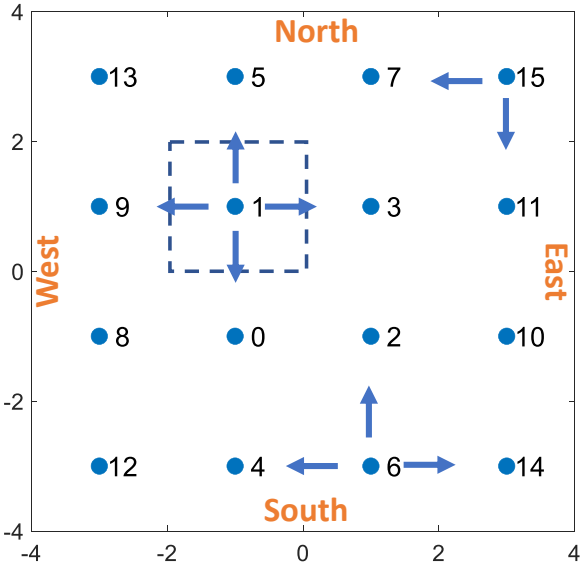


Figure 2-1: 16-QAM constellation symbols labeled with Gray code; Example hard detection region shown for symbol "1"; Error directions marked for internal symbol "1", edge symbol "6" and corner symbol "15".

For descriptive convenience, we refer to symbols inside the constellation plane as "interior" or "internal" symbols, for example "0" and "2" in Fig. 2-1. Symbols on each of the four sides of the constellation are called "edge" symbols, with "4", "5", "9" and "10" as examples. The symbols on the four corners, i.e. "12", "13", "14" and "15", are dubbed "corner" symbols. In NNE channels there are four error directions, "north", "south", "east", and "west" for internal symbols of the constellation, but there are exceptions for edges and corner symbols. For example, the edge symbol "6" in Fig. 2-1 has 3 possible

error directions and remains unchanged if noise moves the received signal south, while the corner symbol "15" has two possible error directions and stays unchanged in noise moves the received signal east or north. The transition probability is identical in all four error directions even though in some directions hard demodulation of edge or corner symbols leaves them unchanged. A transmitted symbol perturbed by mean-zero, uncorrelated, complex Gaussian noise can be described with a 2-D Gaussian distribution centered at the symbol. By computing the likelihood of the hard-detection region of the transmitted symbol, the following formula relates the channel SNR to the error transition probability p of the corresponding NNE channel:

$$p = 1 - \left(\mathrm{Q}\left(-\frac{D}{2}\sqrt{\mathrm{SNR}} \right) - \mathrm{Q}\left(\frac{D}{2}\sqrt{\mathrm{SNR}} \right) \right)^2 \tag{2.5}$$

where $D = \sqrt{6/(M-1)}$ is the distance between adjacent symbols when the average energy of the constellation is normalized to one [85]. The stationary probability of error in each direction is therefore $p/4$. To evaluated decoding performance in QAM modulation, we further map SNR to E_b/N_0 by

$$\frac{E_b}{N_0} = \frac{\mathrm{SNR}}{R\log_2(M)} \tag{2.6}$$

where R is the channel code's rate. Eq. (2.5) and (2.6) enable comparison of decoding performance between channels, modulation orders and coding rates.

In the presence of uncorrelated noise, the NNE channel is memoryless. When narrow-band noise is introduced, where the magnitude and phase change slowly during the transmission of several symbols, hard detection would result in a burst of error symbols with an identical error direction. The on and off nature of bursts can be readily modeled with the same two-state Markov chain as in Section 2.2.1. Within each burst, all symbols transition to their nearest neighbors in the same direction, which can be any one of the four error directions with equal probability. Edge or corner symbols are hard decision demodulated without error if the error direction is from the constellation center toward the edges. The resulting Markov NNE channel is similar in spirit to a two-dimensional version of the binary Markov channel in Section

2.2.1.

We extend the GRAND algorithm from operating on bits to operating on symbols by altering the construction of its query target. Rather than seeking binary error sequences directly, search patterns are generated on the symbols. Inverting the effect of a putative symbol error pattern from the received symbol sequence results in a candidate symbol sequence that is de-mapped to the bit sequence by the Gray code and the resulting binary sequence is then checked for membership of the code. If it is not a code-book member, the next most likely symbol error pattern is tested. The first instance where a code-book member is identified is the maximum likelihood decoded sequence. Guessing noise directly on symbols provides a performance improvement that is not achievable with conventional decoders as they are not aware of error directions, or of edge or corner symbols.

2.3 GRAND-MO Error Pattern Generation

2.3.1 Binary Error Pattern Generation

In order to create putative error patterns, we need to understand when, for a current pattern of m bursts with a total length of l_m flipped bits, the next most likely pattern has the same m and $l_m \mapsto l_m+1$ or we need to interlace with a pattern having an extra burst, $m \mapsto m+1$ with $l_{m+1} \geq m+1$. For bursty channels the chain will be positively correlated and so we make this determination by setting $P_0(m, l_m) = P_2(m+1, l_{m+1})$, from which we obtain

$$\Delta l = l_m - l_{m+1} = \frac{\log\left(\frac{b}{g}\right)}{\log\left(\frac{1-g}{1-b}\right)} - 1, 0 \leq \Delta l \leq n. \tag{2.7}$$

When patterns with m bursts contain $l_m > \Delta l + m + 1$ flipped bits, patterns with $m+1$ bursts are interlaced into the generative order, starting with an initial $l_{m+1} = m+1$ flipped bits. For clarity, we illustrate the operation of the pattern generation algorithm in Fig. 2-2.

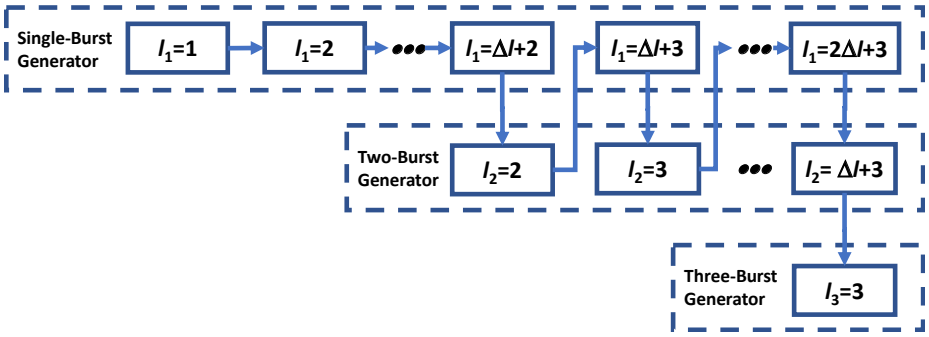


Figure 2-2: Illustration of Markov noise sequence generator

Fig. 2-3 provides a visualization of the Markov query order for strings of length $n = 6$ with $\Delta l = 2$ where each column corresponds to a putative noise sequence, a dot corresponds to a flipped bit, i.e. a 1, and the absence of a dot corresponds to a 0. Sequences are ordered in terms of likelihood from left to right. If the channel was memoryless, i.e. $\Delta l = 0$, these sequences would have non-decreasing numbers of bits flipped from left to right, as illustrated in Fig. 2-4. In a Markov channel, however, owing to the bursty structure of the noise, note that sometimes sequences of higher Hamming weight are queried before those with lighter weight. This important feature explains why the minimum Hamming distance of a code, which is an essential metric in a binary symmetric channel (BSC) and lies at the heart of much of classical code construction, has significantly less relevance for bursty channels.

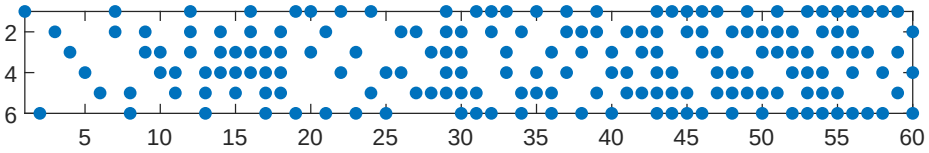


Figure 2-3: Order of patterns generated for use in GRAND-MO for a code-word of length $n = 6$, $\Delta l = 2$, maximum $m = 3$ bursts, and last number of flipped bits $l = 3$. Columns indicate patterns, which are ordered left to right in decreasing likelihood.

GRAND's query abandonment criterion for a BSC channel can be set using the random error correcting capability of the code-book [63]

$$l_{max} \geq \lfloor d/2 \rfloor \,, \tag{2.8}$$

where d is the minimum Hamming weight of the code-words in $\mathcal{C}$ and $l_{\max}$ is the

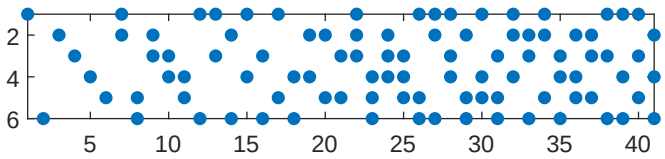


Figure 2-4: GRAND-MO pattern order for BSC channel, corresponding to $\Delta l = 0$, for a code-word of length $n = 6$, maximum $m = 3$ bursts, and final number of flipped bits $l = 3$.

maximum flipped bit in generated noise patterns. This results in GRAND making no more than $AB_{\text{BSC}} = \sum_{i=0}^{l_{\max}} \binom{n}{i}$ queries before abandoning. We then use AB_{BSC} as the abandonment condition for Markovian channels, owing the fact that correlated noise has a lower entropy than BSC noise. Therefore, in a bursty channel it is more probable for GRAND to hit a correct code-word before reaching the abandonment threshold, as confirmed in later simulations.

Algorithm 2 GRAND-MO Pseudo-Code

Input: y^n , b, g, $\mathcal{C}$

Output: $c^{n,*}$ or ABANDON

$\Delta l \leftarrow \lfloor \frac{\log(b/g)}{\log\left(\frac{1-b}{1-g}\right)} - 1 \rfloor$
$z^n \leftarrow 0, m_0 \leftarrow 1, m_1 \leftarrow 1$
$l_1 \leftarrow 1, l_2 \leftarrow 2, \ldots, l_{\lfloor d/2 \rfloor} \leftarrow \lfloor d/2 \rfloor$
if $y^n \in \mathcal{C}$ **then**
  return y^n
end if
while $m_1 \leq \lfloor d/2 \rfloor$ AND $l_{\lfloor d/2 \rfloor} \leq \lfloor d/2 \rfloor$ **do**
  for $m = m_0$ TO m_1 **do**
    $\mathcal{Z}[m, l_m] \leftarrow$ array of binary strings of length n with m bursts and l_m ones
    for all z^n in $\mathcal{Z}[m, l_m]$ **do**
      $c^{n,*} \leftarrow y^n \oplus z^n$
      if $c^{n,*} \in \mathcal{C}$ **then**
        return $c^{n,*}$
      end if

end for

$l_m \leftarrow l_m + 1$

end for

if $l[m_0] = n$ **then**

$m_0 \leftarrow m_0 + 1$

end if

if $l[m_1] = \Delta l + m_1 + 1$ **then**

$m_1 \leftarrow m_1 + 1$

end if

end while

return ABANDON

Algorithm 2 describes GRAND-MO, in which the generation of the pattern set $\mathcal{Z}[m, l_m]$ can be achieved in multiple ways depending on implementation requirements. Here we propose a method that enables evaluation of the size of the resulting pattern sets. It is performed in two steps:

- Generate all combinations that partition a collection of l_m 1s into m segments, where the number of combinations that result is $\binom{l_m-1}{m-1}$.
- Mix each of the m segments of 1s with the remaining $n - l_m$ 0s to form a test pattern, z^n, with at least one 0 bit between segments. The task can be accomplished by listing $n-l_m$ 0s in a string and inserting segments of 1s between 0 bits or at the ends, resulting in $\binom{n-l_m+1}{m}$ combinations. The total size of set $\mathcal{Z}[m, l_m]$ is, therefore, $\binom{l_m-1}{m-1}\binom{n-l_m+1}{m}$.

In general, Δl increases with channel memory via Eq. (2.7). For a channel that is memoryless, we have that $\Delta l = 0$, corresponding to a BSC. The decoder terminates when either $y^n \ominus z^n$ is in the code-book or the abandonment criterion is reached.

2.3.2 Symbol Error Pattern Generation

For higher-order modulations, GRAND-MO directly operates on symbols. With $n_s = n/\log_2(M)$ being the number of symbols, we denote by Υ^{n_s} the hard detected received

symbol sequence. The first step of guessing is to generate Markov-ordered burst formations in decreasing order of likelihood, which are defined by parameter pairs $\{[m_i, l_i], i = 1, 2, 3, \ldots\}$ with m_i and l_i representing burst number and total number of symbols in bursts, and i as the index of likelihood order. Burst formations follow the same generation steps of $z^n \in \{\mathcal{Z}[m, l_m]\}$ in Algorithm 2 for binary, with the dynamic parameter pair $[m, l_m]$ and its ordering now represented by $[m_i, l_i]$, and with the sequence size now being n_s. For clarity, we use $\{\mathcal{U}[m_i, l_i], i = 1, 2, \ldots\}$ to represent the Markov ordered sets of burst formations, in which each bit of a formation $u^{n_s} \in \{\mathcal{U}[m_i, l_i], i = 1, 2, \ldots\}$ represents a symbol, with the bit value of 0 meaning no burst and 1 indicating the symbol is in an error burst. The rule is that each segment of contiguous ones in u^{n_s} represents a burst and the corresponding segment of received symbols are erroneous in the same error direction. Burst formations in the i-th set $\mathcal{U}[m_i, l_i]$ have the same likelihood, and the order of all burst formation sets follows their Markov ordered parameter pairs $\{[m_i, l_i], i = 1, 2, \ldots\}$.

With number of bursts m, length of each burst and their locations specified in a burst formation u^{n_s}, sets of all available (up to 4 in QAM NNE model) burst patterns in each individual burst, denoted as $\mathcal{V}_i$, for $i = 1, 2, ..., m$, are collected to construct the set of combined burst patterns $\mathcal{V}[m, l]$ for u^{n_s} by Cartesian product, i.e.

$$\mathcal{V}[m, l] = \mathcal{V}_1 \times \mathcal{V}_2 \ldots \times \mathcal{V}_m. \tag{2.9}$$

Each combined burst pattern $v^l \in \mathcal{V}[m, l]$ takes the form of

$$v^l = \{v_1^{L_1}, v_2^{L_2}, \ldots, v_m^{L_m}\} \tag{2.10}$$

where $v_i^{L_i} \in \mathcal{V}_i$, for $i = 1, 2, \ldots, m$, with L_i being the number of symbols in the i-th burst, and $l = L_1 + L_2 + \ldots + L_m$. With each combined burst pattern $v^l \in \mathcal{V}[m, l]$, a testing symbol sequence t^{n_s} can be constructed by replacing corresponding symbols in Υ^{n_s} with each burst pattern $v_i^{L_i}$ at its location specified in u^{n_s}, for $i = 1, 2, \ldots, m$.

Not every generated u^{n_s} is a valid burst formation when considering the received symbol sequence. Inside a burst assumed by u^{n_s}, when all received symbols are

internal symbols, 4 patterns are possible in the burst, corresponding to the four error directions for the QAM NNE model. If a received symbol is on an edge or corner, then error directions from the edge or corner toward the constellation center are incompatible with the burst. This observation reduces the number of valid error directions for a burst, and, indeed, invalidates the whole burst formation if there are received symbols residing on all four edges in the assumed burst.

Despite eliminating error directions, edge symbols can introduce more burst patterns. For example, the symbol "6" on the south edge in Fig. 2-1 cannot be the result of error direction to the north, but can be the result of a transition from any of "4", "14" and "2" in error directions of east, west and south, respectively. The fourth possibility is that it was originally symbol "6" unchanged if the error direction is south. Similar arguments apply to the received corner symbol "15" for which two error directions are applicable and each direction comes with two cases, either with a transition or being unchanged. When e_j symbols reside on the j-th edge with valid error direction from the edge outward, there are 2^{e_j} combinations of patterns for the error direction, and the total number of patterns for the burst is $\sum_{j=1}^{h} 2^{e_j}$ with $h \leq 4$ being the number of valid error directions for the burst. Owing to independence of errors among bursts, the total number of testing patterns for a valid burst formation u^{n_s} is

$$\prod_{k=1}^{m} \sum_{j=1}^{h_k} 2^{e_{j,k}}, \tag{2.11}$$

where $h_k \leq 4$ is the number of valid error directions in the k-th burst and $e_{j,k}$ is the number of edge symbols on the j-th valid edge in the k-th burst. Since we have the usual communications model of equal transmission probability of constellation symbols and from the noise model yielding equal probability of error directions, all these patterns have equal likelihood and can be generated in arbitrary order. The instantiation of order generation is determined by the specific implementation algorithm. In the situation when all received symbols are inside the constellation, all four error directions are valid but there are no edge symbol combinations, and Eq. (2.11) reduces to 4^m, which is the minimum size of the testing pattern space for a burst

formation for QAM NNE.

With all the above pattern generation rules considered, the higher-order GRAND-MO algorithm is given in Algorithm 3, assuming that the Markov ordered burst formation sets $\{\mathcal{U}[m_i, l_i], i = 1, 2, \ldots\}$ have been obtained using Algorithm 2 following the steps for $\{\mathcal{Z}[m_i, l_i], i = 1, 2, \ldots\}$, from which the abandonment condition is inherited.

Algorithm 3 High-order Modulation GRAND-MO Pseudo-Code

Input: Υ^{n_s}, $\{\mathcal{U}[m_i, l_i], i = 1, 2, \ldots\}$, $\mathcal{C}$

Output: $c^{n,*}$ or ABANDON

1: De-map Υ^{n_s} to y^n
2: **if** $y^n \in \mathcal{C}$ **then**
3: **return** y^n
4: **end if**
5: **for** each $i \in \{1, 2, 3, ...\}$ **do**
6: **for** each burst formation $u^{n_s} \in \mathcal{U}[m_i, l_i]$ **do**
7: $m \leftarrow m_i$, $l \leftarrow l_i$
8: **if** any bursts in Υ^{n_s} defined by u^{n_s} contains symbols on all 4 edges **then**
9: Discard u^{n_s} and go to line 6
10: **else**
11: **for** j from 1 to m **do**
12: Generate all possible burst patterns $\mathcal{V}_j$ for the j-th burst of u^{n_s}
13: **end for**
14: Construct $\mathcal{V}[m, l]$ by Eq. (2.9)
15: **end if**
16: **for** all $v^l \in \mathcal{V}[m, l]$ **do**
17: Construct the testing sequence t^{n_s} from v^l
18: De-map t^{n_s} to $c^{n,*}$
19: **if** $c^{n,*} \in \mathcal{C}$ **then**
20: **return** $c^{n,*}$

end if

end for

end for

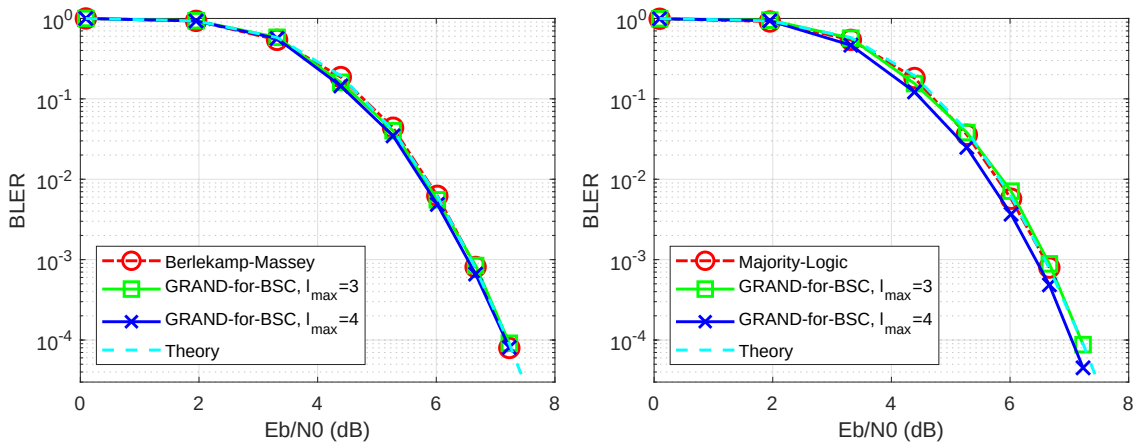
end for

return ABANDON

2.4 Performance Evaluation of Binary Systems

In the simulations, we map the stationary bit flip probability of the channel energy per transmitted information bit via the usual AWGN BPSK formula $p = b/(b+g) = \mathrm{Q}\left(\sqrt{E_b/N_0\, 2R}\right)$, where R is the coding rate.

2.4.1 Comparison with standard decoders



(a) BCH[127,106] (b) RM[128,99]

Figure 2-5: Performance of GRAND-MO in an idealized BSC channel compared to standard decoders, and with reference to theoretical random-error-correcting capability.

We first evaluate the performance of GRAND-MO in a memoryless BSC setting corresponding to idealized infinite interleaving, i.e. $b = 1 - g$. We use two distinct, standard code-book constructions, BCH and Reed-Muller (RM), for which there exists well-established BSC hard detection algorithms, Berlekamp-Massey [21, 71] for BCH and Majority-Logic [75] for RM. Fig. 2-5 reports their performance along with

standard theoretical bounds and GRAND-MO with an abandonment condition based on a BSC channel. This figure in effect serves as confirmation of the effectiveness of GRAND. Note that the random-error-correcting capabilities of B-M decoding for BCH[127,106] and of Majority-Logic decoding for the RM[128,99] code are both 3 bit flips, [63]. As GRAND-MO is a maximum likelihood decoder, in this setting it marginally outperforms both well-known decoders by going slightly further and correctly decoding some transmissions that experienced 4 bit flips. We choose $l_{max} = 4$ as the abandonment condition for BSC channel and the corresponding query number $AB_{\text{BSC}} = 1.1 \times 10^7$ as the abandonment condition for simulations in Markovian channels.

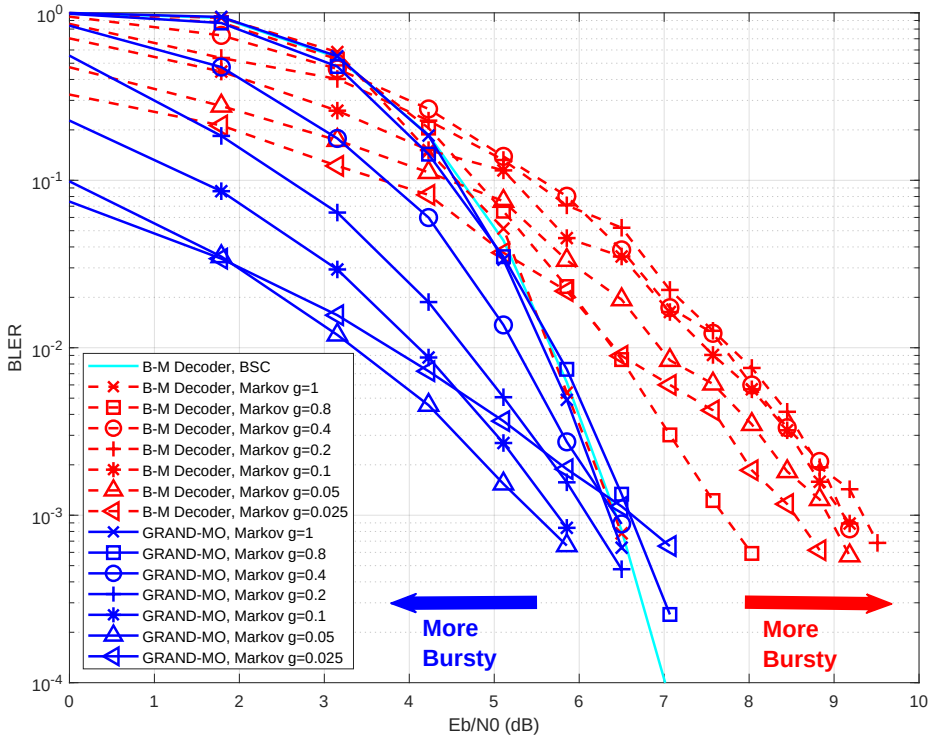


Figure 2-6: Performance of BCH[127,106] with the Berlekamp-Massey (B-M) and GRAND-MO decoders in BSC and Markov channels.

When channels have memory, however, as seen for the BCH[127,106] code in Fig. 2-6, the B-M decoder's performance degrades rapidly as the channel becomes more bursty, even with weak memory at $g = 0.8$. As B-M expects a BSC, that is not surprising. In contrast, GRAND-MO, which is adapted to the channel, resists

degradation, and instead gains substantially from additional structure of bursty noise as g increases.

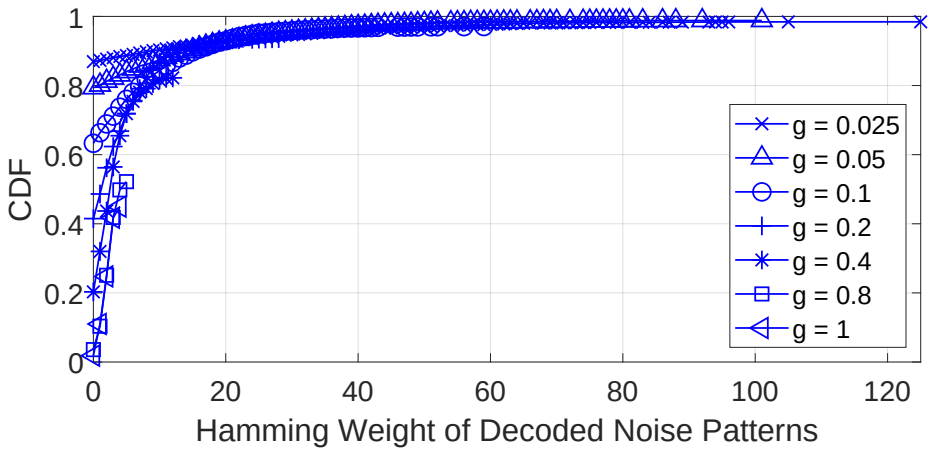


Figure 2-7: Empirical CDF of Hamming weights of noise patterns of successful GRAND-MO decoding of BCH[127,106] code in a Markov channel at $E_b/N_0 = 3.15dB$.

GRAND-MO's performance improvement is the result of its capability of decoding noise patterns far beyond the random-error-correcting capability of standard decoders. To illustrate this effect, we provide the empirical cumulative distribution function of Hamming weights of noise patterns that resulted in successful decoding in Fig. 2-7 for a single E_b/N_0 value. CDF curves starting with non-zero values indicate successful decoding that is achieved with the all-zero noise pattern. With up to AB_{BSC} query number, GRAND-MO can correct at least 4 transmission errors, and occasionally more than 100. In bursty channels, intuition from the BSC does not carry over.

The weaker performance of BCH[127,106] at $g = 0.025$ is caused by the all-one code-word in BCH code-book. Similarly, in RM codes, code-words contain groups of 1s, akin to bursts in noise. One expects this means that RM codes are not robust in bursty channels as adding a putative noise sequence that is as a burst of 1s maps more often to an existing, but incorrect, code-word. Fig. 2-8, illustrates the performance of both GRAND-MO and Majority-Logic decoders for RM[128,99] in Markov channels. A slight decrease of value of g from 1 to 0.8 results in more than 1.5dB loss in Majority-Logic decoder at a target BLER of 10^{-3}. GRAND-MO performs slightly better, but still incurs almost a 1dB loss. With more memory in channels, corresponding to

lower values of g, both decoders suffer performance degradation. This demonstrates that code-words that have patterns that are well-structured for BSC channels are not equally good when used in bursty channels, even with an optimal decoder that is matched to the channel conditions.

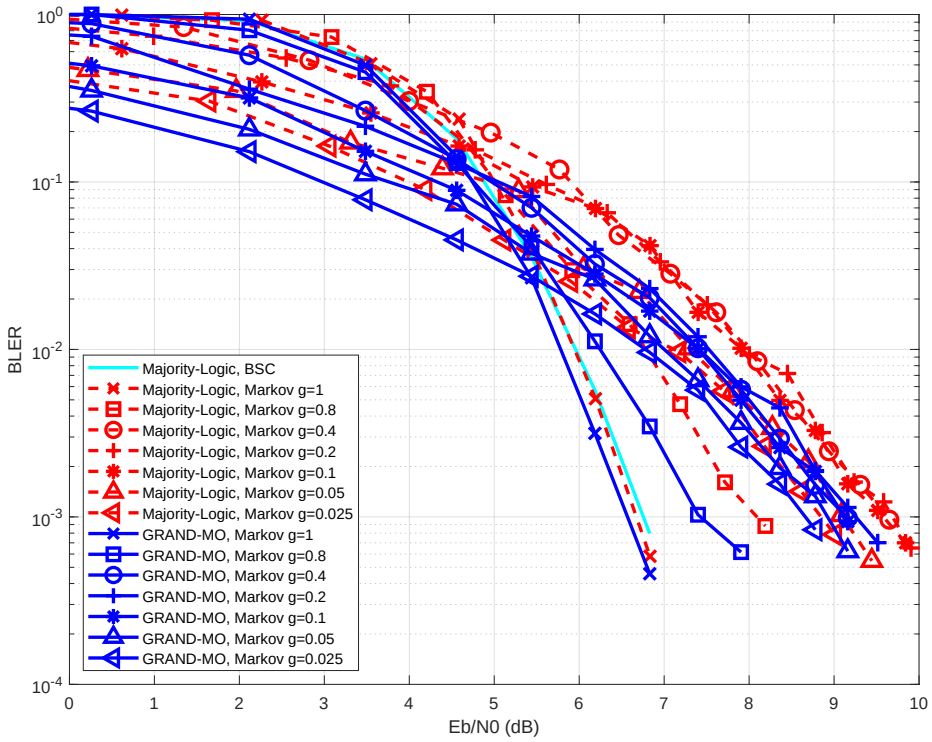


Figure 2-8: Performance of RM[128,99] with Majority-Logic decoder in BSC and Markov channels, and with GRAND-MO decoder in Markov channels

2.4.2 Performance of Random Linear Codes

The above arguments suggest that the structure of BCH codes, which have fairly even distribution of 1s, is able to benefit from decoding that takes into account channel memory but that the structure of RM codes, which exhibit groupings of 1s, is ill-suited to such decoding. In effect, RM codes are highly tailored to the BSC assumption, whereas BCH codes, which bear some resemblance in their distribution to random codes, are well suited to Markov channels and can be taken advantage of by GRAND-MO.

In this vein, we explore the use of Random Linear Code (RLCs). RLCs are linear

block codes that have long been used for proofs of capacity achievability, but have been heretofore not considered practical in terms of decodability. That is entirely understandable as one requires a universal decoder to be able to decode random codes. For GRAND-MO, RLCs are no more challenging to decode than any other linear code.

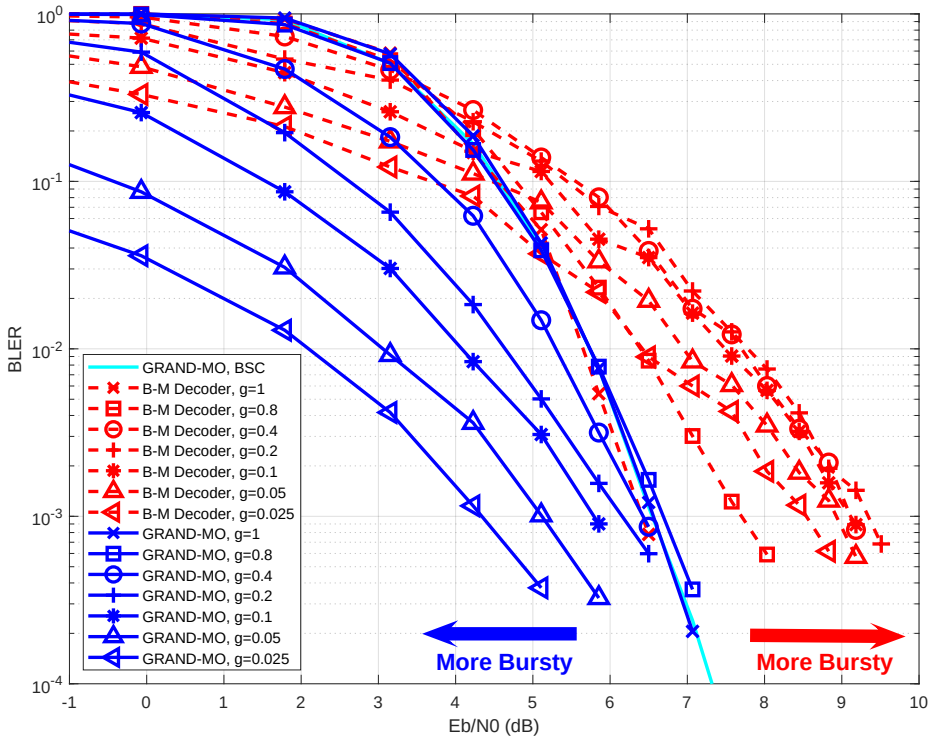


Figure 2-9: Performance of RLC[127,106] with GRAND-for-BSC (GRAND-MO with $\Delta l = 0$, red lines) and channel adapted GRAND-MO (blue lines)

Unlike structured codes that only exist for certain (n, k) combinations and so provide a limited set of rates, RLCs inherently provide complete flexibility of n and k by adjusting dimensions of the generator matrix. This desirable property allows complete flexibility in rate-matching to channel conditions. For the results in Fig. 2-9, we randomly produced a single generator matrix creating an RLC[127,106], which enabled performance comparison with the BCH[127,106] code used for Fig. 2-6. Unlike the RM code, the RLC, which inherently has limited structure in code-word patterns, proves to be robust to bursty noise. Moreover, it provides near identical error protection to the same rate BCH code, apart from when bursts are sufficiently long

on average that the BCH code's performance degrades and the RLC outperforms it. This suggests that RLCs may be well-suited to enabling URLLC.

2.4.3 Latency of Interleavers

Interleavers effectively construct long code-words by grouping shorter code-words, resulting in significant, undesirable additional latency. Moreover, by seeking to emulate the behavior of a BSC channel, using interleavers precludes the decoding benefits, shown in Section 2.4, that taking into account memory can provide. Here we investigate the extent of interleaving required to make a Markov-channel look sufficiently BSC-like so that a standard decoder can perform acceptably and evaluate the cost, in terms of loss of E_b/N_0, that comes with it.

In theory, interleaving is best achieved by a random permutation, but, in practice, matrix or block interleavers [12] are most commonly employed. By that method, a square matrix is created where coded bits awaiting transmission are written into the matrix as columns and then read out by rows. The result is that bits that are close to each other in the original order being well-separated at transmission. The de-interleaver awaits receipt of all interleaved data, and then undoes the permutation before passing the correctly ordered bits to the decoder.

As a function of the number of BCH[127,106] coded packets collected by the interleaver, Fig. 2-10 presents performance in term of the loss of E_b/N_0 experienced by the B-M decoder in the Markov channel at a BLER of 10^{-3} with respect to the corresponding BSC. The gain of GRAND-MO's channel-matching performance is presented under the same condition. By not interleaving, GRAND-MO sees a consistent 1-4dB gain over the highly interleaved B-M decoder. Moreover, to obtain that BSC performance it is necessary to interleave over hundreds of packets, amounting to tens of thousands of bits, which necessitates substantial unwanted delays. By ditching interleavers and using GRAND-MO, we gain both improved BLER performance and have eliminated an unwanted source of delay in the process.

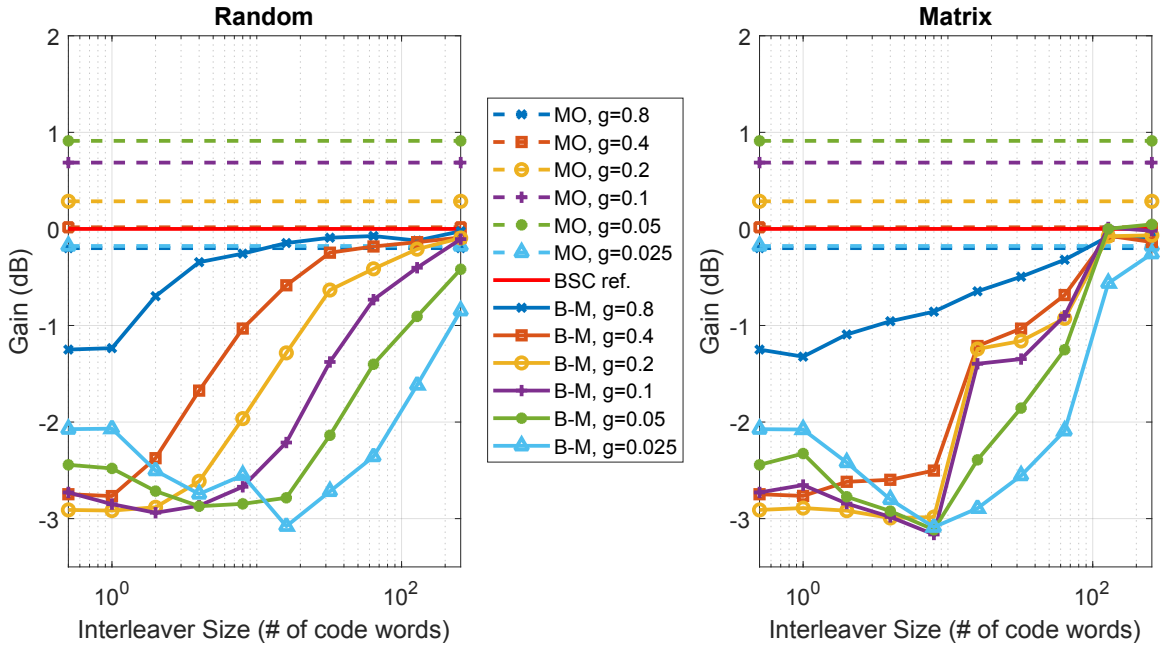


Figure 2-10: Performance of B-M decoders with random and matrix interleavers in Markov channels relative to the BSC and un-interleaved GRAND-MO decoding. Plotted is E_b/N_0 at BLER $= 10^{-3}$ for a BCH[127,106].

2.5 Performance Evaluation of High-Order Modulation Systems

2.5.1 Memoryless Channels

Following the approach in Section 2.4.1, we first investigate the performance applying GRAND to higher-order modulation in memoryless NNE channels, which we name GRAND-for-NNE. We may consider this as a special case of GRAND-MO where the burst formation in the decoder is configured by setting $b = 1 - g$. We choose the BCH[127,106] code for evaluation and compare the performance of GRAND-for-NNE to the standard B-M decoder operating on bits de-mapped from symbols. In NNE channels, the use of a Gray code ensures that a symbol error results in a single bit error, allowing us to set the abandonment condition to be the same as that in section 2.4.1. Fig. 2-11 presents the performance of both decoders for 16-QAM, 64-QAM and 256-QAM modulations with the symbol transition probability p mapped to E_b/N_0 according to Eq. (2.5) and (2.6).

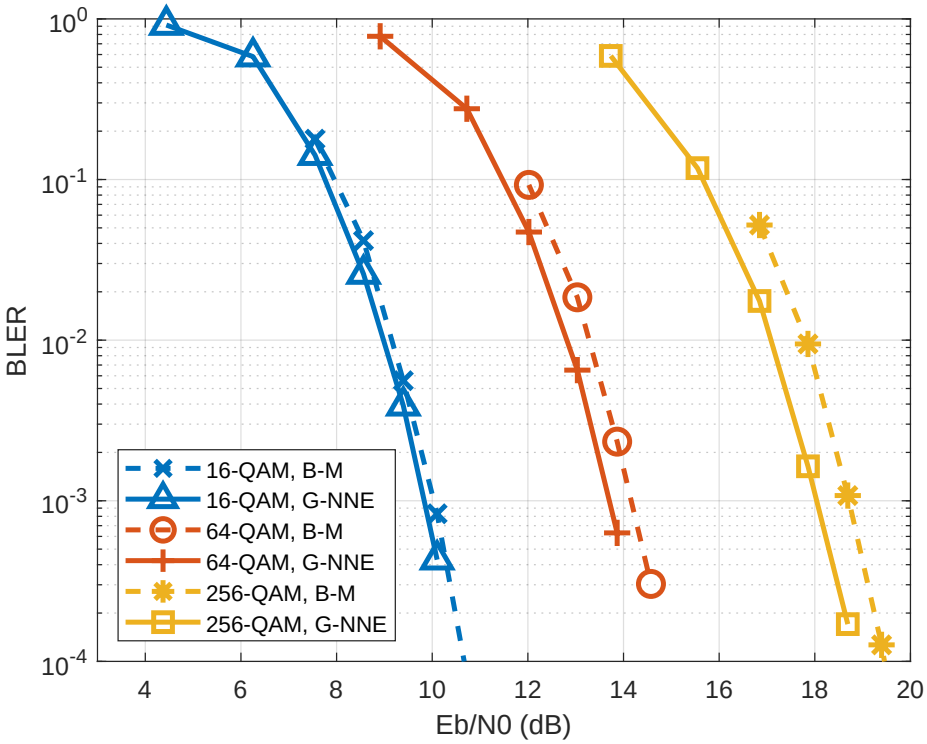


Figure 2-11: Performance of GRAND-for-NNE in memoryless channels for 16-QAM, 64-QAM and 256-QAM modulation orders, as compared to B-M decoders performance in conventional mapping/de-mapping systems

The essentially identical performance of GRAND-MO and B-M decoders as in Fig. 2-5 no longer holds for higher-order modulation. As shown in Fig. 2-11, there is an observable decoding gain for GRAND-for-NNE over the B-M decoder in 16-QAM. The gain, which stems from directly working on symbols, increases with the modulation order reaching $\approx 0.7dB$ in 256-QAM. As confirmation of the mechanism, the B-M decoder is compared with GRAND-for-BSC in Fig. 2-5, which also decodes binary sequences de-mapped from higher-order symbols. The curves in Fig. 2-12 confirm that the performance gain we observe is due to taking modulation into account. The unique capability of GRAND to operate directly on symbols rather than their de-mapped bits enables it to make use of the symbol level information that is lost in the process of de-mapping.

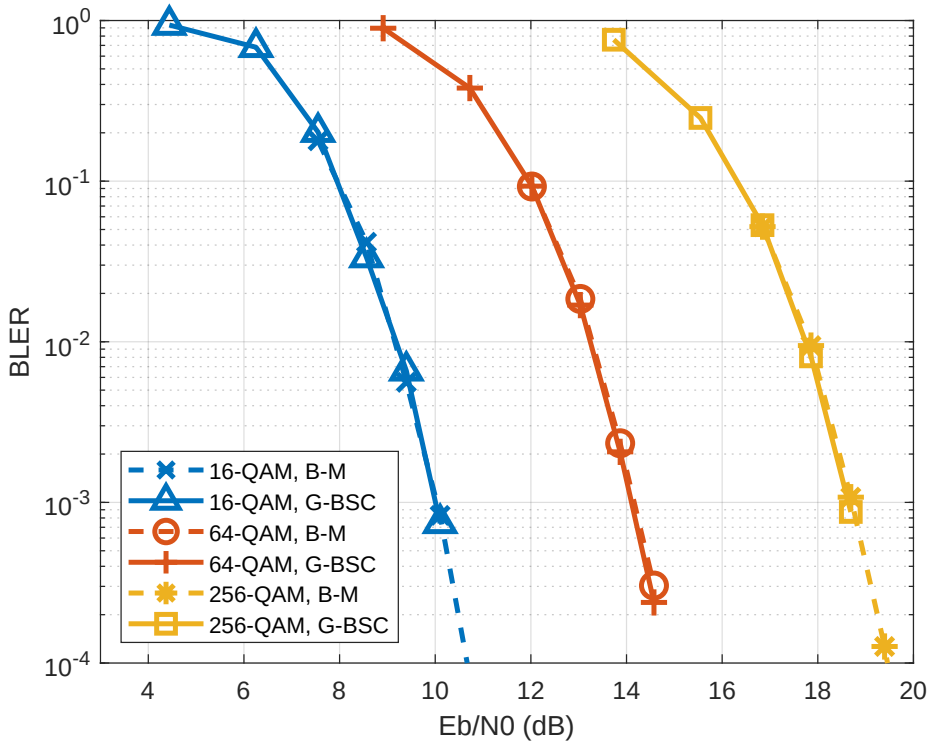


Figure 2-12: Performance comparison of GRAND-for-BSC and B-M decoders in memoryless channel for 16-QAM, 64-QAM and 256-QAM modulation orders; Both operate on binary sequences after de-mapping

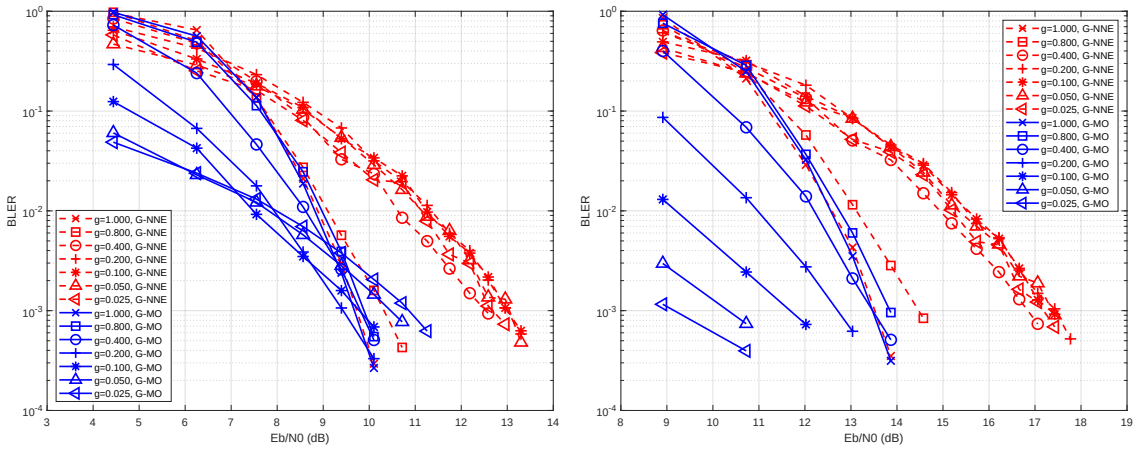
The information that conventional decoders cannot avail of is reflected in the reduced size of the potential error space when the data is considered as modulated symbols. For a n-bit code-word transmitted in BPSK, the size of hard detection

error space is $2^n - 1$. In a system with M-order modulation and NNE channel, a n-bit code-word is mapped to a symbol sequence of length $n_s = n/\log_2(M)$ for transmission and ignoring edge and corner symbols, the size of symbol error pattern space is $5^{n/\log_2(M)} - 1 = 2^{n\log_M(5)} - 1$, where each received symbol can be the result of 5 possibly transmitted symbols. When $M > 5$, the symbol error space is exponentially smaller than the bit error space, and the space size continues to shrink with increasing M, explaining the increasing decoding gain in Fig. 2-11. Edge symbols and corner symbols further reduce the size of the error space by eliminating some error directions, contributing to the decoding gain. When the received symbols are de-mapped to bits, the space of potential error pattern expands to $2^n - 1$, leading to the loss of information available to the decoder.

2.5.2 NNE Channels With Markovian Bursts

When we introduce memory in errors in NNE channels, we must adapt the pattern generator of higher-order GRAND-MO to match the channel statistics. Instead of binary B-M decoders we use GRAND-for-NNE as a reference to reflect the effect of adapting to bursts of errors. Simulation results for 16-QAM are presented on the left side of Fig. 2-13. When configured for a memoryless channel, the performance of GRAND-for-NNE degrades as channel burstiness increases. The higher-order modulation GRAND-MO, when having knowledge of burst statistics, is significantly more robust, although there is still some performance loss when the channel memory is significant. Any reduction in performance is, however, overcome in 64-QAM with a greater than 5dB gain at a BLER of 10^{-3} when $g = 0.025$, as shown on the right side of Fig. 2-13.

The barrier to enhanced performance for 16-QAM is the relatively high proportion of edge and corner symbols in the constellation, which results in a dramatic expansion in the size of the test pattern space for a given burst formation as indicated by Eq. (2.11). There is, therefore, an increased likelihood of hitting incorrect code-word in a GRAND search and a consequent reduction in performance. Once the constellation size is increased to 64-QAM, the fraction of edge symbols or corner symbols decreases



(a) 16-QAM (b) 64-QAM

Figure 2-13: Performance comparison of high order GRAND-MO and GRAND-for-NNE decoders in bursty channel for (a) 16-QAM and (b) 64-QAM modulation schemes.

sufficiently to enable significant performance gains. Increasing the modulation order results in Eq. (2.11) approaching 4^m, the smallest space of test vectors, and leads to further performance improvements. This is confirmed by the 256-QAM results shown in Fig. 2-14, where the total decoding gain achieved by GRAND-MO is multiple dB over the conventional binary Berlekamp-Massey decoder.

While high-order GRAND-MO can operate in the absence of a Gray code, its inclusion improves the decoding performance, especially in higher SNR conditions. Further improvement of performance is expected if GRAND-MO is applied to TCM/BCM systems, for which Gray code can be viewed as the simplest scheme. Interestingly, when decoded with symbol-based GRAND, the BCH[127,106] does not suffer performance degradation caused by ultra long binary bursts encountered in Fig. 2-6, a behavior due to the use of a Gray code, which helps to break the format of the BCH all-one code-word. The Gray code guarantees that there is only 1 bit error for each symbol error, and therefore avoids chains of bit errors from being predicted by investigating symbol errors. Such behavior avoids the pitfalls of the original binary searching, further buttressing our statements on code-word format in Section 2.4.1. As Gray codes are widely applied in higher-order modulation schemes, we deduce that most existing channel codes can still be adapted to URLLC applications without employing

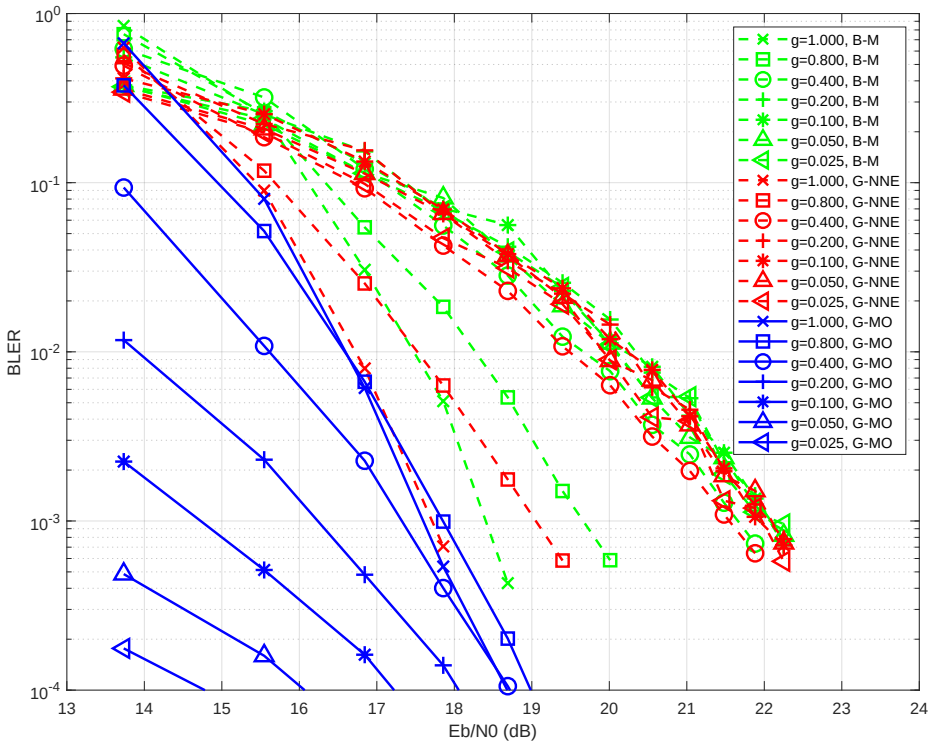


Figure 2-14: Performance comparison of GRAND-MO, GRAND-for-NNE and B-M decoders in bursty channel for 256-QAM modulation orders

interleavers if GRAND is adopted as the decoding solution.

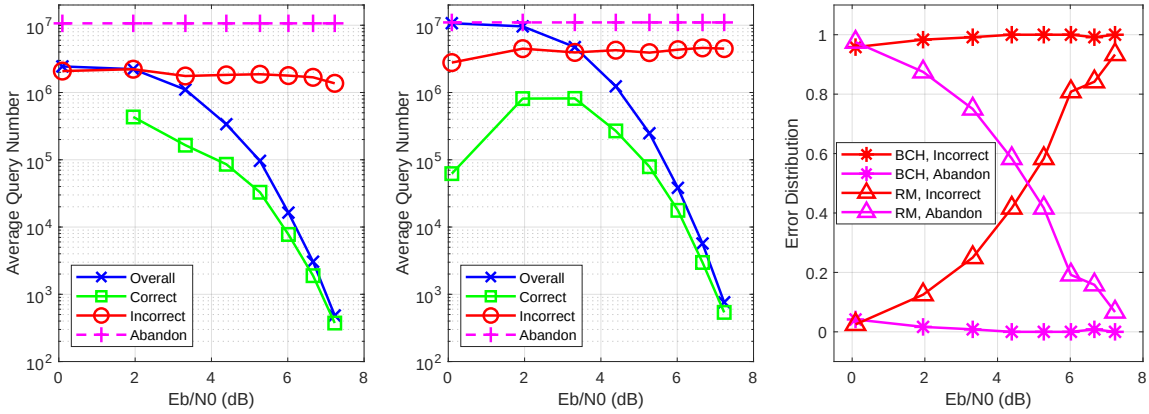
2.6 Computational Complexity of GRAND-MO

In GRAND framework, two core operations of decoding, the noise pattern generation and the code-book checking, consume the major computation of code-book query. Therefore, the average number of code-book checks, named as the computational complexity of the decoder, directly determines the average energy consumed in decoding, and is an essential metric for practical implementations of GRAND decoders.

Here we want to clarify the difference between algorithm complexity and computational complexity. The two notations are normally exchangeable in conventional decoders, where every received sequence goes through the same decoding processes in spite of channel conditions. For GRAND, they are no longer equivalent for complexity measurement. Algorithm complexity is related to the complication and scale

of circuit implementations, which should be in practical scale for GRAND variants to be implementable, especially with welled designed pattern generator. Computation complexity, on the other hand, is measured by the amount of time (or cycles) to accomplish the decoding computation, which may change not only between code-words, but also among various channel conditions.

Overwhelming algorithm complexity of an algorithm, such as SGRAND, makes it unfeasible for implementations due to the impractical size and complexity of VLSI circuits. For algorithms with algorithm complexity appropriate for implementations, such as GRAND-MO, the power consumption is, then, determined by the computational complexity.

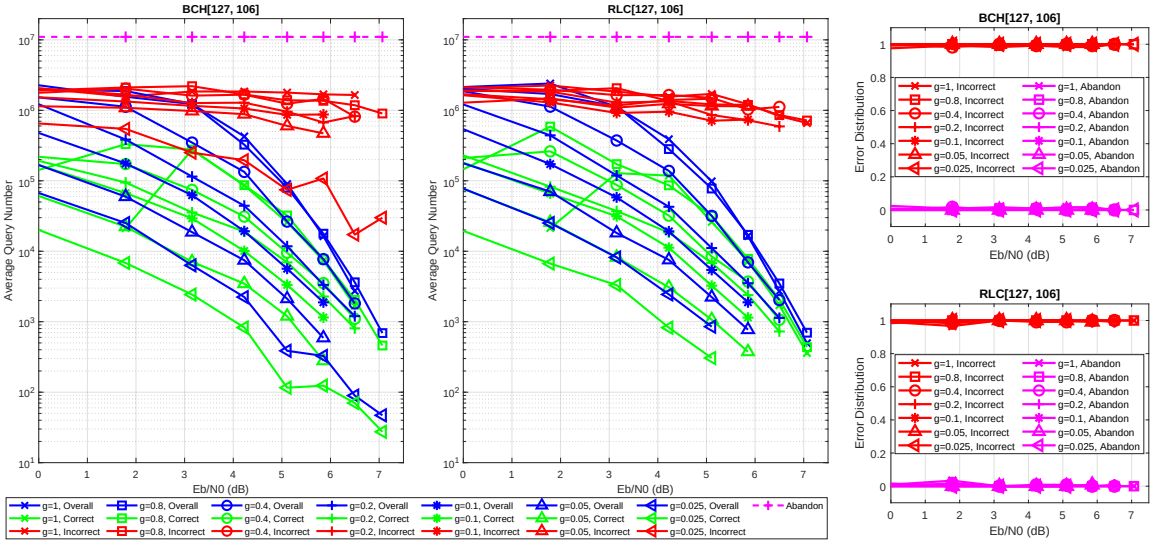


(a) BCH[127,106] (b) RM[128,99] (c) Error distribution

Figure 2-15: Complexity of GRAND-MO for (a) BCH[127,106] and (b) RM[128,99] for simulations with $l_{max} = 4$ in Fig. 2-5; "Overall" for any code-word, "Correct" for correctly decoded code-words, "Incorrect" for incorrectly decoded code-words, and "Abandon" for the abandonment condition; (c) Distribution of errors in the incorrectly decoded and the abandoned for both codes

As a baseline, we begin with the computational complexity investigation for the BSC scenario evaluated in Fig. 2-5, as reported in Fig. 2-15. A key feature of all GRAND algorithms is that their complexity decreases as channel condition improves. Fig. 2-15c shows that almost no abandonment is observed for the BCH[127,106] code before it enters the operating region where BLER $\leq 10^{-3}$, indicating that most errors are contributed by incorrect decoding and essentially maximum performance has been achieved with the given abandonment condition. In contrast, the RM[128,99]

decoding experiences more abandonment due to the code's higher minimum distance of 8, but its proportion quickly reduces as SNR increases. At a BLER of 10^{-3}, the BCH[127,106] code has an average complexity of about $3,000$ queries which can be efficiently accomplished with sequential or parallel VLSI circuits. The complexity further reduces to around 500 as the BLER improves to 10^{-4}. This feature makes GRAND a good candidate for ultra-low power decoding. The RM[128,99] code shows exhibits higher query numbers due to its lower code rate, but the complexity also quickly drops to a low level, facilitating practical implementation.



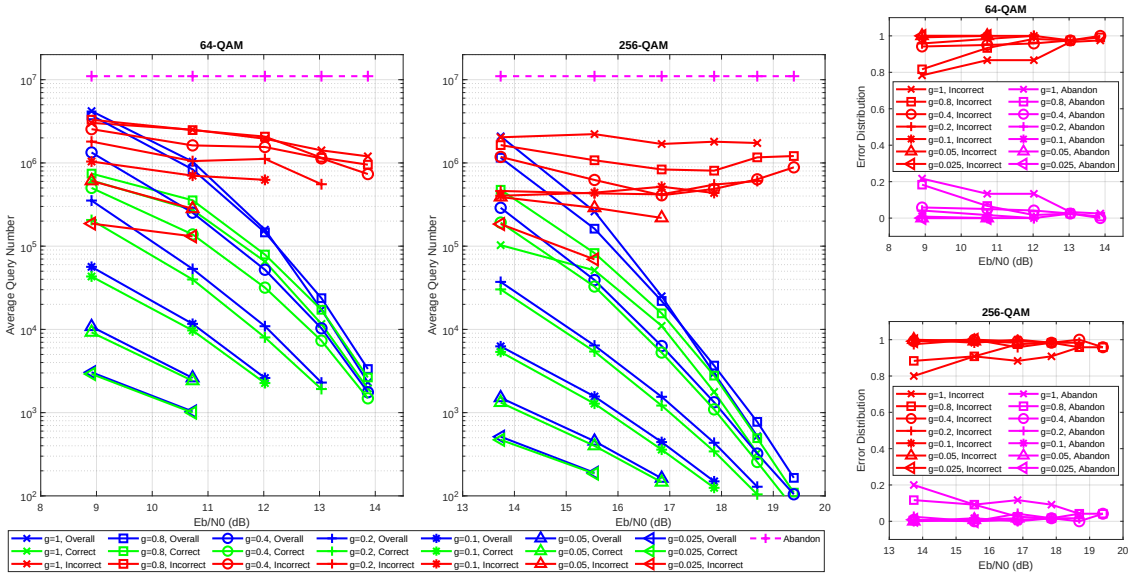
(a) Computational Complexity (b) Error distribution

Figure 2-16: (a) Computational complexity of GRAND-MO for the scenario in Fig. 2-6 and Fig. 2-9, in term of the average number of code-book queries until a decoding is found. (b) Error distribution between abandoned code-words and incorrect decoding.

Fig. 2-16 plots the computational complexity and error distribution of decoding BCH[127,106] and RLC[127,106] in Markov channels. There is almost no decoding abandonment in all simulated region, suggesting there is little room for improvement with AB_{BSC} as the abandonment condition. As in BSC channel, at a BLER of 10^{-3} the average query number in decoding the BCH[127,106] is approximately $3,000$ irrespective of the value of g, which is indicative of the fact that the computational complexity of GRAND algorithms are bound to their BLER performance rather than channel conditions. In essence, a more accurate query order can achieve the target

BLER at lower E_b/N_0 leading to more coding gains. A notable exception is with $g = 0.025$, where incorrect decoding with low query number results in less computation at BLER of 10^{-3} that, however, is achieved at higher E_b/N_0. In contrast, RLC[127,106] has a steady complexity of around $3,000$ at a BLER of 10^{-3}, with all values of g, and with constantly improving decoding gain.

The ready parallelization of code-book queries has been exploited in circuit designs to establish that BSC versions of GRAND can deliver universal high-throughput, or lower power, decoding [5,6,91] for all moderate redundancy codes. GRAND-MO also benefits from that potential parallelization and circuit designs have been proposed for its implementation on binary signals [4]. In addition, by circumventing the need for interleavers, simultaneously improving decoding performance while also saving significant memory and computation, GRAND-MO is a promising practical solution to URLLC applications.



(a) Computational Complexity (b) Error distribution

Figure 2-17: (a) Computational complexity of high-order GRAND-MO on BCH[127,106] with 16-QAM and 256-QAM for simulations in Fig. 2-13b and Fig. 2-14 respectively. (b) Error distribution between abandoned code-words and incorrect decoding

The complexity evaluation of high-order GRAND-MO is presented in Fig. 2-17 for 64-QAM and 265-QAM, where abandonment hits are observed for some channels

with weak memory. However, the close distance between "overall" and "correct" curves suggests a low proportion of abandoned or incorrectly decoded code-words, indicating that little performance improvement is expected with more generous abandonment conditions. Referring to the performance in Fig. 2-13b and Fig. 2-14, it is found that at BLER of 10^{-3}, the average overall query number is approximately 3000 for both 64-QAM and 256-QAM, similar to binary systems in BSC and Markovian channels as shown in Fig. 2-15 and Fig. 2-16 respectively. As in reports for binary systems and other GRAND algorithms, the observation demonstrates that the GRAND's computational complexity decreases as its BLER decreases for these channel conditions and modulation schemes. Due to the large decoding gain, under the same noise condition, the achievement of low BLER also leads to low complexity. Combined with the computation saved from removing de-mapper and interleaver, high-order GRAND-MO is, we suggest, a good candidate for low-power and high-throughput decoder solutions for URLLC applications.

2.7 Augmented Constellation

As indicated by Eq. 2.11 in Section 2.3.2, edge or corner symbols enlarge the search space of test patterns, making it the barrier to performance enhancement of lower modulation orders in which those symbols are not trivial, as shown in Fig. 2-13 with 16-QAM for example. In fact, the demodulator is able to see if a received signal lands inside or outside the constellation when performing hard-detection. Due to the procedure of de-mapping symbols to bits required by conventional decoding algorithms, the demodulator has to link any received signals outside the constellation to either edge or corner symbols, discarding error direction information, and then leading to the expansion of testing pattern space for GRAND-MO.

Similarly, in binary systems a demodulator can easily tell how reliable a received signal is when it is mapped to a hard detection bit. This information can be quantized to a single bit and conveyed to the decoder with trivial cost of complexity. Even with the single-bit information, SRGRAND [38, 40], a variant of GRAND, has its decoding

performance enhanced by $0.5 - 0.75$ dB.

Following the same principle, we propose the augmented constellation in demodulator to preserve and hand over the above mentioned error direction information to GRAND-MO decoder. As shown in Fig. 2-18, additional symbols are added surrounding the original constellation, and are labeled by the indices of their nearest edge or corner symbols with the direction information attached. In the augmented constellation, all edge and corners symbols in original constellation become internal symbols and therefore have the same decision regions for internal symbols. Symbol "15", originally a corner symbol, now has 4 neighbors, with "15N" and "15E" as new neighbors to the north and the east respectively. When the received signal falls outside of the hard-detection region of "15" to the north, the demodulator delivers symbol "15N" to GRAND-MO, from which the decoder not only knows the existence of error but also its direction.

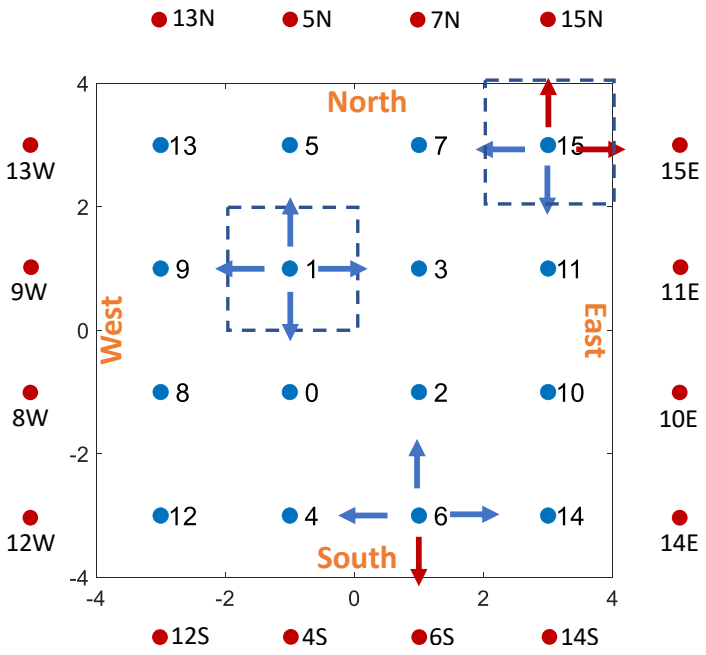


Figure 2-18: Augmented 16-QAM constellation formed by appending augmented symbols to the original 16-QAM constellation; Additional symbols are labeled following their closest constellation symbols with error directions attached.

Armed with the augmented constellation, the high-order GRAND-MO algorithm can be updated with the validation of a generated burst formation u^{n_s} adjusted as

follows. In each assumed burst of a burst formation, if multiple augmented constellation symbols are found from different error directions, then the burst formation is invalidated. Otherwise, only one error direction is possible for the burst which is determined by the augmented constellation symbols. A burst without augmented constellation symbol has four possible error directions. With these updated rules, the total number of testing patterns for a valid burst formation u^{n_s} becomes,

$$\prod_{k=1}^{m} 4^{e_m} \leq 4^m, \text{ where } \begin{cases} e_m = 0, \text{ if valid augmented symbols exist} \\ e_m = 1, \text{ if no augmented symbol exists} \end{cases} \tag{2.12}$$

Eq. (2.12) indicates that the number of resulted patterns can be much smaller than 4^m, which is the minimum size achievable with ordinary constellation, according to Eq. (2.11). Thus, not only is the set size of all possible burst formations reduced by the stricter validation rules imposed by the augmented constellation, but also the number of all possible testing patterns for each validated burst formation can be smaller than the minimum size achievable with the ordinary constellation, indicating a significant reduction of pattern search space, and consequently considerable performance improvement.

The expected performance improvement is confirmed in Fig. 2-19, where the simulation in Fig. 2-13 is repeated with the augmented constellation. The barrier to performance improvement on 16-QAM shown in Fig. 2-13 is completely removed and multiple-dB decoding gain is obtained. Even for 64-QAM and 256-QAM, which have already exhibited excellent performance with their original constellations as shown in in Fig. 2-19 and Fig. 2-20 respectively, further decoding gain are produced with the adoption of the augmented constellation, especially in channels with strong memory. The amount of additional decoding gain in 256-QAM is relatively less than 64-QAM due to the lower portion of edge and corner symbols in its original constellation.

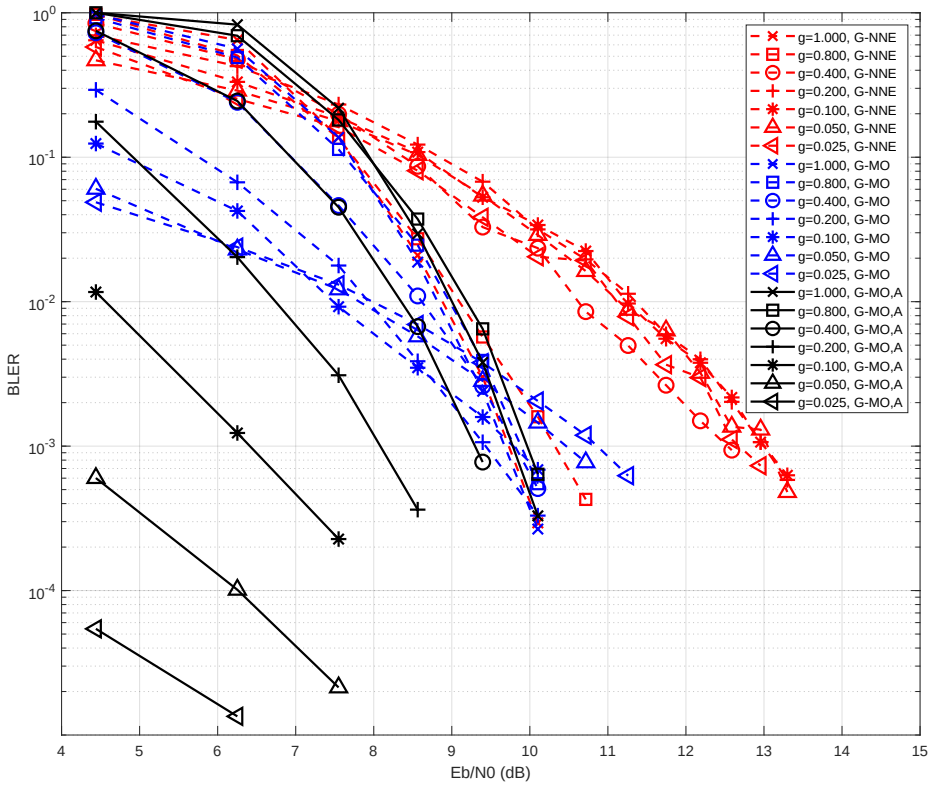


Figure 2-19: Performance of high order GRAND-MO with augmented constellation (G-MO,A), as compared to GRAND-for-NNE and GRAND-MO decoders in bursty channel for 16-QAM modulation scheme

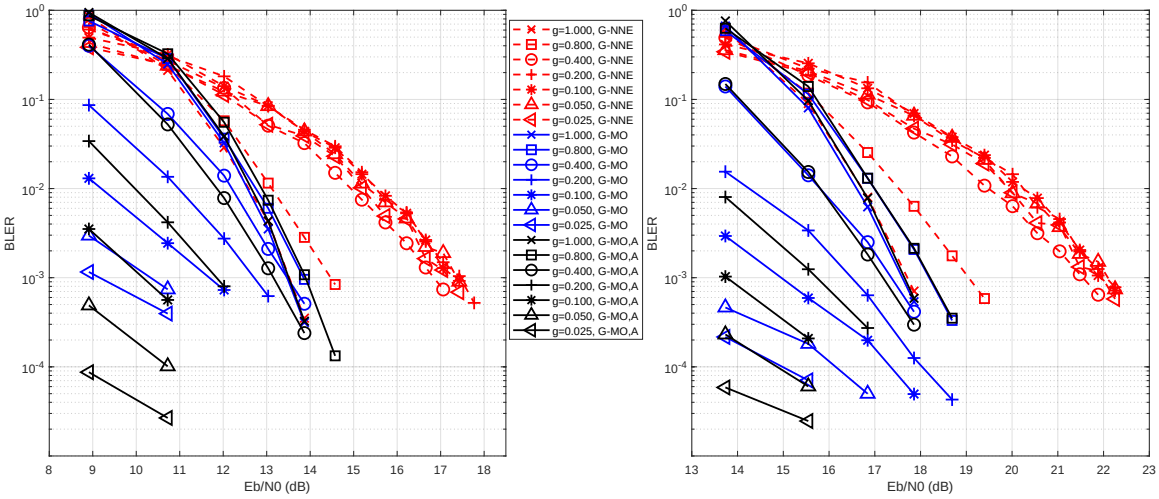


Figure 2-20: Performance of high order GRAND-MO with augmented constellation (G-MO,A)in bursty channel for 64-QAM and 256-QAM modulation schemes

2.8 Summary

The drive to enable ultra reliable low latency communications for modern applications is hindered by the assumption, common to standard decoders, that channels are memoryless. To match that assumption, interleavers buffer large volumes of data before transmission and de-interleavers must wait for that data to be delivered before decoding can begin, resulting in unwanted delays. To circumvent interleavers requires decoders that can directly manage channels subject to burst errors.

As one potential solution, here we further develop the GRAND approach to make it suitable for use in channels with bursty noise. By analyzing a Markov model of channel noise, we develop an efficient algorithm for sequentially producing putative noise patterns in order of their likelihood. Using those patterns in GRAND results in GRAND-MO, a hard detection decoder that can accurately decode any moderate redundancy code. Using it we establish that not only are there latency reductions to be had by ditching interleavers, but there are also significant BLER performance gains to be availed of by the retention and use of statistical structure of noise bursts.

Using GRAND-MO's ability to decode any code, we find that performance with structured code-word patterns of Reed-Muller and BCH ultimately degrades as channels become more bursty. Random Linear Codes, which have been little investigated outside of theory, however, appear robust to channel conditions. As a result, one possible way forward is to ditch interleavers and use RLCs decoded by GRAND-MO to enable accurate, high-rate ultra-low latency communications. The complexity analysis presented here further supports that proposal.

A distinctive aspect of GRAND algorithms that had previously only be discussed theoretically [36] is that they can operate directly on higher-order symbols. We realize that potential here by exploiting this feature to operate directly on higher-order modulated symbols rather than their binary de-mapping. Higher-order modulation schemes are typically adopted in high throughput URLLC applications as well as in backbone optical networks connecting wireless endpoints, whose contribution to end-to-end latency is often overlooked. Working on symbols unlocks performance gains of

≈ 0.7 dB over optimal maximum likelihood decoding of demodulated binary strings in memoryless channels. This comes about as the size of the potential error space is maintained instead of exponentially increased when received signals are regarded as higher-order symbols.

When interleavers are removed and GRAND-MO is extended to higher-order modulation, statistically aligning putative noise queries with the channel, gains are compounded and are significant. With GRAND-MO's complexity decreasing with BLER performance, the large decoding gain also leads to significant drop in computation for a given noise condition. Interestingly, widely used Gray codes help to diminish the negative impact of structured format of code-words, making higher-order GRAND-MO directly applicable in many existing systems. As a result, GRAND-MO seems particularly well suited to high throughput, low-latency designs through the removal of interleavers and to direct operation on symbol level information.

With the augmented constellation introduced in the receiver, the barrier of performance improvement is removed for lower order constellations that have considerable portion of edge or corner symbols. The significant improvement is demonstrated with 16-QAM. Higher modulation orders such as 64-QAM and 256-QAM also benefit with extra multi-dB gain from the augmented constellation in addition to their already excellent performance with GRAND-MO. This is similar to SRGRAND, which, with a single-bit reliability information easily obtained from demodulator, achieves significant decoding gains with little complexity increase.

The algorithms introduced here result in significant, multi-dB, gains by exploiting the statistical correlation structure of noise and modulation information, demonstrating GRAND's potentiality in hard detection decoding, which is far beyond what conventional decoders can achieve. In addition, its practicality is justified with computational complexity analysis, indicating its feasibility to practical implementations.

Chapter 3

Soft Detection Decoding with ORBGRAND

3.1 Introduction

The advantage of employing soft information in decoding is a well-known fact, which is also confirmed in GRAND approach with SRGRAND [40] that achieves significant decoding gain with a single bit of reliability information and trivial complexity cost. The achievement of ML performance in GRAND approach has been effectively demonstrated by SGRAND [98]. However, the amount of computation involved in its execution makes it more appropriate for an off-line performance reference than a real-time decoder solution.

Ordered Reliability Bits GRAND (ORBGRAND) bridges the gap between SRGRAND and SGRAND by obtaining the decoding accuracy of the latter in an algorithm that is, by design, suitable for implementation in circuits. A preliminary version of ORBGRAND that provides near-ML performance for arbitrary length, moderate-redundancy codes and block error rates (BLER) greater than 10^{-3} was presented at IEEE ICASSP in 2021 [42]. Its promise for a highly parallelized hardware realization has already resulted in VLSI architecture being proposed [6] that focuses on maximum throughput and minimizing worst-case latency.

The performance limitation of the basic ORBGRAND is due to its utilization

of only the ordering of received bits by their reliability, and discarding all other information, which also explain its algorithm simplicity. To retain the performance achieved by SGRAND, more soft information need be included, while the algorithm complexity must be watched to avoid complexity expansion to the level of SGRAND. The algorithm development is a trade-off between performance and complexity, which is the common practice for the design of practical decoders.

The chapter is organized as follows. In Section 3.2, we first explain the rationale behind ORBGRAND and its practical implementation, which leads to the basic and full versions of ORBGRAND. Performance evaluation results that demonstrate ORBGRAND's effectiveness are presented in Section 3.3. The practicality of ORBGRAND is supported by the analysis of its computational complexity in Section 3.4. A set of algorithm complexity control techniques are proposed in Section 3.5 to further facilitate hardware implementations. Then similar to GRAND-MO, Section 3.6 extends the algorithm to high-order modulation systems. Section 3.7 closes with the summary.

3.2 ORBGRAND

We first introduce the principle behind ORBGRAND's design, before explaining how the basic and full variants are implemented.

3.2.1 ORBGRAND Principles

An n-bit binary block code-word $c^n \in \{0,1\}^n$, is modulated to $\text{mod}(c^n) \in \{-1,1\}^n$ by $\text{mod}(c_i) = 2c_i - 1$, transmitted and impacted by independent continuous additive noise, $N^n \in \mathbb{R}^n$, resulting in a random received signal $Y^n = \text{mod}(c^n) + N^n$, from which the hard decision sequence $y^n = \text{demod}(Y^n)$, an estimate of c^n, is obtained. The noise effect is the difference between what the transmitted binary code-word and the demodulated received signal, $Z^n = c^n \ominus y^n$. All soft detection GRAND algorithms make queries seeking to identify the noise effect, Z^n, rather than the

original continuous noise on the channel N^n. With the log-likelihood ratio defined as

$$\mathrm{LLR}(Y_i) = \log \frac{P(Y_i|c_i = 1)}{P(Y_i|c_i = 0)},$$

the hard detection y_i is obtained from Y_i by $y_i = (\mathrm{sign}(\mathrm{LLR}(Y_i))+1)/2$, and $|\mathrm{LLR}(Y_i)|$ is referred to as the reliability of y_i.

While there are many ways to quantitatively capture the soft information in Y^n, for ORBGRAND it is instructive to represent it as a sequence, $B^n = (B_1, B_2, \ldots, B_n)$, where B_i is the *a posteriori* probability that the hard decision bit y_i is in error, which can be written as

$$B_i = P(y_i \neq c_i|Y_i) = \frac{P(y_i \neq c_i, Y_i)}{P(Y_i)} = \frac{P(y_i \neq c_i, Y_i)/P(y_i = c_i, Y_i)}{1 + P(y_i \neq c_i, Y_i)/P(y_i = c_i, Y_i)},$$

and so B_i can be expressed in terms of the bit reliabilities as

$$B_i = \frac{e^{-|\mathrm{LLR}(Y_i)|}}{1 + e^{-|\mathrm{LLR}(Y_i)|}} \in [0, 1/2], \tag{3.1}$$

where B_i monotonically decreasing with $|\mathrm{LLR}(Y_i)|$. From B^n we can evaluate the *a posteriori* probability of a binary noise-effect sequence z^n and establish it satisfies

$$\begin{aligned} P(Z^n = z^n) &= \prod_{i:z_i=0} (1 - B_i) \prod_{i:z_i=1} B_i = \prod_{i=1}^{n} (1 - B_i) \prod_{i:z_i=1} \frac{B_i}{1 - B_i} \\ &\propto \prod_{i:z_i=1} \frac{B_i}{1 - B_i} = \exp\left(- \sum_{i:z_i=1} |\mathrm{LLR}(Y_i)| \right) = \exp\left(- \sum_{i=1}^{n} |\mathrm{LLR}(Y_i)| z_i \right) \end{aligned} \tag{3.2}$$

Therefore, the relative likelihood of a putative noise effect sequence z^n is determined by the sum of the reliabilities of hard-detected bits being flipped, $\mathrm{Rel}(z^n) = \sum_{i:z_i=1} |\mathrm{LLR}(Y_i)|$, which can be viewed as its Hamming weight weighted by the reliability of those bits. To rank order putative noise sequences, z^n, in decreasing likelihood, it is, therefore, sufficient to rank order them by increasing reliability sum, $\mathrm{Rel}(z^n)$.

If no soft information is available, defining $|\text{LLR}(Y_i)|$ to be an arbitrary positive constant for all i, $\text{Rel}(z^n)$ is proportional to the Hamming Weight of z^n, $w_{\text{H}}(z^n) = \sum_{i=1}^{n} z_i$. In this case, putative noise sequences would be rank ordered in increasing Hamming weight, as used in the original hard detection GRAND for a binary symmetric channel. SRGRAND filters $|\text{LLR}(Y_i)|$, setting it to be $+\infty$ if it is above a threshold that depends on SNR and code-redundancy, and to a finite, positive constant if below that threshold, resulting in putative noise sequences being be rank ordered in increasing Hamming weight within the masked region of finite reliability bits. Armed with $\{|\text{LLR}(Y_i)| : i \in \{1, \ldots, n\}\}$, true soft ML decoding is achieved by SGRAND [98] using a dynamic algorithm that recursively generates a max-heap for each set of reliabilities to generate z^n with increasing $\text{Rel}(z^n)$. Our goal with ORBGRAND is to obtain comparable performance with an algorithm that is amenable to efficient implementation by design.

For notational simplicity, we shall assume that the reliabilities, $\{|\text{LLR}(Y_i)| : i \in \{1, \ldots, n\}\}$, happen to be received in increasing order of bit position, so that $|\text{LLR}(Y_i)| \leq |\text{LLR}(Y_j)|$ for $i \leq j$. In practice, for each received block we sort the reliabilities and store the permutation, $\pi^n = (\pi_1, \ldots, \pi_n)$, such that π_i records the received order index of the i^{th} least reliable bit. The permutation π^n enables us to map all considerations back to the original order that those bits were received in.

The core of the approach underlying ORBGRAND is the development of statistical models of the non-decreasing sequence $\{|\text{LLR}(Y_i)| : i \in \{1, \ldots, n\}\}$ that are accurate, robust, and lead to computational efficient algorithms for generating rank ordered putative noise sequences. The approach can be most readily understood with the example of a channel using BPSK modulation that is subject to Additive White Gaussian Noise (AWGN), where $\text{LLR}(Y) \propto Y$. As constants of proportionality will prove to have no impact on ORBGRAND's order, from here on we will refer to $L_i = |Y_i|$ as the reliability of the i-th bit. Sample rank ordered reliability values $\{L_i : i \in \{1, \ldots, n\}\}$ are plotted in Fig. 3-1 for various SNRs. At lower SNR, the reliability curve is near linear with a zero intercept, while for high SNR the intercept is non-zero and there is notable curvature, particular for the least reliable

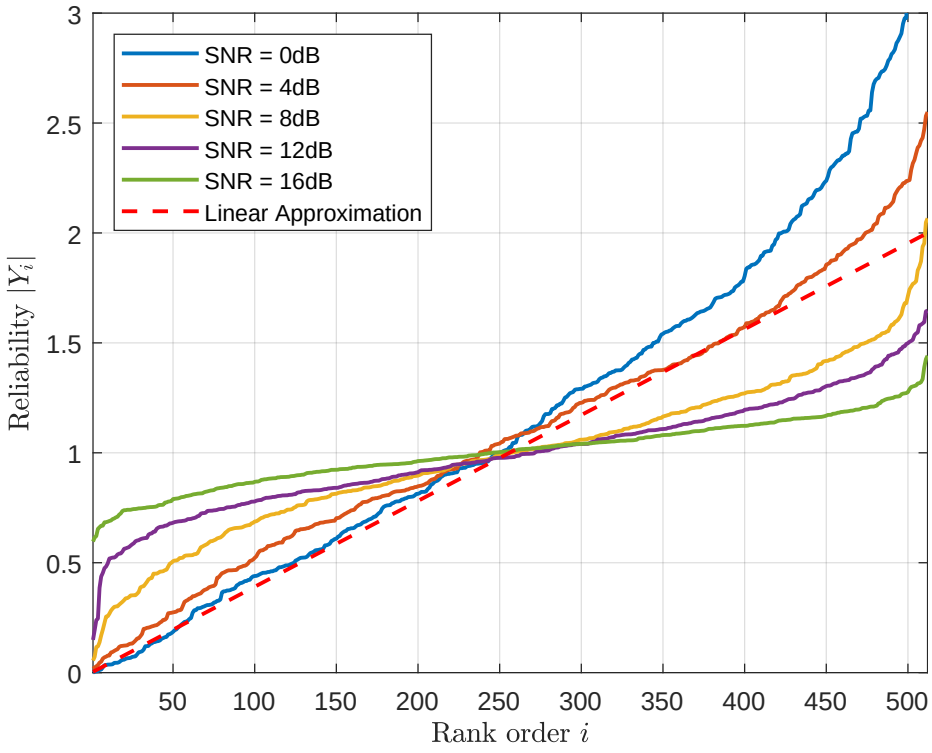


Figure 3-1: Samples of ordered reliability of 512-bit sequences in AWGN channel with given SNRs.

bits, which are most significant for generating an accurate query order. Different levels of approximation to the reliability curve lead to distinct decoding complexity and performance, as will be explored in the following sections.

3.2.2 Basic ORBGRAND - The Low SNR Model

The simplest statistical model, λ^n, for the reliability curve is a line through the origin with slope $\beta > 0$,

$$\lambda_i = \beta\, i, \text{ for } i = 1, 2, \ldots, n. \tag{3.3}$$

This model is illustrated by the dashed line in Fig. 3-1, where it can be seen to provide a good approximation at lower SNR. For the zero-intercept linear model,

$$\mathrm{Rel}(z^n) \approx \sum_{i:z_i=1} \lambda_i = \beta \sum_{i=1}^{n} i z_i = \beta w_{\mathrm{L}}(z^n), \tag{3.4}$$

where we define is the sum of the positions that are flipped in z^n

$$w_{\mathrm{L}}(z^n) = \sum_{i=1}^{n} i z_i, \tag{3.5}$$

to be the *Logistic Weight* of the binary sequence z^n. Thus, in this model the likelihoods of putative noise effect sequences are ordered in increasing logistic weight and hence the value of β need not be estimated.

As a result, for any β, the first putative error sequence always corresponds to no bits being flipped, which has $w_{\mathrm{L}} = 0$. The second query corresponds to z^n with the least reliable bit flipped, having $w_{\mathrm{L}} = 1$. The third corresponds to only the second least reliable bit of z^n flipped, which has $w_{\mathrm{L}} = 2$. The next query is either the noise-effect where only the third least reliable bit is flipped, or the one where the least reliable and second least reliable bits are both flipped, both having $w_{\mathrm{L}} = 3$, with the tie broken arbitrarily. The ordering proceeds in that fashion as illustrated in Fig. 3-2, which describes the noise-effect sequence generator in basic ORBGRAND [41]. Thus, for its operation, ORBGRAND based on this statistical model only requires the permutation recording the positions of the rank ordered reliabilities of the received bits, π^n, from which the algorithm proceeds deterministically.

Note that for a binary string of length n, the maximum logistic weight is achieved by the sequence of all 1s giving $w_{\mathrm{L}}(1, \ldots, 1) = n(n+1)/2$, and so what remains to do for basic ORBGRAND is to develop an efficient algorithm that sequentially generates putative noise sequences in terms of increasing logistic weight. To do so, we must be able to identify all allowable noise-effect sequences for each given logistic weight $W \in \{0, \ldots, n(n+1)/2\}$,

$$\mathcal{S}_W = \{z^n \in \{0,1\}^n : w_{\mathrm{L}}(z^n) = W\}. \tag{3.6}$$

That objective can be fractionated by conditioning on the Hamming weight, w, of

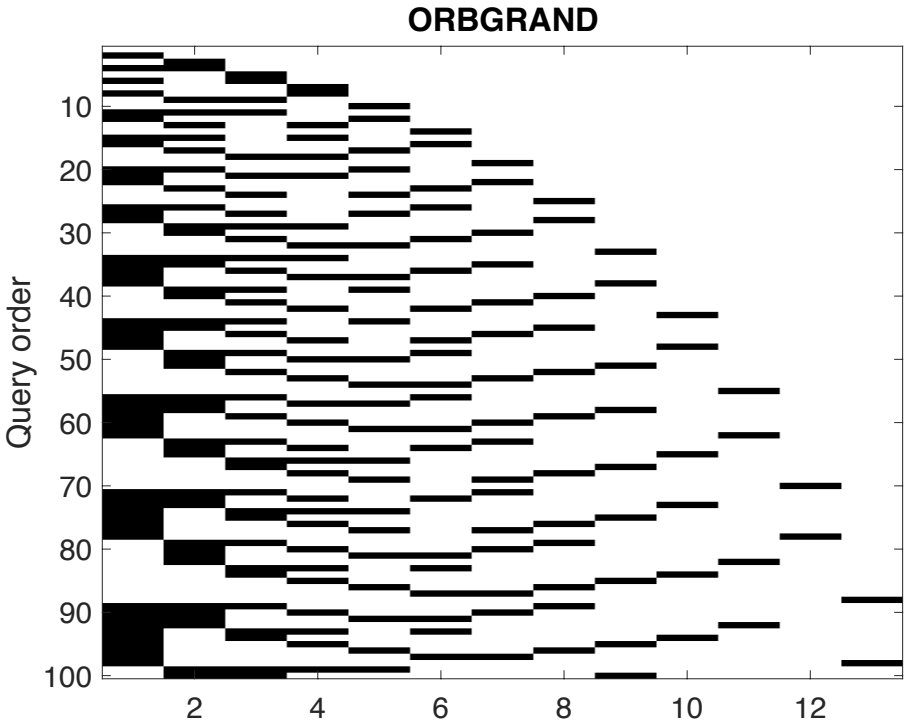


Figure 3-2: First 100 ORBGRAND noise effect queries where bit positions are in increasing order of hard-detection reliability. Each row is a noise sequence with white being no bit flip and black corresponding to a bit flip.

the sequences, giving

$$\mathcal{S}_W = \bigcup_{w=1}^{\lfloor(\sqrt{1+8W}-1)/2\rfloor} \{z^n \in \{0,1\}^n : w_{\mathrm{H}}(z^n) = w, w_{\mathrm{L}}(z^n) = W\}\,, \tag{3.7}$$

where the upper-bound on the union stems from the fact that if the Hamming weight of z^n is w, the smallest logistic weight that z^n can have is from the sequence with flipped bits in the first w positions of z^n, giving a logistic weight of $w_{\mathrm{L}}(z^n) = w(w+1)/2 \leq W$.

Consider a single set in the union in Eq. (3.7) for Hamming weight w. Determining

$$\{z^n \in \{0,1\}^n : w_{\mathrm{H}}(z^n) = w, w_{\mathrm{L}}(z^n) = W\}$$

is equivalent to finding all integer-valued vectors of length w satisfying

$$\left\{ v^w \in \mathbb{N}^w : 1 \leq v_1 < \ldots < v_w \leq n, \sum_{i=1}^{w} v_i = W \right\}, \tag{3.8}$$

where v^w contains the indices of the flipped bits in z^n, which amounts to finding all integer partitions of W of size w with non-repeating positive parts subject to a maximum value of n. By setting

$$v_i = i + u_i, \text{ for } i = 1, 2, \ldots, w, \tag{3.9}$$

it is possible to reformulate the set in Eq. (3.8) in one final way in terms of the u_i, as the integer partitions of $W' = W - w(w+1)/2$ into w not-necessarily distinct, non-negative parts no larger than $n' = n - w$ [110]. That is, determining all the elements in the set Eq. (3.8), is equivalent to finding all integer vectors u^w such that

$$\left\{ u^w \in \mathbb{Z}_+^w : 0 \leq u_1 \leq u_2 \leq \ldots \leq u_w \leq n', \sum_{i=1}^{w} u_i = W' \right\}. \tag{3.10}$$

Here we introduce an efficient algorithm for determining all sequences that are in the partition, which is suitable for implementation in hardware. It will form an essential component of the full ORBGRAND, which uses a more sophisticated model than described in Eq. (3.3).

3.2.3 Integer Partition Pattern Generator

Integer partitions can be represented by diagrams [26] as illustrated in Fig. 3-3, where each column represents an integer part with its value, u_i, equaling to the number of cells in the column and the total number of cells in the diagram equaling the integer to be partitioned $\sum_i u_i$. Here we use a mirror image of a Ferrers Diagram, where the parts are listed in the increasing order to assist in the description of the algorithm.

A function $d : \{1, \ldots, w\} \mapsto \{0, \ldots, W'\}$ records the "drop" between adjacent i,

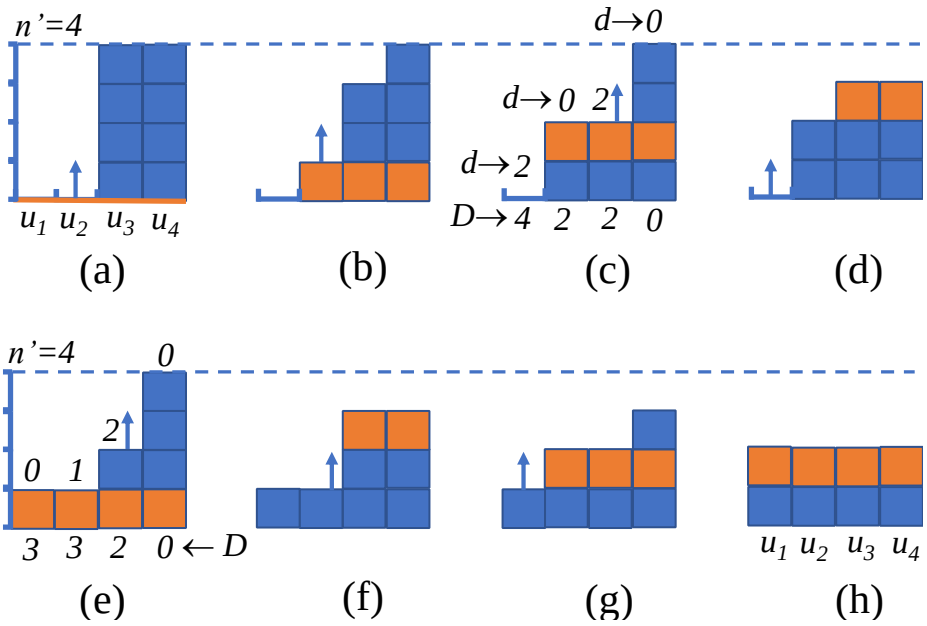


Figure 3-3: Procedure for partitioning $W' = 8$ into $w = 4$ non-negative, non-decreasing parts, each no larger than $n' = 4$. The upward arrow indicates the corresponding part is to be increased by 1 in the next step. (c) and (e) mark the values of $d(i)$ and $D(i)$ for $1 \leq i \leq 4$.

$i + 1$ parts

$$d(i) = \begin{cases} 0 & \text{if } i = w \\ u_{i+1} - u_i & \text{if } i \in \{1, \ldots, w-1\} \end{cases}$$

from which the accumulated drop function is defined by $D(i) = \sum_{j=i}^{w} d(j)$, where $D(1)$ records the total drop in the integer partition u^w. Examples of $d(i)$ and $D(i)$ are shown in Fig. 3-3 (c) and (e),

Fig. 3-3 (a) represents an extreme case in which the minimum number of non-zero integer parts is achieved by pushing cells to the right with part values maximized. Another extreme case is that cells are spread to maximum number of parts achieving the minimum number of rows, or equivalently, satisfying $D(1) \leq 1$, as illustrated in Fig. 3-3 (h). All partitions for the setting of $W' = 8$, $w = 4$ and $n' = 4$ are obtained in the migration procedure from (a) to (h), which can be accomplished with Algorithm 4, the Landslide algorithm.

The algorithm heavily relies on the Build-mountain routine, in which a partial partition is performed to push unallocated cells to the right-most parts, akin to building the steepest, highest mountain possible on the right side of the diagram. For example, in Fig. 3-3 (e), $u_1 = 1$ is determined from step (d), the remaining 7 cells are to be assigned to u_2, u_3 and u_4. The assignment can be accomplished by first making u_2, u_3 and u_4 identical to $u_1 = 1$, and then assigning the remaining 4 cells to the right-most parts, with u_4 maximized and u_3 increased by 1.

In general, when the values of $\{u_1, u_2, \ldots, u_k\}$ have been specified, the allocation of the remaining cells to $\{u_{k+1}, u_{k+2}, \ldots, u_w\}$, or the Build-mountain routine, is carried out as follows:

1. $u_i \leftarrow u_k$, for $k + 1 \leq i \leq w$
2. $W'' \leftarrow W' - \sum_{i=1}^{w} u_i$
3. Obtain q and r such that $W'' = q(n' - u_k) + r$
4. if $q \neq 0$, $u_i \leftarrow n'$, for $w - q + 1 \leq i \leq w$
5. $u_{w-q} \leftarrow u_{w-q} + r$

The initial partition in Fig. 3-3 (a) is obtained with the same method by simply assuming a dummy part $u_0 = 0$. With the Build-mountain routine explained, the Landslide algorithm is described as in Algorithm 4.

Algorithm 4 The Landslide Algorithm

Input: W', w, n'
Output: $\{u^{w,j}, j = 1, 2, \ldots\}$

1: Build-mountain for initial partition
2: $j \leftarrow 1$
3: $u^{w,j} \leftarrow u^w$
4: Update $D(i)$ for $1 \leq i \leq w$
5: **while** $D(1) \geq 2$ **do**

Locate the largest k such that $D(k) \geq 2$

$u_k \leftarrow u_k + 1$

Build-mountain from u_k

Update $D(i)$ for $1 \leq i \leq w$

$j \leftarrow j + 1$

$u^{w,j} \leftarrow u^w$

end while

Return $\{u^{w,1}, u^{w,2}, u^{w,3}, \ldots\}$

Using the same example from Fig. 3-3, the procedure of Algorithm 1 is illustrated in Fig. 4, along with the mapping from partition $u^{w,i}$ to $v^{w,i}$ according to Eq. (3.9). The diagram indicates the potential for efficient implementation of the Landslide algorithm. While one routine generates partitions for one Hamming weight w at a time, multiple parallel routines can generate partitions for different Hamming weights, providing sufficient noise-effect sequences for highly-parallelized code-book checking.

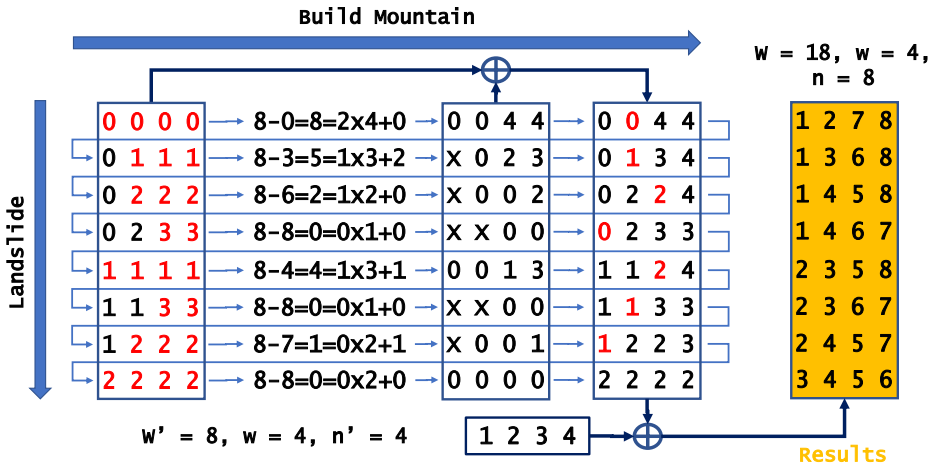


Figure 3-4: The Landslide algorithm is applied to achieve partitioning $W = 18$ into $w = 4$ distinguished parts with maximum value of $n = 8$; The partition problem is first converted to partitioning $W' = 8$ into $w = 4$ repeatable parts with maximum value of $n' = 4$; Mapping from the latter partition to the former one is simply achieved by adding 1, 2, 3, 4 individually

3.2.4 The Full ORBGRAND Algorithm

The zero-intercept, linear statistical model for rank-ordered bit reliabilities that underpins basic ORBGRAND in Eq. (3.3) requires no input beyond a rank ordering of received hard-detection bits by increasing reliability and provides a good approximation to the reliability curve in low SNR conditions. It is, however, evidently a poor description at higher SNR in Fig. 3-1. That mismatch results in basic ORBGRAND's query order diverging from true likelihood order at higher SNR, with corresponding performance loss. By expanding the statistical model used to describe the reliability data to a piece-wise linear one for full ORBGRAND, we retain the algorithmic efficiencies of generating integer partition sequences while improving block error rate performance at higher SNR.

As illustrated in Fig. 3-5, with $I_0 = 0$ and $I_m = n$, the m-segment statistical model curve is represented as

$$\lambda_j = J_{i-1} + \beta_i(j - I_{i-1}), \text{ for } I_{i-1} < j \leq I_i, \tag{3.11}$$

where $1 \leq i \leq m$ is the segment index. The anchor indices $\{I_i : i \in \{0, 1, \ldots, m\}\}$ define the domain of each segment, while $J_{i-1} \in \mathbb{Z}$ and $\beta_i \in \mathbb{N}$, respectively, determine the initial value and the slope of the i-th segment. That J_{i-1} and β_i are restricted to being integers is crucial to enabling efficient algorithmic implementation producing rank ordered putative noise sequences, and results will demonstrate that no loss in performance results from this constraint. Note that the model used for basic ORBGRAND, Eq. (3.3), is a special case of Eq. (3.11) with $m = 1$, $I_1 = n$ and $J_0 = 0$.

The approximate reliability sum of z^n, namely the reliability weight, based on the

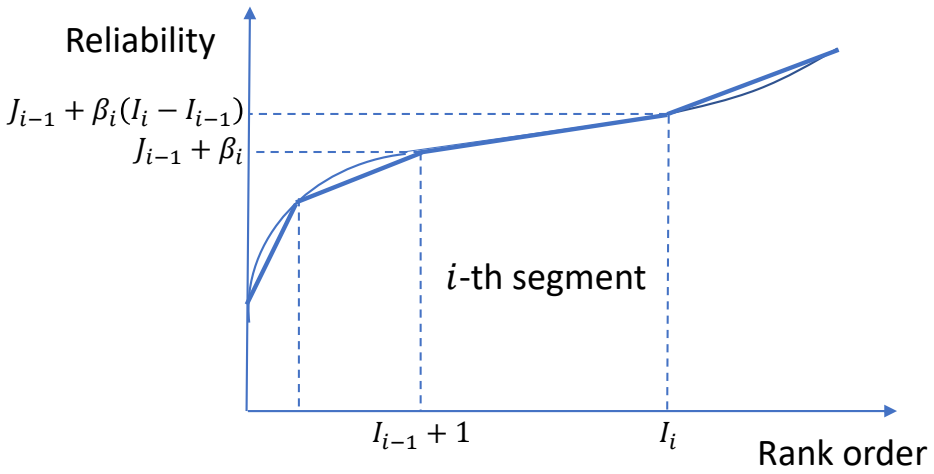


Figure 3-5: Full piece-wise linear statistical model to the ordered reliability curve used in ORBGRAND, with the start and end indices indicated for the i-th segment.

full model is then

$$\begin{aligned}\mathrm{Rel}(z^n) &\approx \sum_{i:z_i=1} \lambda_i = \sum_{i=1}^{n} \lambda_i z_i = \sum_{i=1}^{m} \sum_{j=I_{i-1}+1}^{I_i} \lambda_j z_j \\ &= \sum_{i=1}^{m} J_{i-1} w_{\mathrm{H}}(z_{I_{i-1}+1}, \ldots z_{I_i}) + \sum_{i=1}^{m} \beta_i w_{\mathrm{L}}(z_{I_{i-1}+1}, \ldots z_{I_i}) \in \mathbb{Z}_+,\end{aligned}$$

and the likelihood of noise effect sequences decreases with increasing reliability weight. With this new approximation, the set of noise-effect sequences for a weight of W becomes

$$\begin{aligned}\mathcal{S}_W &= \left\{ z^n \in \{0,1\}^n : \sum_{i=1}^{m} \sum_{j=I_{i-1}+1}^{I_i} \lambda_j z_j = W \right\} \\ &= \bigcup_{W^m : \sum_{i=1}^{m} W_i = W} \left(\Psi^1_{W_1} \times \Psi^2_{W_2} \times \cdots \Psi^m_{W_m} \right), \end{aligned} \tag{3.12}$$

where

$$\Psi^i_{W_i} = \left\{ (z_{I_{i-1}+1}, \ldots, z_{I_i}) : \sum_{j=I_{i-1}+1}^{I_i} \lambda_j z_j = W_i \right\}$$

for $i \in \{1, \ldots, m\}$ and $\times$ represents Cartesian product. Thus, to generate all elements

of $\mathcal{S}_W$ in Eq. (3.12), we identify the set of all possible splitting patterns of W, denoted by

$$\Xi_W = \left\{ W^m \in \mathbb{Z}_+^m : \sum_{i=1}^{m} W_i = W \right\}, \tag{3.13}$$

using Algorithm 5, explained later.

For a given $W^m = (W_1, \ldots, W_m) \in \Xi_W$, consider the generation of the partial sequence set $\Psi_{W_i}^i$ defined in Eq. (3.12). Recalling Eq. (3.11), each partial sequence must satisfy

$$W_i = \sum_{j=I_{i-1}+1}^{I_i} (J_{i-1} + (j - I_{i-1})\beta_i) z_j = J_{i-1} w_{\mathrm{H}}(z_{I_{i-1}+1}, \ldots z_{I_i}) + \beta_i w_{\mathrm{L}}(z_{I_{i-1}+1}, \ldots z_{I_i}), \tag{3.14}$$

which, defining $w_i = w_{\mathrm{H}}(z_{I_{i-1}+1}, \ldots z_{I_i})$ and with $v_k = j_k - I_{i-1}$ being the relative indices of the flipped bits, is equivalent to

$$w_{\mathrm{L}}(z_{I_{i-1}+1}, \ldots z_{I_i}) = \sum_{k=1}^{w_i} v_k = \frac{W_i - w_i J_{i-1}}{\beta_i}. \tag{3.15}$$

Eq. (3.15) indicates that, with the partial reliability weight W_i and Hamming weight w_i specified for the i-th segment, the partial noise-effect sequence generation reduces to the integer partition problem that is efficiently solved by the Landslide algorithm in section 3.2.3.

Splitting a reliability weight value of W into m parts, as defined in Eq. (3.13), is essentially another integer partition problem, which we call the integer splitting problem for differentiation. The difference here lies in that the same group of parts with different orders are distinct splitting patterns. A common approach to finding all splitting patterns in Ξ_W is given in Algorithm 5, which starts with sweeping W_1 from 0 to W. For a given value of W_1, W_2 is swept from 0 to $W - W_1$. For each fixed W_1 and W_2, W_3 is swept and the nested loop reaches W_{m-1}. Then W_m is computed as $W - \sum_{j=1}^{m-1} W_j$, ensuring the sum of all parts is W. The size of the set Ξ_W obtained

from the algorithm is $\xi_W = \binom{W+m}{m-1}$ (See Appendix A for proof).

Algorithm 5 The Integer Splitting Algorithm

Input: W, m

Output: $\{W^{m,k} : k = 1, 2, \ldots, \xi_W\}$

```
k ← 0
for W_1 = 0 To W do
    for W_2 = 0 To W − W_1 do
        ..... (nested loops over W_i, i = 3, 4, ..., m − 2)
        for W_{m−1} = 0 To W − Σ_{l=1}^{m−2} W_l do
            W_m ← W − Σ_{l=1}^{m−1} W_l
            k ← k + 1
            W^{m,k} ← {W_1, W_2, ..., W_m}
        end for
        ..... (nested loops over W_i, i = 3, 4, ..., m − 2)
    end for
end for
return {W^{m,k} : k = 1, 2, ..., ξ_W}
```

Eq. (3.15) indicates that the actual number of valid splitting patterns is, however, much smaller than ξ_W, owning to the requirement that each element W_i of a valid W^m must satisfy all of:

$$\begin{cases} W_i = 0 \text{ or } W_i - w_i J_{i-1} \geq \frac{(1+w_i)w_i}{2} \\ W_i - w_i J_{i-1} \leq (I_i - I_{i-1} + 1)w_i - \frac{(1+w_i)w_i}{2} \\ W_i - w_i J_{i-1} \text{ is divisible by } \beta_i. \end{cases} \tag{3.16}$$

Therefore, any non-zero element W_i in W^m must be associated with a non-empty set of partial Hamming weights $\{w_i\}$, such that Eq. (3.16) is satisfied. Otherwise W^m is invalid and should be discarded. The associated set for W_i can be obtained with Algorithm 6.

Algorithm 6 The collection algorithm for valid partial Hamming weights

Input: W_i, J_{i-1}

Output: $\{w_{i,k} : k = 1, 2, \ldots\}$ or FAIL

1: $k \leftarrow 0$
2: **for** $w = 1$ To $\lfloor \frac{\sqrt{1+8W_i}-1)}{2} \rfloor$ **do**
3: **if** W_i, w and J_{i-1} satisfy Eq. (3.16) **then**
4: $k \leftarrow k + 1$
5: $w_{i,k} \leftarrow w$
6: **end if**
7: **end for**
8: **if** k is 0 **then**
9: **return** FAIL
10: **else**
11: **return** $\{w_{i,k} : k = 1, 2, \ldots\}$
12: **end if**

The FAIL return from Algorithm 3 invalidates W_i as well as the whole split pattern W^m. In Algorithm 2, each new value of W_i is checked against Algorithm 3. A return of FAIL discard the current value of W_i and force the loop to jump to the next iteration with a new value of W_i. Only when W_i is validated, can the follow-up nested loop over W_{i+1} continue. Each returned partial Hamming weights set $\{w_{i,k}, k = 1, 2, \ldots\}$ should also be saved for the later generation of partial noise-effect sequences.

In addition to the validation from Algorithm 6, more measures are available for further reduction of the set size of Ξ_W. For example, after the initial value of 0, W_i can jump to $J_{i-1} + \beta_i$ omitting all values in between. Generally, due to the small segment number m in practice, the generation of splitting patterns W^m has limited impact on the overall efficiency of the ORBGRAND algorithm, which is instead dominated by the efficient Landslide algorithm.

A significant complexity reduction is, however, available if J_{i-1} is divisible by β_i. In this case, W_i must also be divisible by β_i in order to have Eq. (3.16) satisfied. This

can be achieved by sweeping W_i in steps of size β_i. Then the validation of a partial Hamming weight w_i is straightforward, forsaking the need of Algorithm 6. The extra restriction on J_{i-1} logically leads to a potential performance loss. As demonstrated by later simulations, the trivial performance change not only justifies the complexity reduction measure but also demonstrates the robustness of ORBGRAND algorithm, allowing more potential complexity control techniques.

Given parameters of the statistical model in Eq. (3.11), all the components necessary to create the full ORBGRAND algorithm have been described. The likelihood order of generated noise-effect sequences is governed by the increasing value of reliability weight. For each specified weight value W, Algorithm 2 (or its optimized version) is used to generate Ξ_W, the set of valid splitting patterns. Each splitting pattern $W^m \in \Xi_W$ has its element (or partial reliability weight) W_i assigned to the i-th segments. In each segment, Eq. (3.15) indicates that the Landslide algorithm can efficiently generate Ψ_{W_i}, the set of all possible partial noise-effect patterns, as defined in Eq. (3.12). Cartesian product over partial sequence sets, as shown in Eq. (3.12), is performed to create the set of noise-effect sequences for the current splitting pattern W^m. Finally, the union operation in Eq. (3.12) forms $\mathcal{S}_W$, the full set of noise-effect sequences for the reliability weight of W. Parallel implementation can be achieved at several levels, such as jointly generating partial sequences for multiple segments, or concurrently generating noise-effect sequences for multiple splitting patterns. What remains now is determining the parameters in the piece-wise linear model.

3.2.5 Piece-wise Linear Fitting and Quantization

Key to ORBGRAND's practical complexity is that it operates on $\lambda^n = (\lambda_1, \ldots, \lambda_n)$, an approximation to the original rank ordered reliability curve for a given received code block $(L_1, \ldots, L_n)$. The approximation level determines the trade-off between algorithmic complexity and the decoding performance. The simplest statistical model is a line through the origin, which only requires rank ordering of the received bits by their reliability, but results in degraded performance at higher SNR scenarios. The model underlying the full ORBGRAND necessitates two stages: piece-wise lin-

ear fitting and quantization. While there are numerous approaches for either higher accuracy or lower complexity, here we introduce a method with moderate algorithmic complexity that serves as a reference design and demonstrates the robustness of ORBGRAND.

Given an independent and identically distributed set of random variables, $\{A_i : i \in \{1, \ldots, n\}\}$, drawn from a cumulative distribution F_A, results from the theory of Order Statistics [32] tell us that rank ordering from least to greatest, so that $A_{(i)}$ is the i-th smallest value, leads to

$$A_{(i)} \approx F_A^{-1}\left(i/n\right) \tag{3.17}$$

for $1 \leq i \leq n$ and large n. $F_A^{-1}(\cdot)$ is a monotonically increasing function and serves as the functional mean of rank ordered ensembles of observations $\{A_i : i \in \{1, \ldots, n\}\}$. For rank ordered reliabilities of blocks of bits received from the channel, $(L_1, \ldots, L_n)$, this serves as guidance for a fitting procedure for the statistical model.

We can, therefore, use the edge point at index $I_{a,0} = 1$ and the center point at index $I_{a,1} = n/2$ on L^n as the initial set of anchor points from which other anchor points for segmentation can be found, as illustrated in Fig. 3-6. A straight line is drawn linking the anchor points at $I_{a,0}$ and $I_{a,1}$. The maximum vertical gap between the straight line and the reliability curve determines the location of the new anchor point with its index marked as $I_{a,2}$. New anchor points can be found between adjacent anchor points in the same way. A rule of thumb is that more points should be located in the high-curvature area near the edge. The indices of anchor points define the segmentation of the reliability curve, and the linear lines linking adjacent anchor points form a piece-wise linear fitting to the reliability curve.

If the curve L^n is close to a straight line between two anchor points an additional segment is unnecessary, or, a casually added segment has little impact to the performance except for some overhead in the splitting of logistic weight. The same fitting technique can be applied to the high reliability area near the right edge, however, as shown in Fig. 3-6, we choose to extend the central line to cover the area. In low SNR

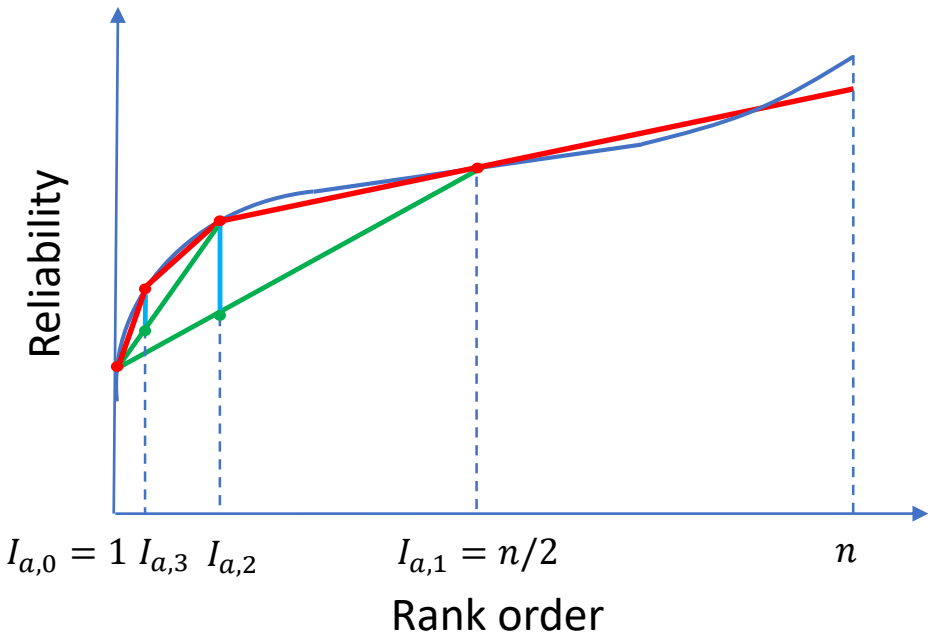


Figure 3-6: Identification of anchor points for segmentation of the reliability curve.

cases, the extended straight line by itself is a good approximation, and in high SNR cases, high reliability bits have little influence on the generation order of noise-effect sequences. The assertion has been verified with simulations.

From Eq. (3.11), the piece-wise linear approximating curve is defined with three sets of non-negative integer parameters: $I_i \in \mathbb{Z}_+, 0 \leq i \leq m$, the indices for segmentation; $J_i \in \mathbb{Z}, 0 \leq i \leq m-1$, the offset of each linear segment; and $\beta_i \in \mathbb{N}, 1 \leq i \leq m$, the slope of each segment. When $m+1$ anchor points on L^n have been obtained, their indices are used as the segmentation indices and is denoted as $I_i, i = 0, 1, 2, \ldots, m$, where $I_0 = 0$ and $I_m = n$. We further use the smallest slope of fitted linear lines to quantize parameters, which is computed as

$$Q = \min\left\{\frac{L_{I_1} - L_1}{I_1 - 1}, \min_{i\in\{2,\ldots,m\}}\left\{\frac{L_{I_i} - L_{I_{i-1}}}{I_i - I_{i-1}}\right\}\right\} \tag{3.18}$$

where the slope of the first segment is specially treated due to the lack of L_0. The

quantized parameters of linear lines are then computed as,

$$\begin{cases} \beta_1 = \left[\frac{L_{I_1} - L_1}{(I_1 - 1)Q}\right], \text{ for } i = 1; \beta_i = \left[\frac{L_{I_i} - L_{I_{i-1}}}{(I_i - I_{i-1})Q}\right], \text{ for } 2 \leq i \leq m \\ J_0 = \left[\frac{L_1}{Q}\right] - \beta_1, \text{ for } i = 0; J_i = \left[\frac{L_{I_i}}{Q}\right], \text{ for } 1 \leq i \leq m-1, \end{cases} \tag{3.19}$$

where [ ] is the rounding operation. Again, β_1 and J_0 are specially treated for the first segment. A complexity reduction technique in section 3.2.4 requires J_{i-1} to be integer multiples of β_i, which can be easily achieved with operation $[J_{i-1}/\beta_i]\,\beta_i$. The segmentation method in Fig. 3-6 and line parameters obtained from Eq. (3.19) complete the piece-wise linear fitting and quantization.

3.3 Performance Evaluation

As stated in our motivation, ORBGRAND is particularly well suited to low to moderate redundancy. Such redundancy regimes can be achieved by having short length codes, or longer codes with sufficiently high rate to make the number of redundancy bits low to moderate. Commonly used codes have structures that limits their operating range, but Random Linear Codes (RLCs) have no such limitation and can be constructed for any length and rate. As an illustration, Fig. 3-7 shows as heat map of block error rates (BLERs) for different code lengths and rates for RLCs decoded using basic ORBGRAND.

With the range of suitable rates and lengths for ORBGRAND in mind, we can now explore the performance. A key feature of all GRAND algorithms is that they provide excellent decoding performance for all moderate redundancy codes, regardless of length or structure, and so can be used to identify the best code structures. Our first comparison is naturally CA-Polar codes, which are the state of the art codes which are both moderate length and high rate and which are designed for decoding with soft information, with dedicated CA-SCL decoders. We consider CA-Polar[256, 234], which has 22 parity bits and uses the 11-bit CRC specified for 5G NR up-link control channels. We set at the CA-SCL decoder the list size set to 16, and apply the

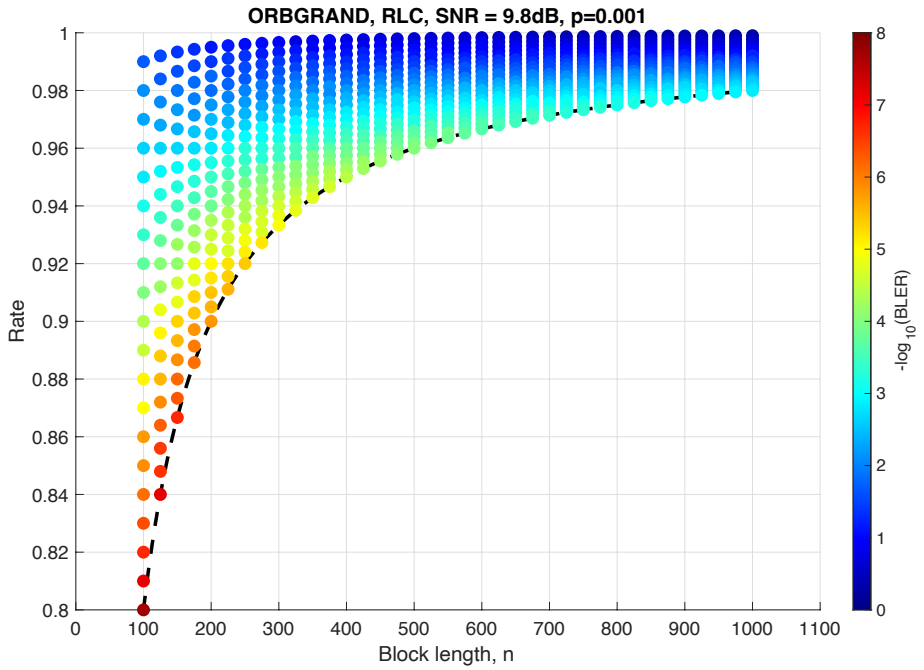


Figure 3-7: Decoding performance of ORBGRAND applied to RLCs of different length n with up to $n - k = 20$ redundant bits over AWGN channels using BPSK at an SNR of 9.8 leading to a hard detection bit flip probability of $p = 10^{-3}$.

CA-SCL decoder from the AFF3CT toolbox [27] as our performance reference. Non-standard codes include BCH codes, which can be well designed for low to moderate redundancy but are not designed for decoding with soft information. CRCs, which are designed for error detection rather than correction but that are being considered for error correction using GRAND [10, 62]. CRCs present desirable low complexity in encoding and code-book checking. Finally, we also consider RLCs, whose use with GRAND is also being explored [9, 10, 41, 79, 91].

Our comparison is in Fig. 3-8, where the 3-line version of ORBGRAND is used to decode. All ORBGRAND algorithms in our figures below have been set to abandon searching and record a block error if no code-book element is identified within 5×10^6 code-book queries.

One of the main innovations of this paper over the original conference paper that introduced ORBGRAND [41] is the multi-line approach. We next explore the effect of the number of lines, and of the selection of the intervals for those lines. To do so, we envisage three CA-Polar code configurations, CA-Polar[256, 234], CA-Polar[512,

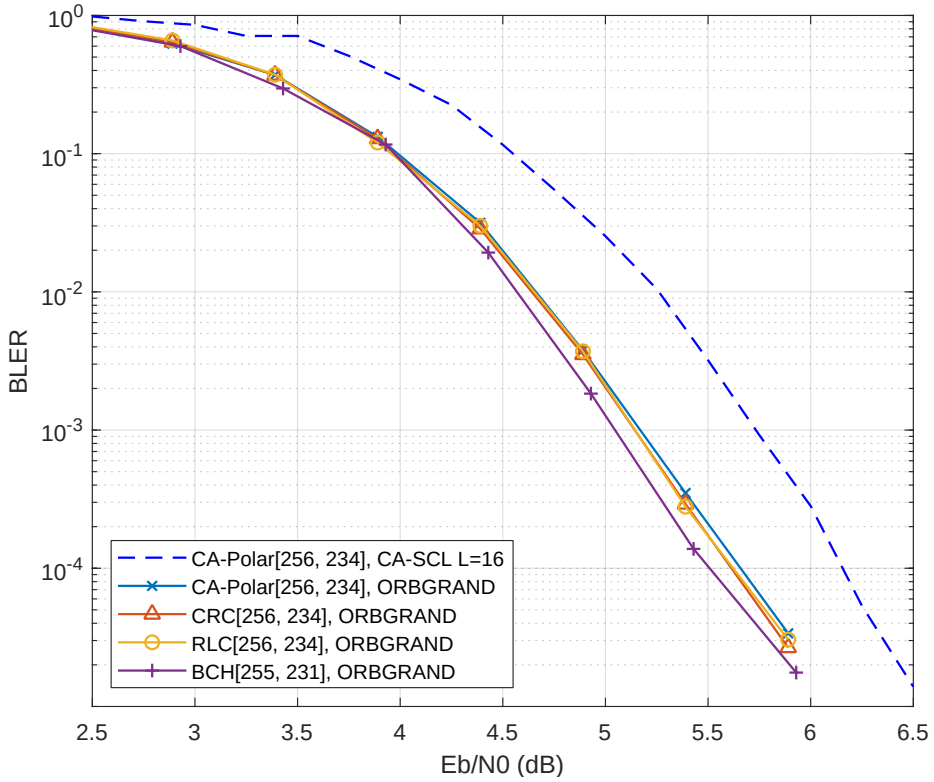
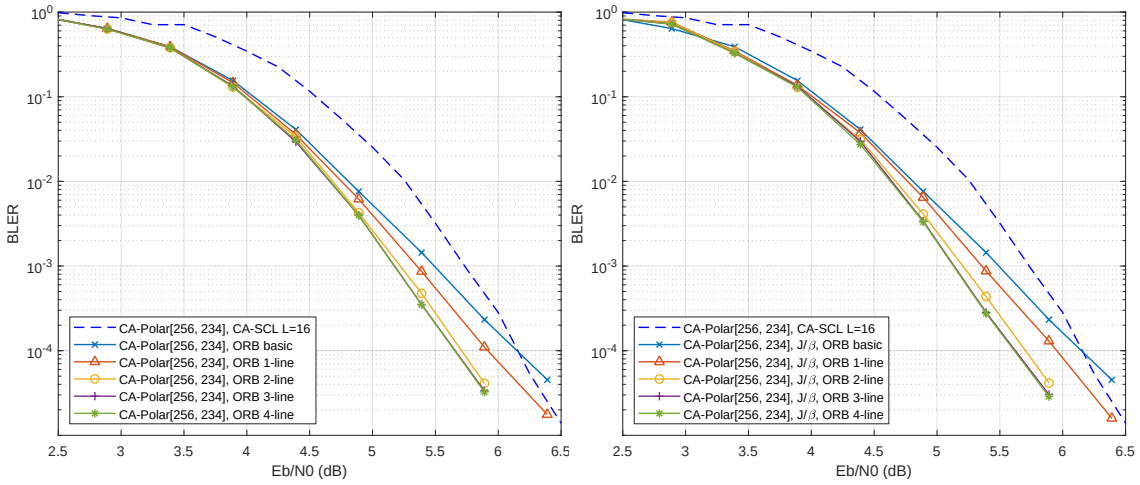


Figure 3-8: Decoding performance of ORBGRAND applied to CA-Polar[256, 234], CRC[256, 234], RLC[256, 234] and BCH[255,231] codes.



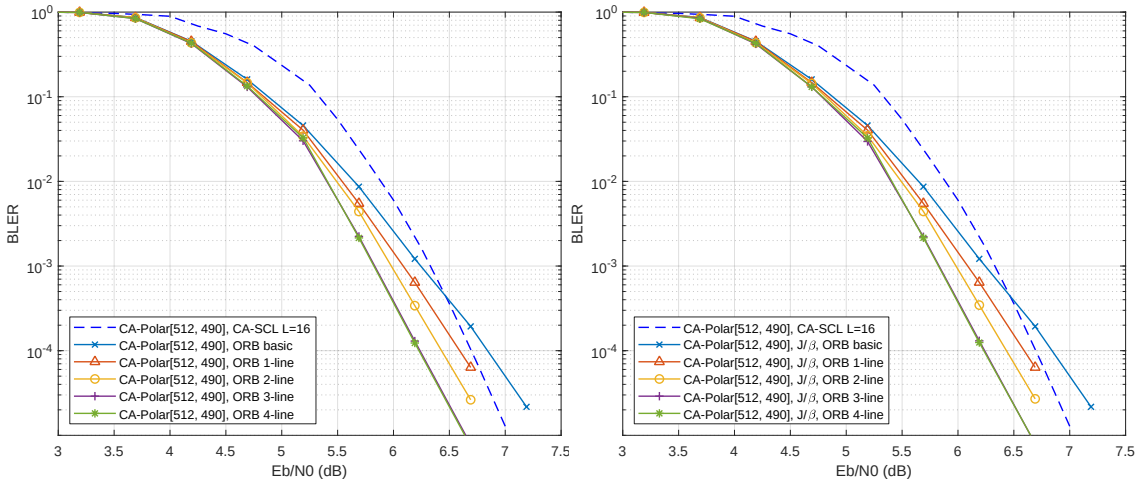
(a) (b)

Figure 3-9: Performance evaluation of CA-Polar[256, 234] as decoded with CA-SCL, with a list size of 16, and ORBGRAND variants. (a) Normal ORBGRAND quantization. (b) J_{i-1} divisible by β_i.

490] and CA-Polar[1024, 1002], which all have 22 parity bits and the 11-bit CRC specified for 5G NR up-link control channels.

Fig. 3-9a presents the simulation results of CA-Polar[256, 234]. For lower SNR, all ORBGRAND variants exhibit substantially better performance than CA-SCL because of the incomplete utilization of CRC bits for error correction in the CA-SCL algorithm [10]. For a BLER of 10^{-4} or below, CA-SCL outperforms the basic variant of ORBGRAND as its model does not produce putative noise sequences in near-ML order at higher SNR. ORBGRAND with 1-line fitting provides an observable, but limited, improvement over the basic version because their only difference is that the 1-line version starts from the quantized value of L_1 instead of the origin in the basic version. With the 2-line version, the curvature in the low reliability region is captured by the two fitted lines, essentially resulting in the elimination of performance loss, and leaving only a small room of improvement for the 3-line version, which in turn overlaps with the 4-line version.

Similar observations can be seen in the simulation results for CA-Polar[512, 490] and CA-Polar[1024, 1002] codes in Fig. 3-10a, 3-10b and Fig. 3-11a, 3-11b respectively, except that with longer block lengths, the 3-line and 4-line versions have more



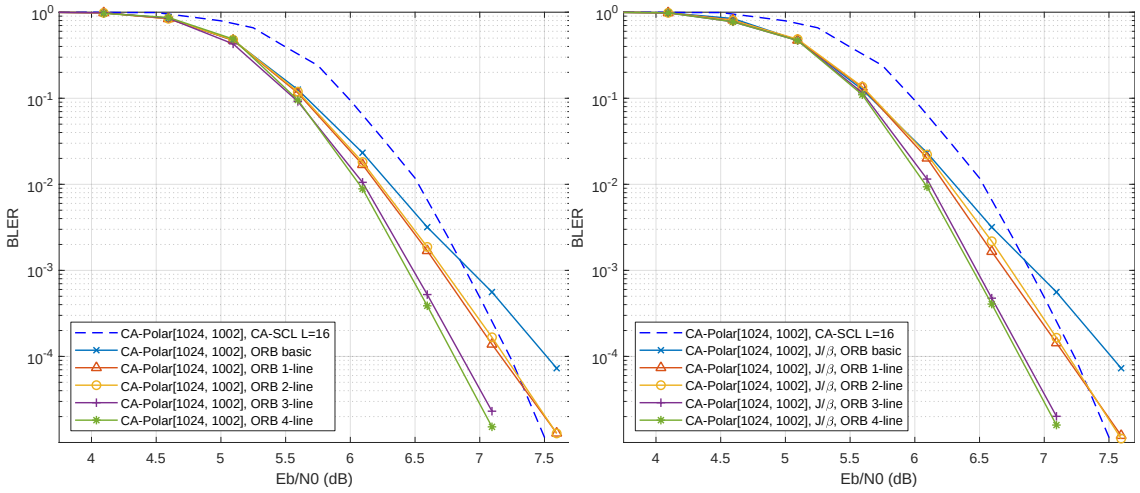
(a) (b)

Figure 3-10: Performance evaluation of CA-Polar[512, 490] decoded with CA-SCL, list size of 16, or ORBGRAND variants. (a) Normal ORBGRAND quantization. (b) J_{i-1} divisible by β_i.

significant decoding improvement. Also, with the same number of parity bits, the loss of performance of the basic version occurs at a higher BLER, as shown in Fig. 3-11a, where CA-SCL surpasses the basic version before BLER of 10^{-3}.

Up to now the simulations have demonstrated the effectiveness of the multi-line ORBGRAND in maintaining its performance advantage over the state-of-art CA-SCL decoders. In the next step we evaluate the influence of complexity control methods on the algorithm performance, which can bring significant advantages for ORBGRAND in practical implementations.

An example complexity control measure is to have J_{i-1} in Eq. (3.15) to be an integer multiple of β_i. As discussed in Section 3.2.4, the advantage is that Algorithm 3 in Fig. 6 is no longer needed, improving the efficiency of Algorithm 2 in Fig. 5. As shown in Fig. 3-9b, Fig. 3-10b and Fig. 3-11b, with the factor of J_{i-1}/β_i joined in, there is trivial change of performance between decoders with corresponding segmentation, demonstrating the robustness of ORBGRAND.



(a) (b)

Figure 3-11: Performance evaluation of CA-Polar[1024, 1002] decoded with CA-SCL, having a list size of 16, or ORBGRAND algorithms. (a) Normal quantization. (b) J_{i-1} divisible by β_i.

3.4 Computational Complexity Analysis of ORBGRAND

So far, evaluations of ORBGRAND performance all consider the abandonment condition $AB \geq 5 \times 10^6$ to ensure optimal decoding performance. In practice, it is undesirable to have a generous abandonment condition without deserved performance improvement. To illustrate, Fig. 3-12 presents the decoding performance and overall complexity of ORBGRAND performed on CA-Polar[256,234] under various abandonment conditions. The ORBGRAND configuration is identical to the 3-segment scenario in Fig. 3-9a. It is observed that the original abandonment condition is rather generous, considering that there is trivial performance loss when the abandonment condition is reduced down to $AB > 10^6$. Correspondingly, the overall complexity has significant reduction in low SNR regions and the reduction is observable even in the operating region at BLER of 10^{-3}. Therefore, in practice the abandonment condition is a necessary knob to achieve lower complexity averaged over all SNR regions. As BLER drops down to 10^{-4} with increased SNR, the complexity saving is no longer significant, suggesting that the abandonment condition has little effect in high SNR region. And the average test number has dropped down to around 300 checks per code-word, promising ultra-low power implementations of ORBGRAND.

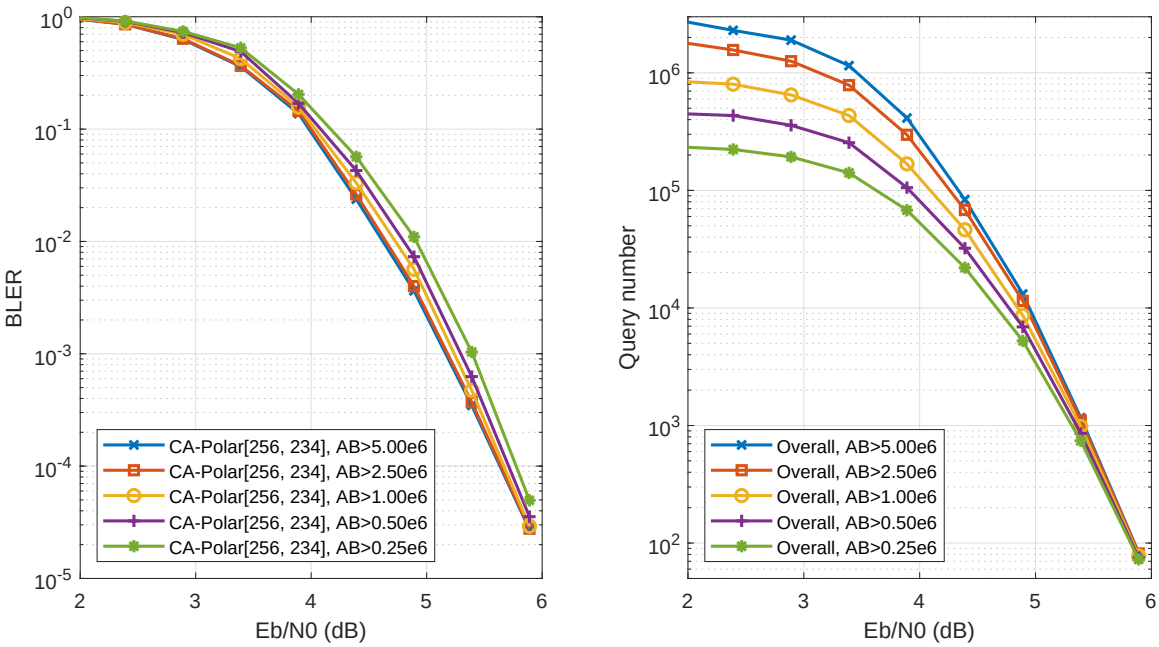


Figure 3-12: Computation complexity of ORBGRAND evaluated with CA-Polar[256,234] code in AWGN channels for various abandonment conditions; The decoding performance is presented for reference on the left; The decoding is performed with 3-segment full ORBGRAND.

As discovered in Section 2.6, GRAND-MO has an attractive property that both decoding performance and computational complexity are improved at a given SNR when the decoding algorithm is enhanced. The property is also observed in the design process of full ORBGRAND. Corresponding to the performance improvement in Fig. 3-9a, the computation complexity of each version of ORBGRAND is presented in Fig. 3-13a. While the basic ORBGRAND is proposed orienting the minimum algorithm complexity, in high SNR region, not only is its performance inferior to the full ORBGRAND, but also more computation is required at the same SNR. Even with the full ORBGRAND, as more segments are included with slight increase of the algorithm complexity, better performance and lower query number are achieved. Fig. 3-13b presents the relationship between the computational complexity and BLER, showing almost a linear relationship between their logarithmic representations. All versions of ORBGRANDs have their curves almost overlapped with each other, indicating that the same computation is required for a given BLER target in spite of the different decoding efficiency achieved by those versions. Therefore, like GRAND-MO, in

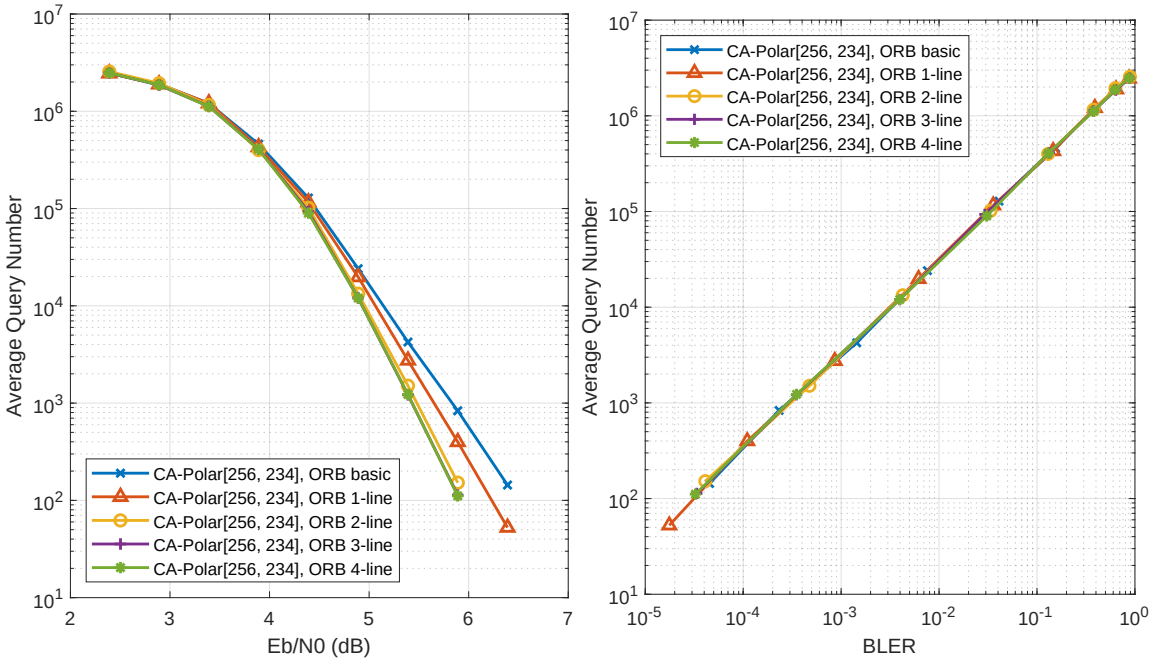


Figure 3-13: Average code-book query number for CA-Polar[256, 234]: (a) query number v.s. E_b/N_0; (b) query number v.s. BLER

the same noise condition an ORBGRAND version with better decoding performance requires less computation, leading to lower power consumption. In conclusion, the algorithm enhancement of GRAND variants improves both decoding performance and computational complexity, doubling the return of the design effort.

3.5 Algorithm Complexity Control Techniques

An essential algorithm complexity control method has been proposed in Section 3.2.4, in which the i-th segment has its parameter J_{i-1} divisible by β_i, eliminating the need of Algorithm 6, and significantly simplifying Algorithm 5, the integer splitting algorithm. Simulations in section 3.3 show that this restriction has trivial impact on the decoder performance, demonstrating the feasibility of the technique and the robustness of ORBGRAND. This successful example of algorithm complexity reduction, with a large amount of W_i values saved, suggests potentially more complexity control techniques at negligible cost of the decoding performance.

3.5.1 Static Segmentation

Compared to the dynamic segmentation method proposed in Section 3.2.5, a static segmentation method, in which the indices of anchor points are pre-determined and fixed during decoding process, is more welcome due to considerable complexity savings in many aspects, albeit the dynamic method, based on the instant information of the reliability curve, has performance advantage. If the decoding performance loss is tolerable in applications, the static segmentation is naturally the choice in practice.

We use AWGN channel to illustrate a reference method for static segmentation, which, due to the wide adoption of AWGN for channel modeling, has immediate practical significance. With Gaussian noise, the reliability value of the i-th rank ordered bit is $L_i = |Y_i|$, as illustrated in Fig. 3-1. A received signal $Y = \text{mod}(c) + N$ is a Gaussian distributed random variable with the BPSK modulated signal $\text{mod}(c)$ as its mean. In high SNR scenario, which is in our main interest, due to the small variance of Y, $|Y|$ is nearly a Gaussian distributed random variable with its mean at $|\text{mod}(c)| = 1$. Eq. (3.17) indicates that the reliability curve formed by $|Y_i|, 1 \leq i \leq n$ from rank ordered bits approaches to $F^{-1}_{|Y|}(i/n)$, where $F_{|Y|}$ can be viewed as Gaussian distributed cumulative distribution function (CDF) of the random variable $|Y|$ representing the reliability of a received bit. We use this property to determine the anchor points for static segmentation.

It is noted that the shape of the reliability curve is preserved after the transformation

$$V = (|Y| - 1)/\sigma. \tag{3.20}$$

where σ is the standard deviation of |Y|. Therefore, equivalent result can be obtained if the rank ordering and pattern generation are based on the transformed reliability V, whose distribution is close to a unit normal distribution with variance of 1, i.e.

$$f_V(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}, \tag{3.21}$$

Eq. (3.21) indicates that a segmentation based on V is SNR independent. The conclusion holds even in low SNR cases, where the distribution of $|Y|$ deviates from Gaussian distribution and approaches to a straight line, as illustrated by reliability curves for low SNRs in Fig. 3-1. Segments obtained from the unit normal distribution are helpful in fitting curvatures of the reliability curve, and have no negative impact when the real reliability curve is close to a straight line, except for some extra overhead in pattern generation. Therefore, it is expected that a segmentation obtained from Eq. (3.21) is applicable to any SNR scenarios in AWGN channel, that is, the resulted segmentation can be static.

From the PDF in Eq. (3.21), the CDF of V is obtained as,

$$r = F_V(s) = \int_{-\infty}^{s} f_V(t)dt = \int_{-\infty}^{s} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \tag{3.22}$$

Referring to Eq. (3.17), the relation $s = F_V^{-1}(r)$ suggests that the segmentation on the curve of s can be obtained by finding the r values of anchor points. The principle of segmentation is that more segments should be assigned to high curvature area, which can be located with the derivative of s, i.e.

$$\frac{ds}{dr} = \frac{1}{\frac{dF_V(s)}{ds}} = \frac{1}{f_V(s)} = \frac{1}{\tau} \tag{3.23}$$

where

$$\tau = f_V(s) \text{ or } s = f_V^{-1}(\tau), \text{ and } \tau \in \left(0, \frac{1}{\sqrt{2\pi}}\right]. \tag{3.24}$$

It is noted that small values of τ correspond to high curvature areas of s and can thus be used to determine anchor points. We propose to set

$$\tau_i = \frac{1}{b^{m-i}} \frac{1}{\sqrt{2\pi}}, \tag{3.25}$$

where $b > 1$ and $1 \leq i \leq m$. With a selected value of b, $\tau^m = \{\tau_1, \tau_2, \ldots, \tau_m\}$ can be obtained by Eq. (3.25), which lead to $s^m = \{s_1, s_2, \ldots, s_m\}$ by Eq.(3.24) and further

determines $r^m = \{r_1, r_2, \ldots, r_m\}$ by Eq. (3.22). Finally, indices of anchor points are obtained as,

$$I_i = nr_i, \text{ for } i = 1, 2, .., m. \tag{3.26}$$

With the inclusion of the fixed indices $I_0 = 0$ and the replacement of I_m with n, the segmentation with m segments is accomplished. The process of generating anchor points for $m = 3$ is illustrated in Fig. 3-14 with b set to 2. Here we are only interested

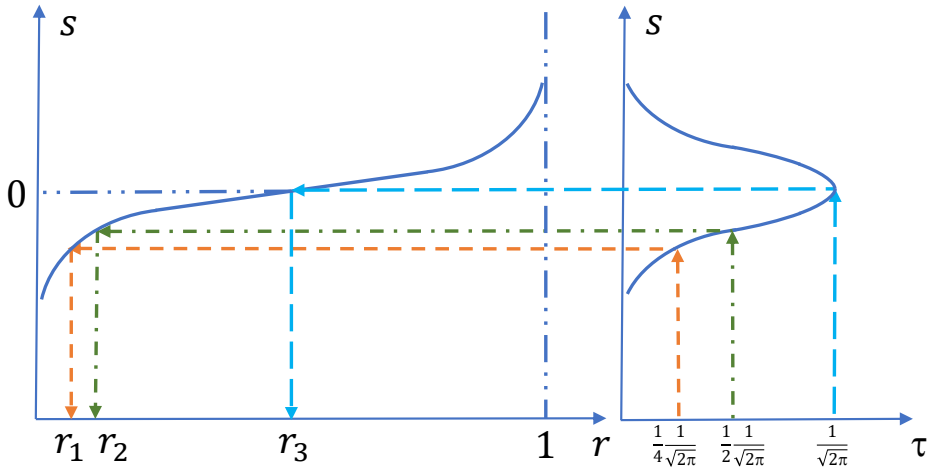


Figure 3-14: Example of static segmentation with $m = 3$ and $b = 2$; Arrows indicate the order of steps from values of τ to r

in assigning segments in low reliability region. As explained section 3.2.5, the high reliability region can be covered by the extension of the last segment, accomplished by replacing $I_m = nr_m$ with $I_m = n$. Fig. 3-14 also indicates that, with increased value of b, segments are more concentrated in the low reliability region where more segments are generally desired to fit the high curvature there. Therefore, while parameter m determines the segment number, the value of b can be used to adjust the segment density in high curvature region on low SNR side.

Fig. 3-15 presents the performance of static segmentation combined with the integer J/β condition. The performance is compared to the dynamic segmentation, both with the number of segmentation set to $m = 3$. With the SNR-independent static segmentation, its performance consistently matches the dynamic method in

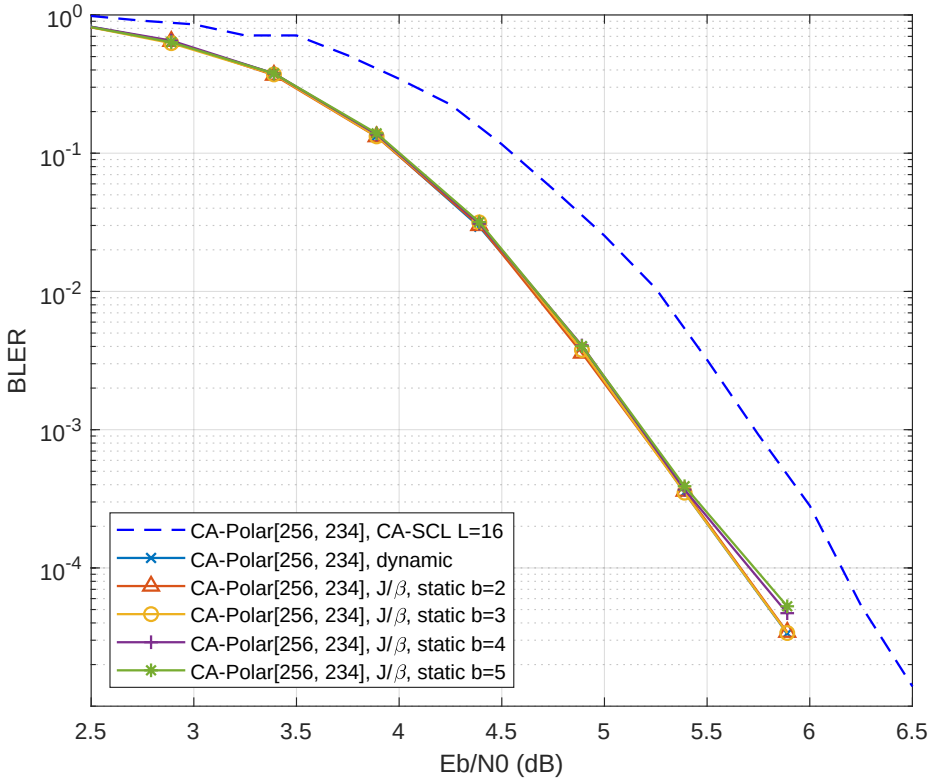
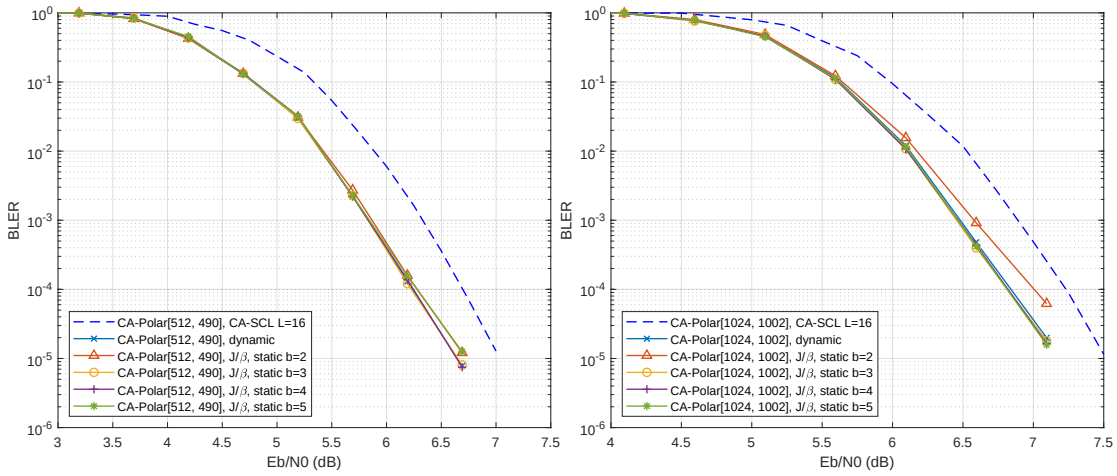


Figure 3-15: performance evaluation of static segmentation with $m = 3$ segments, and $b = 2, 3, 4, 5$, in addition to the integer J/β condition; Performed on CA-Polar[256,234] code with dynamic segmentation as reference



(a) (b)

Figure 3-16: performance evaluation of static segmentation with $m = 3$ segments, and $b = 2, 3, 4, 5$, in addition to J/β condition; Performed on CA-Polar[512,490] code in (a) and CA-Polar[1024,1022] code in (b); Dynamic segmentation is used as reference

most low SNR conditions in spite of the value of b. In high SNR region, $b = 4$ and $b = 5$ lead to slight performance degradation as compared to the dynamic method, but other values of b maintain comparable performance. Therefore, with b set to 2 or 3, there is almost no performance loss with algorithm complexity control methods of both integer J/β and static segmentation. The conclusion is further verified with CA-Polar[512,490] and CA-Polar[1024,1002] codes as shown in Fig. 3-16a and Fig. 3-16b respectively. For longer code-word lengths, due to the denser segment requirement in high curvature area, degenerated performance of $b = 2$ becomes observable, as compared to $b = 3$ with its performance always maintained.

3.5.2 Extra Quantization on Segment Slopes

The integer J/β condition, i.e. J_{i-1} divisible by β_i, significantly improves the efficiency of Algorithm 5, the integer splitting algorithm. This is achieved by increasing W_i at the step of β_i, making W_i naturally divisible by β_i, and eliminating the requirement of Algorithm 6 that specifically collects eligible partial Hamming weights for each value of W_i in order for Eq. (3.16) to be satisfied. As a result, the range of

W_i can be pre-determined as,

$$w_i J_{i-1} + \frac{(1+w_i)w_i}{2} \leq W_i \leq w_i J_{i-1} + (I_i - I_{i-1} + 1)w_i - \frac{(1+w_i)w_i}{2}, \tag{3.27}$$

where $w_i = 0, 1, 2, \ldots$ are values of Hamming weights. And Algorithm 5 can be modified with the following adjustments.

- Sweep W_i in the ranges determined by Eq. (3.27)
- Increase W_i by the step value of β_i.

These adjustments enables the execution of Algorithm 5 without checking Eq. 3.16, largely reducing the complexity of the integer splitting algorithm. However, there is still an efficiency issue that need further investigation. In Algorithm 5, the upper limit of the sweeping range for W_i is,

$$W - \sum_{l=1}^{i-1} W_l \tag{3.28}$$

which is in fact the residue integer to be split in the current and following parts W_j for $i \leq j \leq m$. If $W - \sum_{l=1}^{i-1} W_l <= J_i$, then the only splitting option for the remaining parts is $W_i = W - \sum_{l=1}^{i-1} W_l$ and $W_j = 0$ for $i+1 \leq j \leq m$. If $W_i = W - \sum_{l=1}^{i-1} W_l$ is not divisible by β_i, then this splitting option is not valid, meaning that the whole partition has to be discarded. The ratio of discards in all generated splitting patterns affects the efficiency of the integer splitting algorithm, and can be large in Algorithm 5. Moreover, the lack of guidance for increasing the reliability weight W worsens the situation. To guide pattern generation in the order of high-to-low likelihood, the governing reliability weight W is supposed to start from 0 and increase with the step of 1. With the additional rules applied in the integer splitting algorithm, the discarding ratio can be high or even 100% for some values of W, for which few or no splitting pattern is available.

Although the high efficiency of the Landslide algorithm, which dominates the complexity of ORBGRAND, makes it tolerable to have some low efficiency in the

split pattern generation, there are issues in VLSI implementations that make them unwelcome. Firstly, checking conditions and making decisions are not convenient operations to implement in hardware, resulting in circuit design complexity inappropriate for high throughput or low power design techniques. Secondly, discarding output leads to timing complexity in synchronization between components, again resulting in difficulties in the application of high throughput techniques such as parallelism and pipelining. Therefore, it is desirable to produce an efficient sweeping scheme for W and an effective algorithm of split pattern generation for each value of W.

The high-efficiency scheme presented in the next section requires an additional restriction to parameter quantization for each segment such that,

$$\frac{\beta_i}{\beta_{i+1}} \in \mathbb{Z}_+, \text{ for } 1 \leq i < m \tag{3.29}$$

Here we assume $\beta_i \geq \beta_{i+1}$ for $1 \leq i \leq m-1$. The assumption is valid when we ignore the curvature in the high-reliability region of the reliability curve and represent it with an extended straight line, as illustrated in Fig. 3-6. The additional restriction imposed on parameter quantization may result in performance loss to the algorithm, which is demonstrated to be negligible as shown in Fig. 3-17. The new restriction, labeled as β/β, is applied in addition to all previously proposed algorithm complexity control techniques including integer J/β restriction and static segmentation. Performance curves without those restrictions are also plotted for reference, showing trivial performance change with each new restriction added.

Fig. 3-18a and Fig. 3-18b present simulations on longer code-words confirming the little impact of β/β restriction on decoding performance. Again, the robustness ORBGRAND is demonstrated, allowing future additional algorithm complexity control techniques.

3.5.3 High-efficiency Split Pattern Generation

An efficient split pattern generation must be accompanied with a guided increment procedure of the reliability weight W. The simple procedure of starting with 0 and

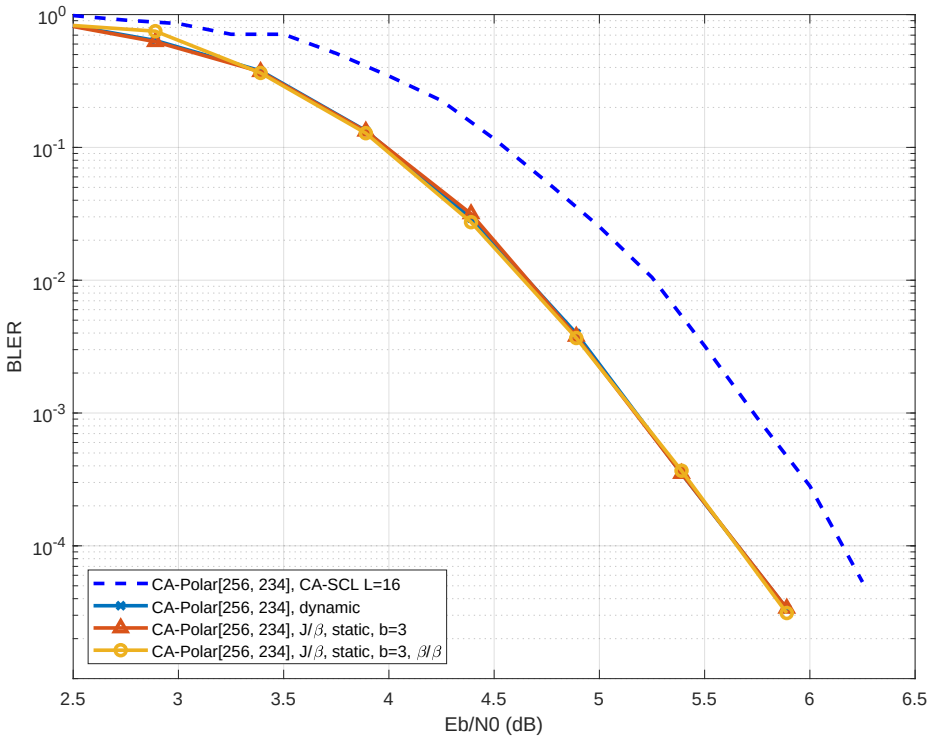
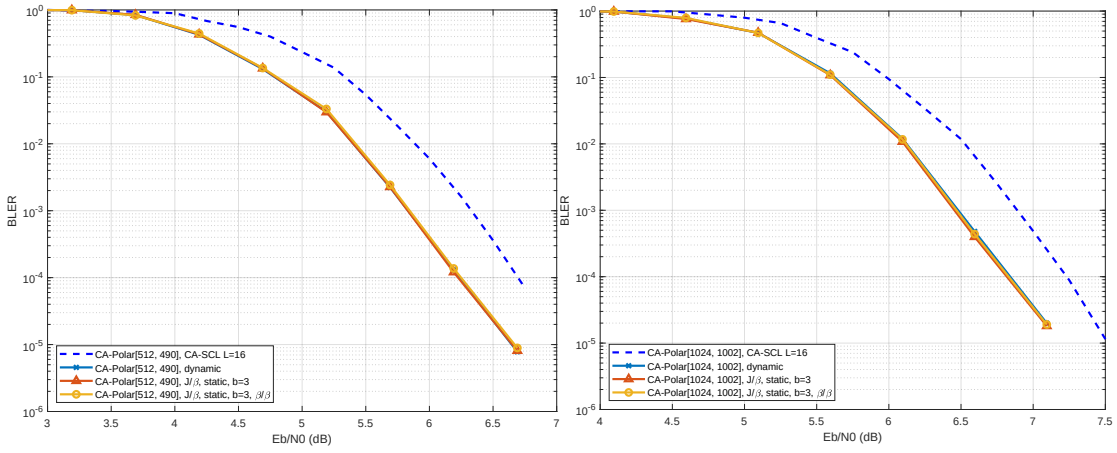


Figure 3-17: performance evaluation of β/β restriction in addition to J/β and $m = 3$ static segmentation with $b = 3$; Performed on CA-Polar[256,234] code; Dynamic segmentation without restrictions is used as reference



(a) (b)

Figure 3-18: performance evaluation of β/β restriction in addition to J/β and $m = 3$ static segmentation with $b = 3$; Performed on CA-Polar[512,490] code in (a) and CA-Polar[1024,1022] code in (b); Dynamic segmentation without restrictions is used as reference

increasing by 1 can result in a large ratio of W values with no split pattern, thus negatively impacting the efficiency of the split pattern generation algorithm. Even with some values of W that have valid split patterns, Algorithm 5 can still frequently encounter invalid patterns in its procedure of finding valid ones. Therefore, the increment procedure of W should be designed together with the split pattern generation algorithm in order to maximally lower the probability of empty or invalid split patterns and achieve the maximal algorithm efficiency.

We assume that the segmentation is performed as in Fig. 3-6 with the segment in center region extended to high-reliability region, and the quantization is performed with the integer J/β rule and the integer β/β restriction. The resulted m segments and their line parameters are defined in Eq. (3.11), with the following properties,

$$\begin{cases} \beta_i \geq \beta_{i+1}, \text{ for } 1 \leq i < m \\ J_{i-1} \leq J_i, \text{ for } 1 \leq i < m \end{cases}. \tag{3.30}$$

A split pattern $W^m = (W_1, W_2, \ldots, W_m)$ for W have its elements assigned to m segments. The i-th segment has partial noise patterns generated, each of which has its reliability weight summed to W_i. The value of W_i can be zero or non-zero. We define a vector $a^m = (a_1, a_2, \ldots . a_m)$ to label the segments having zero or non-zero reliability weights, i.e.

$$a_i = \begin{cases} 1, \text{ for } W_i > 0 \\ 0, \text{ for } W_i = 0 \end{cases} \tag{3.31}$$

In fact, the 1s in a^m indicate the combination of segments whose non-zero reliability weights sum up to W. There are a total of $2^m - 1$ possibilities of a^m ignoring the case of all-zero vector that corresponds to $W = 0$. Each of the m segments, the i-th one

for example, has its minimum and maximum achievable partial reliability weights as,

$$\begin{cases} W_{i,min} = J_{i-1} + \beta_i \\ W_{i,max} = J_{i-1}(I_i - I_{i-1}) + \beta_i \frac{(1+I_i-Ii-1)(I_i-I_{i-1})}{2} \end{cases}. \tag{3.32}$$

For a given label vector a^m, the minimum and maximum reliability weights achievable by segments labeled by 1s from a^m are,

$$\begin{cases} W_{min,a^m} = \sum_{i=1}^{m} a_i W_{i,min} \\ W_{max,a^m} = \sum_{i=1}^{m} a_i W_{i,max} \end{cases}, \tag{3.33}$$

which gives the valid range of reliability weight W allowed by the segment combination, i.e $W \in [W_{min,a^m}, W_{max,a^m}]$. Another property of the value of W allowed by the segment combination is its granularity G_{β,a^m}, which must be able to divide W. If $a_{i_e} = 1$ is the last non-zero element in a^m, then the granularity of the segment combination is

$$G_{\beta,a^m} = \beta_{i_e}, \tag{3.34}$$

which is also the smallest β value among segments labeled by 1. Collectively, in order for a reliability weight W to be valid for the segment combination specified by a^m, the following condition must be satisfied,

$$\begin{cases} W \in [W_{min,a^m}, W_{max,a^m}] \\ \frac{W}{G_{\beta,a^m}} \in \mathbb{N}. \end{cases} \tag{3.35}$$

If we list all combinations of a^m in the increasing order of W_{min,a^m}, resulting in $A(m) = \{a^{m,1}, a^{m,2}, \ldots, a^{m,2^m-1}\}$, then the guidance for the increment procedure of W is that, for each increased value of W, there must be a least one combination of $a^m \in A(m)$ such that Eq. (3.35) is satisfied. When there are multiple segment combinations available for a value of W, their order of treatment can be arbitrated

by the implementation convenience. W can start its value with $W_{min,a^{m,1}}$. As W falls into the range of $[W_{min,a^{m,i}}, W_{max,a^{m,i}}]$, for $1 \leq i \leq 2^m - 1$, W can increase by the granularity of $G_{\beta,a^{m,i}}$ and the resulted reliability weight is ensured to have split patterns over those segments marked by 1 in $a^{m,i}$. In practice, due to the small value of m for segment number, $2^m - 1$ combinations of segments are manageable and can be all generated at once.

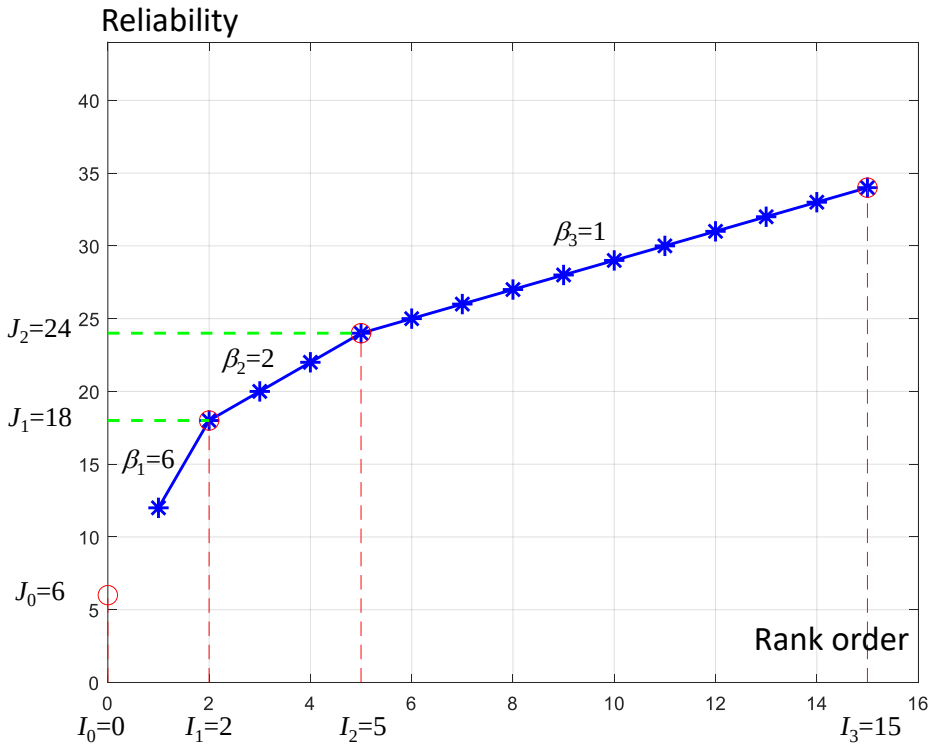


Figure 3-19: Example for the illustration of high-efficiency integer splitting algorithm

It is convenient to illustrate with an example as shown in Fig. 3-19. The example considers $m = 3$ segments that are commonly selected for segmentation in practice. The segment combinations labeled by $a^{m,i} \in A(m)$ and ordered by W_{min,a^m} are listed in Table 3.1. Guided by the information from the table, the value of W can jump to 12 after the initial value of 0. Then W is associated with segment combination $a^{m,1} = (1, 0, 0)$ and should increase by its granularity of 6. After 18, the next value of W should be 20 and W falls into the range of both $a^{m,1} = (1, 0, 0)$ and $a^{m,2} = (0, 1, 0)$. Then W increases by 2, the lower value of the granularities of two ranges, but $a^{m,1}$ is involved only when W is divisible by 6. When W reaches 25, $a^{m,1}$, $a^{m,2}$ and $a^{m,3}$ are

all involved and the increasing granularity becomes 1.

Index i	$a^{m,i}$	W_{min,a^m}	W_{max,a^m}	G_{β,a^m}
1	$(1,0,0)$	12	30	6
2	$(0,1,0)$	20	66	2
3	$(0,0,1)$	25	295	1
4	$(1,1,0)$	32	96	2
5	$(1,0,1)$	37	325	1
6	$(0,1,1)$	45	361	1
7	$(1,1,1)$	57	391	1

Table 3.1: List of segment combinations $A(m) = \{a^{m,1}, a^{m,2}, \ldots, a^{m,2^m-1}\}$ for the example in Fig. 3-19 with $m = 3$.

Each valid reliability weight W is associated with at least one segment combination specified by a^m. In fact, W is to be split over those segments marked by 1 in a^m. Let the number of 1-marked segments be K_s, and their indices in a^m are $1 \leq i_j \leq m$ for $1 \leq j \leq K_s$, then $a_{i_j} = 1$. Constrained by a^m, W is to be split into K_s non-zero partial weights $W_1, W_2, \ldots, W_{K_s}$ that are to be assigned to the corresponding K_s 1-marked segments respectively. In the solution to this K_s-part integer splitting problem, the first step is to transform W to

$$W' = W - W_{min,a^m}. \tag{3.36}$$

where W_{min,a^m} is defined in (3.33). Then the problem is converted to the one that splits W' into K_s non-negative integers $W'_i \in \mathbb{Z}_+, 1 \leq i \leq K_s$ with their granularity as $\{\beta_{i_1}, \beta_{i_2}, \ldots, \beta_{i_{K_s}}\}$ respectively. The high-efficiency splitting algorithm for W' over the K_s segments specified by a^m is presented in Algorithm 7. Each generated split set $\{W'_1, W'_2, \ldots, W'_{K_s}\}$ is then converted to the desired split set $\{W_1, W_2, \ldots, W_{K_s}\}$ via,

$$W_j = W'_j + W_{i_j,min}, \text{ for } 1 \leq j \leq K_s. \tag{3.37}$$

where $W_{i_j,min} = J_{i_j-1} + \beta_{i_j}$ is defined in Eq. (3.32).

Algorithm 7 The Integer Splitting Algorithm with Granularity

Input: W', $\{\beta_{i_1}, \beta_{i_2}, \ldots, \beta_{i_{K_s}}\}$

Output: $\{W'^{K_s,k} : k = 1, 2, \ldots\}$

$k \leftarrow 0, W'_1 \leftarrow 0$
while $W'_1 \leq W'$ **do**
  $W'_2 \leftarrow 0$
  while $W'_2 \leq W' - W'_1$ **do**
    $W'_3 \leftarrow 0$
    ..... (nested loops over W'_i, $i = 3, 4, \ldots, K_s - 2$)
    $W'_{K_s-1} \leftarrow 0$
    while $W'_{K_s-1} \leq W' - \sum_{i=1}^{K_s-2} W'_i$ **do**
      $W'_{K_s} \leftarrow W' - \sum_{i=1}^{K_s-1} W'_i$
      $k \leftarrow k + 1$
      $W'^{K_s,k} \leftarrow \{W'_1, W'_2, \ldots, W'_{K_s}\}$
      $W'_{K_s-1} \leftarrow W'_{K_s-1} + \beta_{i_{K_s-1}}$
    end while
    ..... (nested loops over W'_i, $i = 3, 4, \ldots, K_s - 2$)
    $W'_2 \leftarrow W'_2 + \beta_{i_2}$
  end while
  $W'_1 \leftarrow W'_1 + \beta_{i_1}$
end while
return $\{W'^{K_s,k} : k = 1, 2, \ldots\}$

Consider a value $W = 48$ which is valid to $a^{m,i}, 2 \leq i \leq 6$. For $a^{m,4}$, $W = 48$ is first converted to $W' = 16$, then W' is split between segment 1 and segment 2 with granularity of 6 and 2 respectively. From Algorithm 7, the splitting patterns $\{W'_1, W'_2\}$ are: $\{0, 16\}$, $\{6, 10\}$ and $\{12, 4\}$. They are then converted to desired patterns $\{W_1, W_2\}$ as: $\{12, 36\}$, $\{18, 30\}$ and $\{24, 24\}$. Similarly, for $a^{m,5}$, $W = 48$ is converted to $W' = 11$ and the splitting patterns $\{W'_1, W'_3\}$ are $\{0, 11\}$ and $\{6, 5\}$, with the desired patterns $\{W_1, W_3\}$ as $\{12, 36\}$ and $\{18, 30\}$.

The integer J/β and β/β rules, the properties in Eq. (3.30) and the rule for W in Eq. (3.35) ensure the validation of each generated split set W^{K_s}, making Algorithm 7

the desired high-efficiency split pattern generation algorithm, which, along with Eq. (3.37), is to substitute Algorithm 5.

3.6 Extension to High-order Modulation

In Chapter 2, the discussion of high-order GRAND-MO assumes that hard detected symbols have been available from the demodulator, where the process of hard detection discards a considerable amount of soft information, resulting in the loss of potential performance improvement. The proposal of augmented constellation is a simple effort of preserving some information and has been demonstrated to effectively enhance the decoding performance. A full soft decoding, therefore, with all soft information preserved, is expected to bring in significant performance improvement to the decoder. [63].

3.6.1 Conventional Soft Bit Decomposition

In high-order modulation systems, an n-bit binary block code-word $c^n \in \{0,1\}^n$ is modulated to n_s symbols by mapping every m_s bits into a symbol, i.e. $\text{mod}(c^n) = x^{n_s} \in \chi^{n_s}$, where $n_s = n/m_s$ and χ is the set of complex constellation with size $|\chi| = M = 2^{m_s}$. The symbol sequence is transmitted and impacted by independent additive complex noise, $N^{n_s} \in \mathbb{C}^{n_s}$, resulting in the received signal sequence as $Y^{n_s} = x^{n_s} + N^{n_s}$. With a soft demapper, Y^{n_s} is decomposed to a sequence of soft metrics $\gamma^n \in \mathbb{R}^n$, which is then passed over to the soft decoder for decoding process.

The procedure of soft demapping a received signal sequence $Y_i, 1 \leq i \leq n_s$, following the conventional log-domain Maximum *a posteriori* (Log-MAP) demapping algorithm [44], is a high-complexity process, which not only involves exponential and logarithmic computations, but also has the number of operations at the order of $O(M)$. In Log-MAP demapper, for each received signal Y the resulted soft metrics $\gamma^{m_s} = \{\gamma_1, \gamma_2, \ldots, \gamma_{m_s}\}$ are the LLRs of the modulated m_s-tuple bit set

$c^{m_s} = \{c_1, c_2, \ldots, c_{m_s}\}$, which can be obtained as [107],

$$\gamma_i = \log \frac{P(Y|c_i = 1)}{P(Y|c_i = 0)} = \log \frac{\sum\limits_{x \in \chi_i^{(1)}} P(Y|x)}{\sum\limits_{x \in \chi_i^{(0)}} P(Y|x)}, \text{ for } 1 \leq i \leq m_s, \tag{3.38}$$

where $\chi_i^{(1)}$ represents the set of constellation symbols with their pre-modulation bit set c^{m_s} all having the i-th bit being 1, i.e. $c_i = 1$. The remaining constellation symbols, with their i-th bit being 0, fall into the set $\chi_i^{(0)}$ and $|\chi_i^{(0)}| = |\chi_i^{(1)}| = M/2$. Therefore, M operations of probability computation are required to obtain the soft information of each code-word bit in the demapper, leading to the complexity order of $O(M)$.

In AWGN channels, the conditional PDF $P(Y|x)$ takes the form of

$$P(Y|x) = \frac{1}{\pi N_0} e^{\frac{-|Y-x|^2}{N_0}}, \tag{3.39}$$

where N_0 is the complex noise density. Using the well-known max-sum approximation [92], Eq. (3.38) becomes [107]

$$\gamma_i \approx \log \frac{\max\limits_{x \in \chi_i^{(1)}} P(Y|x)}{\max\limits_{x \in \chi_i^{(0)}} P(Y|x)} = -\frac{1}{N_0} \left(\min_{x \in \chi_i^{(1)}} |Y - x|^2 - \min_{x \in \chi_i^{(0)}} |Y - x|^2 \right) \tag{3.40}$$

The Max-Log-MAP demapper [92] resulted from the max-sum-approximation has the arithmetic complexity reduced avoiding logarithm and exponential operations, but the $O(M)$ operation complexity still remains. The reduction of the number of operations has been the focus of a number of soft demapping algorithms [7,28,67,102], all at the cost of potential performance loss.

ORBGRAND, as described in previous sections, is designed to work on LLR soft information. Like conventional soft decoders, it can naturally operate on soft demapped LLRs obtained from Eq. (3.38) or Eq. (3.40), or on soft bits provided by any other low-complexity soft demapping algorithms. Thus, existing systems

with soft demappers deployed can have the binary ORBGRAND decoder seamlessly adopted. On the other hand, as emphasized in Chapter 2, the GRAND algorithm has the unique capability of working directly on symbols, which is inherited by its variant, ORBGRAND, making it possible to eliminate the need of the complex soft demapper.

3.6.2 ORBGRAND Extension to Symbols

In binary systems, there is only one possible error for a transmitted bit in detection, i.e. 1 detected for the transmitted 0 or vice versa. A transmitted high-order symbol, however, can have any symbol in the constellation except itself as the wrongly detected symbol. Therefore, it is no longer convenient to check the likelihood of error patterns as in Eq. (3.2) for binary systems. Instead, it is more advantageous to evaluate the probability of a demodulated symbol sequence being the correct one, and directly search for the most probable symbol sequence. This is similar to the symbol decoding in Algorithm 3, the high-order GRAND-MO algorithm, except that we are now designing the high-order pattern generator for ORBGRAND.

In the procedure of decoding directly over symbols, a sequence of constellation symbols guessed from the received signal sequence Y^{n_s} is demapped to a bit sequence that is then checked against the code-book $\mathcal{C}$. Here the hard detection demapping is an operation as simple as a table lookup. If the check passes, then the transmitted symbol sequence x^{n_s} is viewed as being recovered, otherwise the next most likely symbol sequence is selected for checking. The optimally decoded symbol sequence $\widehat{x^{n_s}} \in \chi^{n_s}$ should satisfy,

$$\widehat{x^{n_s}} = \arg \max_{t^{n_s} \in \chi^{n_s},\ \mathrm{demap}(t^{n_s}) \in \mathcal{C}} P(t^{n_s}|Y^{n_s}) \tag{3.41}$$

Assuming all symbol sequences in χ^{n_s} are transmitted with equal probability, we have,

$$P(t^{n_s}|Y^{n_s}) = P(Y^{n_s}|t^{n_s}) \propto \prod_{i=1}^{n_s} e^{\frac{-|Y_i - t_i|^2}{N_0}} \propto -\sum_{i=1}^{n_s} |Y_i - t_i|^2, \tag{3.42}$$

where $|Y_i - t_i|^2$ is the squared Euclidean distance between a candidate constellation symbol and a received signal in the complex plane. The hard-detection sequence Υ^{n_s} is obtained by choosing the closest constellation symbol to each element of the received signal sequence Y^{n_s}.

Therefore, unlike the binary ORBGRAND that generates noise patterns z^n ordered by their reliability sum $\text{Rel}(z^n)$, the high-order ORBGRAND, instead, generates candidate symbol sequence $t^{n_s} \in \chi^{n_s}$ following the increasing order of the squared distance to the received sequence Y^{n_s}, i.e.

$$\text{Dis}(t^{n_s}) = \sum_{i=1}^{n_s} |Y_i - t_i|^2. \tag{3.43}$$

For a received signal sequence Y^{n_s}, the ORBGRAND decoding is adjusted as follows,

- For each element Y_i of the received signal sequence Y^{n_s}, identify $n_n + 1$ closest neighboring symbols in constellation that are sorted by their squared distance to Y_i and are denoted as $\{\tilde{x}_{i,j} \in \chi, 0 \leq j \leq n_n\}$. Symbol $\tilde{x}_{i,0}$ is the closest one to Y_i and is thus its standard hard-detection symbol. Other neighboring symbols all have longer squared distance to Y_i than $\tilde{x}_{i,0}$, with the extra difference named as the exceeding distance, and computed as $\delta_{i,j} = |\tilde{x}_{i,j} - Y_i|^2 - |\tilde{x}_{i,0} - Y_i|^2$ for $1 \leq j \leq n_n$.

- The hard detection sequence Υ^{n_s} is constructed with $\{\tilde{x}_{i,0}, 1 \leq i \leq n_s\}$ and its demapped bit sequence is checked against the code-book. If passed, $\widehat{x^{n_s}} = \Upsilon^{n_s}$ and the decoding is successful.

- The remaining symbols $\{\tilde{x}_{i,j}, 1 \leq i \leq n_s, 1 \leq j < n_n\}$ form a re-indexed set $\Omega = \{\tilde{x}_i, 1 \leq i \leq n_s n_n\}$, with all members sorted by their exceeding distance $\{\delta_i, 1 \leq i \leq n_s n_n\}$ re-indexed together with $\tilde{x}_i$, leading to the exceeding distance curve, where δ_i is plotted along the index.

- If Υ^{n_s} fails the code-book check, some of its elements can be replaced with symbols selected from Ω, forming a new symbol sequence for the next checking. The resulted symbol sequence has a distance to Y^{n_s} larger than Υ^{n_s}, with the

difference equaling to the exceeding distance sum of replacing symbols, which in fact determines the likelihood of the resulted symbol sequence. Thus, we can use the exceeding distance sum to govern the order of replacing symbol patterns.

- For a given exceeding distance weight Δ, the generated replacing symbol patterns $\{\tilde{x}_{I_j} \in \Omega, j = 1, 2, \ldots\}$ must have their exceeding distances satisfy $\sum_j \delta_{I_j} = \Delta$. Symbols in a pattern are then used to replace the corresponding symbols in Υ^{n_s} and form a candidate symbol sequence t^{n_s} for code-book check. The increasing procedure of Δ and the corresponding pattern generation can follow exactly the strategy of ORBGRAND for binary systems, as described in previous sections.

- In each generated pattern, it is possible that two or more symbols belong to the same neighboring set of a received signal. The situation has no impact on the decoding performance, however, it is recommended to discard the pattern to save computations.

Like in binary systems, the high-order ORBGRAND algorithm is a near-optimal solution with the approximation to the distance curve having little impact on the decoding performance. The algorithm has no complex arithmetic operations involved, so its complexity is determined by the number of neighboring symbols, $n_n + 1$, for each element of the received symbol sequence. The maximum value of $n_n + 1$ can be as large as M, with all constellation symbols considered. However, as shown in later simulations, $n_n = 8$ is sufficient for 16-QAM or higher order modulation without observable performance loss, which suggests a significant complexity advantage as compared to conventional soft demapper algorithms. Therefore, the high-order ORBGRAND provides a potentially low-complexity decoding solution as compared to conventional decoders requiring complex soft demappers.

3.6.3 Simulation and Analysis

We consider the CA-Polar[128,99] code and 256-QAM modulation to evaluate the performance of high-order ORBGRAND algorithm. For management simplicity the dynamic segmentation with 3 segments is selected without additional algorithm complexity control rules, and the scheme, as shown in Section 3.5, achieves near optimal ORBGRAND performance.

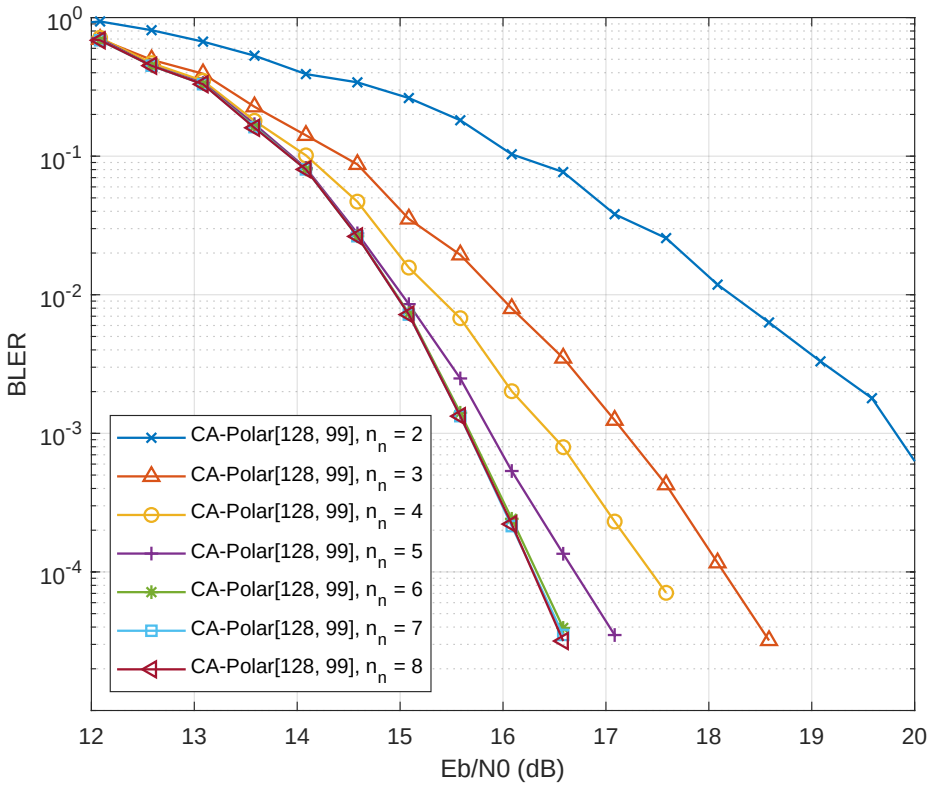


Figure 3-20: High-order ORBGRAND simulations evaluating the influence of the number of neighboring symbols n_n to decoding performance; Simulations performed on CA-Polar[128, 99] code with 256-QAM modulation; Dynamic segmentation with 3 segments

When the total number of included neighboring symbols $n_n + 1$ increases, the performance of the high-order ORBGRAND is expected to improve due to the more information included in decoding, as demonstrated in Fig. 3-20, with n_n ranging from 2 to 8. As n_n reaches 6, there is little performance improvement when its value increases further. Therefore, the inclusion of at most $n_n + 1 = 9$ nearest symbols is sufficient for optimal decoding performance of high-order ORBGRAND, which are

practically conveniently to gather. For a received signal, its hard detection symbol can be found with simple quantization of the real valued complex components on to the constellation grid, from which the 8 other nearest neighboring symbols are straightforward on the grid. This observation coincides with a complexity reduction scheme for conventional soft demappers, from which a number of soft demapper algorithms [102, 107] are proposed to have the complexity reduced to $O(\log_2 M)$.

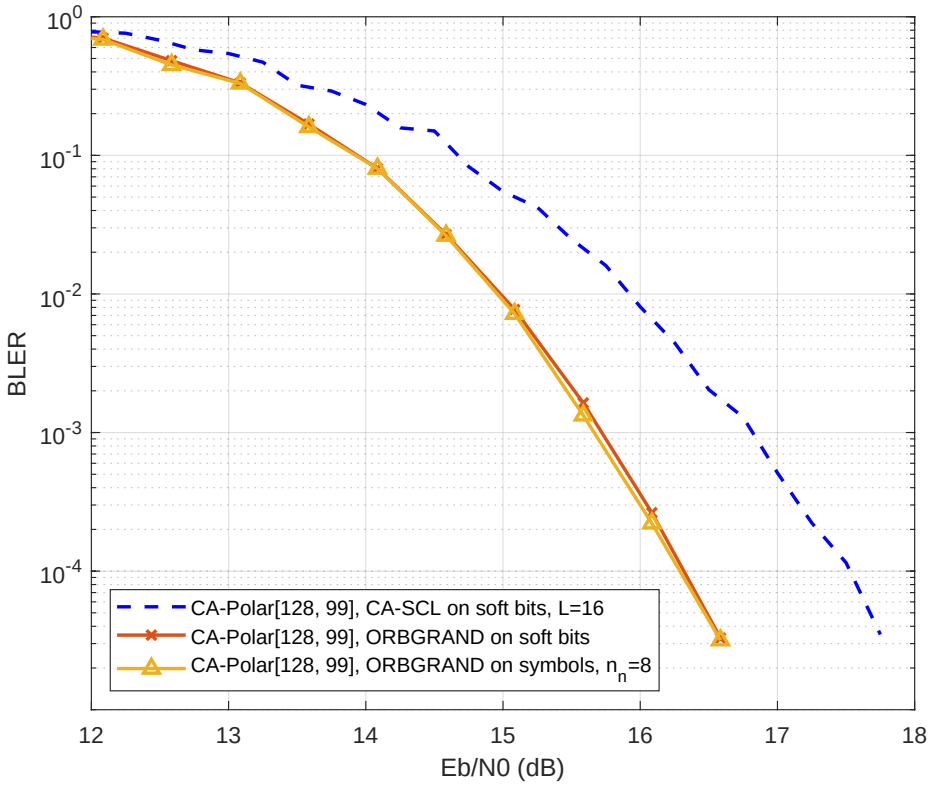


Figure 3-21: Performance evaluation of CA-Polar[128, 99] code with 256-QAM modulation; Comparison is made between the CA-SCL decoder with soft demapper, the binary ORBGRAND with soft demapper, and the high-order ORBGRAND on symbols with $n_n = 8$

The high-order ORBGRAND performance on CA-Polar[128,99] code and 256-QAM modulation is compared to the binary ORBGRAND and the CA-SCL decoders with soft demappers, as presented in Fig. 3-21. The CA-SCL decoder with list size of 16 is evaluated with the open source AFF3CT toolbox [27]. The soft demapper for the binary ORBGRAND takes its complete form in Eq. (3.38) with the full complexity of $O(M)$. Fig. 3-21 shows that there is little performance difference between the binary

ORBGRAND and the high-order ORBGRAND, which indicates that the full soft demapper in Eq. 3.38 preserves all the soft information from received signals. It also means that the choice of $n_n = 8$, though largely reducing the complexity, resulting in almost no performance loss. On the other hand, there is more than 1 dB decoding gain in ORBGRANDs at BLER of 10^{-3} than the standard CA-SCL decoder, which is a significant performance improvement.

Conventional soft decoders such as CA-SCL decoders assume the Gaussian distribution of channel noise in real-valued signals received from binary transmitted bits. However, in high-order modulation systems, received complex signals are contaminated by complex Gaussian noise, and after soft demapper the noise in soft bits no longer follows Gaussian distribution. The complete soft information thus cannot be effectively captured by CA-SCL decoders. The binary ORBGRAND, relying on the reliability of received bits, and decoding by removing noise error from bits, can still capture the soft information preserved by the soft demapper. The high-order ORBGRAND, working by directly guessing complex Gaussian noise, explores the original noise information and can thus naturally achieve optimal performance without the complexity and interference from soft demappers. In this sense, GRAND is universal not only to code-books, but also to noise distributions.

3.7 Summary

In the framework of GRAND, soft detection decoding, due to its known advantage in decoding performance, is required for the evaluation of old and new candidate codes in short code solutions to URLLC applications. We have introduced ORBGRAND, a practical soft detection variant of guessing random additive noise decoding, with which it is possible to decode any moderate redundancy code with near optimal performance.

ORBGRAND offers a range of algorithm complexities with its basic version being the simplest and requiring the least soft information. The core algorithm of the basic ORBGRAND generates integer partitions, for which we proposed the Landslide

algorithm, which is suitable for efficient hardware and real-time implementation. That algorithm is an essential component for the full ORBGRAND, which has higher design complexity, but can better exploit soft information at higher SNRs for additional decoding gains. Simulation results show that the performance of ORBGRAND is directly dependent on how well the reliability curve is approximated, and the basic ORBGRAND adopts the simplest one. Inspired by the finding, we proposed the piece-wise linear approximation to the reliability curve, which optimizes ORBGRAND across all SNRs.

The ORBGRAND algorithm, curve fitting techniques and its robustness with complexity improvement are established with simulations. The decoding performance is dependent on the design complexity of ORBGRAND, but the 3-line version is capable of maintaining close-to-optimal performance in most scenarios. ORBGRAND is demonstrated to provide close-to-ML soft decoding performance on a variety of candidate short codes including CA-Polar, BCH, RLC and CRC codes, making it possible to select the best candidate for a target application. A number of algorithm complexity control methods are proposed that largely enhance the efficiency of practical implementation, and are demonstrated to have little impact on decoding performance, demonstrating the robustness of ORBGRAND, and facilitating VLSI implementations.

Originally designed for binary systems, ORBGRAND is seamlessly ported to high-order modulations. Directly working on symbol sequences, ORGRAND eliminates the need of high-complexity soft demappers for conventional decoders. Instead of seeking the most probable bit error patterns, ORBGRAND searches for the most likely symbol sequence whose hard-demapped bits satisfy the code-book check. Simulations show more than 1 dB gain of ORBGRAND at BLER of 10^{-3} in the settings of CA-Polar[128,99] and 256-QAM modulation.

The ORBGRAND algorithm, with its near-optimal and robust performance, controllable complexity, and portability in systems with any modulation orders, provides a practical universal soft decoding platform for selection and adoption of short code candidates for URLLC applications.

Chapter 4

Error Correction Codes Selection

With the universal optimal (or near-optimal) decoding platform offered by GRAND, in both hard and soft detection, it is possible to evaluate candidate channel codes and pick up the best one for the target application. In this chapter, we evaluate a number of popular short code candidates and investigate the potentiality of Cyclic Redundancy Checks (CRCs) due to their simplicity in both encoding and code-book check. Then, using CRC codes as reference, we evaluate the performance of Random Linear Codes (RLCs), which offers the unique security feature to applications.

4.1 Introduction

Owing to the simplicity of their implementation and their powerful error detection capability, CRC codes [81] are widely adopted in applications in storage, networking, and communications. CRCs are sometimes used not only to error detect, but also to assist in error correction. For example, to assist in early stopping of Turbo decoding iterations [30], or joint decoding of CRCs and convolutional codes [108]. In list decoding architectures, a CRC is commonly used for selection from a list of candidates. This technique has been applied to Turbo codes [76] and Polar codes (CA-Polar) [13, 78, 100]. The latter leads to the success of CRC-aided successive cancellation list (CA-SCL or SCL for brief) decoding algorithms. The better performance at shorter code lengths of CA-Polar codes, as compared to other common Shannon

limit approaching codes, such as Turbo codes and LDPC codes [66], has led to their inclusion in 5G-NR control communications where latency is of particular concern [3].

With the error correction capability enabled by CRC, significant benefits can be brought to existing systems, but have heretofore seen only limited use. For example, the Automatic Repeat Request (ARQ) scheme of networking can be upgraded to Hybrid ARQ through the use of CRC's error correction. Similar benefits can be found in IoT or personal area network (PAN). Previous research on the use of CRCs for error correction focused only on fixing one or two erroneous bits [24, 96]. Soft detection techniques such as Belief Propagation (BP) and Linear Programming (LP) [103] have also been considered for decoding CRCs, but their performance is severely limited when used for short and high rate codes. No previous approach leads to a broadly applicable decoding method capable of extracting the maximum error correction performance from CRCs.

Random codes, due to their convenience of modeling, have been mainly used for theoretical capacity research. Some capacity achieving long codes, such as Turbo codes and Turbo product codes, use interleavers to their code-words emulating the features of random codes. Without practically implementable decoders, they have heretofore not been considered for real applications. Even for short random codes, the complexity of a near-optimal decoder is too overwhelming for practical consideration. Due to its easiness of code-book generation, if decodable, RLC can have a unique feature of security with its code-book updated in real time.

With GRAND, a universal optimal (or near-optimal) decoder with both hard detection and soft detection variants, we are able to evaluate the performance of CRC and RLC codes and compare them to other potential short code candidates such as Reed-Solomon Codes [87], BCH codes [23] and CA-Polar code [13, 78, 100]. Using the best published CRC generator polynomials [58] we find that CRCs perform as well as BCH codes at their available settings, but CRCs can operate for a wider range of code lengths and rates. Using GRAND to compare the performance of Polar, CA-Polar and CRC codes, we find that the CRC is not aiding so much as dominating the decoding performance potential. Furthermore, the celebrated CA-

SCL algorithm underperforms for short packets because the CRC bits are only used for error detection.

Featuring even more flexible code-book settings, RLCs possess security potentials by code-book re-randomization, but at very high code-rates RLC performance degrades in comparison to selected CRCs. However, due to the limited code-word spaces of very high-rate codes, searching for good code-book is computational possible, resulting in comparable performance to CRCs.

4.2 Channel Settings and Decoder Configurations

In convention, practical channel codes and their decoders are designed assuming Additive White Gaussian Noise (AWGN) channels or their hard detection equivalence, Binary Symmetric Channels (BSCs). Therefore, for fair comparison, we use BSCs to evaluate hard detection decoding performance, and AWGN channels for soft detection decoding.

For AWGN channels, Signal to Noise Ratio (SNR) is normally used to measure the noise condition, which is simply the ratio of signal power and noise power. To fairly measure the error correction capability, it is more common to use the ratio of information bit energy before encoding and the noise density to measure the noise condition of channels, i.e E_b/N_0. The relationship between SNR and E_b/N_0 is,

$$\frac{E_b}{N_0} = \frac{\text{SNR}}{2R},$$

where R is the rate of channel code. For convenience of comparison, we also substitute E_b/N_0 to the corresponding error probability p in BSC for hard decoding performance evaluation and their relationship is,

$$p = \text{Q}\left(\sqrt{\frac{2RE_b}{N_0}}\right).$$

For hard detection GRAND decoding in BSC channels, the order of testing noise

patterns follows their Hamming weight from low to high. It can be easily achieved with GRAND-MO by setting $\Delta l = 0$, or with a direct and simple implementation as follows. For patterns with identical Hamming weights, we can choose a sweeping order of generation, in which a grouped pattern sweeps from one end of sequence to the other end. A new grouped pattern is obtained by permuting the bits in the group. When permutation exhausts, an extra non-flipped bit is included and the permutation process restarts. This pattern generator gives the decoder some advantages with burst errors which are specifically treated in GRAND-MO and the resulted algorithm is named as GRAND with simple order sweeping (GRAND-SOS). Standard decoders of linear block codes can decode up to $t = \lfloor d/2 \rfloor$ errors where d is the minimum distance of the code-book [63]. Using this limit as abandonment threshold, GRAND is guaranteed to achieve at least equivalent performance to existing code-book dedicated decoders.

ORBGRAND is naturally used for soft detection decoding in AWGN channels. For the goal of performance evaluation of channel codes, we skip most complexity control techniques and adopt dynamic segmentation with 3 segments, which, as indicated by simulations in Section 3.3, have achieved satisfactory performance leaving little space for further improvement. The performance of soft decoders is no longer restricted by minimum distance of code-books. While looser abandonment conditions always lead to performance improvement, for most channel code settings considered in this chapter, ORBGRAND with a moderate complexity can provide better performance than state-of-the-art soft decoders. Therefore, we stick to a single abandon condition of 5×10^6 checks.

4.3 Overview of Channel Codes

4.3.1 CRC Codes

As a type of cyclic code [84], CRC operates in terms of polynomial computation based on Galois field of two elements GF(2). The CRC generator polynomial can be

expressed as $g(x) = \sum_{k=0}^{n-K-1} g_k x^k$, where $n-K$ is the number of CRC bits to be appended to a message sequence, expressed as polynomial $m(x) = \sum_{i=0}^{K-1} m_i x^i$, with n being the code-word length. The CRC code-word is constructed as,

$$c(x) = x^{n-K} m(x) + \text{remainder}\left(\frac{x^{n-K} m(x)}{g(x)}\right) \tag{4.1}$$

with the remainder of polynomial division appended to the shifted message polynomial, the resulting code-word polynomial $c(x)$ is divisible by $g(x)$. The property is used in parity checks to detect errors, and provides an efficient implementation for checking code-book membership in GRAND. While Equation (4.1) presents the systematic format of CRC codes, they can also be constructed as other cyclic codes (such as BCH) as,

$$c(x) = m(x)g(x) \tag{4.2}$$

Equation (4.2) generates the same code-book as (4.1), but with lower computation complexity.

To maximize the number of detectable errors, the CRC maximizes the minimum Hamming distance of its code-book. The topic has been studied extensively and optimal generator polynomials have been published for a wide set of combinations of CRC bit numbers and message lengths [58, 59, 86]. Unlike channel codes for error correction function, the design of CRC codes need not consider decoder implementation, indicating potentially better decoding performance than normal Forward Error Correction (FEC) codes, provided that a decoder is available, i.e. GRAND.

4.3.2 RLC and Other Short Code Candidates

RLCs are linear block codes that have long been used for theoretical investigation, but have heretofore not been considered practical in terms of decodability. With GRAND, RLCs are no more challenging to decode than any other linear codes. Their generator matrices are obtained by appending a randomly generated parity component to the identity matrix, from which the parity check matrix can be computed accordingly.

In evaluation, we update the RLC code-book after the generation of each code-word and the parity check matrix on receiver side is updated correspondingly. Therefore, in addition to its flexibility, the ease of code-book construction grants RLC security features by real-time code-book updates.

The original RLC coding scheme applies no sanity check to the generated matrix, and still provides excellent performance with enough number of parity bits, as demonstrated in later simulations. For very high-rate code setting, due to the small code-word space, it is more probably to encounter closely located code-words in random code-book generation, resulting in performance degradation. In this situation, however, the small code-word space allows computationally tolerable searching of good code-books, as demonstrated with a simple example in later simulations.

Also as a type of cyclic codes, BCH codes can be constructed with Equation (4.2). In this sense, BCH codes can be viewed as a special case of CRC codes. For any positive integers $m > 3$ and $t < 2^{m-1}$, there exists a binary BCH code with code-word length of $n = 2^m - 1$, parity bit number of $n - K \leq mt$ and minimum distance of $d \geq 2t + 1$. A list of available parameters n, K and t can be found in [63]. The standard decoder for BCH codes is Berlekamp-Massey (B-M) decoder [21, 23, 71], which is also used for the widely adopted Reed-Solomon (RS) codes, a special type of BCH codes with operations extended to higher order Galois fields. It should be noted that the original proposal of GRAND has no limitation on the Galois field order [36], Here we focus on binary codes, but the methodology can be easily extended to RS codes in higher order Galois fields.

CA-Polar codes are concatenated CRC and Polar codes. Although they approach the Shannon reliable rate limit [13], Polar codes do not give satisfactory decoding performance for practical code-word lengths, even with the ML successive cancellation (SC) algorithm. A CRC is introduced as an outer code to assist decoding and the CA-SCL decoding algorithm was invented as the key to the success of the resulted CA-Polar codes [78, 100]. In the decoding process, a list of candidate code-words are produced and CRC checks are performed to make the selection with confidence. There is to our knowledge no standard hard detection decoder for Polar or CA-Polar

codes apart from GRAND decoders. For code-book membership checking in GRAND, parity check matrices can be easily derived for generator matrices, which, for linear block codes, can be obtained with a simple method that injects each vector of the K-by-K identity matrix to the encoder and collects the output vectors.

4.4 Simulated Performance Evaluation

Table 4.1 lists the code-book settings considered for CRC evaluation, along with published preferred generator polynomials. Note that the zero order 1 is omitted in the least significant bit [58].

N	K	BCH t	CRC poly.	d
127	120	1	0x65	3
127	113	2	0x212d	5
127	106	3	0x12faa5	7
128	99	N/A	0x13a46755	8
63	57	1	0x33	3
63	51	2	0xbae	5
63	45	3	0x25f6a	7
64	51	N/A	0x12e6	4

Table 4.1: List of code-book settings along with correctable error number t for BCH codes and minimum distances d for CRC codes.

4.4.1 CRC Codes vs BCH Codes

For code-book settings in Table 4.1 available for BCH codes, we measure the performance of the following code/decoder combinations:

- BCH codes with Berlekamp-Massey (B-M) decoders
- BCH codes with GRAND-SOS decoder and $AB > t$
- CRC codes with GRAND-SOS decoder and $AB > t$
- CRC codes with GRAND-SOS decoder and $AB > 4$

$AB > t$ indicates abandonment when the error number in testing patterns is larger than the code-book decoding capability, t. And $AB > 4$ is a loose condition for all available BCH codes listed in Table 4.1, with which we show the influence of the minimum distance of a code-book to its decoding performance. Simulation results for code-word length of 127 are presented in Fig. 4-1.

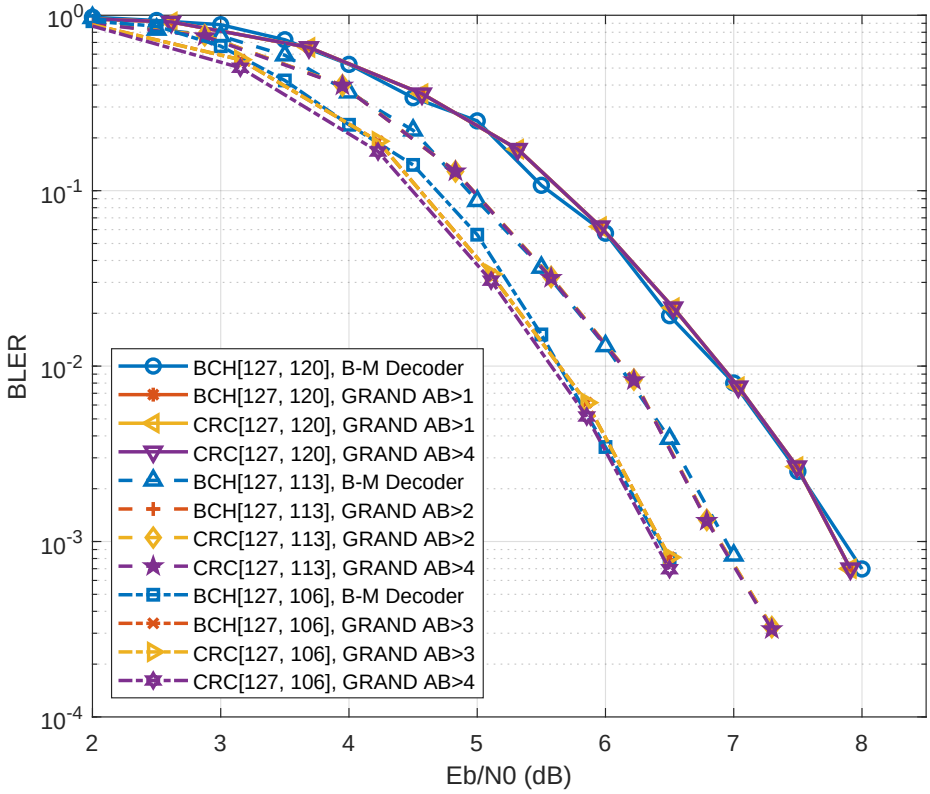


Figure 4-1: Performance evaluation of code/decoder combinations with code-word length of 127 for BCH and CRC codes.

For each code rate, all four code/decoder combinations have performance curves that are close to each other, leading to the following conclusions. For BCH codes, the GRAND-SOS algorithm has the same decoding capability as the standard Berlekamp-Massey decoder. With identical code-book settings, CRC codes have matching performances to BCH codes. Loosening abandonment condition grants little improvement to decoding performance of GRAND, so the use of abandonment to curtail complexity remains attractive. These observations are further confirmed with simulations of code-word length of 63, as shown in Fig. 4-2

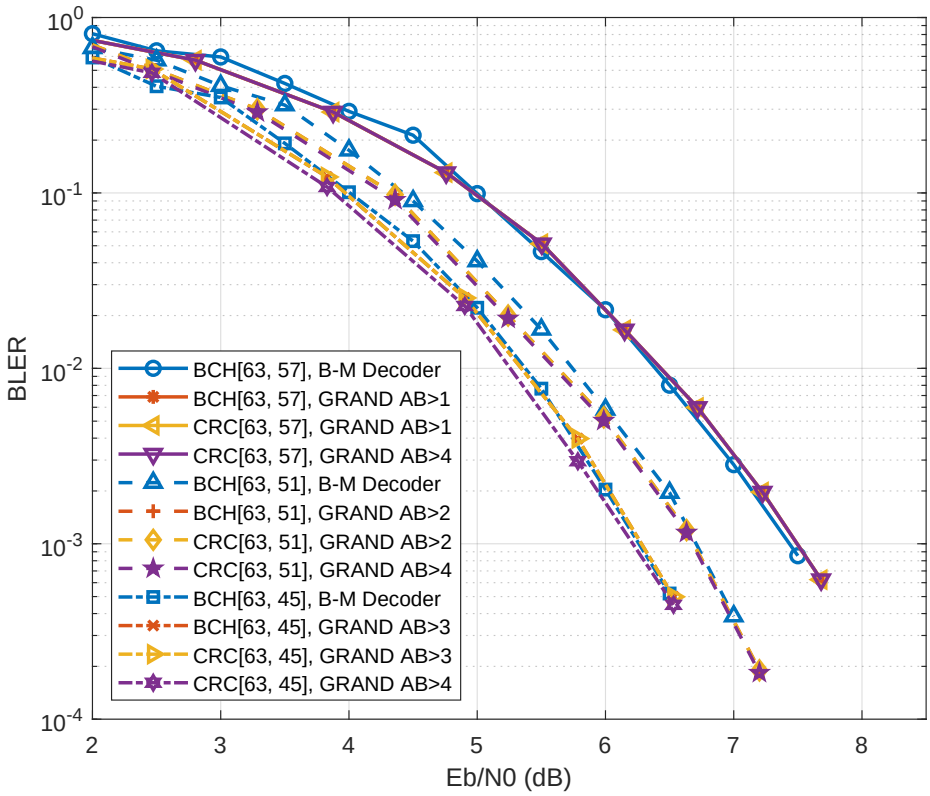


Figure 4-2: Performance evaluation of code/decoder combinations with code-word length of 63 for BCH and CRC codes.

Combined evaluation of curves in Fig. 4-1 and Fig. 4-2 indicates that performance of short codes is sensitive to code-word lengths and rates, therefore the flexibility of CRC renders it attractive vis-à-vis BCH codes in low-latency scenarios.

4.4.2 Comparison with Polar and CA-Polar Codes

In 5G mobile systems, an 11-bit CRC (CRC11) is specified for uplink CA-Polar codes, and a 24-bit CRC (CRC24) for downlink. We evaluate performance of Polar codes with or without aid from CRC11 and CRC24, and compare them to CRC codes from in Table 4.1, for code-book with $[n, K]$ set to [128, 99]. Hard detection ML performance when decoded with GRAND-SOS are presented in Fig. 4-3. A generous abandonment threshold $AB > 5$ is chosen to avoid influence from computation complexity on code performance.

Let us first assess code performance under a hard detection scenario. The Polar[128, 99] code without the CRC shows significantly worse performance than the three other codes. This is reasonable considering that Polar codes are closely related to Reed-Muller codes [75], which are known to be inferior to cyclic codes (such as BCH). The introduction of CRC codes, either CRC11 or CRC24, sufficiently breaks the undesirable structure in Polar code, bringing its performance close to that achieved by CRC29. In this sense, the "aid" from CRC is so effective that it may be preferable just to use the CRC as an error-correcting code, considering that significant encoding complexity can be saved thanks to the simple implementation of CRC encoders.

Having considered performance under hard detection, let us envisage soft detection performance, which is of particular importance given that prevailing decoders for Polar codes are soft decoders, especially when a non-trivial decoding gain is expected [63]. The extra gain comes from decoding errors beyond the hard detection decoding capability t. Therefore, we use the number of code-book checks as the abandonment condition in this case, with $AB > 5e6$ being a generous threshold that ensures optimal decoding performance. A gain of about 1.5 dB from soft decoding at BLER$= 10^{-3}$ can be observed by comparing Fig. 4-3 and Fig. 4-4. Of more interest is that under the same ORBGRAND decoding power four candidate codes follow identical performance

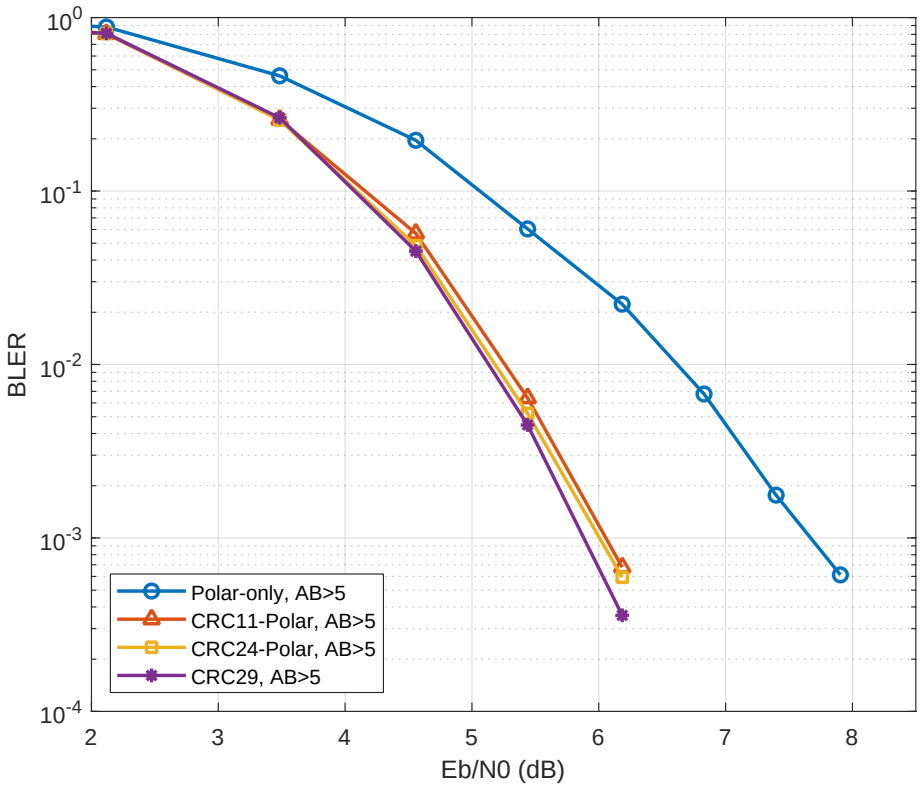


Figure 4-3: Hard detection performance of Polar[128,99], CA-Polar[128,99] with CRC11, CA-Polar[128,99] with CRC24 and CRC[128,99] (i.e. CRC29). All decoded with GRAND-SOS and $AB > 5$.

ranking as their hard detection behaviors, i.e. CRC assists Polar codes to approach the best performance achieved by the CRC code itself.

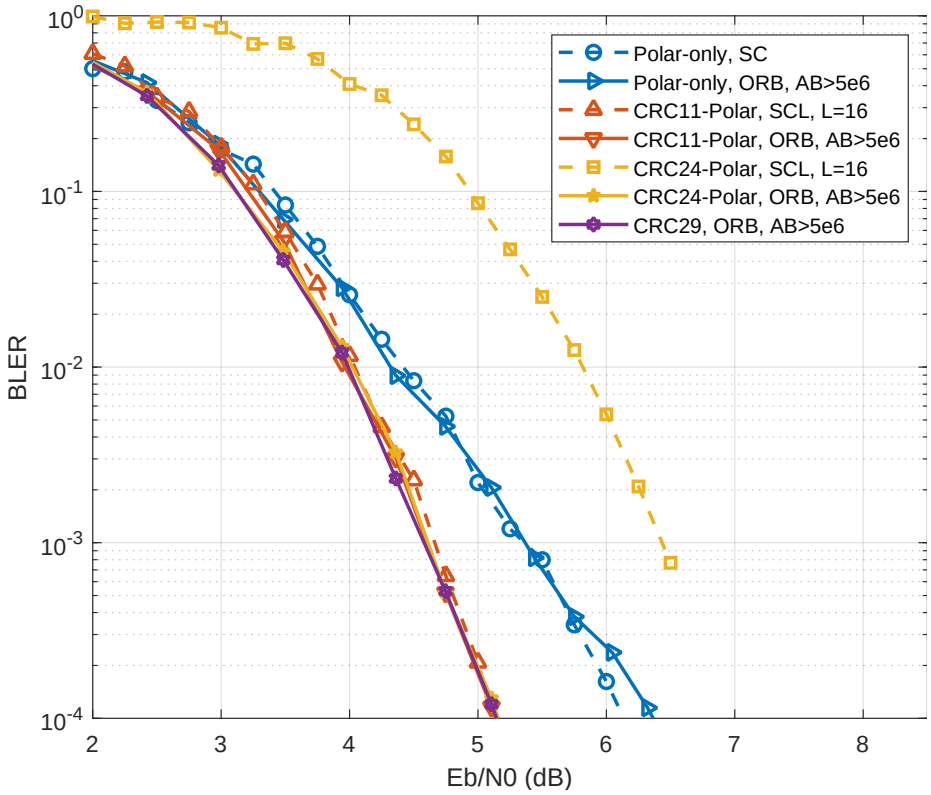


Figure 4-4: Soft detection performance of code-book setting of [128, 99] for Polar, CRC11 aided Polar, CRC24 aided Polar, and CRC29 codes; SC for Polar, CA-SCL with list size of 16 for CA-Polar and ORBGRAND with $AB > 5 \times 10^6$ for all codes; SC and CA-SCL are performed with [27].

Evaluating the performance of the SC and CA-SCL soft decoding algorithms in Fig. 4-4, a surprising phenomenon can be observed. While SC decoding of the Polar code and CA-SCL of the CRC11 aided Polar code provide comparable performance to ORBGRAND, the performance of CA-SCL with the CRC24 aided Polar code dramatically degrades to a level even worse than the hard detection performance of the same code in Fig. 4-3. This reveals a previously unnoticed issue with the CA-SCL algorithm. As a list decoding algorithm, CA-SCL produces a set of candidate code-words in its decoding procedure and use CRC checks to make candidate selection. More CRC bits provide better error detection capability and consequently higher confidence in candidate checks. However, the redundancy introduced by CRC bits

to the code-book is ignored for error correction, leaving the remaining Polar bits to produce the redundancy required for the error correcting list decoding. With a given code-book $[n, K]$ set of parameters, if the number of CRC bits is excessive relative to the limited number of Polar redundant bits, then the decoder may experience limited space of list candidates and, consequently, performance loss, in spite of the enhanced selection confidence from the extra CRC bits. As the key contributor to the error correcting quality of CA-Polar codes, the CA-SCL algorithm has been applied mostly to long codes, in which this shortcoming, which has been unnoticed so far to our knowledge, is far from negligible on these short codes.

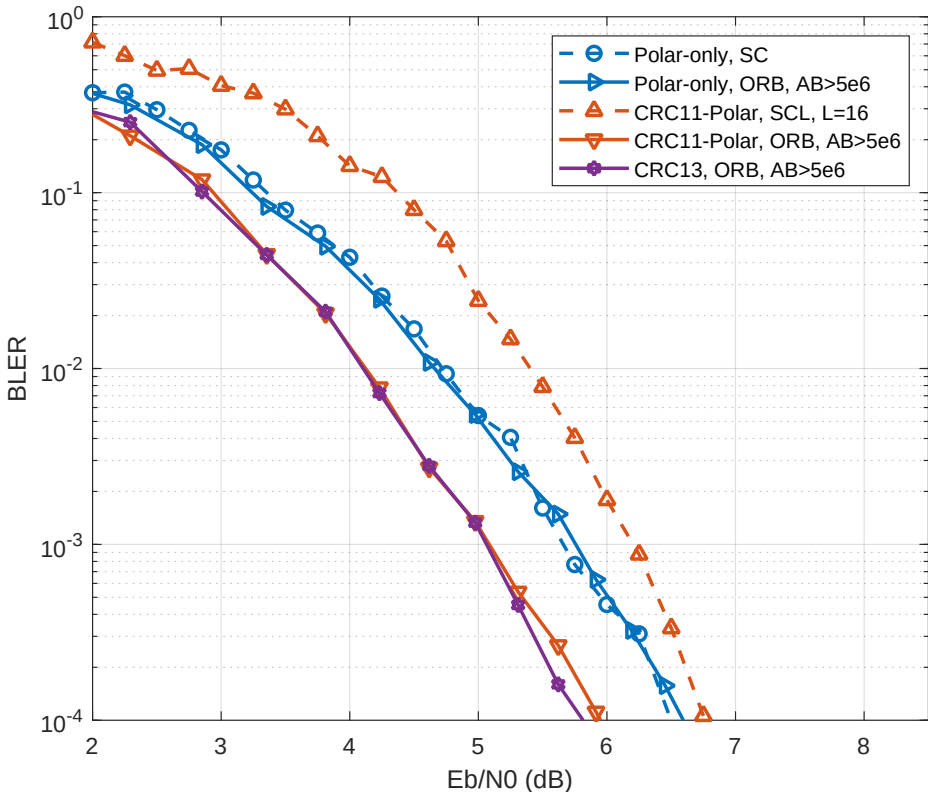


Figure 4-5: Soft detection performance of code-book setting of [64, 51] for Polar, CRC11 aided Polar, and CRC13 codes; SC for Polar, CA-SCL with list size of 16 for CA-Polar and ORBGRAND with $AB > 5 \times 10^6$ for all codes; SC and CA-SCL are performed with [27].

In Fig. 4-4, the CA-Polar[128, 99] with CRC11 does not experience this performance loss, owing to the 18 Polar bits, which are still sufficient to provide enough list candidate space. Unfortunately this success is not guaranteed with shorter code-word

lengths or with higher coding rates. Fig. 4-5 displays the same short-coming of the CA-SCL algorithm with the CRC11 but in CA-Polar[64,51], while ORBGRAND is stably successful.

The superior performance of CRC codes when compared to Polar codes, CRC aided or not, with their ultra-low encoding complexity, and the robust decoding performance of ORBGRAND as compared to CA-SCL algorithm, all suggest that a CRC code with ORBGRAND as its decoder provides a better short code solution than Polar or CA-Polar codes.

4.4.3 CRC Codes v.s. RLCs

To fully explore the decoding capability of RLC codes, we update the generator matrix for every code-word transmission to achieve random selection of code-words. For a fair comparison, we apply the same abandonment threshold for CRC and RLC codes. In Fig. 4-6, the RLC[128, 99] curve overlaps the CRC[128, 99] curve. As the code rate increases, however, the RLC performance degrades in comparison to the CRC code, owing to the shrinking of code-word space that makes it more probable for RLC to pick up code-words close to each other. The generator polynomial of CRC, however, is the product of a selection procedure [59, 86] that thus results in a code-book with stable minimum distance. This suggests that to extract near-CRC code performance for very high rate RLCs, extra structure might be needed.

In generation of RLC generator matrix, for very high-rate, it is possible to perform sanity checks on the randomly produced parity component to ensure the desired minimum distance between code-words. The generator matrix takes the form of $[\mathrm{I}_K|\mathrm{P}]$, where P is the parity component and I_K is a $K \times K$ unity matrix with K being the number of message bits. The parity check matrix is then $\mathrm{H} = [\mathrm{P}'|\mathrm{I}_{n-K}]$, with which the parity check of a code-word $c^n = [c_1, c_2, \ldots, c_n]^T$ is performed as $\mathrm{H}c^n$. If the minimum distance of the code-book is d, then the sum of any combinations of d columns of H cannot be a zero vector [63]. For very high-rate code setting, the achievable minimum distance d is small, and the total number of $\binom{n}{d}$ combinations can be manageable in checking if the produced generator matrix has the desired minimum

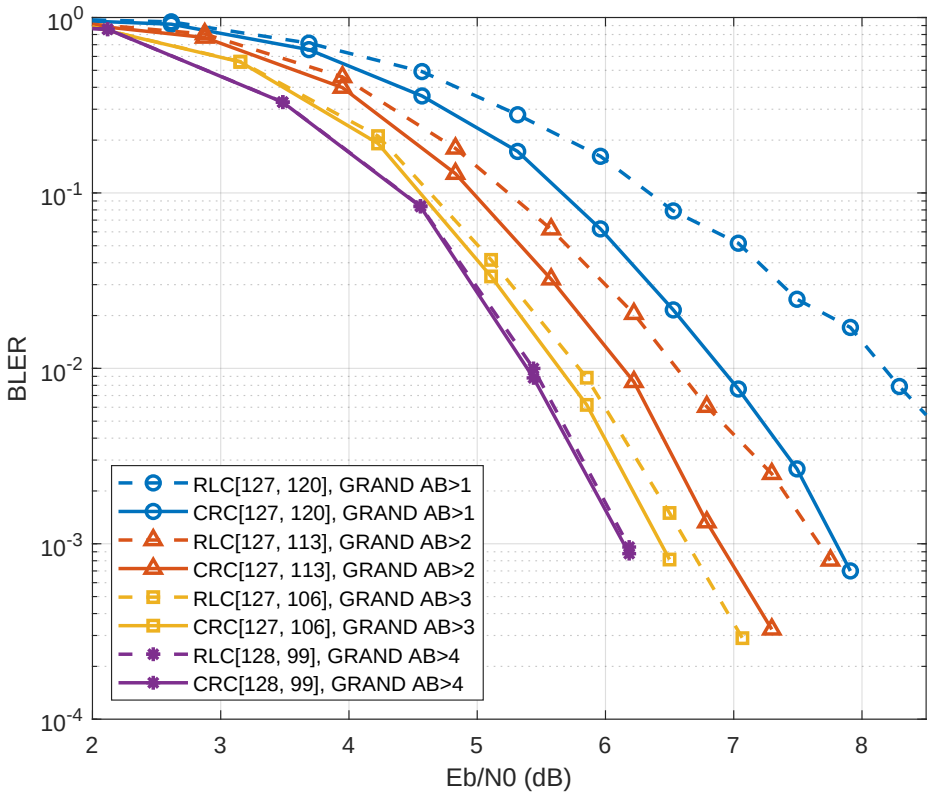


Figure 4-6: Hard detection performance of RLC and CRC codes with selected codeword lengths and rates.

distance.

For example, a code-book with minimum distance of at least 3 can be ensured if the following rules are obeyed in the generation of the parity component P:

- Each row of P has at least two 1's.
- Each row of P is different from other rows of P.

The hard detection performance of RLC[127,120] with its matrix generation following the above rules is presented in Fig. 4-7. Its performance is improved to be identical to CRC[127,120], which has the minimum distance of 3. There are some slight performance improvements in RLC[127,113] and RLC[127,106], suggesting that further checking on their generated matrix is needed to ensure minimum distances of resulted code-books.

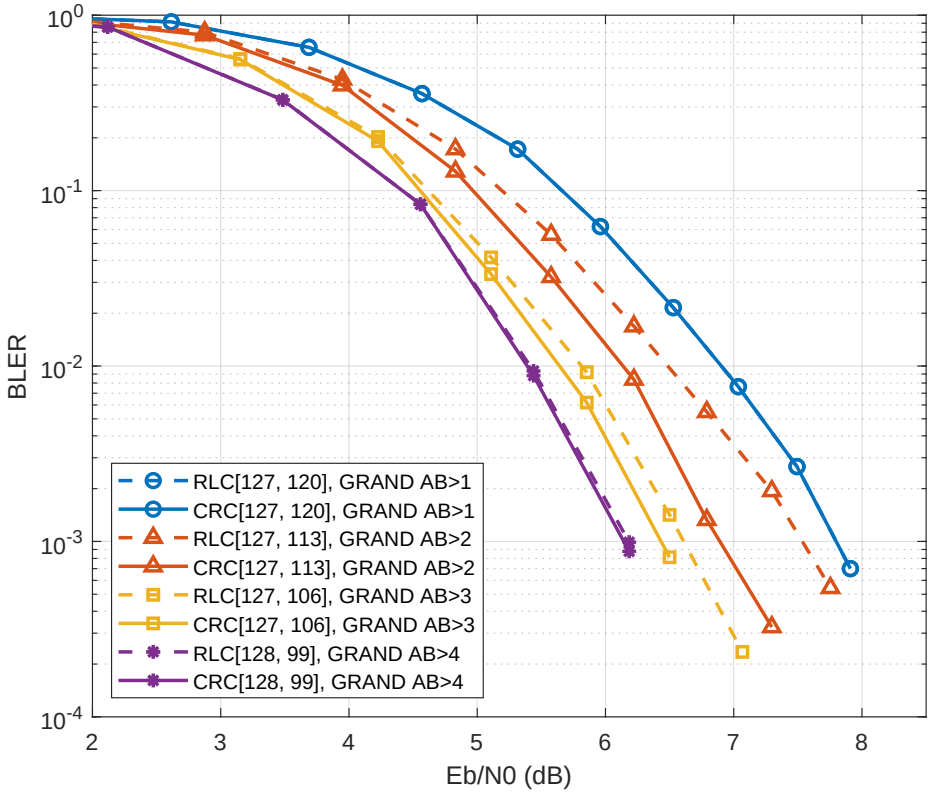


Figure 4-7: Hard detection performance of RLC and CRC codes with selected codeword lengths and rates; RLC matrices are generated with checking of their parity components performed

Fig. 4-8 presents the soft detection decoding performance of these codes. As expected, RLC[128,99] and RLC[127,120] have comparable performance as CRC[128,99] and CRC[127,120] respectively. Moreover, the performance difference of RLC[127,113] and RLC[127,106] from their CRC counterparts is reduced to a level such that it is worthwhile to adopt them in applications where their security feature is more welcome.

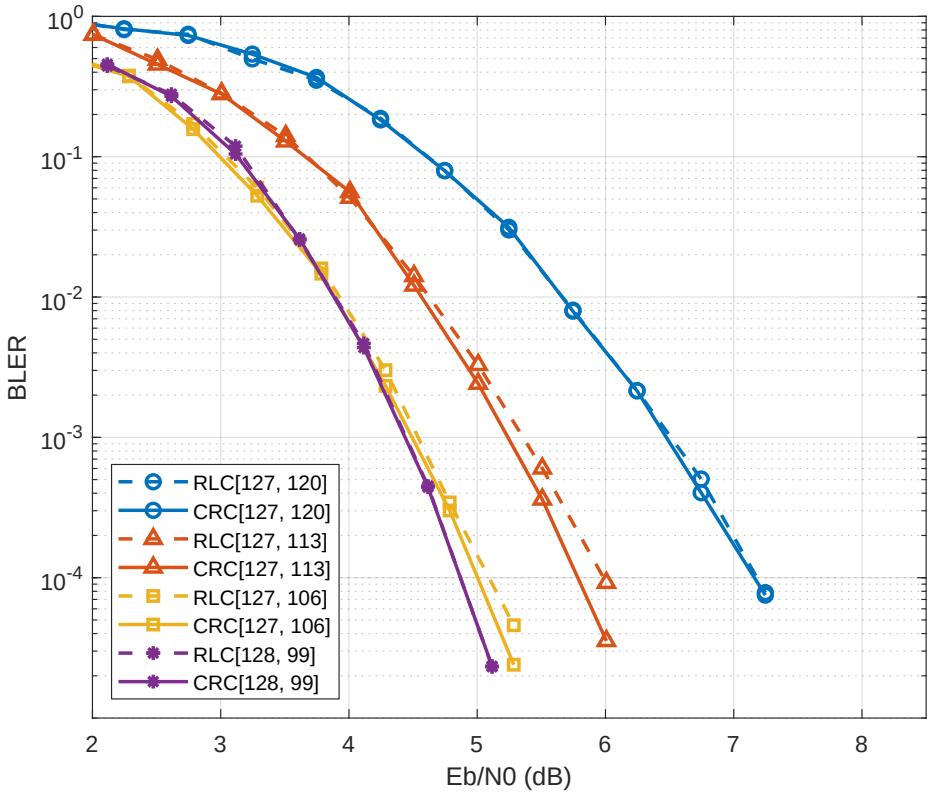


Figure 4-8: Soft detection performance of RLC and CRC codes with selected codeword lengths and rates; RLC matrices are generated with their parity components checking performed

4.5 Computational Complexity of Decoding CRC

Fig. 4-9 presents the computation complexity of GRAND decoding CRC[127,106] whose decoding performance is given in Fig. 4-1, with the abandonment condition set to $AB > 3$, determined by the error correcting capability of the code.

The unique feature of GRAND is demonstrated again that the overall complexity

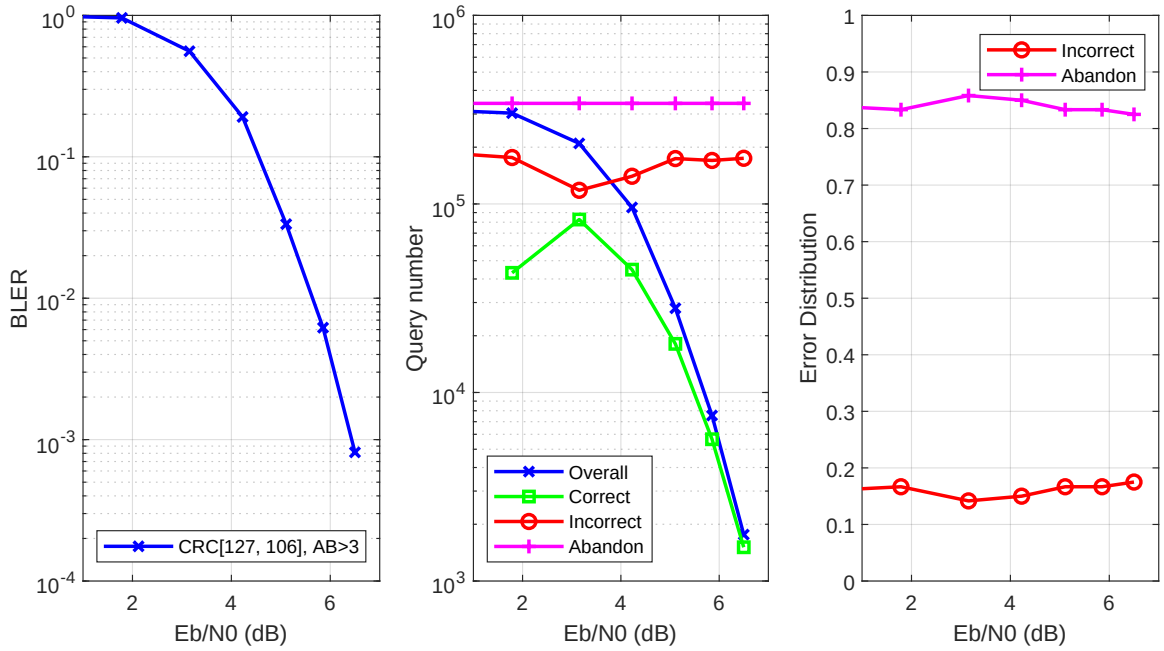


Figure 4-9: Computation complexity of GRAND evaluated with CRC[127,106] code in BSC channel with abandonment AB>3; "Overall" for any code-word, "Correct" for correctly decoded code-words, "Incorrect" for incorrectly decoded code-words, and "Abandon" for the abandonment condition; For reference, the decoding performance plot is presented on the left and the error distribution plot is shown on the right.

reduces as the noise condition improves, approaching the "correct" curve, and indicating that correct decoding dominates in low noise conditions. In the decoder operating region at BLER $\geq 10^{-3}$, the "overall" curve is below $2,000$, suggesting even lower average query number in higher SNR regions, and promising the low-power consumption feature of GRAND in operating region. The "incorrect" and "abandon" complexities are stably around 2×10^5 and 3×10^5, but their portions in "overall" complexity quickly drops as BLER improves with better noise condition, resulting in the merging of the "overall" and "correct" curves. The error distribution shows that abandonment dominates decoding errors in all noise conditions, indicating potential rooms for further decoding improvements.

In Fig. 4-10, the abandonment condition is increased to AB>4, making incorrect decoding the dominant error source. Correspondingly, the average complexity of both incorrect decoding and abandonment increase to higher levels, resulting in the overall complexity increasing to $4,000$ at BLER of 10^{-3}. However, there is only

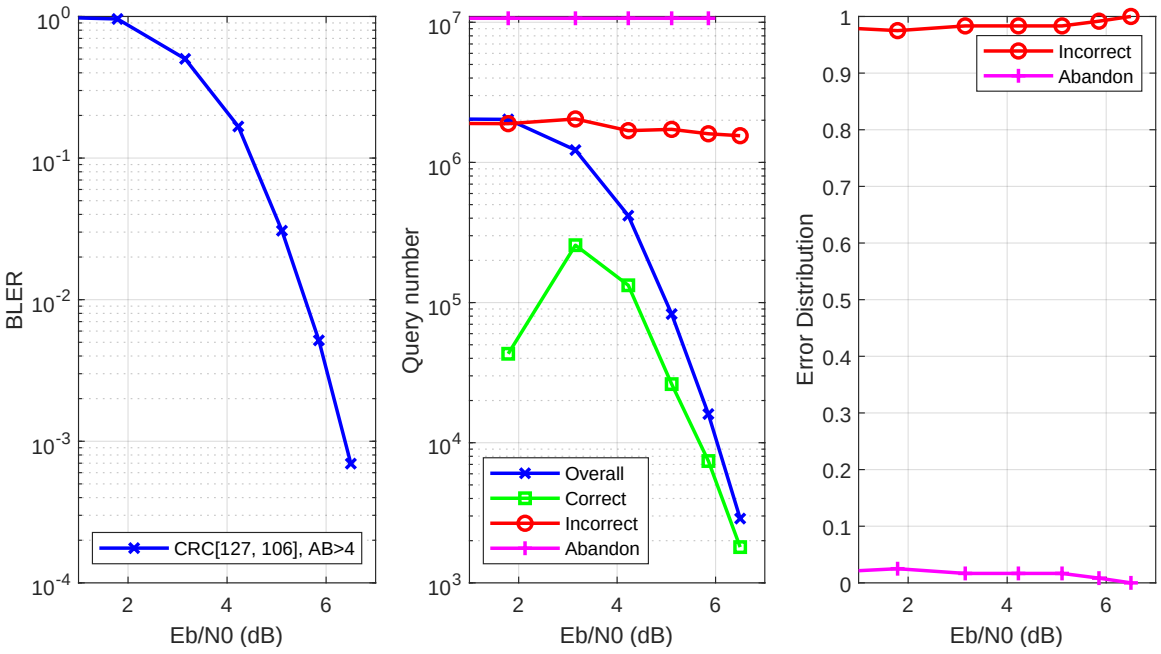


Figure 4-10: Computation complexity of GRAND evaluated with CRC[127,106] code in BSC channel with abandonment AB>4; "Overall" for any code-word, "Correct" for correctly decoded code-words, "Incorrect" for incorrectly decoded code-words, and "Abandon" for the abandonment condition; For reference, the decoding performance plot is presented on the left and the error distribution plot is shown on the right.

limited improvement of decoding performance, indicating the little productivity of the extra computation complexity. Therefore, in practice the abandonment condition is a necessary knob for the achievement of the desired trade-off between performance and complexity.

Similar observations are made in CRC[127,106] decoded with full ORBGRAND, as shown in Fig. 4-11, where various abandonment conditions are tested. As shown in Fig. 4-11, trivial performance loss is observed when AB drops down to $AB > 10^6$. Correspondingly, the overall averaged complexity has observable reduction in low SNR regions where BLER is higher than 10^{-3}. As SNR increases and BLER drops down to 10^{-4}, the complexity difference between abandonment conditions is no longer significant, and the average test number has dropped down to around 300 checks per code-word. Comparing to the hard detection result in Fig. 4-9, ORBGRAND achieves around 1.75dB gain at BLER of 10^{-3} and maintains the similar complexity around $2,000$. It is illustrated again that, in the same noise condition,

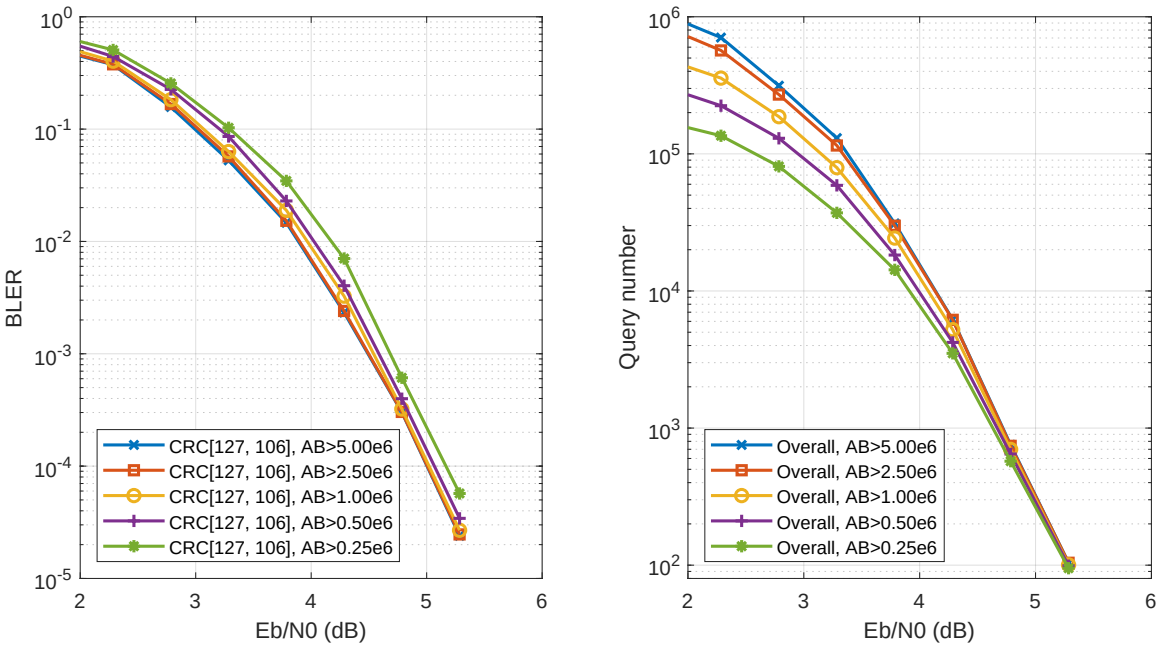


Figure 4-11: Computation complexity of ORBGRAND evaluated with CRC[127,106] code in AWGN channel with various abandonment conditions; The decoding performance is presented for reference on the left; The decoding is performed with 3-segment full ORBGRAND.

the enhancement of GRAND not only improves the decoding performance but also reduces the computational complexity.

4.6 Summary

Despite being the most widely used error detection solution in storage and communication, CRC codes have not been considered seriously as error correcting codes owing to the absence of an appropriate decoder. Instead, their main contribution has been in assisting state-of-the-art list decoders, with CA-Polar codes as the most successful recent example. With the invention of GRAND, the possibility of considering CRCs as error correction codes can be realized.

In this chapter we consider hard and soft detection variants, GRAND-SOS and ORBGRAND, to evaluate the performance of CRC codes in comparison with other candidate short, high-rate codes, including BCH codes, and CA-Polar codes. As short codes, CRCs not only provide at least equivalent decoding performance to

existing solutions, but are superior in various aspects. Compared to BCH codes, CRC codes have no restriction on code lengths and rates. Polar codes are surpassed by CRC codes in both hard and soft detection BLER performance. Aided by CRC, CA-Polar code can provide a matching performance, but the CRC alone still wins with its ultra-low encoding complexity. In addition, CA-SCL decoder, as the key to practical adoption of CA-Polar code, suffers significant performance loss when CRC bits dominate the code's redundancy. The reason is that CA-SCL uses CRC checks for candidate selection in list decoding, but does not avail of the CRC bits for error correction. ORBGRAND, on the other hand, is robust in all scenarios. Therefore, CRC with GRAND as its decoder is a good solution to low latency applications requiring short codes.

RLCs provide comparable performance to CRC codes with sufficient number of parity bits. Their performance degrade at very high rates but can be improved to match CRC by checking the parity component in matrix generation. Moreover, their performance inferiority to CRCs is mitigated in soft detection decoding with ORBGRAND, making them good candidates in applications needing security capability of codes.

The computational complexity analysis of CRC code demonstrates again the low-power potentiality of GRAND, and especially the feature that the algorithm enhancement of GRAND improves both its BLER performance and computational complexity in a given noise condition.

Conventionally impractical codes, such as CRC and RLC, become better channel code candidates to URLLC applications, owing to the universal and optimal (or near-optimal) decoder offered by GRAND. Even with decoders available, existing channel codes often have their performance limited by their code structure, required to facilitate decoder design. GRAND removes this constraint and thus not only enables the consideration of exiting codes like CRCs and RLCs, but also offers application opportunities to future codes designed without decoder constraints.

Chapter 5

Practical Issues for Implementations

So far, we have covered both hard detection and soft detection variants of GRAND appropriate with their algorithm complexity appropriate for hardware implementations. We have also used the developed algorithms to evaluate good short code candidates for URLLC applications. To facilitate implementations of those algorithms on VLSI chips, there are some practical issues to address, and we have them covered in this chapter.

5.1 Signal Quantization

In VLSI implementations, computations are usually performed in fixed-point for savings of bus bandwidth and power consumption. Thus, the soft information bits transferred from demodulator to soft detection decoder always take fix-point data formats. The number of bits considered in fix-point data types has significant influence on size, power consumption and bus width of the chip design, thus the minimum bit number is always desired. This is contradictory to the fact that decoding performance prefers more bits to preserve as much soft information as possible. Therefore, a trade-off must be achieved between hardware investment and decoding performance in the selection of fixed-point bit width, which is normally the first step to accomplish in chip design.

A fixed-point data type consists of an integer part with the sign bit that determines

the dynamic range of the signal, and a fractional part which bit number specifies the resolution that the data type can support. Thus, in selection of the bit number of a fixed-point data type, the location of the decimal point that divides bits between integer and fractional components need also be decided for minimum performance loss. Simulation is the common way to find appropriate fix-point data types, in which each candidate bit width is tested with various locations of the decimal point for optimal bit division. Then all candidate bit widths with their optimal decimal points are compared by their performance and the most appropriate data type is selected.

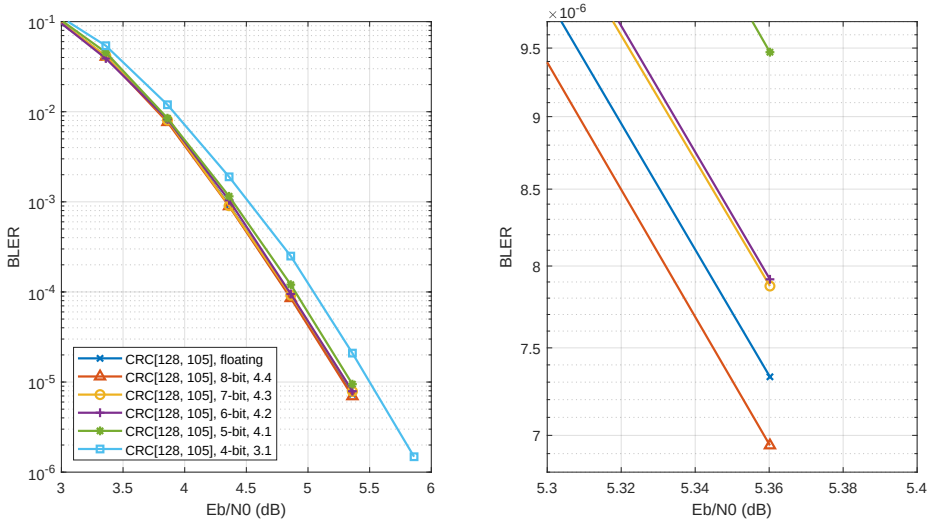


Figure 5-1: Fixed-point data type investigation over CRC[128,105]; Each bit width has had its optimal decimal point location already determined by simulations.

Fig. 5-1 compares the decoding performance of CRC[128,105] with different numbers of quantization bits. The optimal decimal point location has been tested and chosen for each fixed-point bit width. The comparison shows that 5-bit is the minimum bit width that results in negligible impact on the decoding performance. Notably, the zoomed plot shows that 8-bit signal quantization achieves a slightly better performance than the original floating-point data type. The reason is that quantization changes the noise distribution in high SNR region, making it deviate from the original Gaussian distributed noise that is known to have the worst impact on decoding performance.

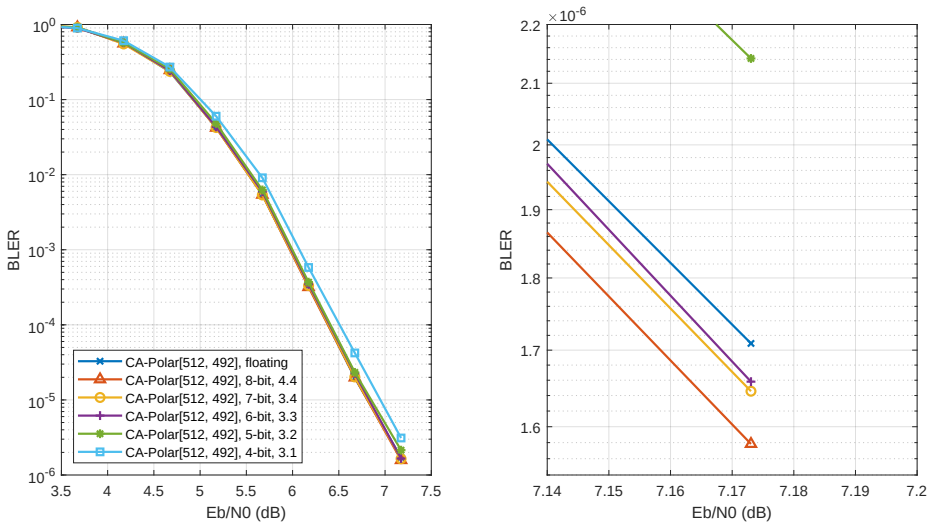


Figure 5-2: Fixed-point data type investigation over CA-Polar[512,492]; Each bit width has had its optimal decimal point location determined.

Evaluation over longer code-words leads to similar conclusion, as shown in Fig. 5-2 and Fig. 5-3 for CA-Polar[512,492] and CA-Polar[1024,1022] respectively. Even for the code-word length of 1024 bits, the 5-bit fixed-point data type is still sufficient to quantize soft bit information with negligible performance loss, albeit only 32 distinct signal levels are provided, which are much smaller than 1024 soft bits to be rank ordered. Referring to the reliability curve in Fig. 3-1, the low-reliability region experiences more reliability changes as compared to the center, and high-reliability regions that have less influence on the decoding performance. Therefore, more quantized numbers are assigned in the low-reliability region that dominates the decoding performance. As reliability increases in Fig. 3-1, the curve becomes flatter, leading to even less impact on the decoding performance. In addition, the redundancy in code-words and the robustness of ORBGRAND allow a certain amount of local misordering. All these factors combined together enable the quanzation bit number as low as 5 to support code-word as long as 1024 bits. This observation is an important support to the feasibility of ORBGRAND for VLSI implementations.

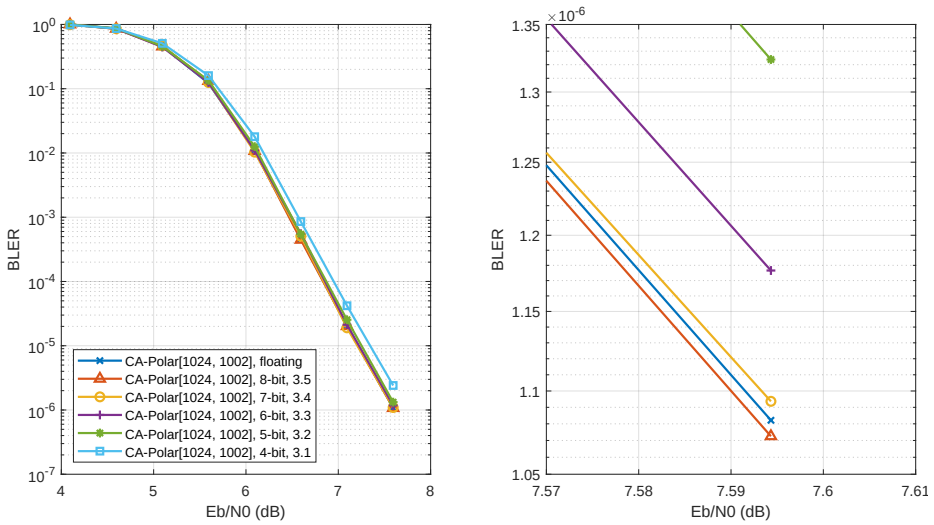


Figure 5-3: Fixed-point data type investigation over CA-Polar[1024,1002]; Each bit width has had its optimal decimal point location already determined with simulations.

5.2 Sorter and Pattern Generator

In the default procedure of ORBGRAND decoding, incoming soft bits first go through hard detection and the resulted hard bits are checked against the code-book. Simultaneously, to avoid delay and idling, the sorter starts to sort the soft bits by their increasing reliability, or their absolution values in case of Gaussian noise. The sorted soft bits are then handed over to the pattern generator to produce testing noise sequences for further decoding. In the meantime, if the hard bits check passes, the decoding is successful with all other ongoing operations aborted, and soft bits for the next code-word are imported for decoding. Otherwise, the decoding procedure continues until success or abandonment.

The time, or the cycle count in hardware design, spent on the sorting operation varies depending on the hardware investment and power consumption. A Bitonic sorter [19] can achieve cycle count as low as $O(\log^2(n))$, but consists of $O(n\log^2(n))$ comparators. For low power design, a sorter with fewer comparators is desired but longer delay is inevitable. For example, the sorter with a comparator tree needs at most n comparators, but it can sort only one element at a time and need n cycles to complete sorting. In this case, it is very inefficient for the pattern generator to be idling and wait for the sorter to completely finish its task. Therefore, it is desirable

that the pattern generator can start together with the sorter without any idling. However, the conflict is that the pattern generator need the result of the sorter, that is, the sorted soft bits, in order to generate patterns in the correct likelihood order.

To address the issue, an approach is to slightly violate the principle of generating noise patterns in the order of their likelihood. Instead, noise patterns with a single flipped bit are generated without following their exact likelihood order, or more conveniently, in their original timing order of arrival. In this strategy, the pattern generator can start immediately by generating single-bit patterns, while soft bits are being imported to the sorter. When single-bit patterns are all generated and the sorted bits are needed, the sorter should have finished its task or at least should be able to provide the needed number of sorted bits.

Incorrect decoding occurs when a generated noise pattern is not the affecting noise but passes the code-book check. When the pattern generation does not follow the likelihood order, the resulted decoding is no longer strictly ML. However, for a code-book with sufficient minimum distance, some local mis-ordering may have little impact on the decoding performance. If the minimum distance of the code-book is d_{min}, and the number of errors in received sequence is at most $d_{min} - 2$, single-error patterns do not cause any incorrect decoding, so their mis-ordering has no impact on the decoding result. Only for received sequences with $d_{min} - 1$ or more errors, can single-bit noise patterns cause incorrect decoding. As d_{min} increases, the probability of correctly guessing noise with $d_{min} - 1$ or more errors drops quickly even in the original correct pattern generation order, suggesting little performance impact of the mis-ordering from single-bit patterns.

Fig. 5-4 shows the performance of the modified pattern generator by evaluating CRC codes with various minimum distance. It is observed that the performance of CRC[127,120] is significantly degraded because its minimum distance is as low as 3. However, for CRC codes with minimum distance of at least 5, the impact on their performance is negligible with the modified pattern generator. Therefore, it is recommended that this chip-design-efficiency oriented pattern generator scheme should be applied to channel codes with a minimum distance $d_{min} \geq 5$.

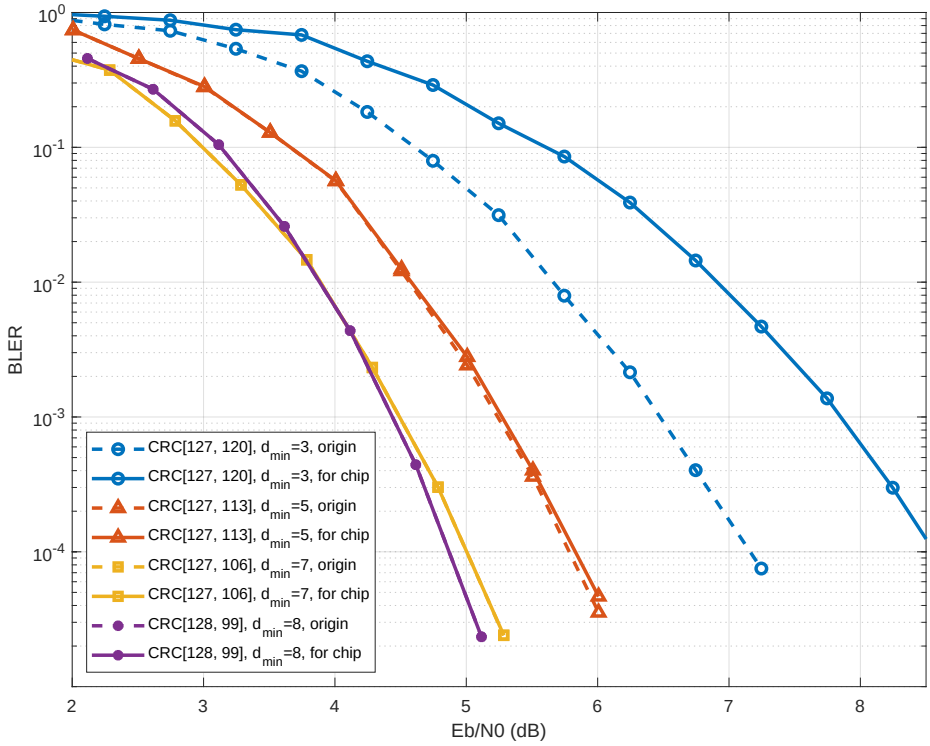


Figure 5-4: Performance evaluation of the modified pattern generation where single-bit patterns are generated first.

5.3 Code-word Re-ordering

A property of GRAND decoders is that the computation applied to each received sequence differs between transmitted code-words and affecting noise patterns. Sequences with no error pass the code-book check and are outputted immediately. Sequences with more errors generally need more computation. This property of GRAND coincides with iterative decoders for capacity-approaching long codes such as Turbo codes and LDPC codes, where the number of iterations differs between received signals.

The feature indicates GRAND's optimization on the assignment of computation resource, which enables the potentiality of ultra-low power designs for requesting applications. However, the feature raises up the issue of timing orders of output sequences. Sequences going through more computation may be available later than sequences needing fewer code-book checks, even when their original timing orders are reversed. For practical applications, this timing order issue need be addressed.

Using the hard detection GRAND chip design in [91] for example, there are three functional blocks that output code sequences. The syndrome block performs code-book check over incoming sequences and output those without errors. Failed sequences are passed over to the primary block where 1-bit and 2-bit error patterns are checked for them, with successfully decoded code sequences as outputs and failed sequences handed over to the secondary block. Patterns of 3-bit errors are checked in the secondary block and ultimately all input sequences reach the output port, either as the decoded or the abandoned. Therefore, all three functional blocks can output sequences that are marked with two possible status, decoded or abandoned, and these sequences are not necessarily in their original timing order.

A straightforward solution to restore timing order is to attach timing index tags to received sequences, which is a common practice in applications of long codes. For short codes, however, the drawback of the approach is that the bit width needed for tagging may be overwhelming. For example, if a group of 64 code sequences are to be reordered, then 6-bit tags are required to mark all sequences, the bit width of which is significant to 64-bit or even 128-bit short codes. Moreover, one additional bit is needed to mark if an output sequence has been decoded or abandoned. Here we propose a marking system with only 2 bits that can not only restore the timing order, but also indicate the abandonment status of output sequences, as described below.

- The syndrome block sends all input sequences to its output port with sequences tagged with "00" indicating that they passed code-book check, and sequences marked with "01" meaning that they need further processing; These "01"-marked sequences are also copied and transferred to the primary block for additional decoding.

- The primary block outputs all received sequences as well, with decoded sequences tagged with "01". The tag "10" on other output sequences indicates that they need further processing, and they are also copied and relayed to the secondary block for further processing.

- In the output of the secondary block, tag "10" indicates that a sequence is successfully decoded, and tag "11" means abandonment.

The marking system relies on the fact that sequences output from the same functional block are in their correct timing orders, which holds true because each functional block must finish the processing of the current sequence before working on the next one. Another fact is that the primary output is always delayed than the syndrome output, so is the secondary output to the primary output. This is also true because a sequence is known undecoded before it is sent to the next functional block. Thus, when a sequence in the syndrome output is marked as "01", it means that the sequence have been further processed and will appear later in the primary output later. Similarly, sequence tagged as "10" in primary output are to appear again later in the secondary output. These relationships are illustrated in Fig. 5-5, where sequences are omitted and only their tags are displayed for convenience. The algorithm, using the tagging system to restore the timing order of output sequences, is given in Algorithm 8.

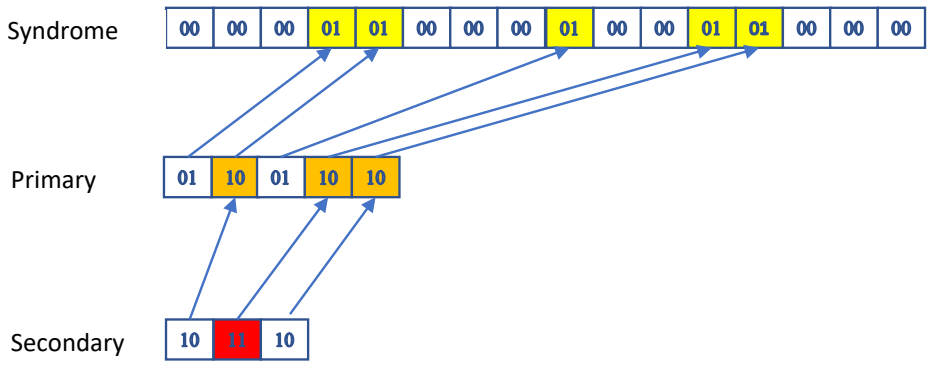


Figure 5-5: The tagging system to facilitate timing order restoring and decoding status marking.

Algorithm 8 The algorithm to restore the timing order of GRAND outputs

Input: Syndrome outputs, Primary outputs, and Secondary outputs

Output: re-ordered items

1: **for** each syndrome item **do**
2: **if** the syndrome item tagged as "01" **then**
3: wait for a primary item

```
    if the primary item tagged as "10" then
      wait for a secondary item
      replace the primary item with the secondary item
    end if
    replace the syndrome item with the primary item
  end if
  collect the item in the re-ordered list
end for
return re-ordered items
```

5.4 Efficient CRC Checking

Enabled by GRAND, CRC codes are found to be a good candidate to URLLC applications, as evaluated in Chapter 4. In addition, the encoding and syndrome check of CRC can be as simple as the polynomial division in Eq. (4.1), which is superior to the parity matrix checking method for other linear block codes in terms of both computational complexity and memory requirement. Polynomial division can be conveniently implemented with Linear Feedback Shift Register (LFSR), a simple digital circuit with low complexity. However, cycle count is the main disadvantage of the design approach, which, in normal implementations, equals to the message length K. With some combinational circuit design techniques, multiple bits may be generated in one cycle, the number of which, however, is limited by the circuit complexity. Moreover, the long critical path of combinational circuit makes it difficult to increase system clock frequency, voiding one of the main techniques for high-throughput design.

In GRAND decoder, code-book checking, beside pattern generation, is the essential component that affects system throughput and power consumption. The cycle count of the LFSR approach, even when the multiple-bit version is considered, imposes limitations to achievable throughput of the system. Moreover, the high cycle count, resulting in high dynamic energy consumption, also inhibits the achievable low-power designs. Here we propose a method for CRC syndrome computation that significantly reduces the cycle count, and consequently the dynamic power consump-

tion, making the technique a good candidate to code-book checking implementation for GRAND-based CRC decoders.

A CRC encoder transforms a K-bit message into a n-bit code-word, denoted as $c(x) = \sum_{i=0}^{n-1} c_i x^i$, and the received hard detected sequence can be represented in polynomial as $y(x) = \sum_{i=0}^{n-1} y_i x^i$. Let $n_g = n - K$ be the number of parity bits, and the generator polynomial is denoted as $g(x) = \sum_{i=0}^{n_g} g_i x^i$. Syndrome checking is the operation to find

$$\text{syn}[y(x)] = \text{remainder}\left(\frac{y(x)}{g(x)}\right).$$

Define an integer m_s such that $m_s \geq n_g$ and $m_p = n/m_s \in \mathbb{N}$. A set of segmented polynomials can be defined as,

$$s_k(x) = \sum_{j=0}^{m_s-1} y_{km_s+j} x^j, \text{ for } 0 \leq k \leq m_p - 1, \tag{5.1}$$

from which we further define a set of nested polynomials as,

$$\begin{cases} y_{(s,m_p-1)}(x) = s_{m_p-1}(x) \\ y_{(s,k)}(x) = y_{(s,k+1)}(x)x^{m_s} + s_k(x), \text{ for } 0 \leq k \leq m_p - 2 \end{cases}. \tag{5.2}$$

Obviously, the hard detected sequence polynomial $y(x) = y_{(s,0)}(x)$. Now we define another set of recursive polynomials as,

$$\begin{cases} y'_{(s,m_p-1)}(x) = s_{m_p-1}(x) \\ y'_{(s,k)}(x) = \text{syn}[y'_{(s,k+1)}(x)x^{m_s}] + s_k(x), \text{ for } 0 \leq k \leq m_p - 2 \end{cases}. \tag{5.3}$$

It can be proved (see Appendix B) that

$$\text{syn}[y_{(s,k)}(x)] = \text{syn}[y'_{(s,k)}(x)], \text{ for } 0 \leq k \leq m_p - 1. \tag{5.4}$$

Thus, the syndrome of $y(x)$ can be computed following the series of operations in Eq.

(5.3) with the last step as,

$$\text{syn}[y(x)] = \text{syn}[y_{(s,0)}(x)] = \text{syn}[y'_{(s,0)}(x)]. \tag{5.5}$$

With $m_s \geq n_g$ selected, the order of $y'_{(s,k)}(x)$ is always smaller than m_s, making it possible to compute $\text{syn}[y'_{(s,k+1)}(x)x^{m_s}]$ with a lookup table composed of the following items,

$$\{\text{syn}[x^i], \text{ for } m_s \leq i \leq 2m_s - 1\}.$$

That is,

$$\text{syn}[y'_{(s,k)}(x)x^{m_s}] = \sum_{i=0}^{m_s-1} y'_{(s,k,i)}\text{syn}[x^{i+m_s}], \tag{5.6}$$

where $y'_{(s,k,i)}$ is the i-th coefficient of polynomial $y'_{(s,k)}(x)$. Therefore, using lookup table, the syndrome computation in each step of Eq. (5.3) can be accomplished in one cycle by Eq. (5.6), with a small critical path that enables high clock frequencies for high-throughput designs. The last step of Eq. (5.5) can also be accomplished in one cycle using another lookup table as $\{\text{syn}[x^i], \text{ for } 0 \leq i \leq m_s - 1\}$, which can be much simplified for the first n_g items, i.e. $\text{syn}[x^i] = x^i$, for $0 \leq i \leq n_g - 1$. In fact, if m_s is selected as $m_s = n_g$, then the computation of the last step in Eq. 5.5 can be as simple as,

$$\text{syn}[y(x)] = \begin{cases} y'_{(s,0)}(x), \text{ if } y'_{(s,0,m_s-1)} = 0 \\ y'_{(s,0)}(x) + g(x), \text{ if } y'_{(s,0,m_s-1)} = 1 \end{cases}. \tag{5.7}$$

Therefore, the syndrome of $y(x)$ can be computed using Eq. (5.3) and Eq. (5.5) with two lookup tables or just one lookup table if m_s is specially selected to be $m_s = n_g$. The total number of cycles is n/m_s, which is much smaller than the LFSR approach, and, due to large savings in dynamic energy consumption, may lead to ultra-low power designs. Moreover, the short critical path of lookup tables removes

the bottleneck for the system clock frequency, which is essential for high-throughput design. The cost of the improvement of the new method is the hardware area invested in lookup tables, which, in modern VLSI designs, is an insignificant issue.

5.5 The "n Choose k" Approach

In the design of GRAND algorithms, pattern generation related problems can often be transformed to the well-known combinatorics "n Choose k" problem, which concerns the choice of k items from n items. The total number of combinations has been widely recognized as the binomial coefficient $\binom{n}{k}$. For GRAND, we need to obtain the set of k-element combinations in addition to its size. With the algorithm to generate "n Choose k" patterns, a number of GRAND related algorithms can be conveniently implemented in this approach, albeit the design might not be as efficient as specifically designed algorithms. An example of pattern generation in "n Choose k" approach can be found in Section 2.3.1 for Algorithm 2 where the two steps in generating $z^n \in \mathcal{Z}[m, l_m]$ both involve the "n Choose k" solution. The advantages of adopting this common pattern generation solution include efficient hardware block management, i.e. reusing the same block across designs, as well as the simplification of algorithm complexity. We first introduce Algorithm 9, the algorithm for generating "n Choose k" patterns, and then discuss more examples of its applications in GRAND designs.

Algorithm 9 The Algorithm for "n Choose k" Pattern Generation

Input: n, k

Output: $\{u^{k,i} : i = 1, 2, \ldots, \binom{n}{k}\}$

1: $i \leftarrow 0$
2: **for** $u_1 = 1$ To $n - k + 1$ **do**
3: **for** $u_2 = u_1 + 1$ To $n - k + 2$ **do**
4: $\ldots\ldots$(nested loops over u_i, $i = 3, 4, \ldots, k - 1$)
5: **for** $u_k = u_{k-1} + 1$ To n **do**

6: $i \leftarrow i + 1$

7: $u^{k,i} \leftarrow \{u_1, u_2, \ldots, u_k\}$

8: **end for**

9: ..... (nested loops over u_i, $i = 3, 4, \ldots, k-1$)

10: **end for**

11: **end for**

12: **return** $\{u^{k,i} : i = 1, 2, \ldots, \binom{n}{k}\}$

In detail, the n items are indexed from 1 to n and each selected pattern can be represented by the indices of selected items $u^k = \{u_1, u_2, \ldots, u_k\}$, in which, indices are listed in increasing order for convenience. As shown in Algorithm 9, the pattern generation procedure is implemented simply using k counters u^k with their initial and ending number configured accordingly. Each instantiation of the counter set contributes to a generated pattern for collection.

The "n Choose k" pattern generator can be immediately applied to the original hard detection GRAND decoder, in which, patterns of flipped bits are generated in the order of their Hamming weights. For n-bit noise patterns with Hamming weight of w_H, the pattern generator selects w_H bits out of n positions to flip. The total number of patterns with this configuration is then $\binom{n}{w_H}$. This pattern generator, however, might not be as efficient as GRAND-SOS, as briefly described in Section 4.2, which can not only handle some bursty errors, but also is feasible for parallel implementation [91].

For GRAND-MO, it has been shown in Section 2.3.1 that the set $\mathcal{Z}[m, l[m]]$ can be obtained via the "n Choose k" pattern generator. For ORBGRAND, the original integer splitting problem, as solved with Algorithm 5, is also a "n Choose k" problem. As explained in Appendix A, using Algorithm 9 for pattern generation is not as efficient as Algorithm 5, but the "n Choose k" approach makes it convenient to evaluate the total number of patterns. With the complexity control technique proposed in Section 3.5.3, the "n Choose k" method is no long applicable, and the new Algorithm 7 achieves the highest efficiency.

So far, it has been seen that all GRAND variants investigated previously have

had their pattern generation related to the "n Choose k" algorithm. Therefore, the algorithm has general applications in GRAND based decoder designs and is expected to facilitate analysis and implementation of future variants of GRAND.

Chapter 6

Conclusions and Outlooks

The dissertation explores an innovative approach to practically select and apply short codes to many emerging applications that require highly reliable and low-latency communications, with URLLC in 5G as a typical class of such applications.

Channel coding is an essential component of a communication system for the achievement of reliability, with short codes specifically required to meet the low-latency requirement. However, conventional channel codes with their standard decoders cannot effectively meet both reliability and low-latency requirements. Firstly, either candidate code-books or their decoding algorithms cannot achieve desired performance in short code regime. Secondly, short codes are vulnerable to channel conditions and can easily loose performance when channels are not memoryless, as most conventional decoders assumed. A common method to restore the desired memoryless channel is via interleaving over multiple code-words, which, however, introduces back the unwelcome long latency due to the permutation among hundreds or thousands of code-words.

An approach to solve the problem is to find a universal optimal decoding algorithm, which can provide a platform for selection of good candidates from available channel codes, and serve as the decoding solution to potential candidate codes that are lack of optimal decoders. Also, the universal decoder is desired to be able to adapt itself to channel conditions eliminating the need of techniques such as interleavers that bring back long latency. GRAND, turns out to be a potential solution,

due to its unique noise-guessing approach of decoding. The theoretical properties of GRAND has been well studied by its inventors and there have been prevailing research carried out on designs and implementations of the new decoding approach. In this thesis, we follow the original GRAND theory, continue to design and investigate GRAND variants, and use these variants as a powerful platform to evaluate candidates of short codes. All these works are carried out with practicality in mind, and have major aspects of algorithms investigated from performance to implementation.

We first study the extension of the hard decoding capability of GRAND in channels with memory. For the well-known Markov modeled channels, we adapt the pattern generator of GRAND to channel statistics, obtaining a GRAND variant called GRAND-MO. Performance evaluation shows that, when being applied to code-books with non-structured code-word patterns, GRAND-MO can not only preserve their decoding performance, but also achieve extra decoding gain that increases with the channel memory, eliminating the need of interleavers and preserving the low latency of short codes. Having naturally random code-word formats, RLCs, made practical by GRAND, present distinguished performance in Markovian channels, and demonstrate to be a good candidate of URLLC solution. GRAND-MO is then extended to high-order modulation systems, for which we introduce NNE channels that are similar to BSC channels for binary systems. Instead of operating on de-mapped binary bits, the high-order GRAND-MO operates directly on received symbols, with the capability originating from GRAND's unique noise guessing decoding approach. The exploration of modulation information brings in decoding gains that are not available to conventional decoders working on de-mapped bits. When being applied to Markovian NNE channels, the high-order GRAND-MO presents excellent performance with modulation orders of 64-QAM and 256-QAM, similar to its counterpart in binary Markovian channels. For 16-QAM and lower modulation orders, due to the large portion of edge and corner symbol, unsatisfactory decoding performance is observed. The drawback is, however, completely dismissed with the introduction of the augmented constellation, which also bring extra decoding gain to higher modulation orders due to the removal of edge or corner symbol effects. Computational analysis

of both versions of GRAND-MO demonstrates the practicality of GRAND-MO, and shows its potentiality for high-throughput and low-power designs, as well as a main feature of GRAND that its complexity is linked to its performance, making the algorithm enhancement double returned with improvements in both performance and complexity.

The investigation of GRAND-MO shows that even hard detection decoding, with additional information from channel statistics or modulation schemes, can achieve significant decoding gains, and completely eliminate the need of interleaver techniques, saving both complexity and latency. The research results suggest a wide range of potential applications of hard detection GRAND in existing or new systems, with the expectation of significant performance improvements.

Next, we move our research focus to soft detection decoding, which is of more interest in practical applications due to the multiple-dB gain from the utilization of soft information. We first explain the rationale behind ORBGRAND, and point out that the basic ORBGRAND is based on a linear approximation to the ordered LLR reliability curve of received signals. We then propose the Landslide algorithm, an integer partition algorithm, to realize a highly efficient and practically implementable pattern generator. For better approximation to the reliability curve, we propose the piece-wise linear approximation and define the reliability weight to govern the order of noise sequence generation, leading to the full ORBGRAND algorithm. The architecture of this algorithm is composed of two integer partition algorithms. The first integer partition algorithm splits a given reliability weight into parts and assign them to each segment of the approximated reliability curve. In each segment, the Landslide algorithm is applied to perform the second integer partition to generate all possible segment patterns. Cartesian product of pattern sets from all segments form the desired set of noise sequences corresponding to the given reliability weight, completing the pattern generator algorithm for the full ORBGRAND. Simulations show that the full ORBGRAND achieves excellent performance in all SNR conditions, beating the state-of-art soft decoding algorithms such as CA-SCL algorithm for CA-Polar codes. A number of complexity control techniques are proposed and

their little impact on the decoding performance demonstrates the robustness of ORGRAND. Computational complexity analysis shows the practicality of ORBGRAND, and again, the major property of GRAND that the algorithm enhancement of ORGRAND, as expected, results in significant improvement in both complexity and decoding performance. Similar to GRAND-MO, the ORBGRAND algorithm is then extended to high-order modulation systems, ditching soft de-mappers that are required by conventional decoders to decompose received signals into soft bits. Like high-order GRAND-MO, high-order ORBGRAND operates directly on a received symbol sequence by guessing candidate symbol sequences in their decreasing likelihood. Simulations shows that significant decoding gain is achieved with high-order ORBGRAND as compared to the state-of-art CA-SCL algorithm when operating on CA-Polar codes modulated with 256-QAM. The same excellent performance is achieved with binary ORBGRAND when being applied to soft bits de-mapped from symbols, due to preservation of all information in soft de-mapper, the complexity of which, however, makes its elimination an advantage of high-order ORBGRAND.

With both hard and soft detection variants of GRAND available, we are ready to evaluate and compare performance between short code candidates. CRC codes, with their decoding capability enabled by GRAND, present good performance in both hard and soft detection evaluations. CRC beats BCH in the sense that BCH can be viewed as a special case of CRC, but has limited code rate and length due to its structure required to facilitate the decoder design. As an aiding component code in CA-Polar code, CRC turns out to dominate the decoding performance and can be used alone without any performance loss. Moreover, ORBGRAND, as a universal soft decoder, is superior to the CA-SCL algorithm, a dedicated decoding solution to CA-Polar codes, which cannot make use of the error correction capability of its component CRC code, and loses performance when the portion of CRC bits is significant. ORBGRAND, on the other hand, is robust and always presents the best performance of any selected channel codes. RLCs, also enabled by GRAND, shows comparable performance to CRCs with sufficient number of parity bits. For very high-rate RLCs, the limited code space causes performance degradation of randomly

generated RLCs, but enables the possibility of checking parity matrix for desired minimum distance. RLCs' soft decoding performance with ORBGRAND makes them good solution to applications needing security features. Computational complexity analysis verifies again the discovered features of GRAND when being applied to CRC decoding, demonstrating that GRAND, as a universal and near-optimal decoder, is a practical solution to short code selections and applications.

The advantages and practicability of GRAND lend itself to feasible VLSI implementations. Some design issues are addressed including signal quantization, timing of sorter and pattern generator, output code-words re-ordering, efficient CRC codebook checking and the "n Choose k" algorithm. The discussion of these issues further facilitates practical implementations of GRAND decoders in high-throughput and low-power VLSI designs. Some discussed techniques have been adopted in on-going VLSI chip projects.

The thesis extends the original GRAND algorithm and investigates major aspects of GRAND from performance to implementation, illustrating the new decoding approach's potentiality in practical applications requiring high reliability and low latency. There are many more extension possibilities of GRAND when various channel types and modulation schemes are considered. For example, extend the hard detection GRAND-MO's burst error correction capability to soft detection regime, or expand the soft detection ORBGRAND to correct burst errors, and evaluate the performance of these further enhanced decoding algorithms in more practical channels, such as channels with Rayleigh fadings or frequency-selective fadings. There are also many specially designed channel codes, such as burst error correcting codes and TCM/BCM codes, that, when decoded with GRAND, may present decoding gains beyond the capability of their default decoders. While high-throughput designs of GRAND variants have been investigated in research papers, GRAND's potentiality in ultra-low power VLSI implementations, as suggested by the computational complexity analysis of various GRAND scenarios in the thesis, is of particular interest in research, especially with the efficient design of CRC syndrome check considered.

In system level of communications, GRAND provides the potentiality of reduc-

ing the energy consumption of each information bit being communicated. For example, the ARQ protocol in MAC layer ensures the communication reliability by re-transmitting packets if errors are detected in them. The detection is often performed with error detection codes such as CRC. With the decoding capability of these codes enabled by GRAND, a large amount of re-transmission can be avoided owing to corrections of those detected errors. The saving can be further enhanced if some reliability information is available so that SRGRAND or even ORBGRAND are deployed. It is an interesting topic to study the system energy savings of those existing systems with the introduction of GRAND.

As a new decoding approach, GRAND opens a wide range of possibilities in designs, implementations, and enhancements of channel codes and communications systems.

Appendix A

Integer Partition with Orders

Dividing integer W into m non-empty parts with their order mattering can be viewed as slicing a string of W balls into m segments, which is equivalent to selecting $m-1$ locations for slicing out of $W-1$ positions between balls. This is a "n Choose k" problem with $n = W-1$ and $k = m-1$. Algorithm 9 can provide all patterns and their total number is $\binom{W-1}{m-1}$.

When parts are allowed to be empty, the above solution is not directly applicable. We can consider slicing a string of $W+m$ balls into m non-empty segments. After slicing, a ball is taken out of each segment of the sliced pattern, and the resulted pattern is a desired combination. Therefore, this is still a "n Choose k" problem with $n = W+m$ and $k = m-1$, and the total number of patterns is $\binom{W+m}{m-1}$.

Although Algorithm 9 can be used to generate all patterns for this problem, the extra step of taking 1 off each part makes it less efficient than the direct solution as provided in Algorithm 5.

Appendix B

Syndrome Operations

Consider a constructed polynomial $p_1(x) = p_2(x)p_3(x)+p_4(x)$. It can be easily proved that

$$\begin{aligned} \text{syn}[p_1(x)] &= \text{syn}[p_2(x)p_3(x) + p_4(x)] \\ &= \text{syn}\left[\text{syn}[p_2(x)]\text{syn}[p_3(x)] + \text{syn}[p_4(x)]\right]. \end{aligned} \tag{B.1}$$

Similarly, another constructed polynomial $p_5(x) = \text{syn}[p_2(x)p_3(x)] + p_4(x)$ has its syndrome as,

$$\begin{aligned} \text{syn}[p_5(x)] &= \text{syn}[\text{syn}[p_2(x)p_3(x)] + p_4(x)] \\ &= \text{syn}\left[\text{syn}[p_2(x)]\text{syn}[p_3(x)] + \text{syn}[p_4(x)]\right]. \end{aligned} \tag{B.2}$$

Now we return to Eq. (5.2) and Eq. (5.3). At $k = m_p - 1$, we have

$$\text{syn}[y_{(s,m_p-1)}(x)] = \text{syn}[y'_{(s,m_p-1)}(x)] = \text{syn}[s_{m_p-1}(x)]. \tag{B.3}$$

At $k = m_p - 2$, Eq. (5.2) and Eq. (B.1) lead to,

$$\text{syn}[y_{(s,m_p-2)}(x)] = \text{syn}\left[\text{syn}[y_{(s,m_p-1)}(x)]\text{syn}[x^{m_s}] + \text{syn}[s_{m_p-2}(x)]\right]. \tag{B.4}$$

And Eq. (5.3) and Eq. (B.2) give,

$$\mathrm{syn}[y'_{(s,m_p-2)}(x)] = \mathrm{syn}\left[\mathrm{syn}[y'_{(s,m_p-1)}(x)]\mathrm{syn}[x^{m_s}] + \mathrm{syn}[s_{m_p-2}(x)]\right]. \tag{B.5}$$

From Eq. (B.3), we have Eq. (B.4) and Eq. (B.5) as equivalent, i.e.

$$\mathrm{syn}[y_{(s,m_p-2)}(x)] = \mathrm{syn}[y'_{(s,m_p-2)}(x)]. \tag{B.6}$$

The deduction continues until $k = 0$, and Eq. (5.4) is proved.

Bibliography

[1] Standards for time-sensitive networking for use in industrial automation networks: time-sensitive networking (TSN) for industry 4.0. Technical report, IEEE Standards Association, 2017.

[2] 3GPP TR 38.824, Study on physical layer enhancements for NR ultra-reliable and low latency case (URLLC). Technical report, 3rd Generation Partnership Project, 2018.

[3] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Multiplexing and Channel Coding, Release 15, V15.6.0. Technical report, 3GPP, 38.212, June 2019.

[4] Syed Mohsin Abbas, Marwan Jalaleddine, and Warren J. Gross. High-throughput VLSI architecture for GRAND Markov order. In *IEEE SiPS*, 2021.

[5] Syed Mohsin Abbas, Thibaud Tonnellier, Furkan Ercan, and Warren J Gross. High-throughput VLSI architecture for GRAND. In *IEEE SiPS*, 2020.

[6] Syed Mohsin Abbas, Thibaud Tonnellier, Furkan Ercan, Marwan Jalaleddine, and Warren J. Gross. High-throughput VLSI architecture for soft-decision decoding with ORBGRAND. In *IEEE ICASSP*, 2021.

[7] E. Akay and E. Ayanoglu. Low complexity decoding of bit-interleaved coded modulation for m-ary qam. In *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, volume 2, pages 901–905 Vol.2, 2004.

[8] W. An, K. R. Duffy, and M. Médard. Ordered reliability bits guessing random additive noise decoding. In *IEEE Trans. Commun.*, submitted.

[9] W. An, M. Médard, and K. R. Duffy. Keep the bursts and ditch the interleavers. In *IEEE GLOBECOM*, 2020.

[10] W. An, M. Médard, and K. R. Duffy. CRC codes as error correcting codes. In *IEEE ICC*, 2021.

[11] W. An, M. Médard, and K. R. Duffy. Keep the bursts and ditch the interleavers. *IEEE Trans. Commun.*, submitted.

[12] Kenneth Andrews, Chris Heegard, and Dexter Kozen. A theory of interleavers. Technical report, Cornell University, New York, 1997.

[13] E. Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inf. Theory*, 55(7):3051–3073, 2009.

[14] Erdal Arıkan, Haesik Kim, Garik Markarian, U Ozgur, and Efecan Poyraz. Performance of short polar codes under ML decoding. *Proc. ICT MobileSummit*, pages 10–12, 2009.

[15] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Trans. Inf. Theory*, 20(2):284–287, 1974.

[16] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg. LLR-based successive cancellation list decoding of Polar codes. *IEEE Trans. Signal Process.*, 63(19):5165–5179, 2015.

[17] Marco Baldi, Nicola Maturo, Enrico Paolini, and Franco Chiaraluce. On the use of ordered statistics decoders for low-density parity-check codes in space telecommand links. *EURASIP J Wirel. Comm.*, (1):272, 2016.

[18] J. Bas et al. Energy and delay analysis of binary BCH codes for machine-to-machine networks with small data transmissions. In *IEEE PIMRC*, pages 1873–1877, 2013.

[19] K. E. Batcher. Sorting networks and their applications. In *Proc. AFIPS Spring Joint Comput. Conf.*, volume 32, pages 307–314, 1968.

[20] E. Berlekamp, R. McEliece, and H. Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Tran. Inf. Theory*, 24(3):384–386, 1978.

[21] Elwyn Berlekamp. *Algebraic coding theory*. World Scientific, 1968.

[22] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *IEEE Int. Conf. on Comm.*, volume 2, pages 1064–1070, May 1993.

[23] R. C. Bose and Dwijendra K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Inf. Control*, 3(1):68–79, 1960.

[24] V. Boussard, S. Coulombe, F. Coudoux, and P. Corlay. Table-free multiple bit-error correction using the CRC syndrome. *IEEE Access*, 8:102357–102372, 2020.

[25] N. Brandonisio, D. Carey, S. Porto, G. Talli, and P. D. Townsend. Burst-mode FEC performance for PON upstream channels with EDFA optical transients. In *ONDM*, pages 190–193, 2018.

[26] Peter J. Cameron. *Combinatorics: Topics, Techniques, Algorithms*. Cambridge University Press, 1 edition, 2001.

[27] A. Cassagne et al. Aff3ct: A fast forward error correction toolbox! *Elsevier SoftwareX*, 10:100345, October 2019.

[28] Chia-Wei Chang, Po-Ning Chen, and Yunghsiang S. Han. A systematic bit-wise decomposition of M-ary symbol metric. *IEEE Transactions on Wireless Communications*, 5(10):2742–2751, 2006.

[29] He Chen, Rana Abbas, Peng Cheng, Mahyar Shirvanimoghaddam, Wibowo Hardjawana, Wei Bao, Yonghui Li, and Branka Vucetic. Ultra-reliable low latency cellular networks: Use cases, challenges and approaches. *IEEE Commun. Mag.*, 56(12), 2018.

[30] J. Cheng and H. Koorapaty. Error detection reliability of LTE CRC coding. In *IEEE 68th Vehicular Technology Conference*, pages 1–5, 2008.

[31] A. Das and N. A. Touba. A new class of single burst error correcting codes with parallel decoding. *IEEE Transactions on Computers*, 69(2):253–259, 2020.

[32] Herbert Aron David and Haikady Navada Nagaraja. Order statistics. *Encyclopedia of statistical sciences*, 2004.

[33] E. I. de Betou, E. Mobilon, B. Angeli, P. Öhlen, A. Lindström, S. Dahlfort, and E. Trojer. Upstream FEC performance in combination with burst mode receivers for next generation 10 Gbit/s PON. In *ECOC*, pages 1–3, 2010.

[34] B Dorsch. A decoding algorithm for binary block codes and J-ary output channels (corresp.). *IEEE Trans. Inf. Theory*, 20(3):391–394, 1974.

[35] K. R Duffy, J. Li, and M. Médard. Guessing noise, not code-words. In *IEEE Int. Symp. on Inf. Theory*, 2018.

[36] K. R. Duffy, J. Li, and M. Médard. Capacity-achieving guessing random additive noise decoding. *IEEE Trans. Inf. Theory*, 65(7):4023–4040, 2019.

[37] K. R. Duffy, J. Li, and M. Médard. Capacity-achieving guessing random additive noise decoding (GRAND). *IEEE Trans. Inf. Theory*, 65(7):4023–4040, 2019.

[38] K. R Duffy and M. Médard. Guessing random additive noise decoding with soft detection symbol reliability information. In *IEEE Int. Symp. on Inf. Theory*, 2019.

[39] K. R. Duffy and M. Médard. Guessing random additive noise decoding with soft detection symbol reliability information - SGRAND,. *IEEE Int. Symp. on Inf. Theory*, 2019.

[40] K. R Duffy, M. Médard, and W. An. Guessing random additive noise decoding with symbol reliability information (SRGRAND). In *IEEE Trans. Commun.*, volume 70, pages 3–18, 2022.

[41] Ken R. Duffy. Ordered reliability bits guessing random additive noise decoding. In *IEEE ICASSP*, pages 8268–8272, 2021.

[42] Ken R. Duffy. Ordered reliability bits guessing random additive noise decoding. In *IEEE ICASSP*, pages 8268–8272, 2021.

[43] Giuseppe Durisi, Tobias Koch, and Petar Popovski. Toward massive, ultrareliable, and low-latency wireless communication with short packets. *Proc. IEEE*, 104(9):1711–1726, 2016.

[44] J. Erfanian, S. Pasupathy, and G. Gulak. Reduced complexity symbol detectors with parallel structure for ISI channels. *IEEE Transactions on Communications*, 42(234):1661–1671, 1994.

[45] T. Etzion and E. Yaakobi. Error-correction of multidimensional bursts. *IEEE Trans. Inf. Theory*, 55(3):961–976, 2009.

[46] Philip Fire. *A class of multiple-error-correcting binary codes for non-independent errors*. PhD thesis, Stanford University, 1959.

[47] M. P. C. Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Trans. Inf. Theory*, 41(5):1379–1396, 1995.

[48] Marc PC Fossorier, Miodrag Mihaljevic, and Hideki Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Trans. Commun.*, 47(5):673–680, 1999.

[49] R. G. Gallager. *Information Theory and Reliable Communication*. John Wiley & Sons, Inc., New York, NY, USA, 1968.

[50] Robert G Gallager. Low density parity check codes, 1963.

[51] David Gazelle and Jakov Snyders. Reliability-based code-search algorithms for maximum-likelihood decoding of block codes. *IEEE Tran. Inf. Theory*, 43(1):239–249, 1997.

[52] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Syst. Tech. J.*, 39(5):1253–1265, 1960.

[53] Andrea Goldsmith. *Wireless communications*. CUP, 2005.

[54] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Tran. Inf. Theory*, 45(6):1757–1767, 1999.

[55] A. Hocquenghem. Codes corecteurs de'erreurs. In *Chiffres, 2*, pages 147–156, 1959.

[56] A. B. Cooper III. Soft decision decoding of block codes. Technical report, U.S. Army Laboratory Command, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, 1988.

[57] T. Kaneko, T. Nishijima, and S. Hirasawa. An improvement of soft-decision maximum-likelihood decoding algorithm using hard-decision bounded-distance decoding. *IEEE Tran. Inf. Theory*, 43(4):1314–1319, 1997.

[58] P. Koopman. Best CRC polynomials. `https://users.ece.cmu.edu/~koopman/crc/`.

[59] P. Koopman and T. Chakravarty. Cyclic redundancy code (CRC) polynomial selection for embedded networks. In *International Conference on Dependable Systems and Networks*, pages 145–154, 2004.

[60] J. K. Kreng, M. M. Ardeshiri, O. C. Barbosa, and Y. Y. Krikorian. Telemetry, tracking, and commanding (TT&C) link considerations for a LEO Sat. In *IEEE Aerospace Conference*, pages 1646–1655, 2005.

[61] M. Leonardon, A. Cassagne, C. Leroux, C. Jego, L.-P. Hamelin, and Y. Savaria. Fast and flexible software polar list decoders. *J. Signal Process. Syst.*, pages 1–16, 2019.

[62] Weisong Liang and Haiyang Liu. Low-complexity error correction algorithm for cyclic redundancy check codes. In *IEEE ICCC*, pages 22–26, 2021.

[63] S. Lin and D. J. Costello. *Error control coding: fundamentals and applications*. Pearson/Prentice Hall, 2004.

[64] R. Lucas, M. Bossert, and M. Breitbach. On iterative soft-decision decoding of linear binary block codes and product codes. *IEEE J. Sel. Areas Commun.*, 16(2):276–296, 1998.

[65] Z. Ma et al. High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals, and enabling technologies. *IEEE Internet Things J.*, 6(5), 2019.

[66] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457–458, 1997.

[67] Juquan Mao, Mahmoud Alfa Abdullahi, Pei Xiao, and Aijun Cao. A low complexity 256QAM soft demapper for 5G mobile system. In *2016 European Conference on Networks and Communications (EuCNC)*, pages 16–21, 2016.

[68] S. Marsili. DC offset estimation in OFDM based WLAN application. In *IEEE GLOBECOM*, volume 6, pages 3531–3535, 2004.

[69] P. Martinelli, E. Cianca, and L. Simone. Comparison of channel codes in presence of pulsed jammers in TT&C links. In *Proc. ASMS/SPSC*, pages 170–173, 2014.

[70] J. Massey. Implementation of burst-correcting convolutional codes. *IEEE Trans. Inf. Theory*, 11(3):416–422, 1965.

[71] James Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf Theory*, 15(1):122–127, 1969.

[72] Robert J McEliece. A public-key cryptosystem based on algebraic. *Deep Space Network Progress Report*, 42-44:114–116, 1978.

[73] Muriel Médard. Is 5 just what comes after 4? *Nature Electronics*, 3(1):2–4, 2020.

[74] Y. Miyamoto and S. Suzuki. Advanced optical modulation and multiplexing technologies for high-capacity OTN based on 100 GB/s channel and beyond. *IEEE Commun. Mag.*, 48(3):S65–S72, 2010.

[75] David E Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE professional group on electronic computers*, (3):6–12, 1954.

[76] K. R. Narayanan and G. L. Stuber. List decoding of turbo codes. *IEEE Trans. Commun.*, 46(6):754–762, 1998.

[77] S. Nihei, M. Umehira, and S. Takeda. A modified DFTs-OFDM with DC subcarrier shift for low PAPR and DC offset error robustness. In *IEEE VTC*, pages 1–6, 2015.

[78] K. Niu and K. Chen. CRC-aided decoding of Polar codes. *IEEE Commun. Lett.*, 16(10):1668–1671, October 2012.

[79] Vivian Papadopoulou, Marzieh Hashemipour-Nazari, and Alexios Balatsoukas-Stimming. Short codes with near-ML universal decoding: Are random codes good enough? In *IEEE SiPS*, pages 94–98, 2021.

[80] Imtiaz Parvez, Ali Rahmati, Ismail Guvenc, Arif I Sarwat, and Huaiyu Dai. A survey on low latency towards 5G: RAN, core network and caching solutions. *IEEE Commun. Surv.*, 20(4):3098–3130, 2018.

[81] W. W. Peterson and D. T. Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, 1961.

[82] P. Pfeifer and H. T. Vierhaus. Forward error correction in wireless communication systems for industrial applications. In *SPA*, pages 14–14, 2017.

[83] Henry D Pfister. A brief introduction to Polar codes. *Supplemental Material for Advanced Channel Coding*, 2014.

[84] E. Prange. *Cyclic Error-correcting Codes in Two Symbols*. AFCRC-TN. Air Force Cambridge Research Center, 1957.

[85] John G. Proakis. *Digital Communications*. McGraw-Hill, New York, NY, USA, 2001.

[86] J. Ray and P. Koopman. Efficient high Hamming distance CRCs for embedded networks. In *International Conference on Dependable Systems and Networks (DSN'06)*, pages 3–12, 2006.

[87] G. Reed and I. Solomon. Polynomial codes over certain finite fields. *J. Soc. Ind. Appl. Math.*, 8:300–304, 1960.

[88] Irving Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, 1954.

[89] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[90] S. Reiger. Codes for the correction of 'clustered' errors. *IRE Trans. Inf. Theory*, 6(1):16–21, 1960.

[91] A. Riaz, V. Bansal, A. Solomon, W. An, Q. Liu, K. Galligan, K. R. Duffy, M. Médard, and R. T. Yazicigil. Multi-code multi-rate universal maximum likelihood decoder using GRAND. In *IEEE ESSCIRC*, 2021.

[92] P. Robertson, E. Villebrun, and P. Hoeher. A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain. In *Proceedings IEEE International Conference on Communications ICC '95*, volume 2, pages 1009–1013 vol.2, 1995.

[93] Gabi Sarkis, Pascal Giard, Alexander Vardy, Claude Thibeault, and Warren J Gross. Fast list decoders for polar codes. *IEEE J. Sel. Areas Commun.*, 34(2):318–328, 2015.

[94] C. E. Shannon. A Mathematical Theory of Communication. *Bell Syst. Tech. J.*, 27:379–423, 623–656, 1948.

[95] Changyang She, Chenyang Yang, and Tony QS Quek. Radio resource management for ultra-reliable and low-latency communications. *IEEE Commun. Mag.*, 55(6):72–78, 2017.

[96] C. Shi-yi and L. Yu-bai. Error correcting cyclic redundancy checks based on confidence declaration. In *6th International Conference on ITS Telecommunications*, pages 511–514, 2006.

[97] M. Shirvanimoghaddam et al. Short block-length codes for ultra-reliable low latency communications. *IEEE Communications Magazine*, 57(2):130–137, 2018.

[98] A. Solomon, K. R Duffy, and M. Médard. Soft maximum likelihood decoding using GRAND. In *IEEE Int. Commun. Conf.*, 2020.

[99] Michal Sybis et al. Channel coding for ultra-reliable low-latency communication in 5G systems. In *IEEE Vehic. Tech. Conf.*, pages 1–5. IEEE, 2016.

[100] I. Tal and A. Vardy. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory*, 61(5):2213–2226, 2015.

[101] S. Tong, D. Lin, A. Kavcic, B. Bai, and Li Ping. On short forward error-correcting codes for wireless communication systems. In *16th int. Conf. on Comp. Comm. and Networks*, pages 391–396, 2007.

[102] F. Tosato and P. Bisaglia. Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, volume 2, pages 664–668 vol.2, 2002.

[103] E. Tsimbalo, X. Fafoutis, and R. J. Piechocki. CRC error correction in IoT applications. *IEEE Transactions on Industrial Informatics*, 13(1):361–369, 2017.

[104] A. Tychopoulos, O. Koufopavlou, and I. Tomkos. FEC in optical communications - a tutorial overview on the evolution of architectures and the future prospects of outband and inband FEC for optical communications. *IEEE Circuits and Devices Mag.*, 22(6):79–86, 2006.

[105] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos. A survey on FEC codes for 100G and beyond optical networks. *IEEE Commun. Surv. Tut.*, 18(1):209–221, 2016.

[106] A. Valembois and M. Fossorier. Box and match techniques applied to soft-decision decoding. *IEEE Trans. Inf. Theory*, 50(5):796–810, 2004.

[107] Qi Wang, Qiuliang Xie, Zhaocheng Wang, Sheng Chen, and Lajos Hanzo. A universal low-complexity symbol-to-bit soft demapper. *IEEE Transactions on Vehicular Technology*, 63(1):119–130, 2014.

[108] R. Wang, W. Zhao, and G. B. Giannakis. CRC-assisted error correction in a convolutionally coded system. *IEEE Trans. Commun.*, 56(11):1807–1815, 2008.

[109] Yingquan Wu and Christoforos N Hadjicostis. Soft-decision decoding of linear block codes using preprocessing and diversification. *IEEE Trans. Inf. Theory*, 53(1):378–393, 2006.

[110] A.M. Yaglom and I.M. Yaglom. *Challenging Mathematical Problems with Elementary Solutions: Volume 1.* Dover Publications, 1 edition, 1964.

[111] C. Yih. Analysis and compensation of DC offset in OFDM systems over frequency-selective rayleigh fading channels. *IEEE Trans. Veh. Technol*, 58(7):3436–3446, 2009.

[112] Chentao Yue, Mahyar Shirvanimoghaddam, Branka Vucetic, and Yonghui Li. A revisit to ordered statistics decoding: Distance distribution and decoding rules. *IEEE Trans. Inf. Theory*, pages 1–1, 2021.

[113] M. Zhan, Z. Pang, D. Dzung, and M. Xiao. Channel coding for high performance wireless control in critical applications: Survey and analysis. *IEEE Access*, 6:29648–29664, 2018.