EPFL

Thèse n° 8989

Symmetry in design and decoding of polar-like codes

Présentée le 4 mars 2022

Faculté informatique et communications
Laboratoire de théorie des communications
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Kirill IVANOV

Acceptée sur proposition du jury

Prof. M. C. Gastpar, président du jury
Prof. R. Urbanke, directeur de thèse
Dr J.-P. Tillich, rapporteur
Prof. I. Dumer, rapporteur
Prof. E. Abbé, rapporteur

École polytechnique fédérale de Lausanne

2022

Dedicated to my family

Acknowledgments

First and foremost, I would like to express the deep gratitude to my advisor, Rüdiger Urbanke. He gives me a lot of freedom in my research. Some chapters of this thesis come from the topics that he advised me to look at and the others from the ideas that I developed on my own, and I am immensely thankful for the opportunity to do both types of research. It happened many times that I come to his office with some troublesome question I'm thinking about and he is almost immediately ready to give a good advice that helps me to look at the problem from a different perspective and get a valuable insight. He also gives full support and encouragement in my activities. Whenever I ask if I should publish some of my results, go to the internship in Huawei Moscow or to the summer school in Brazil, the most common answer is 'yes, absolutely'. Without him, I would not stand here with this thesis.

I am thankful to the IPG personnel who made my life and work substantially easier. Muriel Bardet, our benevolent secretary, is one of the main reasons why we are able to fully enjoy our time in IPG, focusing on research and fun and being able to travel without getting lost in the bureaucracy. She is always here to help with whatever stupid question that we can potentially have. Damir is another reason why everything works so well. Thanks to him, our cluster, workstations and laptops run smoothly. Sometimes he might look quite intimidating but is always helpful nevertheless.

I would also like to thank past and present IPG people. Nicolas Macris kindly introduced me to the world of quantum computing and its brillianses and mysteries. We shared an office with Pierre for the significant part of my PhD time and before the pandemic we spent a lot of time climbing at Totem, playing table football and even skiing. We also had many good times with Erixhen, Eric, Clément, Daria, Reka, Yunus, Amedeo, Andreas and the others. With some of these people we also had fun during ISIT in Paris and ITW in Visby.

Another acknowledgment goes to my friends. Without you, my PhD days would be much more boring and adventureless. Artem is one of the few friends that I know since my first year in Lausanne and we had a lot of fun together. Boris is the main driving force behind the most beautiful and the most dangerous hikes I ever did in Switzerland. Thanks to Valeriya and Polina,

I've got the motivation to master skiing and started enjoying it. I would also like to thank Jenya, Kirill, Matt and our various Russian communities for hikes, skiing trips, parties, barbecues, travels and many other activities. A special round of gratitude is for my Russian friends who live in my homeland. Grisha, Stas, Katya, Kolya, Sonya and Anya, you are the reason why I am always happy to fly home.

Finally, I thank my parents and my sister. They always offer their love and support and believe in me. This thesis is dedicated to you.

Lausanne, 2022

Abstract

Channel coding has rapidly developed since the landmark 1948 paper of Claude Shannon, "A Mathematical Theory of Communication"; it established the largest rate of reliable data transmission through various channels. The main challenge since then has been to find the error-correcting codes that come close to the asymptotic limits. This has been a long journey which includes many brilliant minds making many beautiful theoretical and practical discoveries.

The beginning of 21st century provided us with many answers about how to reach these limits. Irregular low-density parity-check codes achieve a capacity of the binary erasure channel. Polar codes became the first class of codes that provably achieves capacity of any binary memoryless symmetric channel under a low-complexity successive cancellation decoding algorithm. Spatially coupled low-density parity-check (SC-LDPC) codes subsequently followed, being not only capacity-achieving under low-complexity belief propagation decoding but also universal, i.e., good for any channel with the same capacity, which is a stark contrast to polar codes that have channel-dependent construction. Interestingly, polar and SC-LDPC codes take completely different roads to capacity, specifically the polarization effect in channel combining and splitting, and the threshold saturation in iterative message-passing decoding of spatially coupled codes.

Recent developments give us another way that comes from the code symmetry. The Reed-Muller (RM) and extended Bose–Chaudhuri–Hocquenghem (BCH) codes, which are among the oldest families of error-correcting codes with rich algebraic structure, achieve capacity of erasure channels (and are likely to achieve capacity universally) under maximum a posteriori (MAP) decoding. And one of the main ingredients of the proof is the automorphism group of these codes. It is known that short RM and eBCH codes demonstrate excellent performance under optimum decoding that is close to the finite-length limits. However, due to the intractability of MAP decoding for non-erasure channels, we need to develop low-complexity error-correction algorithms. A potentially fruitful way is to consider these codes as polar-like, although the straightforward application of successive cancellation decoding algorithm demonstrates unsatisfactory performance.

In the first part of this thesis, we talk about the performance improvements

that an automorphism group of the code brings on board. We propose two decoding algorithms for the Reed-Muller codes, which are invariant under a large group of permutations and are expected to benefit the most. The former is based on plugging the codeword permutations in successive cancellation decoding, and the latter utilizes the code representation as the evaluations of Boolean monomials. However, despite the performance improvements, it is clear that the decoding complexity grows quickly and becomes impractical for moderate-length codes. In the second part of this thesis, we provide an explanation for this observation. We use the Boolean polynomial representation of the code in order to show that polar-like decoding of sufficiently symmetric codes asymptotically needs an exponential complexity. The automorphism groups of the Reed-Muller and eBCH codes limit the efficiency of their polar-like decoding for long block lengths, hence we either should only focus on short codes or find another way. We demonstrate that asymptotically same restrictions (although with a slower convergence) hold for more relaxed condition that we call partial symmetry. The developed framework also enables us to prove that the automorphism group of polar codes cannot include a large affine subgroup.

In the last part of this thesis, we address a completely different problem. A device-independent quantum key distribution (DIQKD) aims to provide private communication between parties and has the security guarantees that come mostly from quantum physics, without making potentially unrealistic assumptions about the nature of the communication devices. After the quantum part of the DIQKD protocol, the parties share a secret key that is not perfectly correlated. In order to synchronize, some information needs to be revealed publicly, which makes this formulation equivalent to the asymmetric Slepian-Wolf problem that can be solved using binary linear error-correction codes. As any amount of the revealed information reduces the key secrecy, the utilized code should operate close to the finite-length limits. The channel in consideration is non-standard and, due to its experimental nature, it can actually slightly differ from the considered models. In order to solve this problem, we designed a simple scheme using universal SC-LDPC codes and used in the first successful experimental demonstration of DIQKD protocol.

Keywords: polar codes, Reed-Muller codes, eBCH codes, capacity-achieving codes, maximum likelihood decoding, list decoding, device-independent quantum key distribution, spatially-coupled codes, asymmetric Slepian-Wolf problem.

Résumé

Le codage des canaux s'est rapidement développé depuis l'article historique de Claude Shannon de 1948, "A Mathematical Theory of Communication" ; il a établi le plus grand taux de transmission fiable de données par divers canaux. Le principal défi depuis lors a été de trouver les codes correcteurs d'erreurs qui se rapprochent des limites asymptotiques. Ce fut un long voyage au cours duquel de nombreux esprits brillants ont fait de belles découvertes théoriques et pratiques.

Le début du 21e siècle nous a apporté de nombreuses réponses sur la manière d'atteindre ces limites. Les codes irréguliers à contrôle de parité à faible densité atteignent une capacité du canal d'effacement binaire. Les codes polaires sont devenus la première classe de codes qui atteint de manière prouvée la capacité de tout canal binaire symétrique sans mémoire sous un algorithme de décodage à annulation successive de faible complexité. Les codes à contrôle de parité à faible densité et à couplage spatial (SC-LDPC) ont ensuite suivi, car ils permettent non seulement d'atteindre la capacité dans le cadre d'un décodage à propagation de convictions à faible complexité, mais ils sont également universels, c'est-à-dire qu'ils sont bons pour tout canal ayant la même capacité, ce qui contraste fortement avec les codes polaires dont la construction dépend du canal. Il est intéressant de noter que les codes polaires et SC-LDPC empruntent des voies complètement différentes pour atteindre la capacité, notamment l'effet de polarisation dans la combinaison et le fractionnement des canaux, et la saturation du seuil dans le décodage itératif par passage de message des codes à couplage spatial.

Des développements récents nous donnent une autre voie qui provient de la symétrie du code. Les codes de Reed-Muller (RM) et les codes étendus de Bose-Chaudhuri-Hocquenghem (BCH), qui font partie des plus anciennes familles de codes correcteurs d'erreurs avec une structure algébrique riche, atteignent la capacité des canaux à effacement (et sont susceptibles d'atteindre la capacité universelle) dans le cadre d'un décodage maximum a posteriori (MAP). Et l'un des principaux ingrédients de la preuve est le groupe d'automorphisme de ces codes. On sait que les codes RM et eBCH courts présentent d'excellentes performances sous décodage optimal proche des limites de longueur finie. Cependant, en raison de l'intractabilité du décodage MAP pour les canaux non

effacés, nous devons développer des algorithmes de correction d'erreur à faible complexité.Une voie potentiellement fructueuse est de considérer ces codes comme étant de type polaire, bien que l'application directe de l'algorithme de décodage par annulation successive démontre une performance insatisfaisante.

Dans la première partie de cette thèse, nous parlons des améliorations de performance qu'apporte un groupe d'automorphisme du code. Nous proposons deux algorithmes de décodage pour les codes de Reed-Muller, qui sont invariants sous un grand groupe de permutations et qui devraient en bénéficier le plus. Le premier est basé sur l'insertion des permutations de mots de code dans le décodage par annulations successives, et le second utilise la représentation du code comme les évaluations de monômes booléens. Cependant, malgré les améliorations de performance, il est clair que la complexité du décodage croît rapidement et devient peu pratique pour les codes de longueur moyenne. Dans la deuxième partie de cette thèse, nous fournissons une explication à cette observation. Nous utilisons la représentation polynomiale booléenne du code afin de montrer que le décodage de type polaire de codes suffisamment symétriques nécessite asymptotiquement une complexité exponentielle. Les groupes d'automorphisme des codes de Reed-Muller et eBCH limitent l'efficacité de leur décodage de type polaire pour les grandes longueurs de bloc, et nous devons donc nous concentrer sur les codes courts ou trouver une autre solution. Nous démontrons que les mêmes restrictions asymptotiques (bien qu'avec une convergence plus lente) sont valables pour une condition plus relaxée que nous appelons symétrie partielle. Le cadre développé nous permet également de prouver que le groupe d'automorphisme des codes polaires ne peut pas inclure un grand sous-groupe affine.

Dans la dernière partie de cette thèse, nous abordons un problème complètement différent. Une distribution quantique de clé indépendant du pérephérique (DIQKD) a pour but de fournir une communication privée entre les parties et a les garanties de sécurité qui proviennent principalement de la physique quantique, sans faire des hypothèses potentiellement irréalistes sur la nature des pérephérique de communication. Après la partie quantique du protocole DIQKD, les parties partagent une clé secrète qui n'est pas parfaitement corrélée. Afin de se synchroniser, certaines informations doivent être révélées publiquement, ce qui rend cette formulation équivalente au problème asymétrique de Slepian-Wolf qui peut être résolu à l'aide de codes correcteurs d'erreurs linéaires binaires. Comme toute quantité d'information révélée réduit le secret de la clé, le code utilisé doit fonctionner près des limites de longueur finie. Le canal considéré n'est pas standard et, en raison de sa nature expérimentale, il peut en fait différer légèrement des modèles considérés. Afin de résoudre ce problème, nous avons conçu un schéma simple utilisant des codes SC-LDPC universels et utilisé dans la première démonstration expérimentale réussie du protocole DIQKD.

Mots clés : codes polaires, codes eBCH, codes de Reed-Muller, le décodage par probabilité maximale, décodage en liste, codes à couplage spatial, distribution quantique de clé indépendant du pérephérique, codes de capacité

atteindre, problème asymétrique de Slepian-Wolf.

Contents

1 Introduction

The 21st century is an era of information and digital communication. Today, the interaction with zillions of intelligent devices has become an essential part of the daily routine of most human beings; these devices are hidden everywhere, from mobile phones that we use to connect with each other to smart city systems that control power plants, water supply, and transportation. Wireless networks are one of the modern technology cornerstones. They extend our freedom and broaden our mobility at the cost of being subjected to higher noise in the communication links compared to wired configurations. There exist many techniques operating on different abstraction layers of the communication system model; these techniques target achieving the reliable interaction between the parties and satisfy additional constraints such as high data throughput, low latency, and energy consumption.

Channel coding is a crucial component when it comes to dealing with noise. The general idea can be described as sending some additional data along with the payload to compensate for the expected corruption during the transmission. Theoretical foundations of coding date back to 1948 with Claude Shannon and his landmark paper "A Mathematical Theory of Communication", in which he established the possibility of virtually error-free transmission through the noisy channel, as long as the data rate does not exceed a certain characteristic of the channel, which is called the *capacity*. The key ingredient to the recipe of achieving channel capacity is coding. In the remainder of this chapter, we describe the core concepts and models that are actively used in this thesis. We then describe the brief history of coding paradigms and conclude by summarizing our contributions in this thesis.

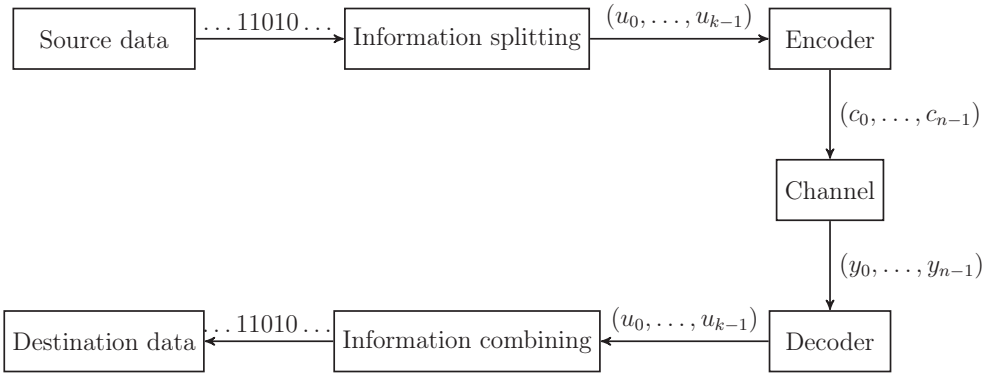


Figure 1.1 – Point-to-point communication system model

1.1 Channel Coding: Main Concepts and Models

Let us consider the following simple point-to-point communication model that is demonstrated in Figure 1.1. We have the transmitter that seeks to reliably send its data to the receiver via the noisy channel. In order to guarantee reliable communication, the transmitter first splits the data bits into *information blocks* of length k and then utilizes a *block code* $\mathcal{C}$ that is known to both parties. Code $\mathcal{C}$ is essentially a rule that maps k-bit information blocks to distinct n-bit sequences called *codewords*. A codeword length is often referred to as the *block length* of code $\mathcal{C}$, and $R = \log_2 |\mathcal{C}|/n$ is called the rate of $\mathcal{C}$. The code rate is a measure of how much information per codeword bit can be transmitted using $\mathcal{C}$. The job of the receiver is to recover, or decode, the original codeword from the corrupted result of its transmission through the noisy channel. It is typically assumed that the channel is perfectly known to the receiver.

The celebrated coding theorem of Shannon [1] says that for a channel W there exists a sequence of rate-R codes $\mathcal{C}$ such that the receiver recovers the original codeword with an arbitrary small probability of failure when the block length goes to infinity. And this is always possible, as long as $R < C(W)$, where $C(W)$ is the capacity of W and is sometimes referred to as the *Shannon limit*. A channel W is characterized by its input $\mathcal{X}$, output $\mathcal{Y}$ and the transitional probabilities given by $W(y|x), y \in \mathcal{Y}, x \in \mathcal{X}$. It will be further assumed that the channel input is binary and it is convenient to take $\mathcal{X} = \{-1, +1\}$, instead of $\{0, 1\}$, by mapping 0 to 1 and 1 to -1. We call a channel memoryless if

$$W(y_0, \ldots, y_{n-1}|x_0, \ldots, x_{n-1}) = \prod_{i=0}^{n-1} W(y_i|x_i),$$

and output-symmetric if $W(y| - x) = W(-y|x)$.

Classic coding theory is mostly centered around three binary memoryless symmetric (BMS) channel models. *Binary erasure channel* with erasure

probability ε, or $BEC(\varepsilon)$, has output alphabet $\mathcal{Y} = \mathcal{X} \cup \{\epsilon\}$ and transition probabilities

$$W(x|x) = 1 - \varepsilon,$$
$$W(\epsilon|x) = \varepsilon,$$

i.e., the transmitted symbol is left untouched with probability $1 - \varepsilon$ and is replaced with an *erasure symbol* ϵ with probability ε. *Binary symmetric channel* with crossover (or bit-flip) probability p, or $BSC(p)$, has output alphabet $\mathcal{Y} = \mathcal{X}$ and transition probabilities

$$W(x|x) = 1 - p,$$
$$W(-x|x) = p,$$

i.e., the transmitted symbol is left untouched with probability $1 - p$ and is flipped with probability p.

Remark. The capacity of channels $BEC(\varepsilon)$ and $BSC(p)$ has very simple expressions, namely $C(BEC(\varepsilon)) = 1 - \varepsilon$ and $C(BSC(p)) = 1 - h_2(p)$, where $h_2(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ is a binary entropy function.

An *Additive white-noise Gaussian channel* with noise variance σ^2, or $AWGNC(\sigma^2)$, has output alphabet $\mathcal{Y} = \mathbb{R}$ and transition probabilities given by

$$W(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-x)^2}{2\sigma^2}},$$

i.e., the received symbol can be expressed as $y = x + \eta$, where $x \in \{-1, +1\}$ and η is a normally distributed random variable with zero mean and variance σ^2. The standard notation to express the noise level in binary-input AWGN channel is given by a signal-to-noise ratio (SNR) per bit E_b/N_0 that is defined as

$$\frac{E_b}{N_0} = 10\log_{10}\frac{1}{2\sigma^2 R},$$

where R is the rate of code $\mathcal{C}$ which is used for the transmission.

The receiver wants to find the most likely codeword that corresponds to the received sequence. The maximum a posteriori (MAP) decoding rule is defined as

$$\mathbf{x}_{MAP} = \arg\max_{\mathbf{x}\in\mathcal{C}} W(\mathbf{x}|\mathbf{y})$$

and minimizes the decoding error probability that is sometimes referred to as *frame error rate* (FER). In case of uniform distribution on codewords, which is a typical assumption in the communication system model, MAP decoding is equivalent to the maximum likelihood (ML) decoding rule that is given by

$$\mathbf{x}_{ML} = \arg\max_{\mathbf{x}\in\mathcal{C}} W(\mathbf{y}|\mathbf{x}). \tag{1.1}$$

In case of BSC and AWGN channels, ML decoding rules admit simpler reformulations that can be utilized in order to develop practical decoding algorithms. Specifically, for binary symmetric channel, the ML solution is given by the codeword with the smallest Hamming distance from the received vector:

$$\mathbf{x}_{ML,BSC} = \arg\min_{\mathbf{x}\in\mathcal{C}} d_H(\mathbf{x},\mathbf{y}), \quad d_H(\mathbf{x},\mathbf{y}) = \sum_i \mathbb{1}\{x_i \neq y_i\},$$

and for the Gaussian channel the ML codeword has the smallest Euclidean distance from the received vector:

$$\mathbf{x}_{ML,AWGN} = \arg\min_{\mathbf{x}\in\mathcal{C}} d_E(\mathbf{x},\mathbf{y}), \quad d_E(\mathbf{x},\mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}.$$

In the case of AWGN channel, it is also convenient to assume that the decoder does not operate with the received sequence but rather with the log-likelihood ratios (LLRs) of the codeword bits

$$l_i = \frac{\log W(y_i|1)}{\log W(y_i|-1)}.$$

1.2 A Brief History of Error-Correcting Codes

The works of Shannon established the theoretical limits of data transmission but left unsolved the question of how to achieve them in practice. The practical implementation of a coding scheme is also concerned with the storage space needed for code description and the complexity of encoding and decoding operations.

A binary linear (n, k, d) block code $\mathcal{C}$ is a k-dimensional linear subspace of n-dimensional binary vector space, such that the Hamming distance between its two distinct elements is at least d. Taking a basis of this subspace and arranging it in $k \times n$ *generator matrix* $\mathbf{G}$ provides the compact description of code $\mathcal{C}$ and the encoding operation simply becomes

$$\mathbf{c} = \mathbf{u}\mathbf{G}.$$

Therefore, any linear code can be described with only kn bits instead of storing all 2^k codewords and the encoding with a linear code can be performed in $O(kn)$ time, which explains why linear codes are extensively used in communications systems. The choice of $\mathbf{G}$ as $k \times n$ binary matrix with uniformly distributed entries gives us a *random linear code* that achieves the Shannon limit with high probability. Are we good? Unfortunately, the decoding problem remains nontrivial and the maximum likelihood decoding of random linear codes

needs time that is exponential in n. Moreover, the general decoding problem for linear codes is NP-complete [2], hence the existence of polynomial-time algorithm is unlikely.

The first nontrivial code is the $(7, 4, 3)$ Hamming code that was invented by Richard Hamming around 1947 and later generalized to the infinite family of codes [3]. The Hamming codes became the beginning of the *algebraic coding* era; algebraic coding remained the dominant research paradigm in coding theory for many decades. The main focus was on seeking codes with a large minimum distance: a measure of the worst-case performance of the code, i.e., the level of noise guaranteed to be tolerated. Algebraic codes are typically characterized by the deterministic construction, which simplifies the estimation of their minimum distance, and they have much internal structure, which is used for efficient encoding and decoding. The Reed-Muller codes are a remarkable infinite algebraic code family. They were introduced in 1954 by David Muller [4], with the subsequent rediscovery by Irving Reed who also proposed an efficient decoding algorithm based on the majority logic [5] that was appealing for hardware implementation. Contrary to the Hamming codes, Reed-Muller codes can be used for transmission with different data rates. However, the Reed-Muller codes' minimum distance grows only as a square root of the block length, which is rather unimpressive. BCH codes, invented by Alexis Hocquenghem in 1959 [6] and independently by Raj Chandra Bose and Dijen Ray-Chaudhuri in 1960 [7], have better minimum distance and are an example of cyclic codes, the class of linear block codes such that a circular shift of any codeword is again a codeword of the same code. A class of non-binary BCH codes, the Reed-Solomon codes, were independently discovered by Irving Reed and Gustave Solomon in 1960 [8]. Since then, they found their applications in many spheres. It is worth noting the all aforementioned algebraic codes have a large group of automorphisms, i.e., there exist many permutations on codeword indices that map codewords to other codewords.

In 1993, there was a significant breakthrough in coding theory, when Claude Berrou presented the turbo codes [9]. Combining two simple convolutional codes with an interleaver demonstrated the performance near the Shannon limit under low-complexity iterative decoding. Turbo codes started the era of *iterative decoding* paradigm. Concurrently, MacKay [10] and Spielman [11] rediscovered low-density parity-check codes, or LDPC codes; originally invented by Gallager in 1960s [12] but forgotten due to the limits of computing power at the time. LDPC codes can also be decoded with an iterative algorithm. Wiberg, Loeliger and Kötter put turbo and LDPC codes together as instances of *sparse-graph codes* and their iterative decoding algorithms are instances of belief propagation [13]. They also rediscovered a paper by Tanner; it introduces a bipartate graph representation of LDPC codes, currently known as *Tanner graph* [14]. A further development of message-passing algorithms on graphs is due to Kschischang, Frey, and Loeliger [15].

The beginning of the new century came with a capacity-achieving result. Luby, Mitzenmacher, Shokrollahi and Spielman constructed *irregular* LDPC codes and showed that these codes can asymptotically approach the capacity of binary erasure channel with low complexity[16, 17]. Meanwhile, Richardson and Urbanke developed density evolution, a powerful tool for the analysis of the performance of LDPC code under iterative message-passing decoding [18] when the transmission takes place over general channels. They demonstrated the excellent performance of optimized LDPC codes [19] and, for the case of AWGN channel, Chung, Forney, Richardson and Urbanke designed irregular codes that achieve 0.0045 dB from the Shannon limit [20].

LDPC convolutional codes, or spatially coupled LDPC (SC-LDPC) codes, are constructed by coupling together several copies of LDPC codes; they were initially proposed by Felström and Zigangirov [21]. Lentmaier, Sridharan, Zigangirov and Costello observed numerically that the properly constructed SC-LDPC codes perform substantially better than LDPC codes under belief propagation decoding when the transmission takes place over the binary erasure channel or the AWGN channel [22]. Kudekar, Richardson and Urbanke formalized these observations and proved that, for the case of BEC [23], the *belief propagation threshold* of SC-LDPC codes asymptotically coincides with the *maximum a posteriori threshold* of the underlying LDPC codes. Later, they proved that the same phenomenon occurs in any BMS channel, hence spatially coupled codes are *universal*, i.e., simultaneously achieve the capacity of all channels with the same capacity [24], under low-complexity iterative decoding. However, despite SC-LDPC being the first universal codes, they were not the first to reach the Shannon limit for general BMS channels under low-complexity decoding..

The polar codes, introduced by Arikan in his 2009 paper [25], target the Shannon limit from the completely different perspective. Arikan used channel-combining and channel-splitting techniques to show that a certain transformation induces synthetic bit channels that are all asymptotically either completely noisy or completely noiseless, which is the essence of the polarization effect. Arikan used this to prove that polar codes achieve the capacity of any BMS channel with low-complexity encoding and *successive cancellation* decoding algorithms. Later, it was demonstrated in [26] that the optimized codes for bitwise multistage decoding, studied by Stolte [27], coincide with polar codes. Stolte did not prove or conjecture, however, the capacity-achieving property. Consequently, his work was largely forgotten.

The development of good finite-length polar coding solutions has been a hot topic ever since their discovery. Tal and Vardy proposed the list version of successive cancellation decoding in order to reach ML performance [28], although, due to the small minimum distance of polar codes, this was insufficient to be competitive with LDPC and turbo codes. The subsequent attempts

to design improved polar-like codes include CRC-aided polar codes [29, 28], parity-check-concatenated polar codes [30] and polar subcodes [31, 32, 33] and polarization-adjusted convolutional codes [34] that outperform other coding schemes and played a crucial role in the adoption of polar codes to a 5G radio standard [35]. The need for reducing the decoding latency gave birth to various ides on utilizing different permutations of the received sequence in order to improve the performance [36, 37, 38, 39, 40, 41, 42, 43, 44].

The success of polar codes brought much attention to the Reed-Muller codes that can be described in the similar way as polar codes. Remarkably, Kudekar, Kumar, Mondelli, Pfister, Şaşoğlu and Urbanke proved that Reed-Muller codes as well as extended BCH codes achieve capacity of erasure channels under bitwise MAP decoding [45]. This result comes from the automorphism group of these codes. Furthermore, Reeves and Pfister recently proved that Reed-Muller codes achieve the capacity of BMS channels but only in terms of bit error probability [46] that used another aspect of the algebraic structure of Reed-Muller codes. Despite the significant progress [47, 48, 49], the complete proof regarding the block error probability remains unknown, as well as any results regarding eBCH codes for general BMS channels. The design of new decoding algorithms for Reed-Muller codes is still an open problem and attracts much research interest. The most notable ideas include [50, 51, 52, 53], where authors take completely different perspectives on the decoding problem.

1.3 Thesis Outline

This thesis can be split essentially into two parts. In the first part, we introduce the framework of Boolean polynomial representation of error-correcting codes in Chapter 2 and study many well-known codes, including the polar, and the Reed-Muller and eBCH codes. This framework is related to a polar-like representation of binary linear codes and a polar-like decoding of these codes. We also introduce the general affine group of permutations; it fits nicely into the Boolean polynomial framework and plays a crucial rule in further analysis. Some of these permutations are useful in the context of polar-like decoding and some of them are not. We follow this road further in Chapters 3 and 4.

Chapter 3 is more practical-oriented, and we focus on the Reed-Muller codes with their invariance under a large group of permutations. We propose two decoding algorithms that exploit the rich algebraic structure of these codes and outperform the standard approaches. The former is for binary erasure channel and achieves an extra acceleration, due to the fact that in erasure channel any non-erasure symbol is perfectly known. The latter is for general channels, but its use is restricted to the subclass of the Reed-Muller codes.

Chapter 4 is more theory-oriented, and we focus on the potential efficiency of polar-like decoding for various codes. We use our framework to define a

specific type of partial code symmetry that serves as a signal that the maximum likelihood performance can be achieve with complexity that grows exponentially with the code length. In the case of the Reed-Muller and eBCH codes, we connect their automorphism groups with this notion of symmetry to prove that these codes are asymptotically ill-suited for polar-like decoding. If we try to construct long codes tailored for polar-like decoding with permutations, the similar effect occurs. By also relating the partial symmetry with the dynamics of polar-like decoding, we can prove that polar codes asymptotically do not have many automorphisms that can be utilized for improving the decoding performance.

Chapter 5 contains the second part of this thesis. We go outside the classic channel coding and talk about the problem of establishing provably secure communication between two parties. The crucial component of this process is the key distribution, and we want to develop the protocol that enables the parties to end up with the identical copies of a secret key, even in the presence of a potential eavesdropper. Device-independent quantum key distribution (DIQKD) provides an elegant solution to this problem; it relies on quantum mechanics with the classical post-processing. We took part in the development of DIQKD protocol that has been successfully implemented in practice. After the quantum part of the protocol, one of the parties has the secret key that is slightly corrupted compared to another. This issue is resolved by revealing some information followed by an error-correction step, and the amount of revealed information reduces the protocol security. We develop a custom coding scheme based on SC-LDPC codes in order to achieve the performance close to the Shannon limit.

2 Preliminaries

2.1 Notations

We use $[n]$ to denote the set $\{0, \ldots, n-1\}$. $\mathbb{F}_q$ denotes the finite field with q elements, and $\mathbb{F}_q^m$ is the m-dimensional vector space over $\mathbb{F}_q$. Note that we can consider the vector space $\mathbb{F}_q^m$ as the finite field $\mathbb{F}_{q^m}$ and vice versa. $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$ is the multiplicative group of $\mathbb{F}_q$. Bold letters are used for matrices and vectors, e.g., $\mathbf{A}$ and $\mathbf{b}$. Given a binary vector $\mathbf{v} = (v_0, \ldots, v_{m-1})$, we consider it as an integer $v = \sum_{i=0}^{m-1} v_i 2^i$ where needed. For $\mathbf{c} = (c_0, \ldots, c_{n-1})$, we define $\mathbf{c}_a^b = (c_a, \ldots, c_b), 0 \leq a \leq b < n$. $R(\mathcal{C})$ denotes the rate of a binary linear code $\mathcal{C}$. Hamming weight $\mathrm{wt}(\mathbf{v})$ of vector $\mathbf{v}$ is defined as the number of its nonzero entries.

2.2 Boolean Functions

Let $\{x_0, \ldots, x_{m-1}\}$ be a collection of m variables taking their values in $\mathbb{F}_2$ and let $\mathbf{v} = (v_0, \ldots, v_{m-1}) \in \mathbb{F}_2^m$ be any binary m-tuple. Then,

$$x^{\mathbf{v}} = \prod_{i=0}^{m-1} x_i^{v_i}$$

denotes a monomial of degree $\mathrm{wt}(\mathbf{v})$.

A function $f(\mathbf{x}) = f(x_0, \ldots, x_{m-1}) : \mathbb{F}_2^m \to \mathbb{F}_2$ is called Boolean. Any such function can be uniquely represented as an m-variate polynomial:

$$f(x_0, \ldots, x_{m-1}) = \sum_{\mathbf{v} \in \mathbb{F}_2^m} a_{\mathbf{v}} x^{\mathbf{v}},$$

where $a_{\mathbf{v}} \in \{0,1\}$ [54]. Its evaluation vector $\mathrm{ev}(f(\mathbf{x})) \in \mathbb{F}_2^{2^m}$ is obtained by evaluating f at all points $\boldsymbol{\alpha}_i$ of $\mathbb{F}_2^m$. Note that any length-2^m binary vector $\mathbf{c}$ can be considered as an evaluation vector of some function f. For the rest of the paper, we assume the standard bit ordering of points, i.e., $\boldsymbol{\alpha}_i$ being the binary expansion of integer i.

2.3 Monomial and Polynomial Codes

Consider a binary linear $(n = 2^m, k, d)$ code $\mathcal{C}$ with generator matrix $\mathbf{G}$. Its *generating set* is given by

$$M_{\mathcal{C}} = \{f_i, 0 \leq i < k | \, \mathrm{ev}(f_i) = \mathbf{G}_{i,*}\},$$

where $\mathbf{G}_{i,*}$ are rows of $\mathbf{G}$. The code $\mathcal{C}$ is called monomial if there exists $M_{\mathcal{C}}$ that contains only monomials and polynomial otherwise. The minimum distance of a monomial code can be calculated [55, Proposition 3] as

$$d_{min}(\mathcal{C}) = 2^{m - \max_{x^{\mathbf{v}} \in M_{\mathcal{C}}} \mathrm{wt}(\mathbf{v})}. \tag{2.1}$$

2.3.1 From Generator Matrix to Generating Set

One can now ask how to construct the generating set from a given generator matrix. The task of determining if a code is monomial also seems nontrivial from the first glance.

Consider the matrix $\mathbf{A}_m = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{\otimes m}$, where $\otimes m$ denotes an m-fold Kronecker product of the matrix with itself. It is easy to verify that $\begin{pmatrix} 1 & 1 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 \end{pmatrix}$ are the evaluations over $\mathbb{F}_2$ of the constant monomial 1 and the monomial x_0, respectively, and from the induction on m it follows that the v-th row of $\mathbf{A}_m$ is an evaluation vector over $\mathbb{F}_2^m$ of the monomial $x^{\mathbf{v}}$. Therefore, an evaluation vector of function $f(\mathbf{x}) = \sum_{\mathbf{v} \in \mathbb{F}_2^m} a_{\mathbf{v}} x^{\mathbf{v}}$ can be computed as

$$\mathrm{ev}(f) = \mathbf{u}^{(f)} \mathbf{A}_m,$$

where $u_v^{(f)} = a_v$. Matrix $\mathbf{A}_m$ is a Kronecker product of a full-rank matrix with itself and therefore invertible, hence we can uniquely recover vector $\mathbf{u}^{(f)}$ that contains the coefficients of the polynomial representation of f from $\mathrm{ev}(f)$. Applying this procedure to $\mathbf{G}_{i,*}$ gives the generating set of $\mathcal{C}$. Note that there are many generator matrices of $\mathcal{C}$ that give different generating sets. In order to get a unique generating set, we can use the technique similar to the one proposed in [31]. Namely, we use that $\mathbf{A}_m$ is non-singular to get the decomposition

$$\mathbf{G} = \mathbf{Q} \mathbf{A}_m, \tag{2.2}$$

for some *precoding matrix* $\mathbf{Q}$. Applying the Gaussian elimination to $\mathbf{Q}$ and transforming it to the reduced row echelon form $\tilde{\mathbf{Q}}$ makes it unique. If the code $\mathcal{C}$ is monomial, the only nonzero columns of the corresponding matrix $\tilde{\mathbf{Q}}$ are the columns of the identity matrix, which can be easily verified. Otherwise, we can write the generating set of $\mathcal{C}$ as

$$\tilde{M}_{\mathcal{C}} = \left\{ \sum_j \tilde{Q}_{i,j} x^{\mathbf{j}} | 0 \leq i < k \right\}. \tag{2.3}$$

2.3.2 Reed-Muller Codes

A Reed-Muller code [5, 4] $\mathrm{RM}(r, m)$ of order r is spanned by the evaluations of m-variate monomials of degree at most r. The $\mathrm{RM}(r, m)$ code has length 2^m, dimension $\sum_{i=0}^{r} \binom{m}{i}$ and minimum distance 2^{m-r}. By definition, Reed-Muller codes are monomial with the generating set $M_{r,m} = \{x^{\mathbf{v}} | \, \mathrm{wt}(\mathbf{v}) \leq r\}$.

2.3.3 Polar Codes

Consider a binary memoryless symmetric (BMS) channel $W : \mathcal{X} \to \mathcal{Y}$ and define the symmetric capacity $I(W)$ as

$$I(W) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} \frac{1}{2} W(y|x) \log_2 \frac{W(y|x)}{\frac{1}{2}(W(y|0) + W(y|1))},$$

that is the highest rate of reliable communication through W given the uniform distribution on its inputs and is equal to the Shannon capacity for BMS channels. Consider the following channel transformation that takes two copies of W and produces two *synthetic channels* $W^{(-)}$ and $W^{(+)}$:

$$W^{(-)}(y_0, y_1|u_0) = \frac{1}{2} \sum_{u_1 \in \{0,1\}} W(y_0|u_0 \oplus u_1) W(y_1|u_1), \tag{2.4}$$

$$W^{(+)}(y_0, y_1, u_0|u_1) = \frac{1}{2} W(y_0|u_0 \oplus u_1) W(y_1|u_1). \tag{2.5}$$

Arikan in [25] proved that the transformation (2.4) preserves the capacity, i.e., $I(W^{(-)}) + I(W^{(+)}) = 2I(W)$, and applying it recursively on $W^{(-)}$) and $W^{(+)}$ leads to the channel polarization effect. Namely, depth-m recursion induces $n = 2^m$ synthetic channels $W^{(i)}$, where $\mathbf{i} \in \{-, +\}^m$, with the following properties:

$$\lim_{n \to \infty} \frac{|\{i | I(W^{(i)}) \notin \{0, 1\}\}|}{n} = 0,$$

$$\lim_{n \to \infty} \frac{|\{i | I(W^{(i)}) = 1\}|}{n} = I(W).$$

Transmitting the information through perfectly reliable channels $W^{(i)}$ guarantees the error-free communication with any rate smaller than $I(W)$ and therefore polar codes achieve the capacity of W.

In the finite-length regime, a $(n = 2^m, k)$ polar code with the set of frozen symbols $\mathcal{F}$ is a binary linear code generated by rows of $\mathbf{A}_m$ with indices $i \in [n] \setminus \mathcal{F}$. The set $\mathcal{F}$ contains the indices of synthetic channels $W(i)$ with the smallest capacities. Our definition of polar codes is slightly different from the conventional, that uses the matrix $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{\otimes m}$, but as already noted in [55], both definitions are equivalent and ours simplifies the polynomial notation. It follows from the correspondence between rows of $\mathbf{A}_m$ and m-variate monomials that polar codes are monomial with the generating set $M_{\mathcal{F}} = \{x^{\mathbf{i}} | i \notin \mathcal{F}\}$.

2.3.4 Extended BCH Codes

A $(2^m - 1, k, d \geq \delta)$ primitive narrow-sense Bose–Chaudhuri–Hocquenghem (BCH) code [56] with design distance δ has a parity check matrix $\mathbf{H}$ with elements

$$H_{j,i} = \alpha_i^j, 0 < i < 2^m, 0 \leq j < \delta - 1,$$

where α_i are distinct elements of $\mathbb{F}_{2^m}^*$. An extended code is formed by adding an overall parity check symbol.

2.4 Symmetries

2.4.1 Automorphism Groups

Consider a permutation π on $[2^m]$ and define its action on the Boolean function f as another Boolean function g obtained by permuting its evaluation points, namely $\pi(f) = g : \text{ ev}(g) = (f(\pi(\boldsymbol{\alpha}_0)), \ldots, f(\pi(\boldsymbol{\alpha}_{2^m-1})))$. The set of permutations that leave code $\mathcal{C}$ invariant, i.e., map its codewords to other codewords, forms the automorphism group of a code denoted by $\text{Aut}(\mathcal{C})$.

Automorphisms and permutations have been studied for many years. The main focus of this thesis is on the general affine group $GA(m, \mathbb{F}_2)$ as well as some of its subgroups. The action of $GA(m, \mathbb{F}_2)$ on the binary representation of $\boldsymbol{\alpha}_i$ can be expressed as the following mapping from $\mathbb{F}_2^m$ to itself:

$$\mathbf{x} \to \mathbf{A}\mathbf{x} + \mathbf{b},$$

where $\mathbf{A} \in \mathbb{F}_2^{m \times m}$ is an invertible matrix and $\mathbf{b} \in \mathbb{F}_2^m$. The result of the action of an element of $GA(m, \mathbb{F}_2)$ on Boolean polynomial $f(\mathbf{x})$ can also be expressed as a change of variables [55]

$$y_i = \sum_j A_{i,j} x_j + b_i. \tag{2.6}$$

Consider now the coordinates $\boldsymbol{\alpha}_i$ as the elements of finite field $\mathbb{F}_{2^m}$. A general affine group $GA(1, \mathbb{F}_{2^m})$ consists of permutations

$$x \to ax + b,$$

where x, a, b are treated as the elements of finite field $\mathbb{F}_{2^m}$, $a \in \mathbb{F}_{2^m}^*$ and the ax is a finite field multiplication.

The notable subgroups of $GA(m, \mathbb{F}_2)$ include the group of translations $\mathcal{T}_m$ that consists of transpositions $\mathbf{x} \to \mathbf{x} + \mathbf{b}, \mathbf{b} \in \mathbb{F}_2^m$ and is also a subgroup of $GA(1, \mathbb{F}_{2^m})$, and the group of variable permutations $\mathcal{P}_m$. The elements of $\mathcal{P}_m$ have form $\mathbf{x} \to \mathbf{P}\mathbf{x}$, where $\mathbf{P}$ is a permutation matrix. The action of $\mathcal{P}_m$ permutes bits in the binary representation of the evaluation points $\boldsymbol{\alpha}_i$, or, as follows from (2.6), permutes variables $\{x_0, \ldots, x_{m-1}\}$ in the polynomial representation of function $f(\mathbf{x})$.

Let us now present some known results regarding the automorphism groups of Reed-Muller and eBCH codes, that are utilized later in the thesis.

Proposition 1 ([56], Ch. 13, Theorem 24)**.** *Reed-Muller codes are invariant under the action of* $GA(m, \mathbb{F}_2)$.

Proposition 2 ([56], Ch.8, Theorem 16)**.** *eBCH codes are invariant under the action of* $GA(1, \mathbb{F}_{2^m})$.

2.4.2 Projections and Derivatives

The derivative in direction $\mathbf{b}$ of the Boolean function f is defined as

$$(D_{\mathbf{b}} f)(\mathbf{x}) = f(\mathbf{x} + \mathbf{b}) - f(\mathbf{x}). \tag{2.7}$$

Given that $\mathbf{g}$ is the evaluation vector of f, from (2.7) it follows that evaluation vector of $D_{\mathbf{b}} f$ can be computed as

$$\mathrm{ev}(D_{\mathbf{b}} f) = (g_{\mathbf{0}} + g_{\mathbf{0} \oplus \mathbf{b}}, \ldots, g_{\mathbf{n-1}} + g_{\mathbf{n-1} \oplus \mathbf{b}}).$$

For monomials, the expression (2.7) becomes

$$D_{\mathbf{b}} x^{\mathbf{v}} = \prod_{i=0}^{m-1} (x_i + b_i)^{v_i} - \prod_{i=0}^{m-1} x_i^{v_i}.$$

When $\mathrm{wt}(\mathbf{b}) = 1$, i.e., when $\mathbf{b} = \mathbf{e}_i$, the directional derivative coincides with the partial derivative $\frac{\partial f}{\partial x_i}$. Note that a partial derivative of a monomial code is also a monomial code.

The derivative in direction $\mathbf{b}$ of the code $\mathcal{C}$ is a binary linear code with generating set

$$M_{\mathcal{C} \to \mathbf{b}} = \{D_{\mathbf{b}} f_i | f_i \in M_{\mathcal{C}}\}. \tag{2.8}$$

By definition, $D_{\mathbf{b}}f_i$ has identical values at coordinates $\mathbf{x}$ and $\mathbf{x}+\mathbf{b}$ for all $\mathbf{x} \in \mathbb{F}_2^m$, so we can write its generator matrix as $G = \begin{pmatrix} G^{(\mathbf{b})} & G^{(\mathbf{b})} \end{pmatrix} P'$, where P' is a column permutation matrix and $G^{(\mathbf{b})}$ is a generator matrix of some $(n^{(\mathbf{b})} = 2^{m-1}, k^{(\mathbf{b})} = \dim M_{\mathcal{C}\to\mathbf{b}}, d^{(\mathbf{b})})$ code $\mathcal{C}^{(\mathbf{b})}$. The codes $\mathcal{C}^{(\mathbf{b})}$ are called the projected codes or the projections. In what follows, we also refer to the codes $\mathcal{C}^{(\mathbf{e}_i)}$ as the partial derivatives or the derivative codes of $\mathcal{C}$.

Observe that the function $f(\mathbf{x}+\mathbf{b})$ can be considered as the action of the element of $\mathcal{T}_m$ on f, which swaps the evaluation points $\mathbf{x}$ and $\mathbf{x}+\mathbf{b}$ (and hence the corresponding entries in $\mathrm{ev}(f)$). Therefore, if $\mathrm{Aut}(\mathcal{C})$ includes the permutation $\mathbf{x} \to \mathbf{x}+\mathbf{b}$, then $D_{\mathbf{b}}\mathcal{C}$ is a subcode of $\mathcal{C}$. Legeay proved that this also puts an upper bound on the dimension of the projected code:

Theorem 1 ([57], Corollary 1)**.** *Consider a binary linear code $\mathcal{C}$ and some permutation π of its codewords. Define the code*

$$\mathcal{C}_\pi = \{\mathbf{c} + \pi(\mathbf{c}) | \mathbf{c} \in \mathcal{C}\}.$$

If π is in the automorphism group of $\mathcal{C}$ and can be decomposed as a product of transpositions, then $\dim \mathcal{C}_\pi \leq \frac{1}{2}\dim \mathcal{C}$.

Remark. Rewriting Theorem 1 in terms of the code rate, we get

$$R(\mathcal{C}_\pi) \leq R(\mathcal{C}). \tag{2.9}$$

All codes considered in this thesis include $\mathcal{T}_m$ in their automorphism group and therefore (2.9) serves as a universal upper bound for the rates of the projected codes.

The partial derivatives of Boolean functions are inherently connected with the Plotkin construction. Namely, any function f can be decomposed as

$$f(x_0, \ldots, x_{m-1}) = g(x_1, \ldots, x_{m-1}) + x_0 h(x_1, \ldots, x_{m-1}), \tag{2.10}$$

where $g(x_1, \ldots, x_{m-1})$ takes identical values for $x_0 = 0$ and $x_0 = 1$, whereas $x_0 h(x_1, \ldots, x_{m-1})$ is nonzero only when $x_0 = 1$. It is straightforward to see that h is a partial derivative of f w.r.t. x_0. This decomposition can be performed w.r.t. any variable x_i.

Proposition 3. *Consider the code $\mathcal{C}$. If all functions $f \in M_{\mathcal{C}}$ can be decomposed in form* (2.10) *such that either $g \equiv 0$ or $h \equiv 0$, then $\mathcal{C}$ can be represented in $(u|u+v)$ form.*

Proof. By assumption the generating set of $\mathcal{C}$ contains only functions that either have identical values for $x_0 = 0$ and $x_0 = 1$ or are nonzero only when $x_0 = 1$. The former generate codewords of type $(u|u)$ and the latter generate codewords of type $(0|v)$, which concludes the proof. □

It follows immediately that any monomial code can be decomposed as $(u|u+v)$. Moreover, in case of Reed-Muller codes this decomposition again brings us Reed-Muller codes. Recall that the generating set of code $\mathrm{RM}(r,m)$ contains all monomials up to degree r. Plugging its elements into (2.10) allows us to conclude that the set of all possible functions h is given by $\mathrm{RM}(r-1,m-1)$ and the set of all possible functions g by $\mathrm{RM}(r,m-1)$ (with the only exception being the case $r=m$ when the code $\mathrm{RM}(m,m)$ is decomposed into two codes $\mathrm{RM}(m-1,m-1)$).

Remark. It might happen that Proposition 3 does not hold for the code of interest for all variables x_i. For example, eBCH codes in general do not admit $(u|u+v)$ representation. In such cases we can consider the *generalized Plotkin decomposition* (GPD), introduced in [31]. For any binary linear $(n=2^m,k)$ code there exist the following decomposition of its generator matrix:

$$\mathbf{G}=\begin{pmatrix}\mathbf{G}_0+\tilde{\mathbf{I}}\mathbf{G}_2 & \tilde{\mathbf{I}}\mathbf{G}_2\\ \mathbf{G}_1 & \mathbf{G}_1\end{pmatrix}, \tag{2.11}$$

where $\mathbf{G}_i$ are $k_i\times 2^{m-1}$ matrices, $k_0+k_1=k, k_2\leq k_0$ and $\tilde{\mathbf{I}}$ is obtained by stacking $(k_0-k_2)\times k_2$ zero matrix and $k_2\times k_2$ identity matrix. If $k_2=0$, we get the standard $(u|u+v)$ construction. In the framework of GPD, a partial derivative $\partial/\partial x_0$ corresponds to the code generated by $\mathbf{G}_0$, and therefore we can proceed as usual.

3 Symmetry-based Decoding of Reed-Muller Codes

Reed-Muller codes are among the oldest known error-correcting codes and among many notable properties, they are invariant under the action of large permutation group. For many years, there have been many successful attempts to utilize their automorphism group in order to achieve better decoding performance.

When trying to classify the main ideas behind various algorithms, it can be seen that they mostly fall into one of three main avenues. The first two correspond to running several successive cancellation or belief propagation decoders with different permutations of the input vector. The former was initially studied by Stolte [27] as well as Dumer and Shabunov [51], with further development for BEC in [40, 58, 59] and for AWGN channel in [41, 60, 61, 62]. The latter was first mentioned in [40], followed by a series of other works [42, 39, 38, 62, 37, 63]. The last avenue can be described as projecting the codeword on multiple lower-order codes and combine the results in order to obtain the solution to the decoding problem. The initial idea is due to Sidel'nikov and Pershakov [50], which was later improved in [64, 65] for the case of second-order codes. Lately Ye and Abbe unbeknownst of works of Sidel'nikov and Pershakov developed the recursive projection-aggregation (RPA) algorithm, which is based on similar principles [52].

The main practical advantages of the symmetry-based decoding paradigm lie in the large degree of parallelization and the simplicity of hardware implementation. All algorithms that are described in this chapter allow the independent processing of permutations or projections and consequently the most computationally demanding steps become simpler in the presence of parallel computing units. The decoding latency can be reduced as well.

This chapter is organized as follows. Section 3.1 begins with the successive cancellation list (SCL) decoder and the factor graph representation of the encoding and decoding process, which is then followed by the description of the permutation-based algorithm that operates in erasure channel and works more efficiently. The proposed algorithm is published in [59]. Section 3.2 describes two state-of-the-art projection-based decoders for the second-order Reed-Muller codes and presents the improvement for the former, which is published in [65].

3.1 Permutation-based List Decoding for Binary Erasure Channel

3.1.1 Successive Cancellation List Algorithm

Assume that a codeword $\mathbf{c}$ is transmitted through the BMS channel W and the received vector is $\mathbf{y}$. The successive cancellation (SC) algorithm [66] performs bit-by-bit estimation of the vector $\mathbf{u}$ s.t. $\mathbf{u}\mathbf{A}_m = \mathbf{c}$ as

$$\tilde{u}_i = \begin{cases} \arg\max_{u_i \in \{0,1\}} W_m^{(i)}(y_0^{n-1}, \tilde{u}_0^{i-1} | u_i), & i \notin \mathcal{F}, \\ 0 & i \in \mathcal{F}, \end{cases} \tag{3.1}$$

where $W_m^{(i)} = W^{(\{-,+\}^m)}$ is computed by the recursive application of channel transformations (2.4) and the initial values of the recursion are $W(y_i|u_i)$. In erasure channel, each bit of $\mathbf{u}$ is either perfectly decoded or erased. If any non-frozen bit is erased, a SC decoding failure is declared. The algorithm runs in $O(n \log n)$ time and needs $O(n)$ space.

The performance of successive cancellation algorithm in finite-length regime is poor since any single failure in the estimation of input bits ruins the decoding process. List decoding algorithm (SCL) [28] overcomes this issue by keeping L most likely continuations. If the current information bit is erased, the decoding path splits in two and the process goes further for both possible values. Paths with nonzero estimations of frozen bits are eliminated. It should be noted that any erasure event happens for all paths and therefore the list size always remains a power of 2. List decoding failure is declared if more than L paths need to be stored at some step. In case of channels with errors, one simply keeps L paths with the best reliability metric value (e.g., Hamming distance, ellipsoidal weight, etc.). The algorithm has time complexity $O(L \cdot n \log n)$ and occupies $O(L \cdot n)$ memory. Note that SCL algorithm is essentially the same as the recursive lists algorithm of Dumer and Shabunov [51] and has similar performance. However, for the sake of convenience with polar coding literature we will stick to SCL notation.

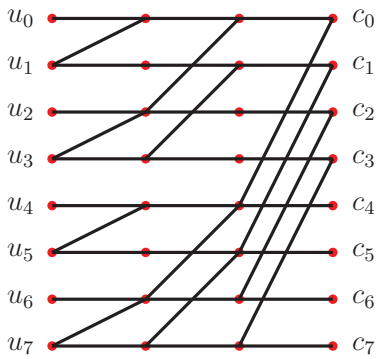


Figure 3.1 – Example of factor graph, $m = 3$

3.1.2 Factor Graph Permutations

The propagation of values during the encoding and decoding process can be represented with an m-layer factor graph [40], which gives a graphical way to represent the action of matrix $\mathbf{A}_m$. Example of this graph for $m = 3$ is presented at figure 3.1. Bit values are propagated left-to-right to form codeword bits c_i from input bits u_i, and probabilities are propagated right-to-left to form bit estimates $\tilde{u}_i$ from the received values y_i. Red dots represent the XOR nodes. A permutation π of the layers of such factor graph changes the order in which the bits are combined. The encoding result remains unchanged since the overall operation is still the multiplication by $\mathbf{A}_m$. However, the things are different when it comes to the successive cancellation decoding since the bit reliabilities depend on the order in which the bits of $\mathbf{u}$ are processed. Figure 3.2 illustrates the change. A permutation π of the factor graph layers corresponds to the change in the processing order of $\mathbf{u}$, defined by

$$u_i \to u_{\pi(i)}, \tag{3.2}$$

where we assume by the slight abuse of notation that $\pi(i)$ permutes bits in the binary representation of i. The standard SC decoding corresponds to $\pi^* = (0, 1, \ldots, m-1)$. Note that the same result can be achieved by either permuting vector $\mathbf{y}$ (and consequently $\mathbf{u}$) according to (3.2) or by changing the processing rules in the decoder. Namely, at layer l the channel transformation is given by combining the indices that differ only in bit π_l.

Denote by $P_e(i)$ the probability that SC decoder makes an erroneous estimation of bit u_i. Applying any layer permutation also permutes these probabilities, i.e., $P_e^{\pi}(i) = P_e(\pi(i))$. In case of polar codes this becomes a problem since the frozen symbols set in the permuted vector $\mathbf{u}$ may no longer contain the indices of subchannels with the largest SC error probabilities and therefore the decoding performance when using π is significantly worse compared to π^*. However, it is not an issue for Reed-Muller codes. The action of π can be expressed as $\mathbf{x} \to \mathbf{Px}$ where $\mathbf{P}$ is a permutation matrix and consequently the set of all layer permutations π coincides with the group $\mathcal{P}_m$ of variable

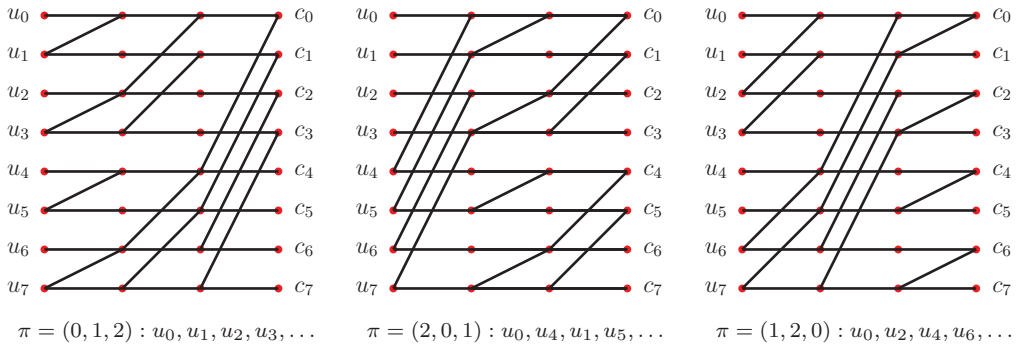


Figure 3.2 – Different factor graph permutations, $m = 3$

permutations, which is a subgroup of $GA(m, \mathbb{F}_2)$. Thus, Reed-Muller codes remain invariant under the factor graph layer permutations and therefore the successive cancellation decoding with any such permutation is expected to have the same error probability.

Remark. The statement regarding SC error probability only implies the similar average decoding performance. A particular noise pattern might be uncorrectable with one permutation and correctable with another and there are many such patterns, which is the reason why permutation decoding works well.

Example 1. *Consider the transmission of all-zero codeword of* $RM(1,3)$ *through the erasure channel and assume that the received vector is* $\mathbf{y} = (0, 0, \epsilon, \epsilon, \epsilon, 0, 0)$. *Recall that* $RM(1,3)$ *is a Plotkin concatenation of* $RM(0,2)$ *which is a* $(4,1,4)$ *repetition code and* $RM(1,2)$ *which is a* $(4,3,2)$ *single parity check code.*

If we take $\pi = \{0,1,2\}$, *then we get* $\mathbf{y}^{(-)} = (y_0+y_4, y_1+y_5, y_2+y_6, y_3+y_7) = (\epsilon, \epsilon, \epsilon, \epsilon)$. *This pattern is uncorrectable in* $RM(0,2)$ *and SC decoding fails. On the other hand,* $\pi = \{2,1,0\}$ *gives* $\mathbf{y}^{(-)} = (y_0 + y_1, y_2 + y_3, y_4 + y_5, y_6 + y_7) = (0, \epsilon, \epsilon, 0)$, *which can be corrected in* $RM(0,2)$ *and SC decoding succeeds.*

3.1.3 Proposed Algorithm

In this section, we present a novel decoding algorithm for Reed-Muller codes when the transmission takes place through binary erasure channel. The algorithm is based on factor graph permutations and utilized the idea similar to the one described in [42]. We pre-select P distinct factor graph permutations $\pi^{(j)}, j \in [P]$. Given the received vector $\mathbf{y}$, we generate P its permuted copies $\mathbf{y}^{(j)}, y_i^{(j)} = \pi^{(j)}(i)$, and use them to initialize P SCL decoders $\mathcal{D}_j$ with list size L. The choice of $\pi^{(j)}$ and the impact of P on the decoding process is discussed in section 3.1.5. Since in erasure channel any decoded symbol $\tilde{u}_i$ is correct, these symbols can be shared among all $\mathcal{D}_j$.

The decoding is performed in an iterative fashion. At each iteration, we

try to find a decoder $\mathcal{D}_j$ that can estimate some unknown symbol $\tilde{u}_i$. The straightforward way is to simply go over all decoders. Due to the different bit estimation orders, it may happen that symbol currently estimated by some $\mathcal{D}_j$ is already known. We treat this symbol as frozen with the corresponding value. Decoding stops when all entries of vector $\tilde{\mathbf{u}}$ are successfully obtained, and the corresponding codeword is returned. A failure is declared if no decoder $\mathcal{D}_j$ can go further.

A pseudocode of the described permutation-based successive cancellation list (PSCL) decoder is presented at algorithm 1. Subroutine `Initialize` is used to initialize a SCL decoder copy with the permutation of the received vector, i.e., sets the initial values for the recursion (2.4) as $W(y_i^{(j)}|0) = W(y_i^{(j)}|1) = 0.5$ if $y_i^{(j)}$ is erased and $W(y_i^{(j)}|y_i^{(j)}) = 1, W(y_i^{(j)}|1-y_i^{(j)}) = 0$ otherwise. Subroutine `GetNextSymbol` performs the standard SCL decoding as described in [28] until some new symbol $\tilde{u}_i$ is decoded or a list decoding failure is declared. In the latter case, we say that the decoder is *stuck*. The obtained value (or the erasure symbol ϵ if the decoder is stuck) is returned to the main processing cycle.

Algorithm 1 Permutation-based list decoder

```
1:  procedure PSCL(y, F, {π^(j)}, {D_j}, L, P)
2:      for j ← 0 to P − 1 do
3:          for i ← 0 to n − 1 do
4:              y_i^(j) ← y_{π^(j)(i)}
5:          D_j.Initialize(L, y^(j), F)
6:      for all ũ_i do
7:          ũ_i ← ϵ
8:      repeat
9:          for j ← 0 to P − 1 do
10:             u_{i_j} ← D_j.GetNextSymbol()
11:             if u_{i_j} ≠ ϵ then
12:                 ũ_{i_j} ← u_{i_j}
13:                 break
14:         if all D_j returned ϵ then
15:             return Failure
16:     until all ũ_i are decoded
17:     return ũA_m
```

The algorithm runs in $O(LP \cdot n \log n)$ time and needs $O(LP \cdot n)$ memory. However, for many erasure patterns only a fraction of decoders $\mathcal{D}_j$ is actually used, so the actual number of iterations might be significantly less than LP.

The performance of the proposed algorithm can be further improved. A position i is *unresolved* if one can find two paths $u^{(l)}$ and $u^{(l')}$ such that $u_i^{(l)} \neq u_i^{(l')}$. Let us keep for any decoder $\mathcal{D}$ its set of unresolved positions $\mathcal{J}$. It

may happen that some of these symbols become resolved by another decoder $\mathcal{D}'$ at some point. Hence, if $\mathcal{D}$ is stuck, it can go over entries of $\mathcal{J}$ and check if some symbols are actually resolved. In such occasion, they are removed from $\mathcal{J}$ and the corresponding paths are eliminated, thus allowing the decoder to take another step forward. Moreover, path elimination may resolve some other symbols as well.

In order to implement this feature, for any unresolved position i, we keep two sets

$$U_i^{(0)} = \{l | u_i^{(l)} = 0\},$$
$$U_i^{(1)} = \{l | u_i^{(l)} = 1\}.$$

The path cloning and killing procedures from [28] need to be modified accordingly, as demonstrated in algorithms 2 and 3. Algorithm 4 implements the resolution procedure, which is called whenever the decoder is stuck.

Algorithm 2 Kill path and check potential symbol resolutions

```
1: procedure KillPathChk(l)
2:     KillPath(l)
3:     for i ∈ J do
4:         if l ∈ U_i^(0) then
5:             U_i^(0).Remove(l)
6:             if |U_i^(0)| = 0 then
7:                 ũ_i ← 1
8:         else
9:             U_i^(1).Remove(l)
10:            if |U_i^(1)| = 0 then
11:                ũ_i ← 0
12:        if ũ_i ≠ ϵ then
13:            J.Remove(i)
```

Algorithm 3 Clone path

```
1: procedure ClonePathChk(l)
2:     l' ←ClonePath(l)
3:     for i ∈ J do
4:         if l ∈ U_i^(0) then
5:             U_i^(0).Add(l')
6:         else
7:             U_i^(1).Add(l')
```

Consider now the decoding failure event. No decoder $\mathcal{D}$ can go further and any unknown symbol $\tilde{u}_i$ is either unresolved for $\mathcal{D}$ or it is stuck before reaching it. We can conclude that we cannot do better in the framework of SCL decoding with the current set of permutations.

Algorithm 4 Check symbol resolutions

```
1: procedure CheckResolutions()
2:     for i ∈ 𝒥 do
3:         if ũ_i ≠ ϵ then
4:             𝒥.Remove(i)
5:             for l ∈ U_i^(ũ_i) do
6:                 KillPathChk(l)
```

3.1.4 Simplified Version

The running time of the algorithm described in the previous section can be significantly reduced. During the simulations we observe that the main loop of algorithm 1 does many redundant operations since many decoders do not contribute to the decoding process, i.e., they recover no symbols $\tilde{u}_i$.

It turns out that if we prohibit going backwards in the main loop, we obtain a great decrease in the average decoding time. More precisely, we start the decoding process with $\mathcal{D}_0$ and proceed until it gets stuck or succeeds. If it is stuck, we keep the decoded symbols and move to $\mathcal{D}_1$, then to $\mathcal{D}_2, \ldots, \mathcal{D}_{P-1}$ until all entries of vector $\tilde{\mathbf{u}}$ are successfully obtained. A decoding failure is declared if some symbols still remain unknown.

It is easy to notice that the complexity of the simplified algorithm in terms of the big O notation remains identical to the one from the previous section. However, our experiments showed more than 10x speedup in the software simulations with negligible performance degradation, so we will present numeric results only for the simplified version.

3.1.5 Performance

Achievability of MAP decoding

The proposed PSCL decoder can be considered as an ensemble of SCL decoders working together and we know that it is possible to reach (near-)MAP performance with list decoding for sufficiently big list size. We also know that permutations can improve the overall performance since for a particular permutation one can find an erasure pattern that is uncorrectable with this permutation but is correctable with some other [58]. On the other hand, the erasure events for different permutations correlate with each other. The theoretic analysis of these correlations goes beyond the scope of this work, yet we can present some numeric results. It should be noted that Kamenev in [67] proposed a method to estimate the failure probability of permutation decoding. However, this method only works for $L = 1$ and sufficiently small values of P, i.e., when the correlations between different permutations can be neglected under the random selection strategy.

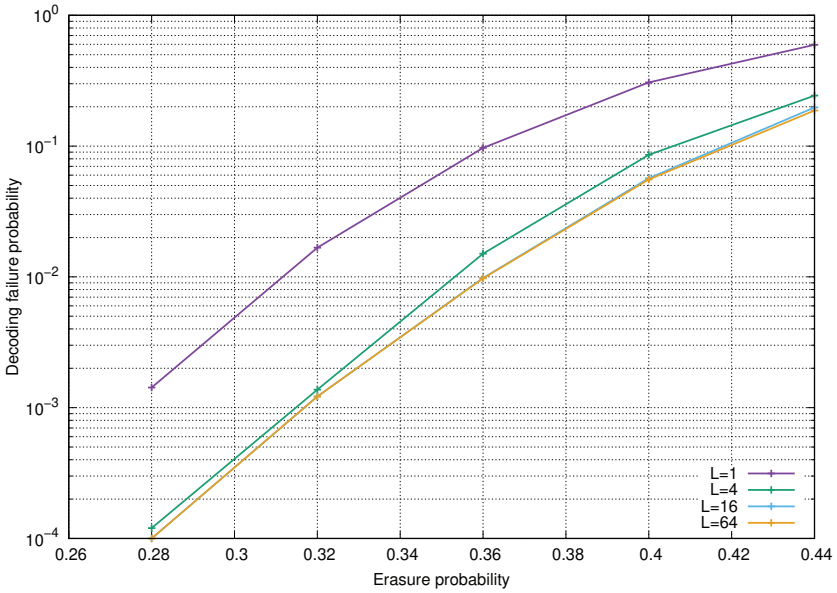


Figure 3.3 – RM$(3, 7)$ with all 7! permutations

Figure 3.3 demonstrates the performance of proposed algorithm for different values of L when all $7! = 5040$ permutations are used. For a small L, even using all permutations is not sufficient to get close to MAP decoding. However, when the list size is increased it is possible to reach MAP performance. For code RM$(3, 7)$, permutation-based decoding achieves near-MAP performance with $L = 16$, whereas pure SCL decoding needs at least $L = 256$.

Selection strategies

The running time and memory consumption of our algorithm for the fixed list size L depend on the number of permutations P. Therefore, one needs to develop a rule on how to choose P permutations so that the performance is maximized. It can be easily shown that taking P first permutations in the lexicographic order is a bad strategy. Indeed, if two permutations π and π' have common length-l suffix, the corresponding decoders have identical estimations of first 2^l symbols of vector $\tilde{\mathbf{u}}$ and therefore if the first decoder is stuck while processing these symbols, the second gets stuck as well. Thus, it is reasonable to select permutations that are sufficiently 'different' from each other to maximize the number of correctable erasure patterns. In the literature, we can find the following:

- Cyclic shifts of factor graph layers [40];
- Permutations with distinct first r factor graph layers [51];
- Random permutations [39].

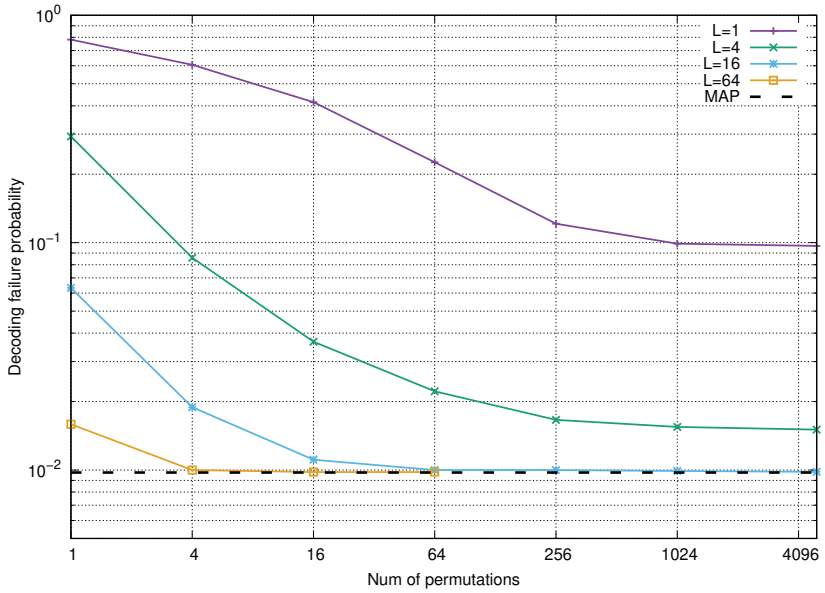


Figure 3.4 – RM$(3, 7)$, $\epsilon = 0.36$

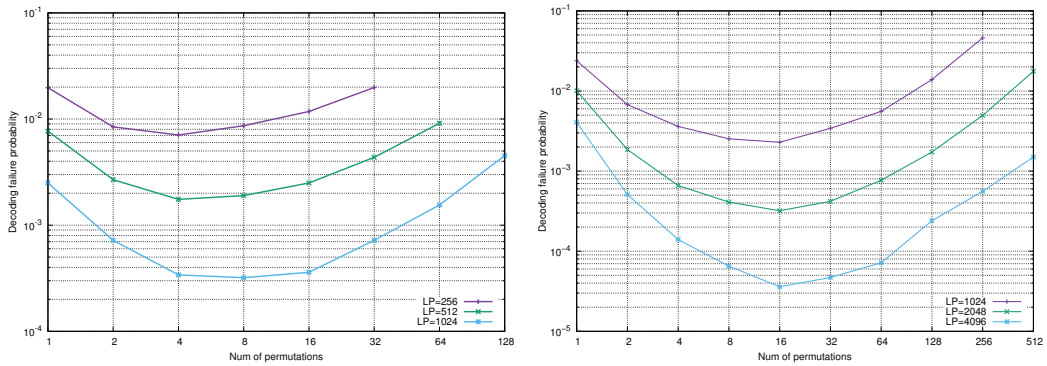
Our experiments showed that all these strategies provide comparable performance for the same number of permutations. In the literature, some good heuristic strategies for AWGN channel are proposed in [41] and [42]. However, our attempts to develop similar ideas for the erasure channel were unsuccessful and hence we resort to the random selection rule.

Figure 3.4 shows how the performance improves with the number of random permutations being used. Different values of the list size L are considered and the erasure probability ϵ is selected so that the probability of MAP decoding failure is around $p^* = 0.01$.

It can be seen that even the small number P of random permutations significantly improves the performance for any fixed list size. However, correlation effects between different permutations start to play more crucial role when P grows. This leads to the performance saturation effect that can be seen at the figure. The saturation threshold goes down with the list size until the MAP performance is reached.

L-P tradeoff

In previous sections we demonstrated that for a fixed list size it is possible to improve the performance with permutations. Now we show how our algorithm performs in comparison to the conventional SCL decoder given the same asymptotic complexity. Since for the given code length the product $\lambda = LP$ defines the time and space complexity of the algorithm, it is reasonable to define (L, P)-PSCL and explore its behavior for the fixed λ.



(a) RM$(4, 9)$, $\epsilon = 0.28$ (b) RM$(5, 11)$, $\epsilon = 0.19$

Figure 3.5 – Performance of (L, P)-PSCL

Figure 3.5 shows how the performance of the proposed algorithm changes with the number of random permutations. Standard SCL decoding, which can be considered as $(\lambda, 1)$-PSCL, corresponds to the leftmost points (with $P = 1$) of the curves. Erasure probability ϵ is selected so that SCL decoding performance of the upper curves is approximately $p^* = 0.02$.

It is clear that our algorithm provides the significant performance gain for different code lengths, which only grows when λ is increased. Whereas for code RM$(4, 9)$ and $\lambda = 256$, the algorithm performs similarly to SCL with $L = 512$, but for code RM$(5, 11)$ and $\lambda = 4096$ we get the improvement around two orders of magnitude and SCL even with list size $L = 65536$ cannot achieve such performance. However, it can be seen that value P needs to be properly adjusted in order to obtain such a substantial improvement and it is unclear how to find the optimum value. In the beginning, when P is increased the gain from combining multiple permutations is rather high. After a certain point we reach the saturation area for small values of L and the overall performance drops down.

Note that $(1, \lambda)$-PSCL demonstrates substantially worse error correction capability compared to $(\lambda, 1)$-PSCL, which is a stark contrast to the AWGN channel, where the permutation-based algorithms without any list are comparable with the conventional SCL decoder or even outperform it [60, 62]. We attribute this difference to the fact that in BEC the list size can also go down by a factor of 2. Namely, when the decoder arrives at the frozen symbol and $W_m^{(i)}(y_0^{n-1}, \tilde{u}_0^{i-1}|u_i) \neq 0.5$, half of the paths has $W_m^{(i)}(y_0^{n-1}, \tilde{u}_0^{i-1}|0) = 1$ and another half has $W_m^{(i)}(y_0^{n-1}, \tilde{u}_0^{i-1}|0) = 0$, so we can safely discard the latter. Another interesting conclusion from the figures is the existence of a single value P which is the best option for all list sizes under the fixed λ.

3.2 Improved Decoding of Second-Order Reed-Muller Codes

The second-order Reed-Muller code $\mathrm{RM}(2, m)$ has generating set

$$M_{2,m} = \{1\} \cup \{x_i | 0 \leq i < m\} \cup \{x_i x_j | 0 \leq i < j < m\},$$

and therefore any codeword can be considered as an evaluation of polynomial

$$f_2(\mathbf{x}) = \sum_{0 \leq i < j < m} B_{i,j} x_i x_j + \sum_{0 \leq i < m} b_i x_i + b,$$

where $B_{i,j}, b_i, b \in \mathbb{F}_2$ are the information symbols. A directional derivative of f_2 can be expressed as

$$D_{\boldsymbol{\alpha}} f_2 = f_2(\mathbf{x}) + f_2(\mathbf{x} + \boldsymbol{\alpha}) = \sum_{i<j} B_{i,j} \alpha_i \alpha_j + \sum_i b_i \alpha_i + \sum_{i<j} B_{i,j} (\alpha_i x_j + \alpha_j x_i) \tag{3.3}$$

$$= b^{(\boldsymbol{\alpha})} + \boldsymbol{\alpha}^T \mathbf{B} \mathbf{x}, \tag{3.4}$$

which is an affine polynomial and consequently is a codeword of $\mathrm{RM}(1, m)$. In fact, from Section 2.4.2 we know that $D_{\boldsymbol{\alpha}} f_2$ consists of two identical copies of $\mathrm{RM}(1, m-1)$ which is used in the practical implementations of the decoder to speed up the computations. There exists an efficient algorithm, which solves the maximum likelihood decoding problem for the first-order codes [68], and it is tempting to utilize it in order to decode the second-order codes.

In this section, we study the efficient algorithms for decoding of $RM(2, m)$ that make use of the projected codes $\mathcal{C}^{(\boldsymbol{\alpha})}$ and run in $O(n^2 \log n)$ time. The first approach is focused on the recovery of information bits and was initially proposed more than 20 years ago by Sidel'nikov and Pershakov [50] with the subsequent improvement by Sakkour in mid-2000s [69]. The second is focused on the recovery of codeword bits and was recently proposed by Min Ye and Emmanuel Abbé [52]. We begin by presenting both algorithms and then propose the improvement of the former.

3.2.1 Optimal Decoding of First-Order Reed-Muller Codes

The first-order Reed-Muller code $\mathrm{RM}(1, m)$ has dimension $m + 1$, minimum distance 2^{m-1} and its generating set is given by

$$M_{1,m} = \{1\} \cup \{x_i | 0 \leq i < m\},$$

i.e., any its codeword is as an evaluation of polynomial

$$f_1(\mathbf{x}) = \sum_{0 \leq i < m} g_i x_i + g,$$

where $g_i, g \in \mathbb{F}_2$ are the information symbols.

Consider $2^m \times 2^m$ Hadamard matrix

$$\mathbf{H}^{(m)} = \begin{pmatrix} \mathbf{H}^{(m-1)} & \mathbf{H}^{(m-1)} \\ \mathbf{H}^{(m-1)} & -\mathbf{H}^{(m-1)} \end{pmatrix}, \mathbf{H}^{(1)} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

and define the mapping

$$\eta(x) = \begin{cases} 0, x \geq 0 \\ 1, x < 0. \end{cases} \tag{3.5}$$

Applying η to the rows of $\mathbf{H}^{(m)}$ gives us half of the codewords of $\mathrm{RM}(1, m)$, and the other half can be obtained by taking complements [56]. Therefore, given the received LLR vector $\mathbf{y}$, the maximum likelihood solution of the decoding problem is given by the row of $\mathbf{H}^{(m)}$ with the largest absolute value of the correlation $\rho(\mathbf{y}, \mathbf{h})$, which is defined as

$$\rho(\mathbf{y}, \mathbf{h}) = \sum_i y_i h_i.$$

Fast Hadamard transform (FHT) allows to compute the correlations of $\mathbf{y}$ with all rows of $\mathbf{H}^{(m)}$ in $O(n \log n)$ time as shown in Algorithm 5. Entries of the returned vector $\mathbf{v}$ satisfy $v_i = \rho(\mathbf{y}, \mathbf{H}^{(m)}_{i,*})$. It remains to obtain $\hat{i} = \arg\max_i |v_i|$ and return the corresponding codeword $\hat{\mathbf{c}} = \eta(\mathbf{H}^{(m)}_{\hat{i},*})$ in order to complete the ML decoding.

Algorithm 5 Fast Hadamard trasform

```
1: procedure FHT(y)
2:     v ← y
3:     for i ← 0 to m − 1 do
4:         for j ← 0 to n/2 do
5:             s_j ← v_{2j} + v_{2j+1}
6:             s_{j+n/2} ← v_{2j} − v_{2j+1}
7:         v ← s
8:     return v
```

Remark. The modification of this procedure to return $l > 1$ most likely codewords is straightforward.

3.2.2 Sidel'nikov-Pershakov Decoder

Assume that the received LLR vector $\mathbf{y}$ corresponds to the noisy codeword of the second-order Reed-Muller code $RM(2, m)$. Sidel'nikov and Pershakov proposed the decoding algorithm, which is based on taking all possible derivatives $D_{\boldsymbol{\alpha}}$ and

applying the FHT-based decoder from the previous section for the first-order codes [50]. The general procedure is summarized in Algorithm 6 and all steps are explained below. First, for each nonzero $\boldsymbol{\alpha}$ the LLR vector

$$\mathbf{y}^{(\boldsymbol{\alpha})} = (y_{\boldsymbol{\alpha}_0} \boxplus y_{\boldsymbol{\alpha}_0+\boldsymbol{\alpha}}, \ldots, y_{\boldsymbol{\alpha}_{2^m-1}} \boxplus y_{\boldsymbol{\alpha}_{2^m-1}+\boldsymbol{\alpha}})$$

that corresponds to the derivative $D_{\boldsymbol{\alpha}}$ is computed, where $\boxplus$ denotes a box-plus operator

$$y_a \boxplus y_b = \log \frac{1+\exp(y_a+y_b)}{\exp(y_a)+\exp(y_b)}$$

[70] or its min-sum approximation

$$y_a \boxplus y_b \approx \operatorname{sgn}(y_a)\operatorname{sgn}(y_b)\min(|y_a|,|y_b|).$$

Remark. The decoder was originally proposed for the binary symmetric channel, whereas in this chapter we consider the AWGN channel. The only difference is in how we compute the elements of $\mathbf{y}^{(\boldsymbol{\alpha})}$.

Algorithm 6 Sidel'nikov-Pershakov decoder

procedure SP2($\mathbf{y}$)
  for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
    for $\boldsymbol{\beta} \in \mathbb{F}_2^m$ **do**
      $y_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} \leftarrow y_{\boldsymbol{\beta}} \boxplus y_{\boldsymbol{\beta}+\boldsymbol{\alpha}}$
    $(v^{(\boldsymbol{\alpha})}, \mathbf{g}^{(\boldsymbol{\alpha})}) \leftarrow$RM1Decode($\mathbf{y}^{(\boldsymbol{\alpha})}$)
  for $i \leftarrow 0$ to $m-1$ **do**
    for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      $t_{\boldsymbol{\alpha}}^{(i)} \leftarrow (-1)^{g_i^{(\boldsymbol{\alpha})}} v^{(\boldsymbol{\alpha})}$
    $(z_i, \mathbf{B}_{i,*}) \leftarrow$RM1SubcodeDecode($i, \mathbf{t}^{(i)}$)
  for $i \leftarrow 0$ to $m-1$ **do**
    for $j > i$ **do**
      if $z_i > z_j$ **then**
        $B_{i,j} \leftarrow B_{j,i}$
  $\hat{\mathbf{c}} \leftarrow \operatorname{ev}(\sum_{i<j} B_{i,j} x_i x_j)$
  for $i \leftarrow 0$ to 2^m-1 **do**
    $\hat{y}_i \leftarrow (-1)^{\hat{c}_i} y_i$
  $(v, \mathbf{g}) \leftarrow$RM1Decode($\hat{\mathbf{y}}$)
  return $\mathbf{B}, \mathbf{g}$

The projected codes are $\mathrm{RM}(1,m)$, so we use the maximum likelihood decoding algorithm from Section 3.2.1. Assume that the information vector, corresponding to the most likely codeword, is given by $(g^{(\boldsymbol{\alpha})}, g_0^{(\boldsymbol{\alpha})}, \ldots, g_{m-1}^{(\boldsymbol{\alpha})})$, and $v^{(\boldsymbol{\alpha})}$ denotes its correlation. Let us also denote by $\mathbf{g}^{(\boldsymbol{\alpha})}$ the vector $(g_0^{(\boldsymbol{\alpha})}, \ldots, g_{m-1}^{(\boldsymbol{\alpha})})$

of information symbols that correspond to the non-constant monomials. It follows from (3.3) that for every $\boldsymbol{\alpha}$ we have $\mathbf{g}^{(\boldsymbol{\alpha})} = \boldsymbol{\alpha}^T\mathbf{B}$.

We will proceed with the recovery of matrix $\mathbf{B}$ in a column-wise fashion. The entry $g_i^{(\boldsymbol{\alpha})} = \boldsymbol{\alpha}^T\mathbf{B}_{*,i}$ can be considered as the value of polynomial $f_i(\mathbf{x}) = \mathbf{x}^T\mathbf{B}_{*,i}$ at point $\boldsymbol{\alpha}$ and consequently the whole vector $(\mathbf{g}_i^{(\boldsymbol{\alpha}_0)}, \ldots, \mathbf{g}_i^{(\boldsymbol{\alpha}_{2^m-1})})$ can be viewed as the evaluation vector of f_i. It is a linear polynomial and therefore is a subcode of $\mathrm{RM}(1, m)$, so we can efficiently perform its ML decoding to recover the column $\mathbf{B}_{*,i}$. Value $v^{(\boldsymbol{\alpha})}$ is considered as the reliability of element $\mathbf{g}_i^{(\boldsymbol{\alpha})}$. Let us denote the correlation of the corresponding codeword as z_i. We know that the true matrix $\mathbf{B}$ is symmetric and therefore for each pair (i, j) the entries $B_{i,j}$ and $B_{j,i}$ should have identical values. We pick the value that corresponds to the largest value between z_i and z_j.

After second-order coefficients are obtained, we subtract their impact from vector $\mathbf{y}$, which is done by computing the evaluation vector of polynomial $f_{\mathbf{B}}(\mathbf{x}) = \sum_{i<j} b_{i,j}x_ix_j$ and changing the signs of elements $\mathbf{y}$ accordingly, and then proceed to decoding first-order code to get the remaining data symbols.

Proposition 4. *Algorithm 6 runs in $O(n^2 \log n)$ time.*

Proof. Subroutines `RM1Decode` and `RM1SubcodeDecode` run in $O(n \log n)$ time. The former is called $n - 1$ times in lines 2-5 and the latter is called $m = \log n$ times in lines 6-9. The complexity of the subsequent lines is dominated by line 17, which is another call to `RM1Decode`. Therefore, the overall complexity is $O((n - 1 + \log n + 1)n \log n) = O(n^2 \log n)$. □

Sidel'nikov and Pershakov also proposed an improvement to Algorithm 6 in the same paper. Observe that for all nonzero $\boldsymbol{\alpha} \neq \boldsymbol{\beta}$ holds the relation

$$\boldsymbol{\beta}^T\mathbf{B} + (\boldsymbol{\alpha}^T + \boldsymbol{\beta}^T)\mathbf{B} = \boldsymbol{\alpha}^T\mathbf{B}, \tag{3.6}$$

and therefore $\mathbf{g}^{(\boldsymbol{\alpha})}$ should be equal to $\mathbf{g}^{(\boldsymbol{\beta})} + \mathbf{g}^{(\boldsymbol{\alpha}+\boldsymbol{\beta})}$. Authors modify the procedure `RM1Decode` to return s most likely pairs $(v^{(\boldsymbol{\alpha}),k}, \mathbf{g}^{(\boldsymbol{\alpha}),k}), 0 \leq k < s$ instead of the single ML solution in line 5. Finally, they propose the heuristic procedure, which is repeated h times before the line 6 of Algorithm 6. This procedure is presented below.

The purpose of these computations is to increase the reliability of the correct values and decrease the probability of the incorrect ones. It is straightforward to verify that it takes $O(hn^2s^3)$ steps and therefore the overall decoder complexity is increased to $O(n^2(\log n + hs^3))$.

for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
  for $k \leftarrow 0$ to $s-1$ **do**
    $\tilde{v}^{(\boldsymbol{\alpha}),k} \leftarrow 0$
for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
  for $\boldsymbol{\beta} \in \mathbb{F}_2^m \setminus \{0, \boldsymbol{\alpha}\}$ **do**
    for $k \leftarrow 0$ to $s-1$ **do**
      $\hat{v}^{(\boldsymbol{\beta}),k} \leftarrow 0$
    $\phi \leftarrow 0$
    for $0 \le k_0, k_1, k_2 < s$ **do**
      if $\mathbf{g}^{(\boldsymbol{\alpha}),k_0} = \mathbf{g}^{(\boldsymbol{\beta}),k_1} + \mathbf{g}^{(\boldsymbol{\alpha}+\boldsymbol{\beta}),k_2}$ **then**
        $\hat{v}^{(\boldsymbol{\alpha}),k_0} \leftarrow \max(\hat{v}^{(\boldsymbol{\alpha}),k_0}, (v^{(\alpha),k_0} + v^{(\boldsymbol{\beta}),k_1} + v^{(\boldsymbol{\alpha}+\boldsymbol{\beta}),k_2})/3)$
        $\phi \leftarrow 1$
    if $\phi = 0$ **then**
      $\hat{v}^{(\boldsymbol{\beta}),k_0} \leftarrow v^{(\boldsymbol{\beta}),k_0}/2$
  $\tilde{v}^{(\boldsymbol{\alpha}),k} \leftarrow \frac{1}{n-2} \sum_{\boldsymbol{\beta}} \hat{v}^{(\boldsymbol{\beta}),k}$
$v^{(\boldsymbol{\alpha}),k} \leftarrow \tilde{v}^{(\boldsymbol{\alpha}),k}$ for $\boldsymbol{\alpha} \in \mathbb{F}_2^m, 0 \le k < s$

3.2.3 Loidreau-Sakkour's Improvement

The key point of Algorithm 6 is the recovery of matrix $\mathbf{B}$, which crucially depends on the number of correct values $\mathbf{g}^{(\boldsymbol{\alpha})}$. In order to increase this number, authors of the original paper proposed to exploit the relation (3.6) as described in the previous section. Loidreau and Sakkour in [64] proposed a simpler procedure, which also turns out to be more efficient in terms of error correction performance. Namely, the idea is to replace the values $\mathbf{g}^{(\boldsymbol{\alpha})}$ by their majority estimates, i.e., set

$$\tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} = \underset{\boldsymbol{\beta} \in \mathbb{F}_2^m}{\mathrm{Maj}} (\mathbf{g}^{(\boldsymbol{\beta})} + \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha})}), \tag{3.7}$$

where Maj is the function that returns the most frequent value, and proceed with $\mathbf{g} = \tilde{\mathbf{g}}$ in the subsequent steps. Note that the majority voting has complexity $O(n^2)$ and therefore the overall running time of the algorithm is not affected. Numeric results from [69, 71] demonstrate that the Loidreau-Sakkour's SPM algorithm outperforms the Sidel'nikov-Pershakov decoder even in its improved version.

3.2.4 Recursive Projection-Aggregation Decoder

Recursive projection-aggregation (RPA) decoder, which utilizes the similar ideas as the one of Sidel'nikov and Pershakov but is conceptually simpler was recently proposed by Ye and Abbe [52]. Algorithm 7 presents its pseudocode for the decoding of second-order Reed-Muller codes in AWGN channel. The initial step is similar, i.e., for all nonzero $\boldsymbol{\alpha}$ the LLR vector $\mathbf{y}^{(\boldsymbol{\alpha})}$ of the corresponding derivative is computed and the FHT-based ML decoder is used to recover the codeword $\mathbf{c}^{(\boldsymbol{\alpha})}$ of the first-order code. However, whereas Sidel'nikov-Pershakov

Algorithm 7 Projection-aggregation decoder for RM$(2, m)$ in AWGN channel

procedure RPA2($\mathbf{y}, N_{it}, \theta$)
  for $i \leftarrow 0$ to $N_{it} - 1$ **do**
    $\mathbf{v} \leftarrow \mathbf{0}$
    for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      for $\boldsymbol{\beta} \in \mathbb{F}_2^m$ **do**
        $y_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} \leftarrow y_{\boldsymbol{\beta}} \boxplus y_{\boldsymbol{\beta}+\boldsymbol{\alpha}}$
      $\mathbf{c}^{(\boldsymbol{\alpha})} \leftarrow$RM1Decode($\mathbf{y}^{(\boldsymbol{\alpha})}$)
      for $\boldsymbol{\beta} \in \mathbb{F}_2^m$ **do**
        if $c_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} = 0$ **then**
          $v_{\boldsymbol{\beta}} \leftarrow v_{\boldsymbol{\beta}} + y_{\boldsymbol{\alpha}+\boldsymbol{\beta}}$
        else
          $v_{\boldsymbol{\beta}} \leftarrow v_{\boldsymbol{\beta}} - y_{\boldsymbol{\alpha}+\boldsymbol{\beta}}$
    $\tilde{\mathbf{y}} \leftarrow \mathbf{y}$
    for $\boldsymbol{\beta} \in \mathbb{F}_2^m$ **do**
      $v_{\boldsymbol{\beta}} \leftarrow \frac{v_{\boldsymbol{\beta}}}{n-1}$
      if $|y_{\boldsymbol{\beta}} - v_{\boldsymbol{\beta}}| > \theta |y_{\boldsymbol{\beta}}|$ **then**
        $\tilde{y}_{\boldsymbol{\beta}} \leftarrow v_{\boldsymbol{\beta}}$
    if $\tilde{\mathbf{y}} = \mathbf{y}$ **then**
      break
    else
      $\mathbf{y} \leftarrow \tilde{\mathbf{y}}$
  return $(\eta(y_0), \ldots, \eta(y_{n-1}))$

algorithm tries to recover the information symbols, RPA targets at the codeword symbols. The LLR vector corresponding to the codeword of RM$(2, m)$ is estimated as

$$v_{\boldsymbol{\alpha}} = \frac{1}{n-1} \sum_{\boldsymbol{\beta} \neq \boldsymbol{\alpha}} (-1)^{c_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})}} y_{\boldsymbol{\alpha}+\boldsymbol{\beta}},$$

and the coordinate $y_{\boldsymbol{\alpha}}$ is updated if the relative change in magnitude of $v_{\boldsymbol{\alpha}}$ w.r.t. $y_{\boldsymbol{\alpha}}$ is at least θ. These steps are repeated iteratively until a stable point is reached or the maximum number of iterations N_{it} is exceeded. According to [52], setting $N_{it} = \lceil \frac{m}{2} \rceil$ and $\theta = 0.05$ is sufficient to achieve good performance in most cases. The running time of RPA is $O(N_{it} n^2 \log n) \approx O(n^2 \log n)$.

3.2.5 Proposed Improvement

Efficiency of majority voting

Let us take a closer look at the majority voting procedure. Consider the transmission of all-zero codeword and assume that for each directional derivative $\mathbf{y}^{(\boldsymbol{\alpha})}$, its s most likely values $\mathbf{g}^{(\boldsymbol{\alpha}),j}$ are returned. The expected number of votes

for the correct value $\tilde{\mathbf{g}}^{(\boldsymbol{\alpha})}$ can be expressed as

$$\begin{aligned}\mathbb{E}[\#\text{ votes for } \tilde{\mathbf{g}}^{(\boldsymbol{\alpha})}] = \\ \mathbb{E}[\#\text{ votes for } \tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} | \mathbf{g}^{(\boldsymbol{\beta})} = \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha})} = 0] \\ +\mathbb{E}[\#\text{ votes for } \tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} | \mathbf{g}^{(\boldsymbol{\beta})} = \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha})} \neq 0].\end{aligned}$$

Assume that there are d correct entries among the returned $s(n-1)$. The first term on the r.h.s. is equal to $\frac{\binom{d}{2}}{n-1}$. The second term, which we will further denotes as *collision votes*, can be estimated using the following proposition:

Proposition 5. *Assume that the incorrect values of* $\mathbf{g}^{(\boldsymbol{\alpha})}$ *are uniformly distributed. Then*

$$\mathbb{E}[\#\ \textit{votes for}\ \tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} | \mathbf{g}^{(\boldsymbol{\beta})} = \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha})} \neq 0] \approx O(s^2).$$

Proof. From the uniformity assumption it follows that for any nonzero $\mathbf{g}'$ the expected number of elements $\mathbf{g}^{(\boldsymbol{\beta})} = \mathbf{g}'$ is $\frac{s(n-1)-d}{n-1}$. Thus, we get

$$\mathbb{E}[\#\text{ votes for } \tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} | \mathbf{g}^{(\boldsymbol{\beta})} = \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha})} \neq 0] = \binom{\frac{s(n-1)-d}{n-1}}{2} \approx \left(s - \frac{d}{n-1}\right)^2 \approx O(s^2).$$

□

Sakkour in [71] states that the expected maximum number of votes for the incorrect value for the case $s = 1$ is $\frac{\log n}{\log \log n}$. We can expect that with $s > 1$ it will grow as $O(s\frac{\log n}{\log \log n})$ under the uniformity assumption.

Remark. The statements above are made under the assumption of the uniformity of the returned values $\mathbf{g}^{(\boldsymbol{\alpha})}$ and their independence from each other. Strictly speaking, both assumptions do not hold in reality. The derivatives clearly correlate with each other and there are no guarantees that the results of their (list) decoding is uniform.

Let us now demonstrate what actually happens in the decoding process. Figure 3.6 demonstrates how the number of collision votes for the correct value changes with d for different values of s for RM$(2, 8)$ and $E_b/N_0 = 1$dB. Although it is clear that its distribution of collision votes is rather far from uniform and has a larger tail towards the higher number of votes, its mean scales quadratically with s. Figures 3.7-3.8 demonstrate how the total number of votes for the correct value and the maximum number of votes for the incorrect value, which we denote as the *best competitor*, changes with d for different values of s. Although the distribution is not uniform with a large tail towards larger values, its mean scales linearly with s. Table 3.1 contains the computed mean values for the collision and the best competitor votes for $s \in \{1, 2, 3, 4\}$.

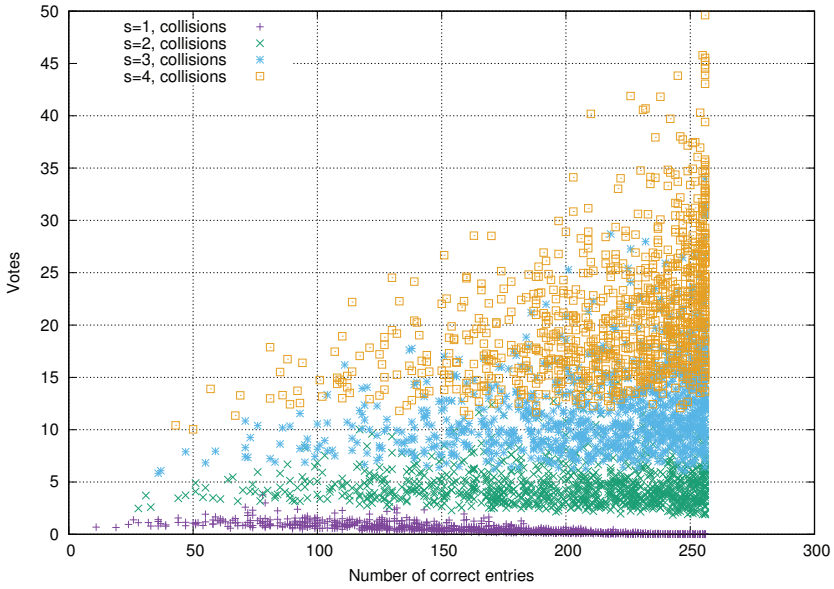


Figure 3.6 – RM$(2, 8)$, collision votes for the correct value

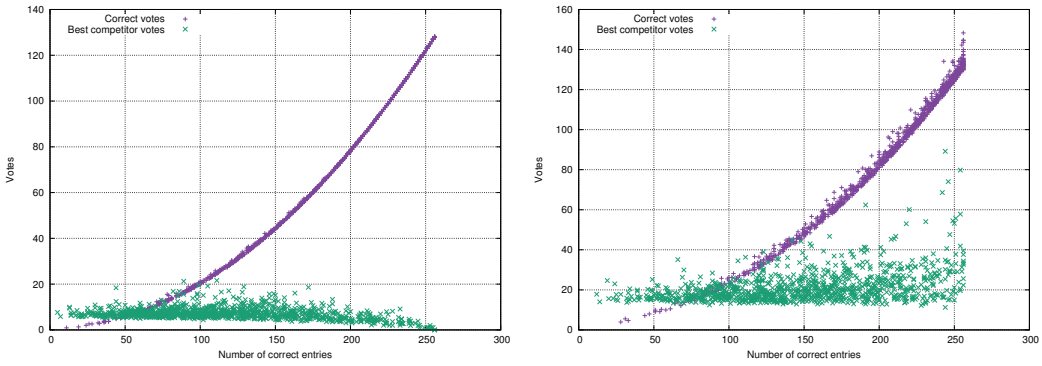


Figure 3.7 – $s = 1$ and $s = 2$

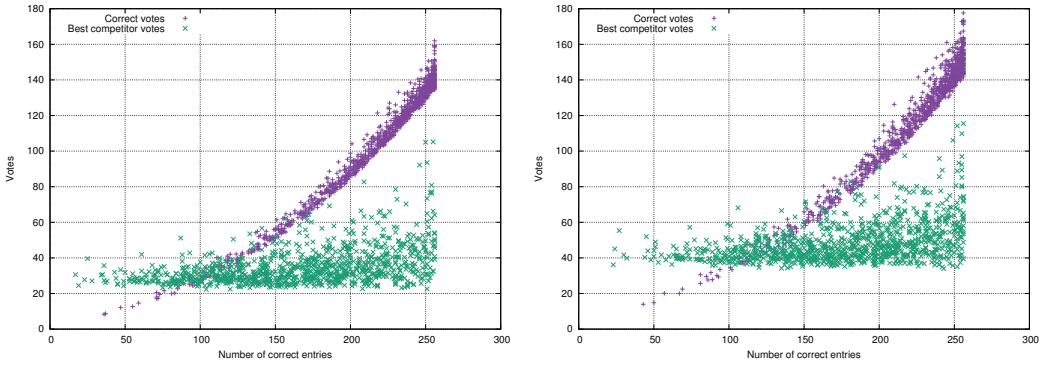


Figure 3.8 – $s = 3$ and $s = 4$

Table 3.1 – Mean votes for the correct value due to collisions and for the best incorrect value.

s	Collision votes, mean	Best competitor votes, mean
1	0.5	7.3
2	4.8	21.3
3	11.9	35.1
4	20.9	48.8

Note that by increasing s we also implicitly increase d since the larger decoder list size increases the chance of returning the correct codeword.

Figure 3.9 demonstrates another aspect of majority voting, namely how the ratio of correct coefficients changes after the voting is completed. Again we consider the code $\mathrm{RM}(2, 8)$ and set $s = 1$. We observe that even through the majority voting is often unable to recover all correct values $\mathbf{g}^{(\boldsymbol{\alpha})}$, the procedure increases their proportion as long as the initial number d of the recovered values is not too small.

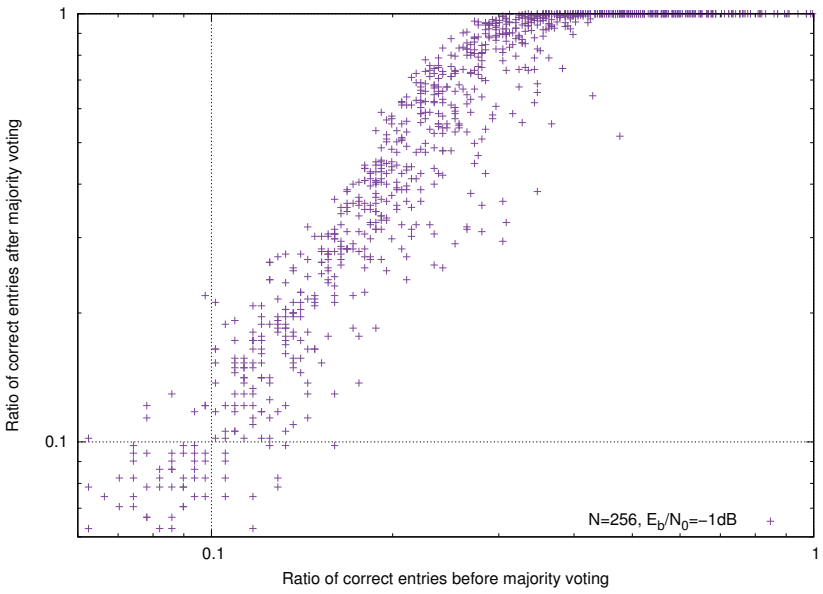


Figure 3.9 – Correction capability of majority voting

Repeated majority voting

We observed in the previous section that returning $s > 1$ most likely values of each $\mathbf{g}^{(\boldsymbol{\alpha})}$ increases the number of correct entries among them and therefore the chances of majority voting succeed. Moreover, the majority voting tends to increase the number of correct entries. This gives an intuition behind the following idea:

- In the first stage, s most likely values $\mathbf{g}^{(\boldsymbol{\alpha}),j}, 0 \leq j < s$ are returned. This step increases the number of correct coefficients d among $s(n-1)$ candidates
- Do the majority voting with $O(s^2 n)$ checks per coordinate
- Repeat the majority procedure h times with $O(n)$ checks using the values obtained on the previous step

Complexity of this procedure is $O(n^2(s^2+h))$, thus the whole decoding runs in $O(n^2(\log n + s^2 + h))$ time. The repeating majority vote is performed with only best estimates because of the complexity concerns, but our experiments also confirm that using all estimates does not improve the performance. Pseudocode of the procedure is given in Algorithm 8.

We also noticed that after a few iterations the majority voting converges, so the process can be stopped after $i < h$ steps if no values $\tilde{\mathbf{g}}^{(\boldsymbol{\alpha})}$ were changed in order to speed up the algorithm.

List decoding

The second improvement can be done at the last stage of the decoding process. When a row $\mathbf{B}_{i,*}$ of matrix $\mathbf{B}$ is obtained, the correct value does not necessarily correspond to the ML estimate. Hence, one can instead return s its most likely values $\mathbf{B}_{i,*}^{(j)}$ along with their correlations $z_i^{(j)}$ for $0 \leq j < s$. Any candidate matrix is associated with a vector $\mathbf{w}$ so that its rows are $\mathbf{B}_{i,*}^{(w_i)}$ and the reliability score of the matrix is $\mu(\mathbf{w}) = \sum_i z_i^{(w_i)}$. We propose to construct L candidate matrices with the largest scores. For each matrix $\mathbf{B}^{(l)}$, the algorithm proceeds further and produces a list of candidate codewords $\mathbf{c}^{(l)}$. The closest codeword to the received vector is returned as the decoding result.

Observe that for every fixed i the sequence $\{z_i^{(j)}\}_{j=0}^{s-1}$ corresponds to the outputs of the procedure `RM1SubcodeDecode`, which are sorted in the decreasing order. Therefore, vector $\mathbf{w} = (w_0, \ldots, w_{m-1})$ has larger score $\mu(\mathbf{w})$ than any vector $\mathbf{w}'$ s.t. $w'_i \geq w_i$ for all $0 \leq i < m$ and there exists an index i' s.t. $w'_{i'} > w_{i'}$. This property is exploited in order to efficiently construct L matrices with the largest scores one-by-one using the priority queue. We start from the matrix associated with an all-zero vector and after processing the matrix $(w_0, \ldots, w_{m-1})$ we add all vectors $(w_0+1, \ldots, w_{m-1}), \ldots, (w_0, \ldots, w_{m-1}+1)$ to the priority queue, discarding duplicates. Thus we ensure that at most m new entries are added to the priority queue and hence the complexity of the candidate matrix generation step is dominated by $O(n \log n)$ term from the decoding of first-order code and therefore the overall complexity is $O((L+n)n \log n)$. Algorithm 9 demonstrates the pseudocode of this procedure.

Let us also briefly mention other attempted strategies:

Algorithm 8 Improved SPM decoder

procedure SPMIMPROVED($\mathbf{y}, h, s$)
  for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
    for $\boldsymbol{\beta} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      $\mathbf{y}_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} \leftarrow y_{\boldsymbol{\beta}} \boxplus y_{\boldsymbol{\alpha}+\boldsymbol{\beta}}$
    $\{(v^{(\boldsymbol{\alpha}),j}, \mathbf{g}^{(\boldsymbol{\alpha}),j}) | 0 \le j < s\} \leftarrow$RM1DECODE($\mathbf{y}^{(\boldsymbol{\alpha})}$)
  for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
    $\tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} \leftarrow \underset{\substack{\boldsymbol{\beta}\in\mathbb{F}_2^m \\ 0\le j_1,j_2<s}}{\mathrm{Maj}} (\mathbf{g}^{(\boldsymbol{\beta}),j_1} + \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha}),j_2})$
  for $i \leftarrow 2$ to h **do**
    for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      $\mathbf{g}^{(\boldsymbol{\alpha})} \leftarrow \tilde{\mathbf{g}}^{(\boldsymbol{\alpha})}$
    for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      $\tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} \leftarrow \underset{\boldsymbol{\beta}\in\mathbb{F}_2^m}{\mathrm{Maj}} (\mathbf{g}^{(\boldsymbol{\beta})} + \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha})})$
  for $i \leftarrow 0$ to $m-1$ **do**
    for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      $t_{(\boldsymbol{\alpha})}^{(i)} \leftarrow (-1)^{\tilde{g}_i^{(\boldsymbol{\alpha})}} v^{(\boldsymbol{\alpha}),0}$
    $(z_i, \mathbf{B}_{i,*}) \leftarrow$RM1SUBCODEDECODE($i, \mathbf{t}^{(i)}$)
  for $i \leftarrow 0$ to $m-1$ **do**
    for $j > i$ **do**
      if $z_i > z_j$ **then**
        $B_{i,j} \leftarrow B_{j,i}$
  $\hat{\mathbf{c}} \leftarrow \mathrm{ev}(\sum_{i<j} B_{i,j} x_i x_j)$
  for $i \leftarrow 0$ to $2^m - 1$ **do**
    $\hat{y}_i \leftarrow (-1)^{\hat{c}_i} y_i$
  $(v, \mathbf{g}) \leftarrow$RM1DECODE($\hat{\mathbf{y}}$)
  return $\mathbf{B}, \mathbf{g}$

- Combine Algorithm 8 and 9. The simulations show that the error-correction capability does not improve. When the repeating majority voting is applied, it pushes vectors $\mathbf{t}_i$ closer to the codewords of first-order code. During this procedure, either the vector is pushed towards the right direction (in this case ML decoding of the subcode returns correct value of $\mathbf{B}_{i,*}$), or it is brought too far from the correct estimate and has sufficiently small correlation to not appear among the several most likely values.

- Replace hard votes in the majority voting by soft estimates $v^{(\boldsymbol{\beta})} \boxplus v^{(\boldsymbol{\alpha}+\boldsymbol{\beta})}$. No observed impact on performance.

Algorithm 9 List SPM decoder

procedure SPMList($\mathbf{y}, s, L$)
  for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
    for $\boldsymbol{\beta} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      $\mathbf{y}_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} \leftarrow y_{\boldsymbol{\beta}} \boxplus y_{\boldsymbol{\alpha}+\boldsymbol{\beta}}$
    $(v^{(\boldsymbol{\alpha})}, \mathbf{g}^{(\boldsymbol{\alpha})}) \leftarrow$RM1Decode($\mathbf{y}^{(\boldsymbol{\alpha})}$)
  for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
    $\tilde{\mathbf{g}}^{(\boldsymbol{\alpha})} \leftarrow \underset{\boldsymbol{\beta} \in \mathbb{F}_2^m}{\mathrm{Maj}}(\mathbf{g}^{(\boldsymbol{\beta})} + \mathbf{g}^{(\boldsymbol{\beta}+\boldsymbol{\alpha})})$
  for $i \leftarrow 0$ to $m-1$ **do**
    for $\boldsymbol{\alpha} \in \mathbb{F}_2^m \setminus \{0\}$ **do**
      $t_{\boldsymbol{\alpha}}^{(i)} \leftarrow (-1)^{g_i^{(\boldsymbol{\alpha})}} v^{(\boldsymbol{\alpha})}$
    $\{(z_i^{(j)}, \mathbf{B}_{i,*}^{(j)}) | 0 \le j < s\} \leftarrow$RM1SubcodeDecode($i, \mathbf{t}^{(i)}$)
  Q.Push($\sum_i z_i^{(0)}, \mathbf{0}$)
  for $l \leftarrow 0$ to $L-1$ **do**
    $(z, \mathbf{w}) \leftarrow$Q.ExtractMax()
    for $i \leftarrow 0$ to $m-1$ **do**
      $\mathbf{w}' \leftarrow \mathbf{w}$
      $w_i' \leftarrow w_i' + 1$
      if $w_i' < s-1$ **then**
        Q.Push($\sum_i z_i^{(w_i')}, \mathbf{w}'$)
    for $i \leftarrow 0$ to $m-1$ **do**
      $\mathbf{B}_{i,*}^{(l)} \leftarrow \mathbf{B}_{i,*}^{(w_i)}$
    for $i \leftarrow 0$ to $m-1$ **do**
      for $j > i$ **do**
        if $z_i^{(w_i)} > z_j^{(w_j)}$ **then**
          $B_{i,j}^{(l)} \leftarrow B_{j,i}^{(l)}$
    $\hat{\mathbf{c}}^{(l)} \leftarrow \mathrm{ev}(\sum_{i<j} B_{i,j}^{(l)} x_i x_j)$
    for $i \leftarrow 0$ to $2^m - 1$ **do**
      $\hat{y}_i^{(l)} \leftarrow (-1)^{\hat{c}_i^{(l)}} y_i$
    $(v, \mathbf{g}^{(l)}) \leftarrow$RM1Decode($\hat{\mathbf{y}}^{(l)}$)
  return $\mathbf{B}^{(l)}, \mathbf{g}^{(l)}$ that correspond to the codeword closest to $\mathbf{y}$.

Table 3.2 – Decoding algorithms for RM$(2, m)$

	Algorithm	Complexity
1	Recursive/SCL [51, 28]	$O(Ln \log n)$
2	RPA [52]	$O(n^2 \log n)$
3	SPM [69]	$O(n^2 \log n)$
4	Proposed improved SPM	$O(n^2(\log n + s^2 + h))$
5	Proposed list SPM	$O((L + n)n \log n)$

Performance

In this section, the performance of proposed algorithms is illustrated. We consider the transmission through AWGN channel with BPSK modulation. The performance of proposed methods is compared with other efficient RM decoding algorithms that can be found in the literature. Frame error rate (FER) is used as a performance metric. All participants of this comparison are presented in table 3.2. The lower bound on the ML performance was computed by running the SCL decoder with sufficiently large list so that whether an incorrect codeword is returned, it is closer to the received vector than the correct codeword and therefore the ML decoder would also be erroneous. We also adjust the parameters of all algorithms so that the complexity is approximately $O(n^2 \log n)$.

Figures 3.10 and 3.11 illustrate the results for codes RM$(2, 8)$ and RM$(2, 9)$, respectively. It can be seen that for the former code, all algorithms perform sufficiently close to each other and within less than 0.25 dB from the simulated ML lower bound. However, for the latter code we get a different picture. SCL algorithm with list size $L = 512$ demonstrates significantly worse results compared to the other algorithms and SPM algorithm also falls behind. On the other hand, both proposed algorithms as well as RPA perform equally well. SCL is outperformed by 0.5 dB and SPM by 0.2 dB.

Remark. A permutation-based version of SCL decoder closes the gap and performs similarly to RPA and the proposed decoders.

Complexity reduction

The dominant factor in the complexity of the projection-based algorithms is the number of decoded projections. One can now ask whether it is possible to take a smaller subset $\mathcal{B}$ of the derivative directions without losing too much in performance. We studied two possible strategies for selecting $\mathcal{B}$:

- Any coordinate $\mathbf{g}^{(\boldsymbol{\alpha})}$ has the same number of check relations (3.6) (which decreases quadratically with $|\mathcal{B}|$). Choosing the elements of $\mathcal{B}$ uniformly at random gives the similar result;

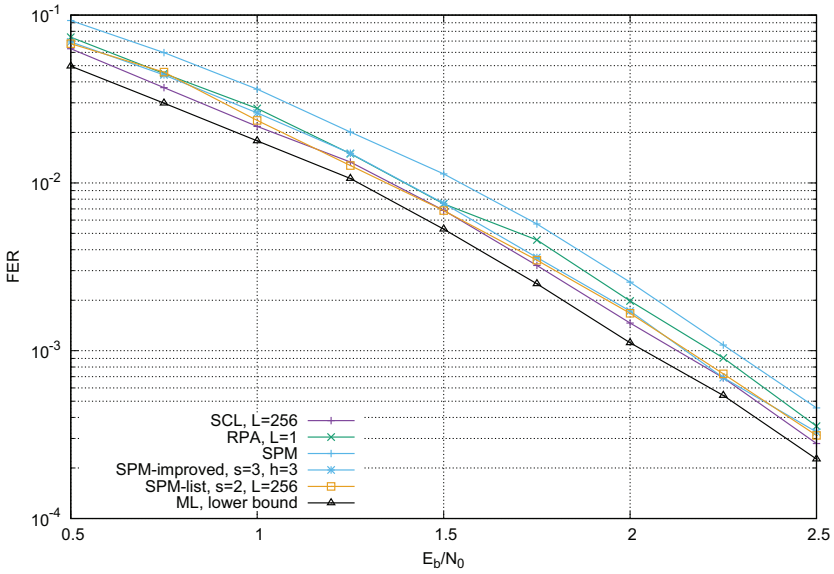


Figure 3.10 – Performance of decoding RM$(2, 8)$

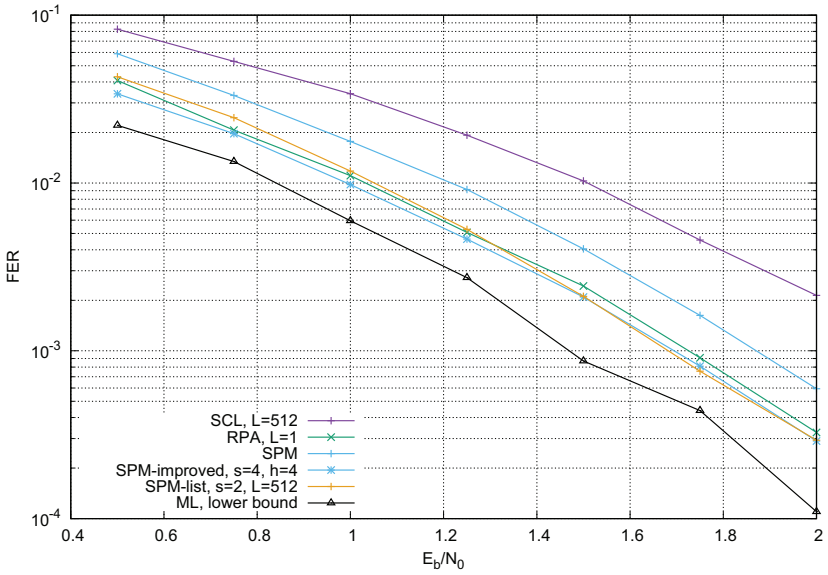


Figure 3.11 – Performance of decoding RM$(2, 9)$

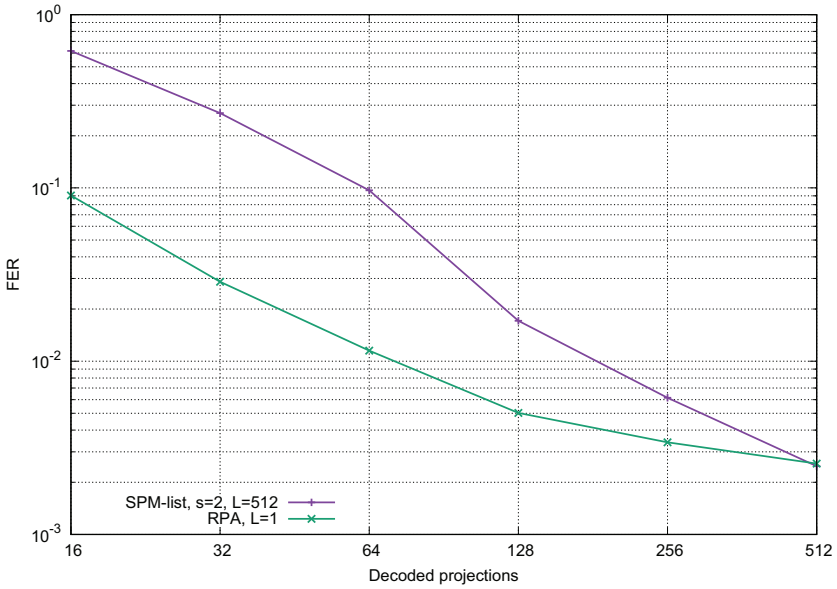


Figure 3.12 – RM$(2, 9)$, projection-based performance degradation

- Some coordinates $\mathbf{g}^{(\boldsymbol{\alpha})}$ have large number of check relations and some have very small; we treat the latter as erasures.

Our experiments showed that the first outcome gives better performance. Figure 3.12 demonstrates how the performance of RPA and the proposed list SPM decoder degrades with $|\mathcal{B}|$ with the uniform selection rule. The code in consideration is RM$(2, 9)$ and the transmission takes place over the AWGN channel with $E_b/N_0 = 1.5$dB.

It can be seen that frame error rate of the proposed decoder grows significantly faster. We attribute it to the iterative fashion of RPA algorithm which makes it more robust to the less reliable estimates.

Remark. The recent paper [72] tackles this problem by running several instances of RPA algorithm with randomly selected sets $\mathcal{B}$ and returning the closest codeword to the received vector, which for second-order codes allows to even outperform the original RPA algorithm and reach the ML performance while having the smaller complexity.

Symmetry-Induced Constraints on Decoding Efficiency

4

In the previous chapter we discussed the decoding of Reed-Muller codes based on their automorphism group and presented two algorithms. Although it is clear that some gain can be achieved compared to the conventional decoders, the experiments demonstrate that near-ML performance quickly becomes computationally unfeasible for large block lengths except the low- and high-rate regimes. The successive cancellation decoding of eBCH codes needs even larger list size to achieve near-ML performance [73]. The similar phenomenon is observed when trying to design codes that are efficiently decodable with permutation-based methods.

In this chapter, we demonstrate that the invariance under general affine group makes the code asymptotically ill-suited for successive cancellation decoding or its variations. We also present the first proof that eBCH and Reed-Muller codes need an exponential list size for near-ML decoding. Moreover, the similar result holds even under significantly milder conditions, i.e., when we only require that several partial derivatives of the code have small rates. In the last section we apply our partial symmetry framework to show that the automorphism group of polar codes cannot include many affine permutations besides the lower-triangular group.

The initial version of this work, which only considers monomial codes, appears in [74] and its further refinement and generalization for the case of arbitrary binary linear codes is published in [75]. This chapter is based on the preprint version [76] of the latter manuscript.

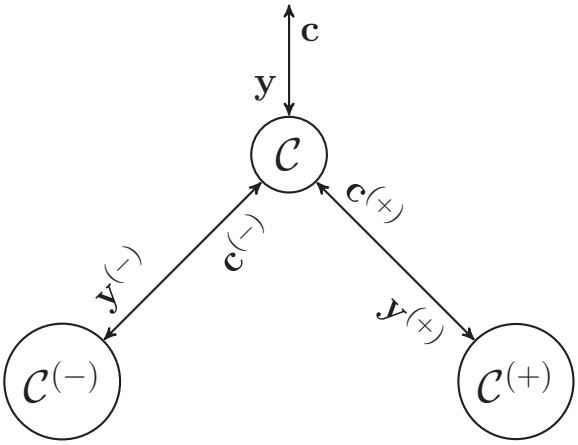


Figure 4.1 – Recursions in successive cancellation decoding

4.1 Partial Derivatives and Decoding Efficiency

Let us begin by explaining the dependence between the rates of the derivative codes and the complexity of the optimal decoding with SCL algorithm.

Proposition 6. *Consider a code $\mathcal{C}$ of length $n = 2^m$ and the transmission through a BMS channel W. If for all $i \in [m]$ holds $R(\mathcal{C}^{(\mathbf{e}_i)}) > I(W^{(-)})$, SC list algorithm needs the list size $L = 2^{\Omega(n)}$ to achieve ML performance.*

Proof. Consider a vector $\mathbf{y}$ that corresponds to the output of BMS channel W after the codeword of $\mathcal{C}$ was transmitted. The recursive application of channel transformations (2.4) in the SC decoding process can be reformulated as follows:

1. Recover $\mathbf{c}^{(-)} = \mathbf{c}_0^{n/2-1} \oplus \mathbf{c}_{n/2}^{n-1}$ from vector $\mathbf{y}^{(-)}$ that corresponds to the output of channel $W^{(-)}$.
2. Recover $\mathbf{c}^{(+)} = \mathbf{c}_{n/2}^{n-1} = \mathbf{c}^{(-)} \oplus \mathbf{c}_0^{n/2-1}$ from vector $\mathbf{y}^{(+)}$ that corresponds to the output of channel $W^{(+)}$, assuming that $\mathbf{c}^{(-)}$ is correct.
3. Return $\mathbf{c} = (\mathbf{c}^{(-)} \oplus \mathbf{c}^{(+)} | \mathbf{c}^{(+)})$.

In this perspective, the decoding is a two-stage process, when we first recover $\mathbf{c}^{(-)} \in \mathcal{C}^{(-)}$, assuming the transmission through the synthetic channel $W^{(-)}$, and then use it to recover $\mathbf{c}^{(+)} \in \mathcal{C}^{(+)}$, assuming the transmission through the synthetic channel $W^{(+)}$. Figure 4.1 illustrates this process.

Observe now that any codeword of the code $\mathcal{C}^{(-)} = \{\mathbf{c}_0^{n/2-1} \oplus \mathbf{c}_{n/2}^{n-1} | \mathbf{c} \in \mathcal{C}\}$ that appears at the first step of the SC recursions can be written as

$$\mathbf{c}^{(-)} = (c_{\mathbf{0}} + c_{\mathbf{0} \oplus \mathbf{e}_0}, \ldots, c_{\mathbf{n/2-1}} + c_{\mathbf{n/2-1} \oplus \mathbf{e}_0}),$$

and therefore the code $\mathcal{C}^{(-)}$ is a partial derivative of $\mathcal{C}$ w.r.t. x_0. Consequently, at each level of the SC recursions we pick a variable x_i and perform the decomposition of $\mathcal{C}$ into codes $\mathcal{C}^{(-)}$ and $\mathcal{C}^{(+)}$, where the former is a partial derivative w.r.t. variable x_i and the latter has generating set $M_{\mathcal{C}^{(+)}} = \{f \in \mathcal{C} | \frac{\partial f}{\partial x_i} = 0\}$. The standard SC decoding implies the selection of variable x_i at level i, where level 0 is the recursion start and at level m we have length-1 codes that correspond to the information and frozen bits.

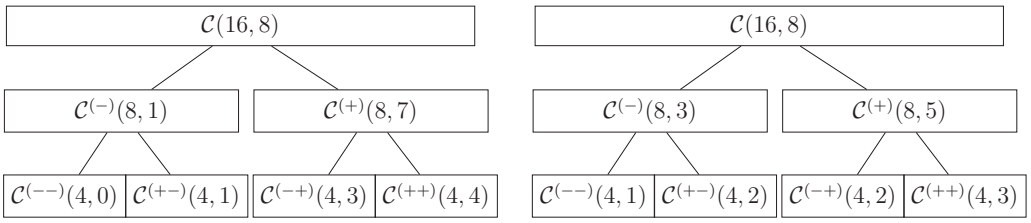
The SCL decoder ensures that at every step of the recursion there are at most L distinct vectors $\mathbf{c}^{(-)}$ or $\mathbf{c}^{(+)}$ and therefore the decoder can only succeed if the correct codeword is in the list. Now, if $R(\mathcal{C}^{(-)}) > I(W^{(-)})$, we are trying to decode above the capacity and therefore need the list at least of size $2^{n(R(\mathcal{C}^{(-)}) - I(W^{(-)}))}$ to succeed [77, eq. (1.6)]. It remains to recall that code $\mathcal{C}^{(-)}$ is a partial derivative $\mathcal{C}^{(\mathbf{e}_i)}$, where i depends on the chosen ordering. □

Remark. In case of SC decoding of an arbitrary polynomial code using the GPD framework (2.11), the final codeword has the form $\mathbf{c} = (\mathbf{c}^{(-)} \oplus \mathbf{c}^{(+)} \oplus \hat{\mathbf{c}} | \mathbf{c}^{(+)} \oplus \hat{\mathbf{c}})$, where $\hat{\mathbf{c}} = \mathbf{u}_0^{n/2-1} \tilde{\mathbf{I}} \mathbf{G}_3$. Since $\mathbf{u}_0^{n/2-1}$ is known after the recovery of $\mathbf{c}^{(-)}$, the required corrections at steps 2 and 3 are trivial.

Remark 2. The condition $R(\mathcal{C}^{(\mathbf{e}_i)}) > I(W^{(-)})$ in Proposition 6 implies that the code $\mathcal{C}^{(\mathbf{e}_i)}$ is capacity-achieving and can be safely replaced with $R(\mathcal{C}^{(\mathbf{e}_i)}) > \hat{I}$, where $\hat{I}$ is the largest capacity of the channel from the same family as $W^{(-)}$ so that the ML decoding of $\mathcal{C}^{(\mathbf{e}_i)}$ almost always succeeds.

As follows from Proposition 6, in order to have good performance with small list size a code should have a derivative with sufficiently small rate. For example, we know that a polar code of rate $R = I(W)$ achieves capacity of W under SC decoding and since by construction it is a $(u|u+v)$ concatenation of two polar codes for the corresponding channels $W^{(-)}$ and $W^{(+)}$, there is a derivative $\mathcal{C}^{(\mathbf{e}_i)}$ of rate $\hat{R}^{(-)} \approx I(W^{(-)})$. By convention we have $i = 0$ (or $i = m - 1$ if the bit-reversal permutation is utilized in the code construction).

The case of permutation-based SCL decoding is significantly more demanding in terms of the code structure. In section 3.1.2 we demonstrated that applying a variable permutation on the received vector changes the order of partial derivatives in the SC recursions. Consequently, different codes $\mathcal{C}^{(-)}$ and $\mathcal{C}^{(+)}$ appear during the decoding process and in order to get good performance we need several derivatives $\mathcal{C}^{(\mathbf{e}_i)}$ with sufficiently small rates.



(a) $\pi = (0,1,2,3)$ (b) $\pi = (3,2,1,0)$

Figure 4.2 – Different decompositions of $(16,8)$ code with $M_{\mathcal{C}} = \{1, x_0, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3\}$.

Example 2. *Consider* $(16,8,4)$ *code* $\mathcal{C}$ *with*

$$M_{\mathcal{C}} = \{1, x_0, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3\}.$$

In case of standard SC decoding we have $\mathcal{C}^{(-)} = \mathcal{C}^{(\mathbf{e}_0)} = \frac{\partial}{\partial x_0}\mathcal{C}$ *that is a* $(8,1,8)$ *code with generating set* $M_{\mathcal{C}^{(\mathbf{e}_0)}} = \{1\}$. *On the other hand, variable permutation* $\pi = (3,2,1,0)$ *corresponds to* $\mathcal{C}^{(-)} = \mathcal{C}^{(\mathbf{e}_3)} = \frac{\partial}{\partial x_3}\mathcal{C}$ *that is a* $(8,3,4)$ *code with generating set* $M_{\mathcal{C}^{(\mathbf{e}_3)}} = \{1, x_1, x_2\}$. *Figure 4.2 demonstrates one more step of the SC recursions.*

4.2 Fully Symmetric Codes

Definition 1. *A* $(2^m, k)$ *code* $\mathcal{C}$ *is fully symmetric if all its partial derivatives have equal dimensions.*

Let us denote the dimension of the derivatives as $\tilde{k}$. In this section, we demonstrate that for fully symmetric codes $\tilde{k}$ is bounded from below and in fact becomes of order $k/2$. We also show that the Reed-Muller and eBCH codes are fully symmetric.

Proposition 7. *Consider a* $(2^m, k)$ *fully symmetric code* $\mathcal{C}$. *If its dimension can be expressed as* $k = \sum_{i=0}^{l-1} \binom{m}{i} + j\frac{\mathrm{lcm}(l,m)}{l}$, *where* $0 \leq j\frac{\mathrm{lcm}(l,m)}{l} < \binom{m}{l}$, *then*

$$\tilde{k} \geq \sum_{i=0}^{l-2} \binom{m-1}{i} + j\frac{\mathrm{lcm}(l,m)}{m} \tag{4.1}$$

We get lcm in the expression due to the full symmetry constraint, which is further explained below. We call the code *optimal* if it satisfies (4.1) with an equality.

Proposition 8. *Consider a sequence of optimal fully symmetric codes* $\mathcal{C}$ *of fixed rate and increasing length* 2^m. *Then for* $i \in [m]$

$$\lim_{m\to\infty} R(\mathcal{C}^{(\mathbf{e}_i)}) = R(\mathcal{C}). \tag{4.2}$$

Proposition 7 implies that list or permutation decoding in any channel W s.t. $\frac{\tilde{k}}{2^{m-1}} > I(W^{(-)})$ needs an exponential complexity to achieve the ML performance, and proposition 8 states that this condition asymptotically becomes $R(\mathcal{C}) > I(W^{(-)})$.

4.2.1 Proof of Proposition 7 for Monomial Codes

Let us start from the rate-1 code $\mathcal{C}_m$. All its derivatives are also rate-1 codes and therefore $\mathcal{C}_m$ is fully symmetric. Any monomial code $\mathcal{C}$ can be constructed by removing $2^m - k$ monomials from $M_{\mathcal{C}_m}$ and we would like to do it in a way such that $\mathcal{C}$ is fully symmetric and $\tilde{k}$ is minimized. Observe that $\frac{\partial x^{\mathbf{v}}}{\partial x_i}$ is nonzero iff $v_i = 1$ and hence removing $x^{\mathbf{v}}$ from the generating set decreases the dimensions of $\mathrm{wt}(\mathbf{v}) = \deg(x^{\mathbf{v}})$ partial derivatives by 1 (or equivalently, the dimensions of all derivatives are decreased on average by $\frac{\mathrm{wt}(\mathbf{v})}{m}$). This implies that the optimum strategy is to remove $2^m - k$ monomials of the largest degrees.

If $k = \sum_{i=0}^{l} \binom{m}{i}, 0 \leq l \leq m$, we simply remove all monomials of degree larger than l and consequently each derivative contains all monomials on $m-1$ variables of degree at most $l-1$, which gives $\tilde{k} = \sum_{i=0}^{l-1} \binom{m-1}{i}$. Otherwise, we can write $k = \sum_{i=0}^{l-1} \binom{m}{i} + p, 0 < p < \binom{m}{l}$, and remove p monomials of degree l in addition. It follows that the dimensions of all derivatives are decreased on average by $\frac{pl}{m}$, and since the code is fully symmetric the actual decrease for any derivative should also be $\frac{pl}{m}$. Therefore, $\frac{pl}{m}$ must be an integer, which is true only if p is a multiple of $\frac{\mathrm{lcm}(l,m)}{l}$, which implies the bound (4.1).

The bound coincides with the parameters of Reed-Muller codes when $j = 0$. Otherwise, the set of monomials to remove can be found by considering a bipartite graph $\mathcal{G} = (V_L, V_R, E)$ with left vertices $h_i \in V_L$ isomorphic to variables $x_i, 0 \leq i < m$ and right vertices $h_v \in V_R$ isomorphic to all degree-l monomials $x^{\mathbf{v}}$. We draw an edge between h_i and h_v if the monomial $x^{\mathbf{v}}$ contains variable x_i. This graph is $(\binom{m-1}{l-1}, l)$-biregular and we want to remove all but $j\frac{\mathrm{lcm}(l,m)}{l}$ of its right vertices so that the graph remains biregular, i.e., find its $(\cdot, l)$-biregular subgraph $\mathcal{G}'$.

Figure 4.3 demonstrates an example of graph $\mathcal{G}$ for $m = 4, l = 2$ and one of its possible $(1, 2)$-regular subgraphs $\mathcal{G}'$ (in red). Such $\mathcal{G}'$ can be found as a maximum flow solution for the network with the source connected to all left vertices with capacity-$\frac{\mathrm{lcm}(l,m)}{l}$ edges, the sink connected to all right vertices with capacity-m edges and all $e \in E$ having the unit capacity.

4.2.2 Proof of Proposition 7 for Polynomial Codes

We start from a simple case and consider an $(2^m, k)$ code $\mathcal{C}_r \subseteq \mathrm{span}\{x^{\mathbf{v}} | \mathrm{wt}(\mathbf{v}) = r\}$ for some fixed r. By definition, $0 \leq k \leq \binom{m}{r}$, and consequently $0 \leq \tilde{k} \leq$

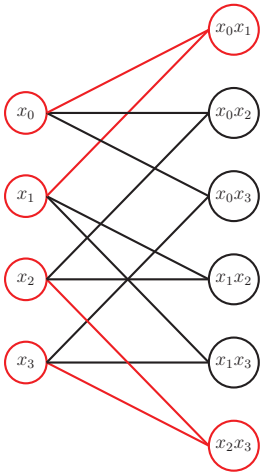


Figure 4.3 – $(3,2)$-regular $\mathcal{G}$ and $(1,2)$-regular $\mathcal{G}'$

$\binom{m-1}{r-1}$. Code $\mathcal{C}_r$ by definition is spanned by k degree-r linearly independent homogeneous polynomials $f_s, 0 \leq s < k$. Assume now a certain ordering on monomials $x^{\mathbf{v}^{(j)}}$, e.g., lexicographic w.r.t. $\mathbf{v}^{(j)}$, and consider the $k \times \binom{m}{r}$ matrix $\mathbf{M}$ such that $M_{s,j} = 1$ if f_s includes $\mathbf{v}^{(j)}$. $\mathbf{M}$ is a basis of the linear space of all polynomials whose evaluations are codewords of $\mathcal{C}_r$ and therefore has full row rank, so we can use Gaussian elimination to transform it into $\tilde{\mathbf{M}} = \begin{pmatrix}\mathbf{I} & * \end{pmatrix} \mathbf{P}$, where $\mathbf{P}$ is a column permutation matrix. Let us further define a vector $\boldsymbol{\phi}$ s.t. $\phi_j = 1$ if j-th column of $\tilde{\mathbf{M}}$ is a columnd of the identity matrix.

Similarly, the generator of the linear space corresponding to the partial derivative $\frac{\partial}{\partial x_i}$ is a matrix $\tilde{\mathbf{M}}_i$ obtained by removing all columns of $\tilde{\mathbf{M}}$ but the ones that correspond to monomials that include $x_i = 1$. Its dimension is equal to $\operatorname{rank} \tilde{\mathbf{M}}_i$. From the construction it follows that $\operatorname{rank} \tilde{\mathbf{M}}_i \geq |\{j | \phi_j = 1 \wedge v_i^{(j)} = 1\}|$. In case of monomial codes we have $\operatorname{rank} \tilde{\mathbf{M}}_i = |\{j | \phi_j = 1 \wedge v_i^{(j)} = 1\}|$, so it only remains to see that the bound (4.1) minimizes $\max_i |\{j | \phi_j = 1 \wedge v_i^{(j)} = 1\}|$. The extension to the general case is straightforward.

Example 3. *Consider $m = 4, r = 2, k = 4$ and assume $\boldsymbol{\phi} = (0,1,1,1,1,0)$, where the ordering on degree-2 monomials is $(x_0x_1, x_0x_2, x_2x_3, x_1x_2, x_1x_3, x_2x_3)$. This vector corresponds to the matrix*

$$\tilde{\mathbf{M}} = \begin{pmatrix} * & 1 & 0 & 0 & 0 & * \\ * & 0 & 1 & 0 & 0 & * \\ * & 0 & 0 & 1 & 0 & * \\ * & 0 & 0 & 0 & 1 & * \end{pmatrix},$$

where $*$ can be any binary value. The partial derivatives correspond to the

matrices

$$\tilde{\mathbf{M}}_0 = \begin{pmatrix} * & 1 & 0 \\ * & 0 & 1 \\ * & 0 & 0 \\ * & 0 & 0 \end{pmatrix} \tilde{\mathbf{M}}_1 = \begin{pmatrix} * & 0 & 0 \\ * & 0 & 0 \\ * & 1 & 0 \\ * & 0 & 1 \end{pmatrix},$$

$$\tilde{\mathbf{M}}_2 = \begin{pmatrix} 1 & 0 & * \\ 0 & 0 & * \\ 0 & 1 & * \\ 0 & 0 & * \end{pmatrix} \tilde{\mathbf{M}}_3 = \begin{pmatrix} 0 & 0 & * \\ 1 & 0 & * \\ 0 & 0 & * \\ 0 & 1 & * \end{pmatrix}.$$

We have $\max_i \operatorname{rank} \tilde{\mathbf{M}}_i \geq 2$ *and we know that the code spanned by monomials* $\{x_0x_2, x_0x_3, x_1x_2, x_1x_3\}$ *has* $\max_i \operatorname{rank} \tilde{\mathbf{M}}_i = 2$.

4.2.3 Proof of Proposition 8

Let m be an odd number and consider an optimal fully symmetric code of rate 1/2. Its dimension can be expressed as $k = 2^{m-1} = \sum_{i=0}^{\lfloor m/2 \rfloor} \binom{m}{i}$ and its derivatives have dimension

$$\tilde{k} = \sum_{i=0}^{\lfloor m/2 \rfloor - 1} \binom{m-1}{i}.$$

Now consider $\left| \frac{\tilde{k}}{2^{m-1}} - \frac{1}{2} \right| = \left| \frac{\binom{m-1}{\lfloor m/2 \rfloor}}{2^m} \right|$, which goes to 0 with $m \to \infty$, and to finish the proof it remains to notice that the bound (4.1) is convex, which gives the same convergence for all values of k. Assume now that $\frac{\partial C}{\partial x_i}$ is a subcode of $\mathcal{C}$, i.e., $\mathrm{Aut}(\mathcal{C})$ contains the permutation $\mathbf{x} \to \mathbf{x} + \mathbf{e}_i$. Since any permutation $\mathbf{x} \to \mathbf{x} + \mathbf{b}$ can be decomposed into a product of transpositions, we have dim $\frac{\partial C}{\partial x_i} \leq k/2$ due to Theorem 1 and therefore the lower bound on the derivative code rate converges to the worst-case value.

4.2.4 Symmetry of RM and eBCH Codes

Proposition 9. *If* $\mathcal{C}$ *is invariant w.r.t. a swap of variables* (x_i, x_j)*, then its partial derivatives w.r.t. these variables are permutation equivalent.*

Proof. Consider Boolean functions f and $\tilde{f}$, where $\tilde{f}$ is obtained from f by swapping variables x_i and x_j. It follows that $\frac{\partial \tilde{f}}{\partial x_j}$ can be obtained from $\frac{\partial f}{\partial x_i}$ by swapping variables x_i and x_j. □

Proposition 10. *Reed-Muller codes are fully symmetric*

Proof. Indeed, by construction $M_{r,m}$ includes all m-variate monomials up to degree r and the generating set of any partial derivative consists of all $(m-1)$-variate monomials up to degree $r-1$. □

Lemma 1. *If* $\mathrm{Aut}(\mathcal{C})$ *contains permutation* π *s.t.* $\pi(\mathbf{x}+\tilde{\mathbf{b}}) = \pi(\mathbf{x}) + \mathbf{b}$ *for some nonzero* $\mathbf{b}, \tilde{\mathbf{b}} \in \mathbb{F}_2^m$*, then the codes induced by the direction derivatives* $D_{\mathbf{b}}$ *and* $D_{\tilde{\mathbf{b}}}$ *are permutation equivalent.*

Proof. Consider some function $f(\mathbf{x})$ which is a codeword of $\mathcal{C}$ and its permutation $g(\mathbf{x}) = f(\pi(\mathbf{x}))$. Take the derivatives in directions $\mathbf{b}$ and $\tilde{\mathbf{b}}$:

$$\begin{aligned} D_{\mathbf{b}} f(\mathbf{x}) &= f(\mathbf{x}) + f(\mathbf{x}+\mathbf{b}) \\ D_{\tilde{\mathbf{b}}} g(\mathbf{x}) &= g(\mathbf{x}) + g(\mathbf{x}+\tilde{\mathbf{b}}) \\ &= f(\pi(\mathbf{x})) + f(\pi(\mathbf{x}+\tilde{\mathbf{b}})) = f(\pi(\mathbf{x})) + f(\pi(\mathbf{x})+\mathbf{b}) \end{aligned}$$

It follows that $D_{\tilde{\mathbf{b}}} g$ can be obtained from $D_{\mathbf{b}} f$ by map $\mathbf{x} \to \pi(\mathbf{x})$. Since both f and g are codewords of $\mathcal{C}$, we can conclude that any codeword of $D_{\tilde{\mathbf{b}}}\mathcal{C}$ can be obtained from a codeword of $D_{\mathbf{b}}\mathcal{C}$ by permutation and vice versa and consequently the derivatives of $\mathcal{C}$ in directions $\mathbf{b}$ and $\tilde{\mathbf{b}}$ lead to permutation equivalent codes. □

Proposition 11. *Affine-invariant codes are fully symmetric*

Proof. A code is called affine-invariant if it is invariant under $GA(1, \mathbb{F}_{2^m})$. Any permutation $x \to ax \in GA(1, \mathbb{F}_{2^m})$ satisfies the conditions of Lemma 1 for all pairs $(b, \tilde{b})$ s.t. $b\tilde{b}^{-1} = a$ and it follows that all derivative codes are permutation equivalent. □

Corollary 1. *eBCH codes are fully symmetric*

Proof. Indeed, by Proposition 2 eBCH codes are affine-invariant and therefore fully symmetric. □

Figure 4.4 shows the actual derivative code rates for eBCH codes of length 512 along with the lower bound (4.1) on the derivative rates for fully symmetric codes (recall that RM codes achieve this bound) compared to the smallest derivative rate for polar codes constructed using the Gaussian approximation [78] for $E_b/N_0 = 2$dB. By convention, the derivative code of polar code with the smallest rate is $\mathcal{C}^{(e_0)}$. The capacity transformation for BEC is given as a reference (if $I(W) = (1-\varepsilon)$, $I(W^{(-)}) = (1-\varepsilon)^2$). Observe that the bound is rather loose for eBCH codes, that are in fact close to an upper bound $k/2$. Potentially a better bound might be derived by taking more structural properties into consideration rather than just full symmetry, which we leave as a direction for the future research. However, this plot provides a good demonstration why the list size for near-ML decoding of the eBCH codes grows substantially faster than for the Reed-Muller codes (and why for both codes it quickly becomes impractical). An interesting observation is that the smallest derivative rate for polar codes is close to the capacity of the erasure channel $W^{(-)}$ despite being constructed for the Gaussian channel.

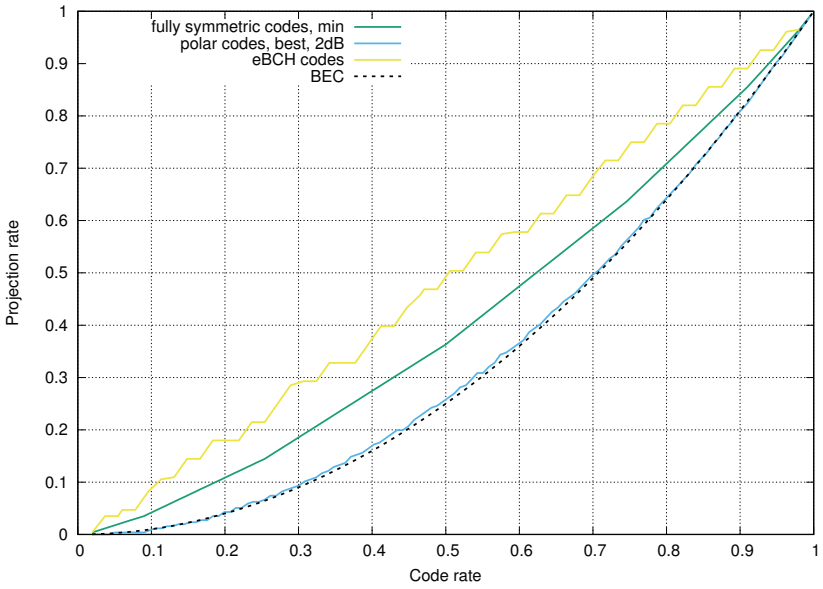


Figure 4.4 – Bounds on the derivative code rates for fully symmetric codes, $n = 512$

4.2.5 Interlude: Derivative-Constrained Polar Codes

Let us now pay a little bit more attention to the fact that the best projection $\mathcal{C}^{(e_0)}$ for polar codes constructed for Gaussian channel has much smaller rate compared to the capacity of the underlying channel. We know that polar codes are optimized for the case of SC decoding without list ($L = 1$) and we can expect that any other code will have inferior performance. However, the situation changes when we consider list decoding with $L > 1$. Reed-Muller codes represent the trivial example, outperforming polar codes by far if the list size is unbounded. Whereas the analysis of the error probability of list decoder for any finite list size $L > 1$ remains an open problem, we propose a simple heuristic construction of polar codes with improved performance under SCL decoding, which can be described as follows.

Consider the construction of (n, k) polar code and assume that we have the vector $\mathbf{r}$ of reliabilities of the corresponding synthetic bit subchannels. In order to obtain a vanilla polar code, we take the frozen set $\mathcal{F}$ as indices of $n - k$ smallest entries of $\mathbf{r}$. The dimension of $\mathcal{C}^{(e_0)}$ can be computed as $|\{i \in \mathcal{F}^c | i < n/2\}|$, i.e., the number of non-frozen indices in the first half of vector $\mathbf{r}$. Let us denote this dimension as $\tilde{k}^*$. A *derivative-constrained* (n, k) polar code with constraint parameter s is constructed by imposing the restriction on set $\mathcal{F}$ so that $\dim \mathcal{C}^{(e_0)} = \tilde{k}^* + s$. Such restriction can be easily implemented by replacing s indices $i'_j < n/2$ in $\mathcal{F}$ with the largest reliabilities with s indices $i''_j \geq n/2$ of non-frozen symbols with the smallest reliabilities.

Figures 4.5 and 4.6 demonstrate the performance of derivative-constrained

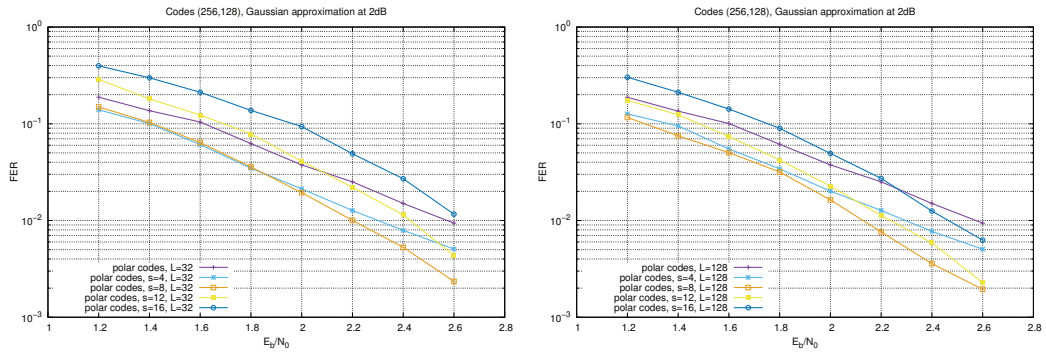


Figure 4.5 – Derivative-constrained polar codes, $L = 32$ and $L = 128$

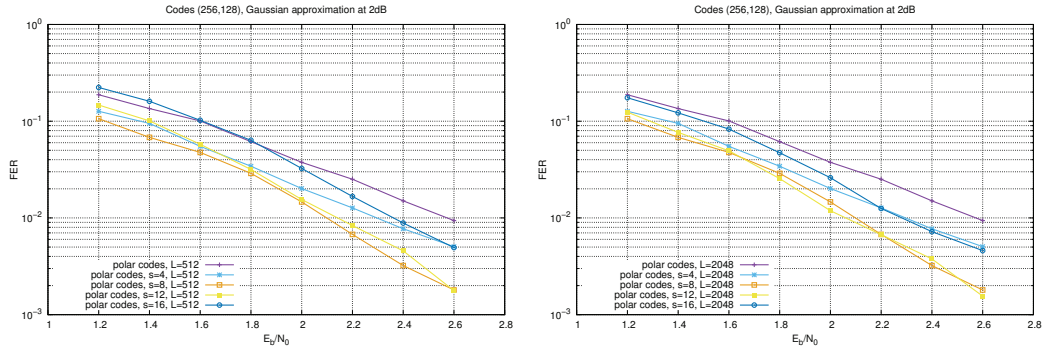


Figure 4.6 – Derivative-constrained polar codes, $L = 512$ and $L = 2048$

$(256, 128)$ codes for $s \in \{4, 8, 12, 16\}$, where Gaussian approximation at $E_b/N_0 =$ 2dB is used to estimate the subchannel reliabilities and SCL algorithm with $L \in \{32, 128, 512, 2048\}$ is used for decoding. The performance plots confirm the intuition behind the construction: whereas for $L = 32$, derivative-constrained codes only for $s = 4$ and $s = 8$ perform strictly better, when L is increased the similar thing happens to the codes with larger values of s, reaching $s = 16$ when the list size $L = 2048$ is used during the decoding. Therefore, we can expect that the optimal value s should grow with L and for unbounded list size the optimal $\dim \mathcal{C}^{(e_0)}$ for the derivative-constrained code most likely coincides with the dimension of the corresponding Reed-Muller code (if k can be expressed as $\sum_{0 \le i \le r} \binom{m}{r}$ for some $r \ge 0$) since Reed-Muller codes in simulations demonstrate the best ML performance among all monomial codes.

Remark. The construction of derivative-constrained polar codes with CRC or polar subcodes has inferior performance compared to the vanilla constructions.

Remark 2. The essentially similar idea (although coming from somewhat different motivation) was recently investigated in [79]. Whereas the idea of derivative-constrained codes is based only on the topmost SC recursion step and just the dimension of code $\mathcal{C}^{(e_0)}$ is modified, authors in [79] perform extensive

simulations combined with the dynamic programming approach to optimize the dimensions of codes $\mathcal{C}^{(-)}$ and $\mathcal{C}^{(+)}$ at all levels of SC recursion. They achieve the performance of CRC-concatenated polar codes without using CRC. However, even after combining with CRC, there is no improvement over vanilla CRC-concatenated codes.

4.3 Partially Symmetric Codes

We demonstrated that the full symmetry puts a rather restrictive lower bound on the dimensions of the derivatives. In this section, we show what happens if we demand fewer symmetries and derive the corresponding bound.

Definition 2. *A $(2^m, k)$ code $\mathcal{C}$ is t-symmetric if t of its partial derivatives have equal dimensions, which we denote as $\tilde{k}_t$, and $m - t$ have dimensions strictly greater.*

In other words, there exists a set of *target derivatives* $\mathcal{H}_t, |\mathcal{H}_t| = t$, such that $\forall \mathbf{e}_i \in \mathcal{H}_t \ \dim \mathcal{C}^{(\mathbf{e}_i)} = \tilde{k}_t$ and $\forall \mathbf{e}_i \notin \mathcal{H}_t \ \dim \mathcal{C}^{(\mathbf{e}_i)} > \tilde{k}_t$. A code is fully symmetric if $t = m$, non-symmetric if $t = 1$ and partially symmetric otherwise. Reed-Muller codes are fully symmetric and polar codes are in general non-symmetric. Without loss of generality, we assume $\mathcal{H}_t = \{\mathbf{e}_i | i \in [t]\}$.

Proposition 12. *If a t-symmetric code $\mathcal{C}$ has dimension $k = \sum_{i=0}^{l-1} \binom{t}{i} 2^{m-t} + j\frac{\operatorname{lcm}(l,t)}{l}$, then*

$$\tilde{k}_t \geq \sum_{i=0}^{l-2} \binom{t-1}{i} 2^{m-t} + j\frac{\operatorname{lcm}(l,t)}{t} \tag{4.3}$$

We get lcm in the expression due to the t-symmetry constraint, which is further explained below. Again we call a partially symmetric code *optimal* if it satisfies (4.3) with an equality.

Proposition 13. *Consider a sequence of optimal t-symmetric codes $\mathcal{C}$ of fixed rate and increasing length 2^m, where t is an increasing function of m. Then for $i \in [t]$*

$$\lim_{m\to\infty} R(\mathcal{C}^{(\mathbf{e}_i)}) = R(\mathcal{C}). \tag{4.4}$$

Therefore, even in more relaxed setting we get the lower bound similar to the one for fully symmetric codes and identical asymptotic result. However, the speed of convergence is different and depends on t. Using the result of Reeves and Pfister that upper bounds the rate difference between $\mathrm{RM}(r, m)$ and $\mathrm{RM}(r, m+1)$ [46, Lemma 7], we can conclude that

Table 4.1 – Impact of monomials on the dimension of code and its target derivatives

l	**Change in** $\dim \mathcal{C}_{m,t}$	**Change in** $\frac{\partial}{\partial x_i} \dim \mathcal{C}_{m,t}$	$\#\{x^{\mathbf{v}} : \tau_{\mathbf{v}} = l\}$
t	1	1	2^{m-t}
$t-1$	$\frac{\mathrm{lcm}(t,t-1)}{t-1}$	$\frac{\mathrm{lcm}(t,t-1)}{t}$	$\binom{t}{t-1}2^{m-t}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	t	1	$t2^{m-t}$

$$R(\mathcal{C}) - R(\mathcal{C}^{(\mathbf{e}_i)}) \leq O\left(\frac{1}{\sqrt{t}}\right) \tag{4.5}$$

4.3.1 Proof of Proposition 12

Define $\tau_{\mathbf{v}} = |\{i \in [t] | v_i = 1\}|$, i.e. the number of variables $\{x_0, \ldots, x_{t-1}\}$ in the monomial $x^{\mathbf{v}}$. We start from rate-1 code $\mathcal{C}_m$ and using the same argument as in section 4.2.1 we conclude that removing $2^m - k$ monomials of the largest $\tau_{\mathbf{v}}$ gives the optimal t-symmetric code. The number of monomials $x^{\mathbf{v}}$ s.t. $\tau_{\mathbf{v}} = l$ is $\binom{t}{l}2^{m-t}$ since l of the variables $\{x_0, \ldots, x_{t-1}\}$ can be selected in $\binom{t}{l}$ ways with any combination of the remaining $m - t$.

Figure 4.7 demonstrates the lower bound (4.3) on the derivative rates of partially symmetric codes for $t > 2$ and $n = 512$. In case of 3-symmetric codes, the bound is close to the BEC curve, which is similar to the best derivative for polar codes, so we can expect rather good list decoding performance. However, it quickly grows with t, so we expect the large list size for near-ML decoding except for the low- and high-rate regions.

Similarly to the section 4.2.2, the bound for monomial codes also holds for polynomial codes.

4.3.2 Proof of Proposition 13

Let t be an odd number and consider an optimal t-symmetric code of rate 1/2. Its dimension can be expressed as $k = 2^{m-1} = 2^{m-t}\sum_{i=0}^{\lfloor t/2 \rfloor}\binom{t}{i}$ and its target derivatives have dimension

$$\tilde{k}_t = 2^{m-t}\sum_{i=0}^{\lfloor t/2 \rfloor - 1}\binom{t-1}{i}.$$

If t is an increasing function of m, similarly to section 4.2.3 the expression $\tilde{k}_t/2^{m-1}$ converges to 1/2 and due to convexity of bound (4.3), the same holds for all values of k. Therefore, any sequence of $(2^m, k)$ binary linear codes with the partial symmetry growing with m is asymptotically bad for polar-like decoding.

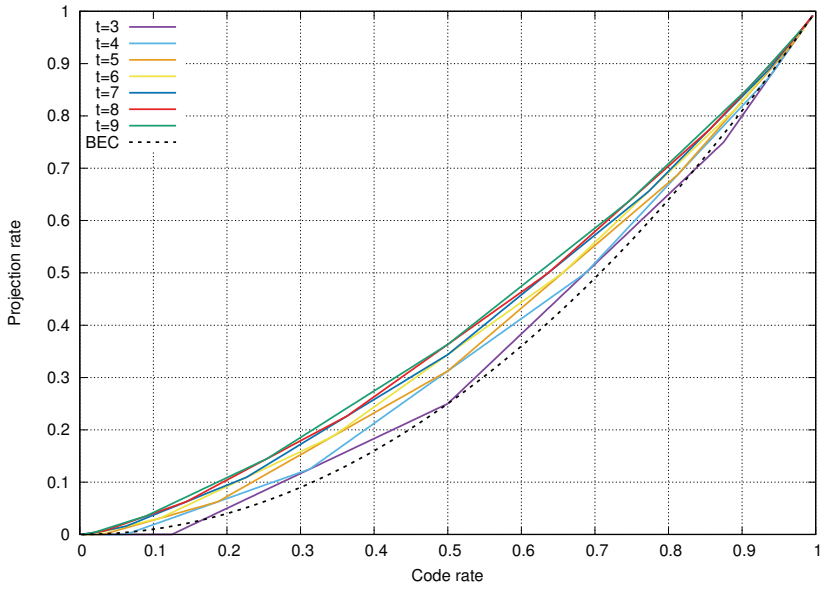


Figure 4.7 – Lower bounds on the derivative code rates for partially symmetric codes, $n = 512$

4.3.3 Code Construction

The construction of the optimal t-symmetric monomial codes, i.e., achieving (4.3) with equality, is summarized in Algorithm 10.

Algorithm 10 Construction of optimal partially symmetric code

Take $M_{\mathcal{C}} = M_{\mathcal{C}_m}$, $k' = |M_{\mathcal{C}_m}| = 2^m$
$\hat{l} \leftarrow t$
while $k' - 2^{m-t}\binom{t}{\hat{l}} \geq k$ **do**
  Remove from $M_{\mathcal{C}}$ all monomials with $\tau_{\mathbf{v}} = \hat{l}$
  $\hat{l} \leftarrow \hat{l} - 1$, $k' \leftarrow k' - 2^{m-t}\binom{t}{\hat{l}}$
$\hat{d} \leftarrow m - t + \hat{l}$
while $k' - \binom{t}{\hat{l}}\binom{m-t}{\hat{d}-\hat{l}} \geq k$ **do**
  Remove from $M_{\mathcal{C}}$ all degree-$\hat{d}$ monomials with $\tau_{\mathbf{v}} = \hat{l}$
  $\hat{d} \leftarrow \hat{d} - 1$, $k' \leftarrow k' - \binom{t}{\hat{l}}\binom{m-t}{\hat{d}-\hat{l}}$
while $k' - \binom{t}{\hat{l}} \geq k$ **do**
  Pick a degree-$(\hat{d} - \hat{l})$ monomial $x^{\mathbf{s}}$ s.t. $\tau_{\mathbf{s}} = 0$
  Remove from $M_{\mathcal{C}}$ all degree-$\hat{d}$ monomials with $\tau_{\mathbf{v}} = \hat{l}$ that contain $x^{\mathbf{s}}$
  $k' \leftarrow k' - \binom{t}{\hat{l}}$
return $M_{\mathcal{C}}$

We can use Algorithm 10 to establish an upper bound on the minimum distance of codes achieving $\tilde{k}^*_{m,t}$.

Table 4.2 – Monomials to remove.

l	**Impact on dimension**	**Monomials**
3	Remove 1 monomial	$x_0x_1x_2x_3$
	$\tilde{k}$ decreases by 1	$x_0x_1x_2$
2	Remove 3 monomials	$x_0x_1x_3$, $x_0x_2x_3$, $x_1x_2x_3$
	$\tilde{k}$ decreases by 2	x_0x_1, x_0x_2, x_1x_2
1	Remove 3 monomials	x_0x_3, x_1x_3, x_2x_3
	$\tilde{k}$ decreases by 1	x_0, x_1, x_2

Proposition 14. *Consider a code $\mathcal{C}_{m,t}$ which achieves the lower bound and assume that Algorithm 10 stopped at some $l = \hat{l}$. Then the minimum distance of $\mathcal{C}_{m,t}$ is at most 2^{m-z}, where $z = \hat{l} + m - t$ if less than $\binom{t}{\hat{l}}$ first entries at stage $l = \hat{l}$ were removed and $z = \hat{l} - 1 + m - t$ otherwise.*

Proof. For any l, the maximal total degree of monomials considered in this stage is $l + m - t$ (l variables from $\mathcal{M}_t$ and all $m - t$ from $\mathcal{M}_t^c$), and there are $\binom{t}{l}$ such monomials. Plugging this into (2.1) completes the proof. □

Remark Algorithm 10 constructs codes with poor minimum distance for small values of t. For example, in case of 2-symmetric codes with dimension $k \geq 2^m - 2^{m-2}$ only at most 2^{m-2} monomials of $\tau_{\mathbf{v}} = 2$ are removed and from Proposition 14 it follows that the code has minimum distance at most 2.

In practice, one can construct t-symmetric codes as subcodes of some Reed-Muller codes $\mathrm{RM}(r, m)$ to guarantee that the minimum distance is at least 2^{m-r}. In this case, at step 1 we start from $M_{r,m}$ instead of $M_{\mathcal{C}_m}$, at step 2 the term 2^{m-t} is replaced with $\binom{t}{\hat{l}} \sum_{i=0}^{\min(m-t, r-\hat{l})} \binom{m-t}{i}$ and at step 3 the initial value of $\hat{d}$ becomes $\min(m - t + \hat{l}, r)$ (since after step 1 all monomials with the degree greater than r are already removed and therefore out of consideration).

Example 4. *Consider $m = 4, t = 3$ and $k = 8$. All monomials with nontrivial $\tau_{\mathbf{v}}$ are listed in Table 4.2 sorted in the removal order. Start from $M_4, k' = 16$ and go to step 2. Set $\hat{l} = 3$. $16 - 2^{4-3}\binom{3}{3} = 14 \geq 8$, so we remove all monomials that contain $x_0x_1x_2$ ($x_0x_1x_2x_3$ and $x_0x_1x_2$), now $k' = 14$ and $\hat{l} = 2$. $14 - 2^{4-3}\binom{3}{2} = 8 \geq 8$, so we remove all monomials that contain x_0x_1, x_0x_2 or x_1x_2 ($x_0x_1x_3$, $x_0x_2x_3$, $x_1x_2x_3$ and x_0x_1, x_0x_2, x_1x_2), now $k' = 8$ and the construction procedure is terminated since $k' = k$.*

The constructed $(16, 8, 4)$ code has generating set

$$M_{\mathcal{C}_{4,3}} = \{x_0x_3, x_1x_3, x_2x_3, x_0, x_1, x_2, x_3, 1\}$$

and all of its target derivatives have the generating set $\{x_3, 1\}$ of cardinality 2.

Assume that k satisfies proposition 12 and Algorithm 10 ends with $k' > k$. This means that for some fixed degree-$(\hat{d} - \hat{l})$ monomial $x^{\mathbf{s}}, \tau_{\mathbf{s}} = 0$ we need to remove $k' - k$ degree-$\hat{d}$ monomials with $\tau_{\mathbf{v}} = \hat{l}$ that contain $x^{\mathbf{s}}$ so that the dimensions of all target derivatives are decreased by $\frac{(k'-k)\hat{l}}{t}$. The set of monomials to remove can be found using the same bipartite graph formulation as in section 4.2.1.

4.3.4 Structure of Partially Symmetric Monomial Codes

Let us define $\check{\mathcal{C}}_{m,t}$ as the code obtained from Algorithm 10.

Proposition 15. *For any $\check{\mathcal{C}}_{m,t}$ holds $\mathcal{T}_m \in \mathrm{Aut}(\check{\mathcal{C}}_{m,t})$.*

This property follows directly from the code construction. Consider a monomial $x^v \in M_{\check{\mathcal{C}}_{m,t}}$ and assume it has $\tilde{l}$ variables in $\mathcal{M}_t$. Any divisor $x^{\mathbf{s}}$ of $x^{\mathbf{v}}$ has $\tau_{\mathbf{s}} \leq \tilde{l}$ and smaller degree, so it cannot be removed from $M_{\check{\mathcal{C}}_{m,t}}$ before $x^{\mathbf{v}}$ in the construction process.

Proposition 16. *$\check{\mathcal{C}}_{m,t}$ is invariant w.r.t. any permutation on sets $\{x_0, \ldots, x_{t-1}\}$ and $\{x_t, \ldots, x_{m-1}\}$.*

It is sufficient to observe that for any d, l all degree-d monomials with $\tau_{\mathbf{v}} = l$ are either in $M_{\check{\mathcal{C}}_{m,t}}$ or its complement.

Note that in some cases codes $\check{\mathcal{C}}_{m,t}$ might coincide with the ones from [36] (which are for some choice of parameters invariant w.r.t. $\{x_0, \ldots, x_{t-1}\}$, but are optimized for SC decoding error probability rather than the projected code dimensions), but contrary to them the construction of $\check{\mathcal{C}}_{m,t}$ is channel-independent.

Proposition 17. *For any $e_i \in \mathcal{H}_t$ code $\check{\mathcal{C}}^{(\mathbf{e}_i)}_{m,t}$ achieves $\tilde{k}^*_{m-1,t-1}$.*

Consider the generating set of code $\check{\mathcal{C}}_{m,t}$. It contains all monomials of two types:

1. With $\tau_{\mathbf{v}} < \hat{l}$;
2. With $\tau_{\mathbf{v}} = \hat{l}$ and total degree less than $\hat{d}$

for some $\hat{l}, \hat{d}$. When we take the derivative in the direction $\mathbf{e}_i \in \mathcal{H}_t$, the monomials of first type now have less than $\hat{l} - 1$ variables in $\mathcal{M}_t \setminus \{x_i\}$ and the monomials of second type now have exactly $\hat{l} - 1$ variables in $\mathcal{M}_t \setminus \{x_i\}$ and total degree less than $\hat{d} - 1$. The only thing left here is to notice that matches the description of code $\check{\mathcal{C}}_{m-1,t-1}$.

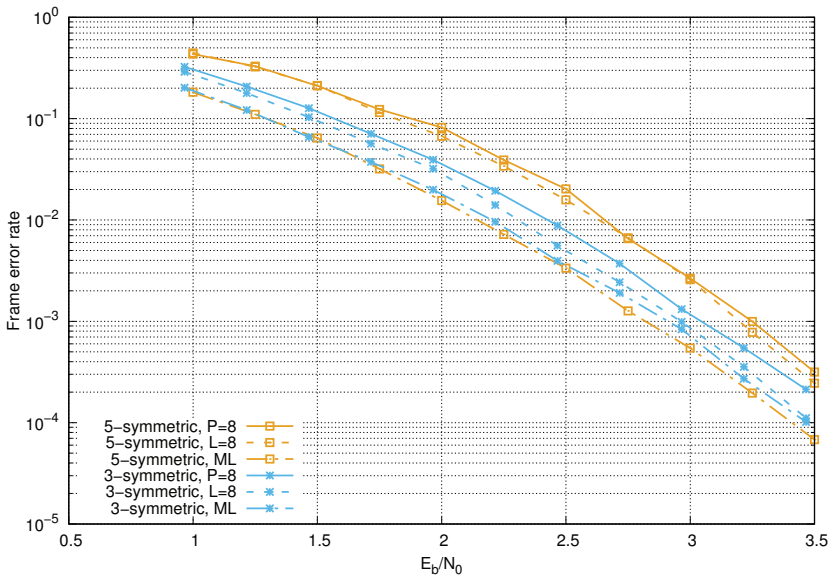


Figure 4.8 – List and permutation decoding performance, $(256, 127)$ 3-symmetric and $(256, 128)$ 5-symmetric monomial codes

4.3.5 Performance of Partially Symmetric Monomial Codes

We consider the transmission via additive white Gaussian noise (AWGN) channel with binary phase shift keying (BPSK) modulation. We compare the list and permutation decoding of optimal $(256, 127)$ 3-symmetric and $(256, 128)$ 5-symmetric monomial codes that are subcodes of RM$(4, 8)$, which are constructed with the proposed algorithm. The set of permutations π_i is selected as in [36], namely by sorting all $m!$ factor graph layer permutations by the SC decoding error probability and picking P smallest such that the Hamming distance between any pair $(\pi_{i'}, \pi_{i''})$ is at least 5 so that they are more likely to correct different error patterns. We observed that this method performs better than randomly choosing from $t!$ layer permutations.

The results are presented at figure 4.8. Despite almost identical ML performance, 3-symmetric codes perform better under SCL decoding. In case of 5-symmetric codes, permutation decoding is as efficient as SCL. A similar behavior for partially symmetric codes is also observed in [43] and [44], where a larger group of permutations is used for the decoding.

We also demonstrate the performance of partially symmetric monomial codes in BEC(ε), where the polynomial-time ML decoding is available. Figure 4.9 shows the frame error rate of codes of length 512 and rate 1/2, constructed for different values of t such that the minimum distance of the constructed codes for $t \leq 7$ is equal to 16 (for $t = 8 = 9$ we get the Reed-Muller code with $d_{min} = 32$). The polar code at the figure is constructed for each value of ε. Its

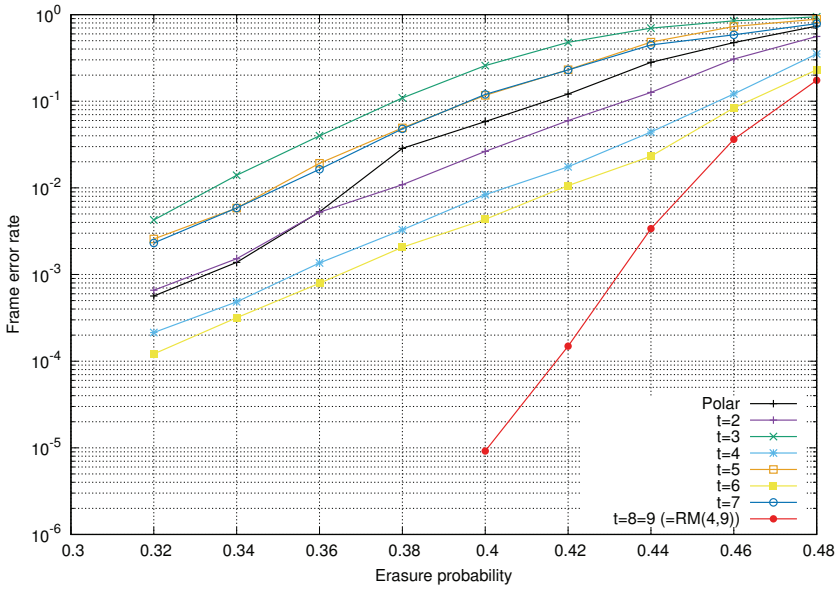


Figure 4.9 – ML performance in BEC, $n = 512, k = 256$.

minimum distance depends on the target erasure probability and jumps from 8 to 16 between $\varepsilon = 0.36$ and $\varepsilon = 0.38$.

One conclusion that can be made from the picture is that the ML performance does not strictly improve with t. However, partially symmetric codes for some values of t demonstrate better performance compared to the polar code, although the gap is not large.

In order to further illustrate the limitations of the projection-based decoding, let us consider a variation of RPA algorithm [52] for decoding arbitrary error-correcting codes in the binary erasure channel, which is summarized in Algorithm 11. For a code $\mathcal{C}$, consider some set $\mathcal{B}$ of the directions. Then in order to recover the erased positions in codeword $(c_0, \dots, c_{2^m-1})$ of $\mathcal{C}$, for each $\boldsymbol{\alpha} \in \mathcal{B}$ we compute the corresponding vectors $\mathbf{y}^{(\boldsymbol{\alpha})}$ and use the bitwise MAP decoder based on the Gaussian elimination in order to (partially) recover the codewords $\mathbf{c}^{(\boldsymbol{\alpha})}$. Any recovered position of $\mathbf{c}^{(\boldsymbol{\alpha})}$ can be written as $c_{\boldsymbol{\beta}} \oplus c_{\boldsymbol{\beta}+\boldsymbol{\alpha}}$, so if only one of the positions $c_{\boldsymbol{\beta}}, c_{\boldsymbol{\beta}+\boldsymbol{\alpha}}$ was erased, we can always recover another one. The process is iteratively repeated until all erasures are recovered. In case if no erasures were corrected in the current iteration, one erasure gets randomly resolved and the decoding proceeds further.

Figure 4.10 shows the performance of the same partially symmetric codes ($n = 512$, rate-1/2, $d_{min} \geq 16$). We choose the set $\mathcal{B}$ to contain 16 (out of 511 total) projections with the smallest dimensions (which we expect are more likely to be decoded successfully). If the ML performance is not achieved, we also plot the corresponding curve.

Algorithm 11 PA decoder for BEC

procedure PADECODEBEC($\mathbf{y}, \mathcal{B}$)
  while There are erasures in $\mathbf{y}$ **do**
    for $\boldsymbol{\alpha} \in \mathcal{B}$ **do**
      for $\boldsymbol{\beta} \in \mathbb{F}_2^m$ **do**
        if $y_{\boldsymbol{\beta}} = \epsilon$ or $y_{\boldsymbol{\beta}+\boldsymbol{\alpha}} = \epsilon$ **then**
          $y_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} \leftarrow \epsilon$
        else
          $y_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} \leftarrow y_{\boldsymbol{\beta}} + y_{\boldsymbol{\beta}+\boldsymbol{\alpha}}$
      $\mathbf{c}^{(\boldsymbol{\alpha})} \leftarrow$DECODEMAP$(\mathcal{C}^{(\boldsymbol{\alpha})}, \mathbf{y}^{(\boldsymbol{\alpha})})$
      for $\boldsymbol{\beta} \in \mathbb{F}_2^m$ **do**
        if $y_{\boldsymbol{\beta}} = \epsilon$ and $y_{\boldsymbol{\beta}+\boldsymbol{\alpha}} \neq \epsilon$ and $c_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} \neq \epsilon$ **then**
          $y_{\boldsymbol{\beta}} \leftarrow c_{\boldsymbol{\beta}}^{(\boldsymbol{\alpha})} + y_{\boldsymbol{\beta}+\boldsymbol{\alpha}}$
    if No erasures were corrected and there exists an erased position y_{e_i} **then**
      Randomly resolve y_{e_i}
  return $\mathbf{y}$

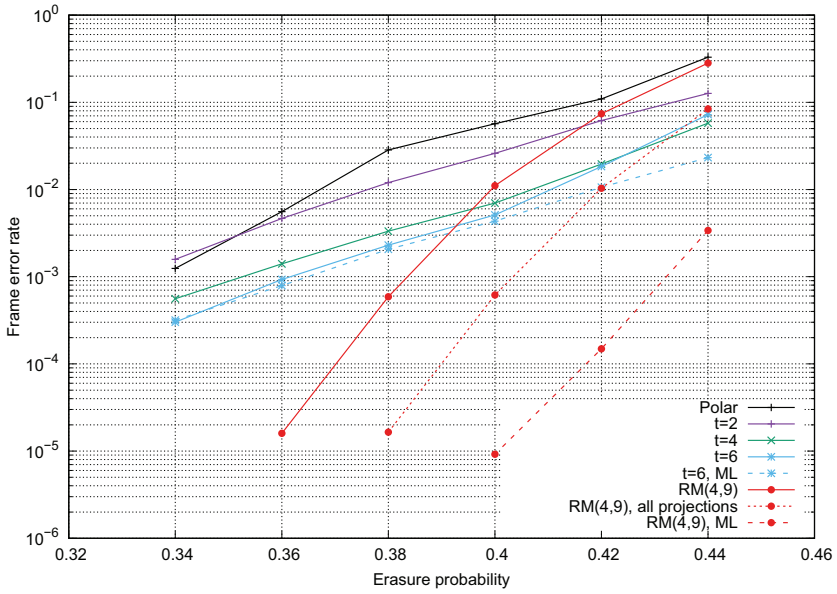


Figure 4.10 – Projection-based decoding in BEC, $n = 512, k = 256$.

It can be seen that for small values of t the projection-based decoding can achieve near-ML performance. However, the projections' dimensions grow with t, so the efficiency of this approach drops down. This is particularly noticeable for the larger values of ε. For the Reed-Muller codes (which are fully symmetric), ML performance is unreachable even when all 511 projections are utilized. Note that the presented partially symmetric codes demonstrate the

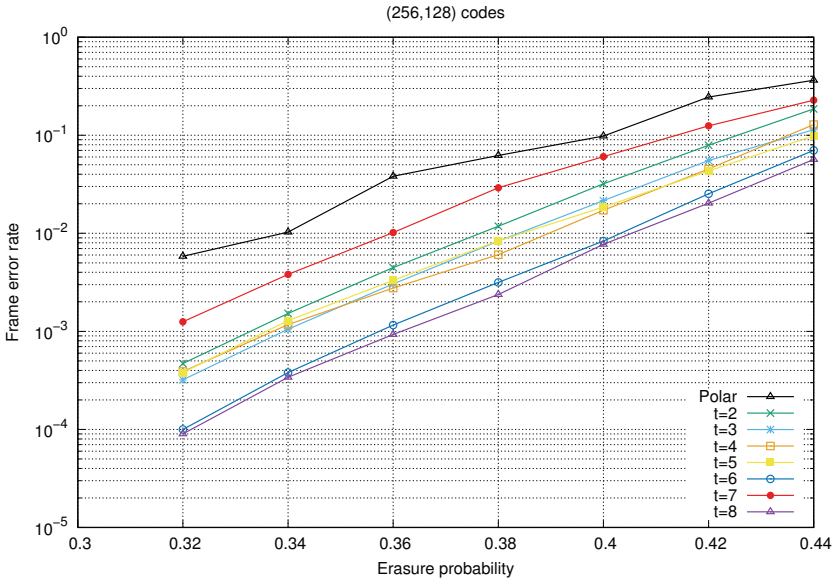


Figure 4.11 – Performance under PA-BEC, part 1

performance superior to the polar code, although the gap is not large.

Figures 4.11 and 4.12 further illustrate the performance of partially symmetric codes with parameters $(256, 128)$ and $(1024, 638)$ when Algorithm 11 is used. As before, 16 projections with the smallest dimensions are selected for decoding. The performance of polar code with the same parameters, constructed for each erasure probability, is also plotted as a reference. All considered codes have $d_{min} = 16$ except 9-symmetric code that is identical to RM$(5, 10)$ with $d_{min} = 32$ and polar code that has $d_{min} = 8$. We can observe that symmetry-based decoding achieves ML performance for $n = 256$. On the other hand, the larger code length leads to near-ML performance only for small values of t and is highly suboptimal for large values of t, especially in the more noisy region. An interesting observation is that polar codes demonstrate ML performance, which implies that a few other projections besides $\mathcal{C}^{(e_0)}$ also contribute to the decoding process.

4.4 Polar Codes Do Not Have Many Affine Automorphisms

Over the past few years, various researchers have studied the automorphism group of polar codes as well as the construction of codes for permutation decoding. These works mostly focus on the subgroups of $GA(m, \mathbb{F}_2)$. In this section, we show how the automorphism groups of polar codes fit into our framework and consequently derive that polar codes cannot be invariant under many affine automorphisms.

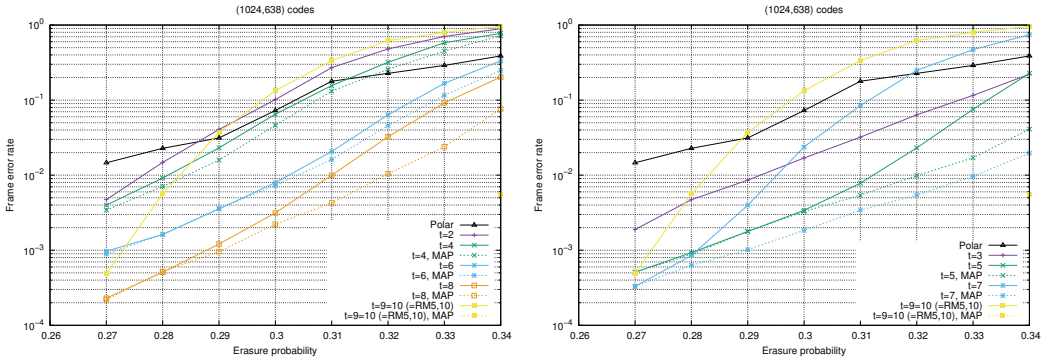


Figure 4.12 – Performance under PA-BEC, part 2

4.4.1 Known Automorphisms of Polar Codes

Definition 3 ([55], Definition 3)**.** *Two monomials of the same degree are ordered as* $x_{i_0} \dots x_{i_{s-1}} \preceq x_{j_0} \dots x_{j_{s-1}}$ *if and only if for all* $l \in [s]$ *holds* $i_l \leq j_l$*. This partial order is extended to the monomials of different degrees through divisibility.*

A monomial code $\mathcal{C}$ is called *decreasing* if for any monomial $x^{\mathbf{v}}$ from $M_{\mathcal{C}}$ all monomials $x^{\mathbf{s}} \preceq x^{\mathbf{v}}$ are also in $M_{\mathcal{C}}$. Theorem 2 in [55] states that an automorphism group of any decreasing monomial code contains the *lower-triangular affine group* $LTA(m, \mathbb{F}_2)$, which is a subgroup of $GA(m, \mathbb{F}_2)$ that includes only matrices $\mathbf{A}$ that are lower-triangular, and polar codes are decreasing monomial codes [55, Theorem 1]. This is an important result that allows to compute the number of minimum-weight codewords of polar codes. However, all permutations from $LTA(m, \mathbb{F}_2)$ correct identical error patterns in the context of SC decoding and therefore cannot bring any performance improvement [62, Theorem 2]. In other words, the group $LTA(m, \mathbb{F}_2)$ is *absorbed by SC decoder*.

The next important step on this road can be attributed to Geiselhart et al., who introduced a larger automorphism group that appears in decreasing monomial codes and is actually useful for the permutation decoding [43]. Namely, decreasing monomial codes are invariant under *block lower-triangular affine group* $BLTA(\mathbf{s}, m)$ for $\mathbf{s} = (s_0, \dots, s_{l-1}), \sum_i s_i = m$, which is another subgroup of $GA(m, \mathbb{F}_2)$, where the matrix $\mathbf{A}$ has form

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{0,0} & & & \mathbf{0} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} & & \\ \vdots & \vdots & \ddots & \\ \mathbf{A}_{l-1,0} & \cdots & & \mathbf{A}_{l-1,l-1} \end{pmatrix}, \tag{4.6}$$

where $\mathbf{A}_{i,j}$ are $s_j \times s_i$ submatrices and all entries that lie above the blocks $\mathbf{A}_{i,i}$ are zero. The entries of vector $\mathbf{s}$ can be determined from $M_{\mathcal{C}}$.

Li et al. later proved that $BLTA(\mathbf{s}, m)$ is equal to the group of affine automorphisms of decreasing monomial codes [80]. Pillet et al. also proved that if $s_0 > 1$, then the group $BLTA((2, 1, \ldots, 1), m)$ is absorbed by SC decoder [81]. The size of $BLTA(\mathbf{s}, m)$ can be computed as

$$|BLTA(\mathbf{s}, m)| = 2^m \prod_{i=0}^{l-1} \left(2^{s_i \gamma_i} \prod_{j=0}^{s_i - 1} \left(2^{s_i} - 2^j \right) \right), \tag{4.7}$$

where $\gamma_i = \sum_{j<i} s_i$ [43]. In case of $\mathbf{s} = (1, \ldots, 1)$ it is equal to $|LTA(m)| = 2^{\frac{m(m-1)}{2} + m}$ and for $\mathbf{s} = (m)$ it coincides with $|GA(m, \mathbb{F}_2)| \approx 0.29 \cdot 2^{m^2+m}$ [56].

4.4.2 New Restrictions on the Size of the Automorphism Group.

The existing results state that the group of affine automorphisms of polar codes is $BLTA(\mathbf{s}, m)$, although no constraints on the values s_i are reported. In this section, we demonstrate that the diagonal blocks cannot grow with m. More precisely, we prove the following result:

Theorem 2 (Polar codes do not have many affine automorphisms)**.** *Consider a BMS channel W and the sequence of polar codes $\{\mathcal{C}^m\}$ of rate $R = I(W)$ with increasing block lengths $n = 2^m$. Then codes $\mathcal{C}^m$ cannot be invariant under $BLTA(\mathbf{s}, m)$ s.t. there exists a block of size s_i that is an increasing function of n.*

Let us first show the correspondence between the invariance under permutations from $BLTA(\mathbf{s}, m)$ and our framework of partially symmetric monomial codes. Consider the matrix $\mathbf{A}'$ which has block-permutation-diagonal form $\mathrm{diag}(\mathbf{P}_0, \ldots, \mathbf{P}_{l-1})$, i.e., all its non-diagonal blocks are zero and all submatrices $\mathbf{A}'_{i,i}$ are some $s_i \times s_i$ permutation matrices $\mathbf{P}_i$:

$$\mathbf{A}' = \begin{pmatrix} \mathbf{P}_0 & & & \mathbf{0} \\ & \mathbf{P}_1 & & \\ \vdots & \vdots & \ddots & \\ \mathbf{0} & \cdots & & \mathbf{P}_{l-1} \end{pmatrix}.$$

Matrix $\mathbf{A}'$ belongs to $BLTA(\mathbf{s}, m)$. Observe that it has a block-permutation structure and consequently is an element of the group $\mathcal{P}_m$ of $m \times m$ permutation matrices. Authors in [43] and [81] already noted that if a code is invariant under $BLTA(\mathbf{s}, m)$, it is s_{l-1}-symmetric, which corresponds to the topmost step of the SC recursion. In fact, we can prove even stronger result:

Theorem 3. *Consider the code $\mathcal{C}$ which is invariant under $BLTA(\mathbf{s}, m)$ and define the set of codes that appear at level $\nu_i = \sum_{j>i} s_j$ of the SC recursion*

$$\phi(\mathcal{C}, i) = \{\mathcal{C}^{(\{-,+\}^{\nu_i})}\}.$$

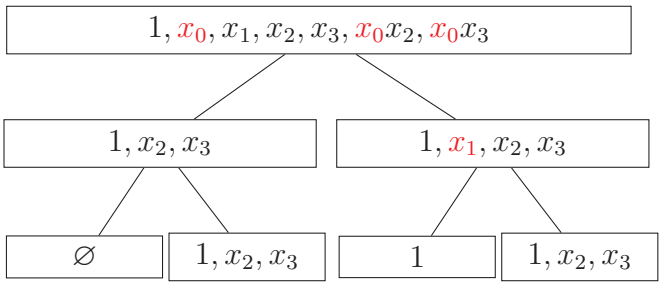


Figure 4.13 – Code decomposition tree

Then for $i < l$ any element of $\phi(\mathcal{C}, i)$ is a s_i-symmetric code.

Proof. Recall that the standard SC schedule implies that the recursion starts from the derivative $\partial/\partial x_0$ and ends with the length-1 codes after taking the derivative $\partial/\partial x_{m-1}$ and the action of group $\mathcal{P}_m$ corresponds to the permutation of variables $\{x_0, \dots, x_{m-1}\}$ or, equivalently, to the change of order in which the SC decoder takes the derivatives. At level ν_i of the standard SC schedule we have a code $\mathcal{C}^{(\{-,+\}^{\nu_i})}$ that is by definition an element of $\phi(\mathcal{C}, i)$ and we need to take the derivative w.r.t. variable x_{ν_i+1}. The invariance under permutations of form $\mathrm{diag}(\mathbf{I}_0, \dots, \mathbf{I}_{i-1}, \mathbf{P}_i, \mathbf{I}_{i+1}, \dots, \mathbf{I}_{l-1})$ tells us that the derivative w.r.t. any of s_i variables $\{x_{\nu_i+1}, \dots, x_{\nu_i+s_i}\}$ gives the same code and therefore $\mathcal{C}^{(\{-,+\}^{\nu_i})}$ is a s_i-symmetric code. □

Example 5. *Consider the $(16, 9)$ code $\mathcal{C}$ with*

$$M_{\mathcal{C}} = \{1, x_0, x_1, x_2, x_3, x_0x_2, x_0x_3\},$$

which is invariant under $BLTA((2, 1, 1), 4)$. Figure 4.13 demonstrates the generating sets of codes $\mathcal{C}^{(--)}, \mathcal{C}^{(-+)}, \mathcal{C}^{(+-)}$ and $\mathcal{C}^{(++)}$ that appear at the second level of SC recursions and form the set $\phi(\mathcal{C}, 0)$. We have $s_0 = 2$ and it is easy to verify that all these codes are invariant under the permutations on variables x_2 and x_3, i.e., are 2-symmetric.

Now we use this result in order to prove Theorem 2.

4.4.3 Proof of Theorem 2.

We prove by contradiction. Assume that code $\mathcal{C}$ is invariant under the action of group $BLTA(\mathbf{s}, m)$ with a diagonal block of size s_i that grows with m. Then from Theorem 3 it follows that any code $\mathcal{C}^{(\mathbf{j})} \in \phi(\mathcal{C}, i), \mathbf{j} \in \{-, +\}^{\nu_i}$ is s_i-symmetric and using (4.5) we conclude that the rate of $\mathcal{C}^{(-\mathbf{j})}$ is smaller that the rate of $\mathcal{C}^{(\mathbf{j})}$ by at most $O\left(\frac{1}{\sqrt{s_i}}\right)$.

Recall that a polar code by construction is a Plotkin concatenation of two polar codes for the corresponding synthetic channels $W^{(-)}$ and $W^{(+)}$, where

$I(W^{(-)}) < I(W)$ and $I(W^{(-)}) > I(W)$ when $I(W) \notin \{0, 1\}$. Applying this definition recursively, we get that code $\mathcal{C}^{(\mathbf{j})}$ needs to be a polar code constructed for the channel $W^{(\mathbf{j})}$ and its derivative w.r.t. x_{ν_i} is a polar code for the channel $W^{(-,\mathbf{j})}$. We know from [82] that the ratio of non-perfectly polarized synthetic channels scales as $\Theta(2^{-m/\mu})$, where μ is a polar scaling exponent. The code $\mathcal{C}^{(\mathbf{j})}$ has length $2^{m-\nu_i}$ and therefore the difference between $R(\mathcal{C}^{(-,\mathbf{j})})$ and $I(W^{(-,\mathbf{j})})$ is at most $\Theta(2^{-(m-\nu_i)/\mu})$.

Let us now compare the statements from two paragraphs. Partial symmetry of code $\mathcal{C}^{(\mathbf{j})}$ gives us an upper bound

$$R(\mathcal{C}^{(\mathbf{j})}) - R(\mathcal{C}^{(-,\mathbf{j})}) \leq O\left(\frac{1}{\sqrt{s_i}}\right). \tag{4.8}$$

On the other hand, from the nested property of polar codes it follows that

$$R(\mathcal{C}^{(\mathbf{j})}) - R(\mathcal{C}^{(-,\mathbf{j})}) \geq I(W^{(\mathbf{j})}) - I(W^{(-,\mathbf{j})}) - \Theta(2^{-(m-\nu_i)/\mu}). \tag{4.9}$$

We can use again the result from [82] that states that for any interval $[a, b]$, where $a, b \in (0, 1)$, the number of channels with capacities that fall into this interval is $\Theta(2^{m(1-1/\mu)})$. Therefore, for any interval $[a, b]$ there exists a channel $W^{(\mathbf{j})}$ s.t. $I(W^{(\mathbf{j})}) \in [a, b]$ and consequently

$$I(W^{(\mathbf{j})}) - I(W^{(-,\mathbf{j})}) \geq \min_{\hat{W}: I(\hat{W}) \in [a,b]} \left(I(\hat{W}) - I(\hat{W}^{(-)}) \right) > 0,$$

where the second inequality holds trivially for any fixed pair of values $a, b \in (0, 1)$. Hence, we have an upper bound (4.8) that converges to zero because by assumption s_i is an increasing function of m and a lower bound (4.9) that is bounded away from zero, which gives us the contradiction.

Coding for Device-Independent Quantum Key Distribution 5

This is a joint work with D. P. Nadlinger, P. Drmota, B. C. Nichol, G. Araneda, D. Main, R. Srinivas, D. M. Lucas, C. J. Ballance, E. Y-Z. Tan, P. Sekatski, R. Renner, N. Sangouard and J-D. Bancal. The preprint version of this work can be found in [83].

Private communication over shared network infrastructure is of fundamental importance to the modern world. Classically, shared secrets cannot be created with information-theoretic security; real-world key exchange protocols rely on unproven conjectures about the computational intractability of certain operations. Quantum theory, however, promises that measurements on two correlated quantum systems can yield identical outcomes that are fundamentally unpredictable to any third party. This possibility of generating secret correlated outcomes at a distance forms the basic idea of quantum key distribution (QKD) [84, 85, 86]. Importantly, the security guarantees provided by QKD are unique in that they do not rely on the assumption that the adversary has limited computational power. Rather, the only required assumptions are that (i) quantum theory is correct, (ii) the parties can isolate their systems to prevent information leaking to an adversary, (iii) they can privately choose random classical inputs to their quantum devices, and (iv) they can process classical information on trusted computers.

Existing QKD systems rely on an additional assumption that is hard to satisfy in practice: they require the devices used to distribute the key to be accurately characterised [85, 86]. Generally, both the structure of the quantum states involved (e.g., their dimension) and the nature of the measurements [87] must be accurately known, i.e., the quantum devices are assumed to be trusted and to maintain perfect calibration. Deviations from the expected

behaviour can be difficult to detect, which has been exploited in a number of demonstrations where real-world QKD systems were compromised [88, 89, 90, 91, 92]. So-called measurement-device-independent QKD protocols permit untrusted measurement devices to be used as part of the system but still require well-characterised sources [93, 94, 95, 96, 97].

Device-independent QKD (DIQKD) protocols [98, 99, 100, 101] make no additional assumptions about the physical apparatus. According to Bell's theorem, we can guarantee that two systems produce outcomes that share exclusive correlations without knowing how these outcomes are produced, while preventing a third party from knowing these results [102]. This remarkable fact can be used to construct key distribution protocols with security guarantees independent of any assumption about the inner workings of the quantum devices. Imperfections, which might lead to key leakage in conventional QKD, instead just result in the protocol aborting. This enhanced security, however, comes at the cost of far more stringent experimental requirements. The certifiable amount of private information directly depends on the size of a Bell inequality violation, necessitating a platform capable of distributing and measuring high-quality entangled states and closing the detection loophole. To successfully extract a shared key by using state-of-the-art devices, a tight theoretical analysis and an efficient classical post-processing pipeline are needed, in particular, regarding finite-size effects resulting from practical limits on the number of measurements. Despite significant theoretical progress [103, 99, 104, 105, 100, 101, 106, 107, 108, 109, 110, 111] a practical demonstration of these protocols has remained out of reach.

The work [83] reports the first experimental demonstration of device-independent quantum key distribution with a concrete protocol. Our contribution is the design of an error-correction scheme that is required at the post-processing step in order to ensure that both parties have the identical keys. In the subsequent sections, we first introduce the DIQKD protocol then proceed to the description of the error-correction scheme.

5.1 Device-Independent Quantum Key Distribution

Our DIQKD protocol is summarized in Fig. 5.1. It follows a similar general structure, as most DIQKD protocols proposed thus far [112, 113, 101]. The protocol involves two distant parties, Alice and Bob, who initially share a secret key $\mathbf{K}_0$. The purpose of the protocol is to create a shared secret key $\mathbf{K}_1$ that is longer than $\mathbf{K}_0$ by performing $n \in \mathbb{N}$ successive measurements on distributed quantum systems that are followed by post-processing steps. The protocol is successful when both parties are confident that their copy of $\mathbf{K}_1$ is sound. The protocol depends on the following parameters that are fixed before its start: the

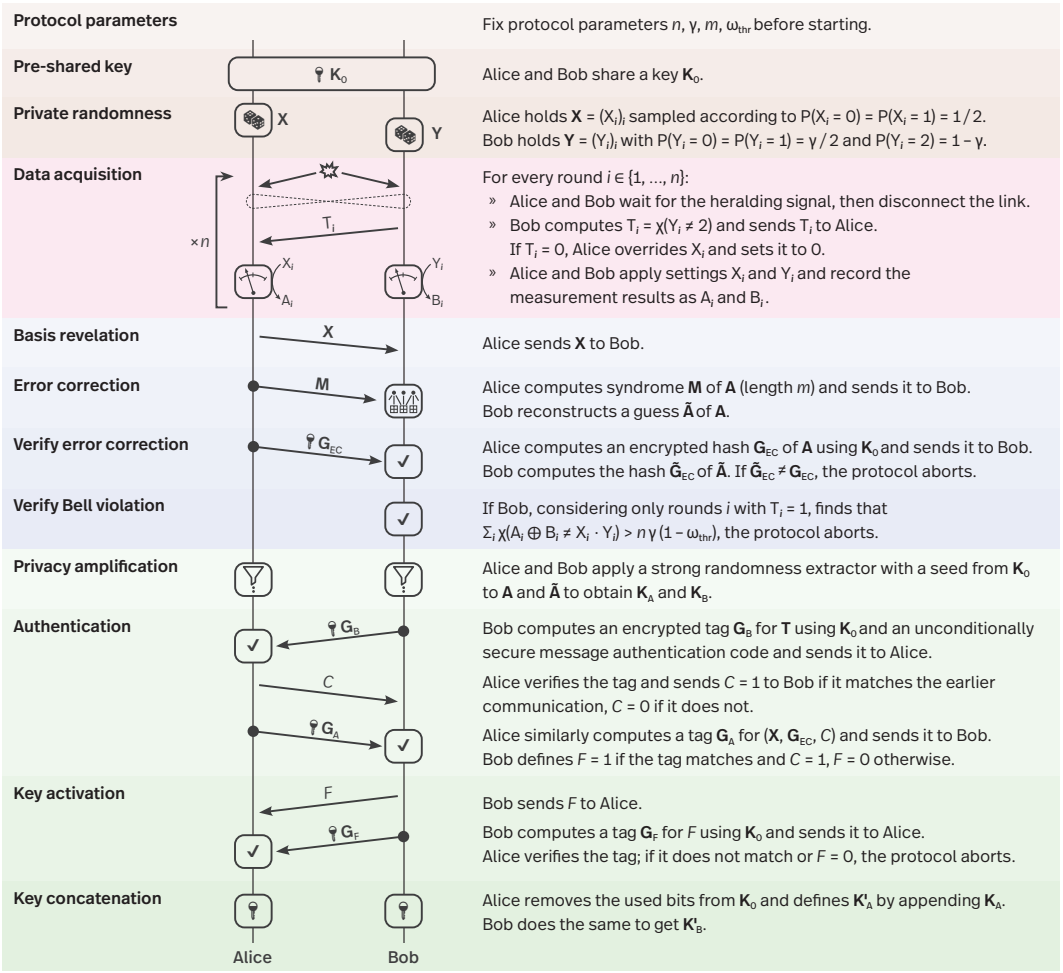


Figure 5.1 – **DIQKD protocol structure.** Before the protocol begins, the number of rounds n, the probability γ of each round being chosen to be Bell test round, an acceptance threshold ω_{thr} for the CHSH winning probability, and the length m of the error-correction syndrome are fixed. An initial key $\mathbf{K_0}$, mostly reusable, is required to seed the privacy amplification and authentication algorithms and is also used as a one-time pad to encrypt a few short messages (indicated using a key symbol). The arrows indicate the classical messages exchanged between the parties, bold-letter strings consisting of multiple bits, χ the indicator function with $\chi(P) = 1$ if P is true and 0 otherwise. Alice's input $X_i = 0$ is chosen both for the key and test generation rounds, whereas Bob's input $Y_i = 2$ is used exclusively for the key generation rounds.

testing probability $\gamma \in (0, 1)$, the total number of rounds $n \in \mathbb{N}$, the threshold CHSH winning probability $\omega_{\text{thr}} \in \left(\frac{3}{4}, \frac{1+1/\sqrt{2}}{2}\right]$, and the syndrome length m. The protocol has three phases that are denoted in Fig. 5.1 in different colors.

The first phase of the protocol is the data acquisition, which consists of n sequential rounds. Any round is either a Bell test round or a key generation

round. For the former, Alice and Bob randomly select inputs $X_i, Y_i \in \{0,1\}$ that are implemented so that the outcomes $A_i, B_i \in \{0,1\}$ maximize the probability of winning the CHSH game[114] $A_i \oplus B_i = X_i \cdot Y_i$. A winning probability ω, customarily expressed in terms of the CHSH score $S = 4\,(2\omega - 1)$, tightly bounds the information any adversary can have about the outcomes. For the latter, the inputs are fixed to $X_i = 0$ and $Y_i = 2$, maximising output correlations as quantified by a low quantum bit error rate $Q = P(A_i \neq B_i | X_i = 0, Y_i = 2)$. Bell test rounds contribute to the security of the protocol, but the corresponding outcomes A_i, B_i are weakly correlated, which increases the amount of information that needs to be revealed. On the contrary, key generation rounds do not participate in the security but have a much higher correlation between A_i and B_i, hence less information is needed for the successful reconciliation. Bob randomly chooses between the round types after the links are disconnected, choosing Bell test rounds with probability γ, and communicates this choice to Alice, which avoids sifting (discarding of rounds with mismatched measurement bases). The parties keep private records of their measurement settings $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_n)$, as well as the outcomes $\mathbf{A} = (A_1, \ldots, A_n)$ and $\mathbf{B} = (B_1, \ldots, B_n)$.

In the second phase, Alice and Bob need to verify the CHSH score and to extract a shared key from the noisy output correlations. Alice openly sends her inputs $\mathbf{X}$ to Bob, along with an extra string $\mathbf{M}$, which enables Bob to reconstruct a copy $\tilde{\mathbf{A}}$ of Alice's outcomes $\mathbf{A}$. Now holding $\mathbf{XY\tilde{A}B}$, Bob is able to verify whether the CHSH score achieved during the Bell test rounds exceeds a pre-agreed threshold. If this is not the case, or if the reconstruction of $\tilde{\mathbf{A}}$ fails, the security might be compromised, hence the protocol aborts. Otherwise, the parties proceed to the third phase and locally process the outcomes $\mathbf{A}$ and $\tilde{\mathbf{A}}$ to extract identical final keys, with a guaranteed, arbitrarily low, bound on the information leaked to any adversary.

The length of the final key depends on the amount of information revealed, as part of the protocol, through the classical messages exchanged by the parties. Therefore, the length of string $\mathbf{M}$ should be as small as possible. Asymptotically, the cost of reconciliation per round is given by $\mathrm{H}(\mathbf{A}|\mathbf{XYB})$; this is the entropy of Alice's outcomes conditioned on the knowledge of inputs and Bob's outcomes. This reconciliation is in the presence of finite statistics. However, it comes at a high price, and the best algorithms are often not realizable in practice. In fact, all finite-statistics DIQKD analyses have, so far, assumed computationally intractable error-correction schemes with decodings involving hash pre-image computations [100, 101, 115, 116].

X_i \ Y_i	0	1	2
0	$\begin{pmatrix} 0.415(1) & 0.0688(5) \\ 0.0961(6) & 0.420(1) \end{pmatrix}$	$\begin{pmatrix} 0.439(1) & 0.0805(6) \\ 0.0735(5) & 0.4068(10) \end{pmatrix}$	$\begin{pmatrix} 0.5017(5) & 0.00339(6) \\ 0.0110(1) & 0.4839(5) \end{pmatrix}$
1	$\begin{pmatrix} 0.3928(10) & 0.0916(6) \\ 0.0851(6) & 0.431(1) \end{pmatrix}$	$\begin{pmatrix} 0.0820(6) & 0.437(1) \\ 0.3970(10) & 0.0841(6) \end{pmatrix}$	—

Table 5.1 – Empirical probabilities of observing classical outcome $(A_i,\ B_i)$ for measurements settings $(X_i,\ Y_i)$. Matrix rows represent Alice's outcomes, and columns those of Bob. The probabilities are estimated from the 1 920 000 total characterisation rounds, with the multinomial standard errors given in parentheses. Alice's classical outcomes were inverted, giving maximum correlations for the key generation settings $X = 0, Y = 2$ and maximizing the probability that $A_i \oplus B_i = X_i \cdot Y_i$ in Bell test rounds.

5.2 Data model

Table 5.1 demonstrates the experimental results after 1 920 000 data acquisition rounds. We have four distributions $P(A, B|0, 0)$, $P(A, B|0, 1)$, $P(A, B|1, 0)$ and $P(A, 1 - B|1, 1)$ corresponding to the test rounds and one distribution $P(A, B|0, 2)$ that corresponds to the key rounds. It is reasonable, from practical point of view, to make a simpler model. The distributions corresponding to the key rounds are similar to each other, and we condense them, i.e., we assume that samples during the key rounds come from the same distribution

$$P'(A, B) = (P(A, B|0, 0) + P(A, B|0, 1) + P(A, B|1, 0) + P(A, 1 - B|1, 1))/4,$$

and we also define $P''(A, B) = P(A, B|0, 2)$. For numeric simulations, we also consider even simpler model, which is characterized two parameters S and Q, respectively, corresponding to the CHSH Bell violation and quantum bit error rate (QBER). In this model, we assume that the statistics in the test rounds satisfy

$$P(A, B|0, 0) = P(A, B|0, 1) = P(A, B|1, 0) = P(A, 1 - B|1, 1) = \frac{1 + (-1)^{A\oplus B} S/4}{4} \quad (5.1)$$

and that the statistics in the key rounds are sampled according to

$$P(A, B|0, 2) = \frac{\delta_{A,B} - (-1)^{A\oplus B} Q}{2}. \quad (5.2)$$

5.3 Error Correction in DIQKD

In the second phase of the DIQKD protocol, described in Section 5.1, Alice and Bob end up with their measurements A_i and B_i, and Alice sends her measurement settings X_i. We also assume that Alice also sends a single message $\mathbf{M}$ of length m, and the goal of Bob is to recover the string $\mathbf{A}$ from the knowledge of $\mathbf{B}, \mathbf{X}, \mathbf{Y}$ and $\mathbf{M}$. The length m of the communicated message

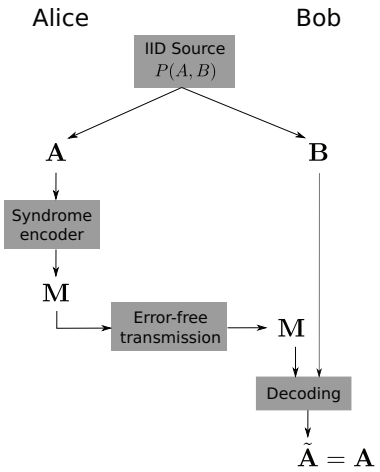


Figure 5.2 – Overall error-correction setting: asymmetric Slepian-Wolf coding. Strings $\mathbf{A}$ and $\mathbf{B}$ are jointly sampled from a distribution $P(A, B)$ and given to two distinct parties, Alice and Bob. The purpose is for Bob to end up with a copy of Alice's string $\mathbf{A}$, while exchanging only a 'short' message $\mathbf{M}$.

has to be as small as possible, as it is revealed in public hence reduces the Alice's key secrecy.

The setting is equivalent to a Slepian-Wolf scheme with asymmetric coding [117], which also states that the *overhead* $\eta = m/n$ can be as small as $H(A|B)$. In [118] it is shown that this limit is achievable with binary linear codes by using the simple procedure presented at Fig. 5.2. $\mathbf{A}$ and $\mathbf{B}$ are considered as samples from the correlated source, which is described by the joint probability distribution $P(A, B)$. Consider a noisy channel transforming A into B, described by the conditional distribution $P(B|A) = \frac{P(A,B)}{P(A)}$ and a binary linear code $\mathcal{C}$, which achieves the capacity of this channel, of length n and rate $R = 1 - m/n$. The message $\mathbf{M}$ is generated by taking a $m \times n$ parity check matrix $\mathbf{H}$ of $\mathcal{C}$ and computing the *syndrome* $\mathbf{M} = \mathbf{HA}$. Then $\mathbf{A}$ can be almost always recovered from the pair $(\mathbf{B}, \mathbf{M})$. Note that this formulation was initially proved for the case, when $P(B|A)$ corresponds to a symmetric channel and the input distribution $P(A)$ is uniform. However, the recent result [119] confirms the achievability of $H(A|B)$ limit for the general setting, i.e., for arbitrary distributions $P(A)$ and $P(B|A)$.

The challenge here is to develop an error-correcting code for the given $P(A, B)$ that performs close to the asymptotic limits under low-complexity decoding algorithm. In Section 5.2, we introduce the simplified model for describing $P(A, B)$. Therefore, another requirement is that the code is rather insensitive to the noise specificities. The universality of the code, which makes it optimal for all distributions $P(A, B)$ with the same value of $H(A|B)$, is highly desirable.

5.3.1 Concrete Setting

As discussed in Sec. 5.2, the honest model for our setup includes five sources of outcomes for Alice and Bob, corresponding to each choice of measurement setting; and the corresponding distributions are combined so that we have only $P'(A,B)$ that corresponds to the test rounds and $P''(A,B)$ that corresponds to the key rounds. In the case when P' and P'' are parameterized in terms of the Bell value S and the QBER Q only, this model defines the transmission through a mixture of two BSCs with different crossover probabilities. The Bell test rounds correspond to a BSC with crossover probability $\delta' = \frac{4-S}{8}$ and are on average sampled γn times. Key generation rounds correspond to a BSC with crossover probability $\delta'' = Q$ that is on average sampled $(1-\gamma)n$ times. Therefore, we need to deal with the transmission of approximately γn samples through a BSC with bit flip probability δ' and with approximately $(1-\gamma)n$ samples through a BSC with bit flip probability δ''. Although the exact number of samples transmitted through each channel is a random variable, the total number of the transmitted samples is guaranteed to be n.

Remark. Note the difference with a classical coding setting, where a fixed-length bit string is transmitted through the channel.

The simplified model enables us to estimate the minimum achievable length m of the syndrome $\mathbf{M}$ for a finite number of samples n. Assume that the transmission takes place through BSC(δ) and that the error-correction scheme fails with probability ε. Then, [120, Eq. (289)] gives the effective finite-size channel capacity $C(n,\delta)$ as

$$\begin{aligned} n \cdot C(n,\delta) =& n(1-h(\delta)) - \sqrt{n\delta(1-\delta)}\log_2\frac{1-\delta}{\delta}Q^{-1}(\varepsilon) \\ &+ \frac{1}{2}\log_2(n) + O(1), \end{aligned} \tag{5.3}$$

where n is the length of the "transmitted" bit string (which in our model is $\mathbf{B}$), $h(x)$ is the binary entropy function, ε is the error-correction failure probability and $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = \frac{1}{2}\left(1 - \mathrm{erf}(x/\sqrt{2})\right)$. The minimum length of the syndrome is then

$$m_{\mathrm{BSC}}(n,\delta) = n(1 - C(n,\delta)). \tag{5.4}$$

Therefore, for our mixture of BSC(δ') and BSC(δ''), the estimated minimum syndrome length is

$$\tilde{m}(n) = m_{\mathrm{BSC}}(\gamma n, \delta') + m_{\mathrm{BSC}}((1-\gamma)n, \delta''). \tag{5.5}$$

In practice, we can choose one of two strategies. We can either construct two codes $\mathcal{C}'$ and $\mathcal{C}''$ to independently process strings that correspond to BSC(δ')

and BSC(δ''), or we can use a single code $\mathcal{C}$ and jointly handle these strings. From (5.3), it is clear that the larger length reduces the required overhead for error-free performance, hence we can expect joint decoding to perform better in practice. Moreover, in our protocol the number of samples corresponding to BSC(δ') is a random variable with mean γn hence, during the protocol after the number of test rounds is established, we need to adapt the length of codes $\mathcal{C}'$ and $\mathcal{C}''$. Using a single code solves this problem since the total number of rounds n is fixed and therefore the length of $\mathcal{C}$.

5.4 Practical Coding Approaches

For a practical application, we need an error-correcting code that is capacity-achieving under low-complexity decoding algorithm. As we operate in the finite-length regime, we would also like to achieve a finite-length performance close to theoretical limits. The simplicity of the code construction is an additional advantage, due to the possible changes in the experimental setting. There are three reasonable choices that we can use: low-density parity-check (LDPC) codes, spatially-coupled LDPC (SC-LDPC) codes, and polar codes.

Polar codes [25] have low-complexity encoding and decoding procedures, but the codes have several issues. Perhaps most importantly, the code construction is channel-dependent, i.e., any change in γ, Q, S or the significant difference in actual distributions $P'(A,B)$ and $P''(A,B)$ from our model will require a separate optimization step. Another problem is that, although the optimal code has gap to the asymptotic value that scales as $\Theta(\frac{1}{\sqrt{n}})$ (it follows from (5.3)), for polar codes it is at least of order $\Theta(\frac{1}{n^{1/3.579}})$ [121]. This is significantly worse and implies that their finite-length performance is rather far from theoretical limits.

The LDPC codes were introduced in the early 1960s [12]. They are described as dual spaces of some sparse matrices that typically have a simple structure. The sparsity of this matrix gives rise to a low-complexity iterative belief propagation (BP) decoding algorithm. It is possible to compute the asymptotic performance of LDPC codes under BP decoding [18]. Although LDPC codes can achieve the capacity of BSC under computationally unfeasible maximum likelihood (ML) decoding [122], their *BP thresholds*, i.e., the largest channel error probability that can be corrected by BP decoder with infinite number of iterations, are typically not capacity-achieving. There exist some code constructions that come close to the asymptotic limits [123], but the optimization for certain channels (such as a mixture of two BSCs in our setting) might be nontrivial. In the case of the binary erasure channel (the receiver either gets the transmitted bit correctly with probability $1-\varepsilon$ or receives an *erasure symbol* "ϵ" with probability ε), the LDPC codes achieve capacity with linear decoding complexity [124, 125, 126]. The optimal scaling behavior of

the LDPC codes is established for the erasure channel and conjectured for the general channels [127].

SC-LDPC codes are constructed as a chain of LDPC codes that are coupled together. It was observed numerically in [22] and proved in [24] that the BP threshold of the SC-LDPC codes converges to the ML threshold of the underlying LDPC code. Furthermore, they achieve capacity universally, i.e., a code of rate R enables the error-free transmission through any channel with a capacity greater than R when the code length grows to infinity. Another advantage of the SC-LDPC codes is the simplicity of the code construction, as it is much easier to find the LDPC codes that are capacity achieving under ML decoding than to optimize LDPC codes so that their BP threshold is close to capacity. Concerning the scaling of the SC-LDPC codes, [128] gives an estimate of $\Theta(\frac{1}{n^{1/3}})$ by using a heuristic argument, but nothing else is known. As the SC-LDPC codes build on LDPC codes, in the following paragraphs, we describe the LDPC codes then proceed to the SC-LDPC codes.

5.4.1 LDPC Codes

An LDPC code of length n and dimension k is defined as a dual space of binary $(n-k) \times n$ full-rank parity-check matrix $\mathbf{H}$ that needs to be sparse. A code is called (d_v, d_c)-*regular* if each row of $\mathbf{H}$ contains exactly d_c ones and each column exactly d_v ones, otherwise it is irregular. The matrix $\mathbf{H}$ is typically represented as a Tanner graph that is a bipartite graph that consists of *check* and *variable* nodes that correspond to the rows and columns of $\mathbf{H}$. If $H_{i,j} = 1$, then check node i and variable node j are connected by an edge. In Tanner graph representation, d_v and d_c become the degrees of variable and check nodes.

One way to construct good LDPC codes of variable length is to use a protograph construction [129]. A protograph with *design rate* $R = 1 - n_c/n_v$ is a small bipartite graph with partitions of size n_c and n_v that correspond to check and variable nodes, respectively. A code of length $n = Mn_v$ is obtained by applying a *lifting* procedure with *lifting factor* M. This procedure can be described as follows. Take the $n_c \times n_v$ biadjacency matrix $\mathbf{Z}$ of the protograph and replace its zeros with $M \times M$ all-zero matrices and ones with random $M \times M$ permutation matrices $\Pi_{i,j}$. The resulting code has length Mn_v and rate $R \geq 1 - n_c/n_v$. In practice, for random matrices $\Pi_{i,j}$ this inequality almost always becomes an equality [129]. The advantage of the protograph-based LDPC codes is that their properties, such as BP decoding threshold, can be derived directly from protographs [129].

Example 6. *Consider the* $(2, 3)$*-regular LDPC code. The corresponding protograph is given at Figure 5.3, and its biadjacency matrix can be written as* $\mathbf{Z} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. *The lifting process for* $M = 3$ *is given at figures 5.4a-5.4b.*

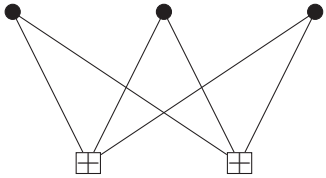
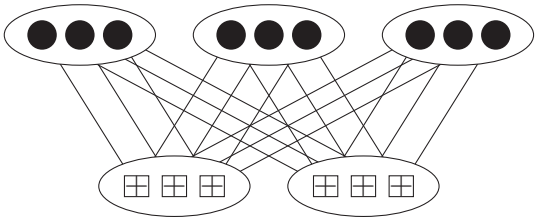
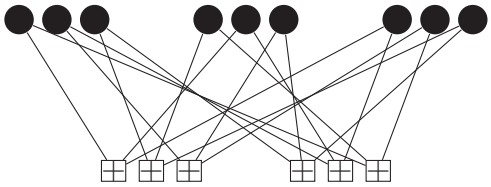


Figure 5.3 – (2,3)-regular protograph. Three vertices are connected to two check nodes.



(a) Lifting with $M = 3$: vertices and check nodes are split into three copies, with connections following the original topology.



(b) Tanner graph of lifted $(2, 3)$-regular LDPC code. The mapping between check and variable nodes, within connections compatible with the protograph, is random.

5.4.2 SC-LDPC Codes

A spatially coupled protograph is obtained by chaining together L copies of LDPC protographs, where L is called the *coupling factor*. We assign time index t to each of these copies. Suppose a check node c_i and a variable node v_j in a protograph are connected with $Z_{i,j}$ edges. Then we spread these edges forward, i.e., we connect $Z_{i,j}$ edges from node v_j at time t to check nodes c_j at time $t, t+1, \ldots, t+w$, where w is called *coupling width*. For a regular protograph, it is reasonable to take $w = d_v - 1$. Figures 5.5 and 5.6 illustrate the process for a (3,6)-regular protograph. An SC-LDPC code is then obtained by applying the lifting procedure described earlier for a spatially-coupled protograph.

Let us now compute the rate of the obtained SC-LDPC code. We take L copies of the original protograph and chain them together. The number of variable nodes is now fixed hence the code length is Ln_v. A coupling width w means that variable nodes of the protograph at position t are also connected to the check nodes of protographs at positions $t+1, \ldots, t+w$, hence we need to add extra wn_c check nodes in addition to the Ln_c that we already have. Therefore, the resulting SC-LDPC code has rate

$$R = 1 - \frac{(L+w)n_c}{Ln_v} = 1 - \frac{(1+(d_v-1)/L)n_c}{n_v}. \tag{5.6}$$

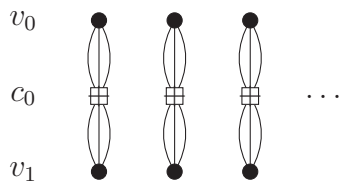


Figure 5.5 – Uncoupled (3,6)-regular protographs. Each variable node has degree 3 and each check node has degree 6. Although there are duplicate edges in the protograph, there will be none in the code after the coupling and lifting steps.

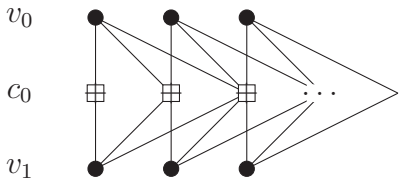


Figure 5.6 – Spatially coupled (3,6)-regular LDPC ensemble, $w = 2$

5.4.3 Belief Propagation Decoding

Assume that we constructed some LDPC code $\mathcal{C}$ with a parity check matrix $\mathbf{H}$. Recall that, in our setting, Bob has the string $\mathbf{B} = (B_1, \ldots, B_n)$ and wants to reconstruct the string $\mathbf{A} = (A_1, \ldots, A_n)$ from the knowledge of $\mathbf{B}$, the syndrome $\mathbf{M} = (M_1, \ldots, M_m) = \mathbf{HA}$, and from the joint distribution $P(A, B)$. Therefore, the goal of the decoder is to find

$$\tilde{\mathbf{A}} = \arg \max_{\hat{\mathbf{A}}:\ \mathbf{H}\hat{\mathbf{A}}=\mathbf{M}} \prod_{i=1}^{n} P(A = \hat{A}_i, B = B_i | X_i, Y_i). \tag{5.7}$$

The straightforward procedure is to check all possible codewords of $\mathcal{C}$ and to return the most probable. However, the number of codewords increases as 2^k, where k is the code dimension. This quickly makes such a procedure unfeasible, hence there is a need for low-complexity algorithms.

Belief propagation is an iterative algorithm that makes use of the sparsity of the parity-check matrix $\mathbf{H}$ and, for constant (d_v, d_c), has running time $O(N_{\text{it}} \cdot n)$, where N_{it} is a number of decoding iterations performed. Let

$$\mathbf{l} = (l_1, \ldots, l_n), \quad l_i = \ln \frac{P(A_i = 0, B_i)}{P(A_i = 1, B_i)} \tag{5.8}$$

be the logarithmic reliability ratios vector corresponding to the vector $\mathbf{B}$. The prior knowledge of which bits have error probability δ' and which have error probability δ'' is incorporated in l_i by appropriately setting the corresponding probabilities $P(A_i = 0, B_i)$ and $P(A_i = 1, B_i)$. The recovery of $\tilde{\mathbf{A}}$ can be

performed by an iterative exchange of messages between variable and check nodes of the Tanner graph of the code. The messages at iteration i are defined as follows [130, 131]:

$$\mu_{v\to c}^{(i)} = \begin{cases} l_v, \ i = 0 \\ l_v + \sum_{c' \in C_v \setminus \{c\}} \mu_{c'\to v}^{(i)}, \ i > 0 \end{cases} \tag{5.9}$$

$$\mu_{c\to v}^{(i)} = 2 \tanh^{-1} \left((-1)^{M_c} \prod_{v' \in V_c \setminus \{v\}} \tanh(\mu_{v'\to c}^{(i-1)}/2) \right), \tag{5.10}$$

where the set C_v contains indices of check nodes, incident to the variable node v, and V_c contains indices of variable nodes, incident to the check node c. After N_{it} iterations we can obtain the string $\tilde{\mathbf{A}} = (\tilde{A}_1, \ldots, \tilde{A}_n)$ as follows:

$$\tilde{A}_v = \frac{1 - x_v}{2}, x_v = \text{sgn}(l_v + \sum_{c \in C_v} \mu_{c\to v}^{(N_{it})}), \tag{5.11}$$

where x_v are the hard-decision values corresponding to the posterior log-likelihood ratios from the decoder. For the SC-LDPC codes with underlying (d_v, d_c)-regular LDPC codes, if $d_v \geq 3$, the error probability is expected to decrease doubly exponentially with decoding iterations [129]. There exist various improvements of this algorithm in the literature for general LDPC codes [132, 133], as well as for SC-LDPC codes [134, 135]. However, in the literature, there is no analysis of the behaviour of the SC-LDPC codes under such decoders for non-erasure channels, hence we use the standard BP algorithm, which is provably good, for our experiments. The similar lack of theoretical analysis exists with early termination schemes that seek to find a criterion when the decoding process can be stopped.

5.5 Simulations

In the asymptotic setting with infinite block length, the smallest achievable overhead is a Slepian-Wolf limit

$$\tilde{\eta}^{\infty} = \gamma H'(A|B) + (1 - \gamma) H''(A|B), \tag{5.12}$$

where H' and H'' are the entropies corresponding to the distributions $P'(A, B)$ and $P''(A, B)$, defined in Section 5.2. Although the existing theory predicts good asymptotic behaviour of the SC-LDPC codes, their finite-length performance is well studied only for the case of the erasure channel. Our setting with the transmission through a mixture of two channels makes the analysis even more complicated. In case of our simplified model, using (5.5) we get

$$\tilde{\eta}(n) = \frac{\tilde{m}(n)}{n}. \tag{5.13}$$

Note that this is the best achievable overhead. The actual code construction can have an overhead loss due to different factors, such as the length and overhead adaptation. Therefore, we use numeric simulations in order to estimate its performance.

The simulation setting for the error-correction part can be described as follows. Assume that the length n and the error-correction overhead $\eta = m/n$ are fixed. Then we choose the base protograph with the design rate close to $1 - \eta$; this is then followed by the coupling and lifting process, where the coupling factor is fixed as $L = 80$ (SC-LDPC codes are optimal for $L \to \infty$, but in practice we just choose a sufficiently high number). Note that the resulting code has slightly larger length n' (as it is a multiple of the lifting factor) and overhead η' (due to coupling). Hence, to finish the code construction, we need to remove some variable and check nodes from the Tanner graph in a way that keeps the spatial coupling advantage (which comes from low-degree check nodes at the sides of the chain). We use the simple and rather efficient strategy that can be described as follows:

1. Remove $n' - n$ variable nodes with the smallest degrees (the node is less reliable if it has a small degree)

2. Choose two sets of $(\eta' - \eta)n$ checks with the largest degrees and perform pairwise merging, i.e., replace every pair of nodes c, c' with a new node c'' s.t. $V_{c''} = V_c \cup V_{c'}$.

Our experiments show that if the target values n and η are not too far from n' and η', this construction performs rather well. Before the decoding process, the string is shuffled so that noisier bits become more evenly distributed in the codeword. Note that the shuffling pattern is known to the decoder, hence it can correctly assign the reliabilities.

5.5.1 Numeric Results

The performance of the error-correction scheme is first tested for the simplified model and then on the more realistic one. For the simplified model, we consider two sets of parameters that will be further denoted as Ψ_0 and Ψ_1:

1. Ψ_0: $\gamma = 0.08, S = 2.6192, Q = 0.015$

2. Ψ_1: $\gamma = 0.08, S = 2.6192, Q = 0.02$.

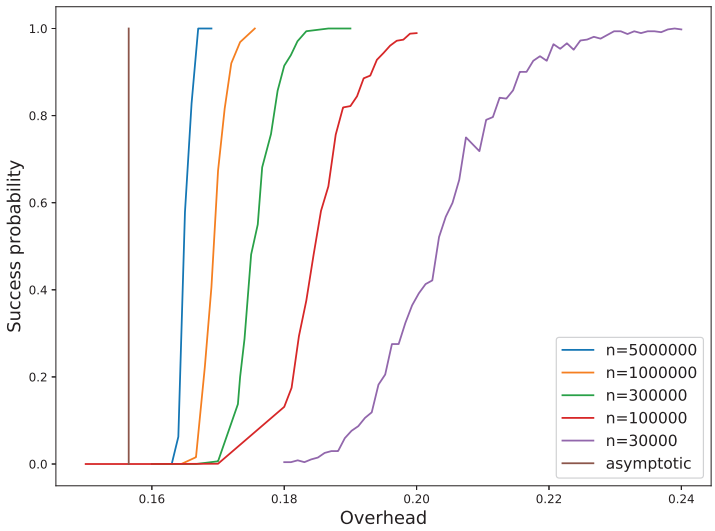


Figure 5.7 – Performance of the error-correction scheme for model Ψ_0

The more realistic model, which we will denote as Ψ_2, has $\gamma = 13/256$ and the probability distributions for test and key rounds are

$$P'(A,B) = \begin{pmatrix} 0.41638569 & 0.08054036 \\ 0.08811974 & 0.4149542 \end{pmatrix}, \tag{5.14}$$

$$P''(A,B) = \begin{pmatrix} 0.4906533 & 0.00669439 \\ 0.01720901 & 0.4854433 \end{pmatrix}. \tag{5.15}$$

The asymptotic overheads for the considered models are $\tilde{\eta}_0^\infty \approx 0.1565, \tilde{\eta}_1^\infty \approx 0.1832$ and $\tilde{\eta}_2^\infty \approx 0.1847$, respectively.

We performed numeric simulations to observe the performance of the designed scheme for our models and total lengths $n \in \{30000, 100000, 300000, 1000000, 5000000\}$. We used two protographs, $(9, 45)$-regular code with design rate $1 - 1/5$ and $(9, 36)$-regular code with design rate $1 - 1/4$, for the code construction. The former is selected for model Ψ_i and length n_j if the largest simulated overhead for this scenario does not exceed 0.2 and the latter does so otherwise. In our experiments, the simulated overheads did not exceed 0.25, hence it is sufficient to consider only these two codes. The total number of BP iterations was set to $N_{it} = 3000$.

Figures 5.7-5.9 demonstrate the results. Note that, whereas in typical coding scenario we have a fixed code that is evaluated over a range of different channel parameters, here we face an opposite situation, where the channel is fixed and the code is evaluated for a range of different overhead values. The curves are not very smooth, which we attribute to our overhead adaptation procedure. It can be seen that the code performance gets very close to the asymptotic limit.

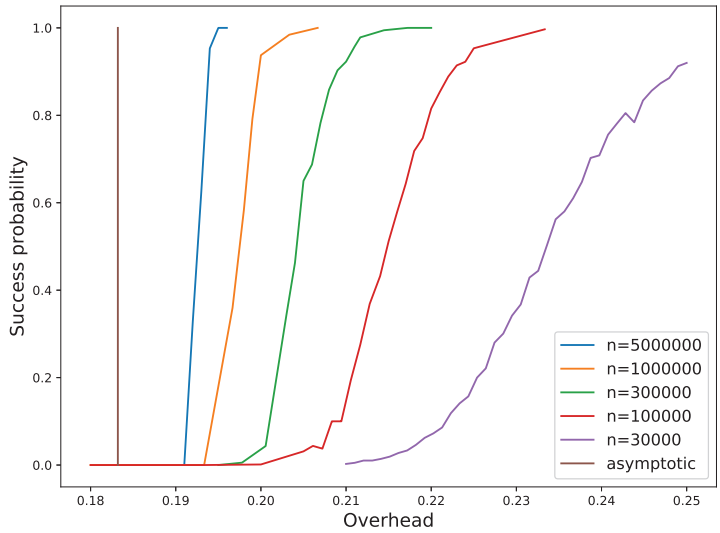


Figure 5.8 – Performance of the error-correction scheme for model Ψ_1

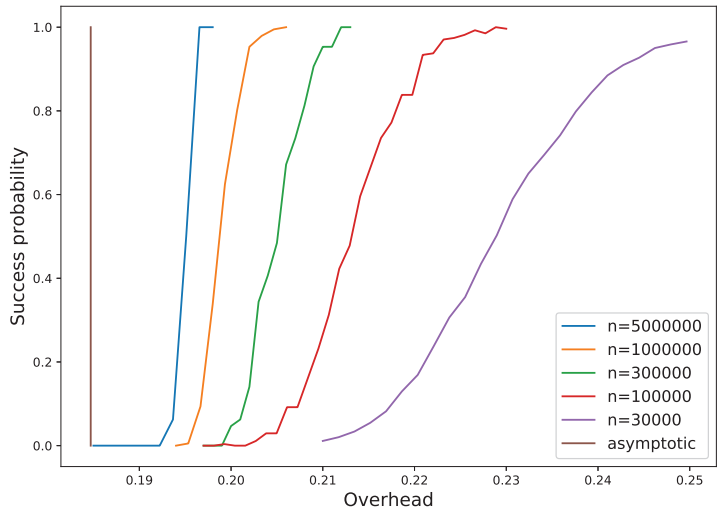


Figure 5.9 – Performance of the error-correction scheme for model Ψ_2

In particular, for the model Ψ_2, we can see that the error-correction procedure succeeds in most cases for overheads larger than $\eta_2^* = 0.196$. This is only 6% larger than the asymptotic value $\tilde{\eta}^\infty \approx 0.1847$, given by Eq. (5.12), and it is 3.7% larger than $\tilde{\eta}(5e6) \approx 0.189$, given by Eq. (5.13).

Let us now demonstrate the threshold performance of the considered code construction, i.e., the amount of "extra" overhead we need, in addition to the asymptotic value for the string reconstruction for finite block length n, and how this amount decreases with n. As we are in the finite-length setting, we define the threshold as the smallest overhead η such that the failure probability of the error-correction scheme is smaller than some fixed value ε^*. Let us denote the gap function as

$$g(\eta) = \eta - \tilde{\eta}^\infty. \tag{5.16}$$

Figure 5.10 demonstrates the behavior of $g(\eta)$ for our models Ψ_0, Ψ_1 and Ψ_2 for the failure probability $\varepsilon^* = 0.1$. Observe that all models demonstrate a similar convergence, despite the differences between them; this which confirms the universality of our code construction. For reference, we also put the curve, corresponding to (5.13) for the setting $\gamma = 13/256, S = 2.6507, Q = 0.0239$, that we can consider as a lower bound on the achievable overhead. We also compare our simulation results to the error-correction schemes previously considered for DIQKD. These schemes do not admit a known efficient decoding algorithm hence are not suited for practical purpose and represent only a theoretical interest. We consider, in particular, the bound from [136, 115]:

$$m_{AFRV} = n \cdot \tilde{\eta}^\infty + \min_{0 \le \varepsilon' \le \varepsilon^*} 4\log(2\sqrt{2}+1)\sqrt{2n\log\left(\frac{8}{\varepsilon'^2}\right)} + \log\left(\frac{8}{\varepsilon'^2} + \frac{2}{2-\varepsilon'}\right) + \log\left(\frac{1}{\varepsilon^* - \varepsilon'}\right) \tag{5.17}$$

and the one from [116]:

$$m_{TSBSRSL} = n \cdot \tilde{\eta}^\infty + \min_{0 \le \varepsilon' \le \varepsilon^*} 2\log(5)\sqrt{n\log\left(\frac{2}{\varepsilon'^2}\right)} + 2\log\left(\frac{1}{\varepsilon^* - \varepsilon'}\right) + 4. \tag{5.18}$$

It is clear, in Fig. 5.10, that our construction achieves overhead smaller than the AFRV bound, and it achieves an overhead comparable to the TSBSRSL bound, which is achieved with the actual numeric experiments and practical low-complexity decoding algorithm. Note that it also follows from the figure that the proposed SC-LDPC scheme has the scaling that is rather close to the heuristic estimate of $O(n^{-1/3})$.

Finally, we note that the results of our simulations in Fig. 5.10 yield a critical threshold that is systematically shifted from the bound given in Eq. (5.13). This can be attributed to the fact that scaling depends on the lifting factor M more than on the total block length n.

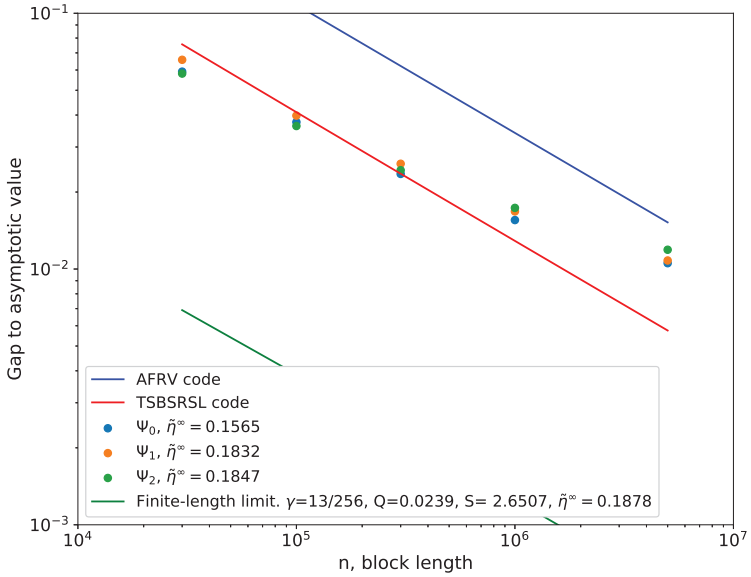


Figure 5.10 – Convergence of the code performance: gap to the asymptotic limit as a function of the block length for the failure probability $\varepsilon^* = 0.1$.

6 Conclusions

In this chapter, we conclude the thesis and discuss the potential research directions.

6.1 Symmetries of Polar-Like Codes

In Chapters 3 and 4, we have studied the framework of polar-like codes; it extends to all length-2^m binary linear codes. Our main tools are the Plotkin $(u|u+v)$ decomposition of the code and the Boolean polynomial representation of the codewords. Both tools are interconnected via a notion of directional derivative of the code; this notion is defined in the Boolean function framework and each direction corresponds to the different $(u|u+v)$ decomposition. Another notable connection is that the invariance under a large subgroup of general affine group of permutations implies the existence of many derivative directions that induce the same code.

In Chapter 3, we have focused on the Reed-Muller codes and talked about the design of low-complexity error-correcting algorithms that utilize the large automorphism group of the code. We have presented a variation of successive cancellation decoder with permutations; this variation operates over the erasure channel and outperforms the vanilla approach. For the second-order codes, we have presented the improved decoder for the general channels.

In Chapter 4, we have taken a more theoretical look at the decoding problem. We have introduced the notion of code symmetry and have shown that it makes the error probability polar-like decoding fall far from the maximum likelihood performance. The most notable examples of the symmetric codes are the

Reed-Muller and eBCH codes that are well-known to have a rich algebraic structure. However, our results extend beyond this and encompass the codes with significantly smaller automorphism groups, which limits the attempts to design long polar-like codes with good performance under permutation decoding. The symmetry framework also enabled us to restrain the size of automorphism group of polar codes. Specifically, it cannot be significantly larger than the lower-triangular affine group.

It should be clear by now that the journey towards practical decoding algorithms with excellent performance is far from being complete. Having a large automorphism group is clearly beneficial for boosting the performance of the Reed-Muller codes, compared to the standard approach, but this is also limits the potential of their polar-like decoding. We also know that the Reed-Muller and eBCH codes are great if we can do maximum likelihood decoding. Therefore, the design of low-complexity algorithms that reach the error probability of an optimal decoder remains a challenging but potentially beneficial open problem. The second direction is related to the design of good codes for permutation decoding. Our proof demonstrates that, in case of long codes and affine permutations, there is not much hope. Nevertheless, the cases of short and moderate block lengths remain feasible, as well as those of non-affine automorphisms.

6.2 Device-Independent Quantum Key Distribution

In Chapter 5, we have talked about the concept of device-independent quantum key distribution. DIQKD enables the fully private communication between parties, where the secrecy is guaranteed by quantum physics and there are no assumptions about the structure and inner working of quantum devices being in use. There has been much progress regarding the theoretical analysis of various aspects of DIQKD, but the experimental demonstration remained out of reach.

We have presented the DIQKD protocol that was successfully implemented in practice. It was a result of collaboration with many physicists responsible for the protocol design, security proofs, and the experimental implementation. After the quantum part of the protocol, two parties have the bit strings that are not perfectly correlated. The issue was solved by disclosing some information and performing the error correction. The induced channel is non-standard and closely resembles a mixture of two BSCs. The amount of revealed information needs to be small in order to satisfy security guarantees. We designed the key reconciliation solution based on SC-LDPC codes that are perfect candidates due to their universality, to the simplicity of rate adaptation, and to the performance close to numeric limits.

The field of potential future research directions is vast concerning DIQKD. Hence, we mention here just two that seem the most relevant to us. The most important are probably related to the large-scale practical implementation of the proposed protocol and to the further improvements of error-correction performance. Despite being close to the finite-length coding limits, there remains a noticeable gap that leaves an invitation for the development of more powerful schemes.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[2] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[3] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[4] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *Transactions of the I.R.E. Professional Group on Electronic Computers*, vol. EC-3, no. 3, pp. 6–12, Sept 1954.

[5] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 38–49, September 1954.

[6] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (Paris)*, vol. 2, p. 147–156, Sep 1959.

[7] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, pp. 68–79, 1960.

[8] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of The Society for Industrial and Applied Mathematics*, vol. 8, pp. 300–304, 1960.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, 1993, pp. 1064–1070 vol.2.

[10] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[11] D. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1723–1731, 1996.

[12] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[13] N. Wiberg, H.-A. Loeliger, and R. Kotter, "Codes and iterative decoding on general graphs," in *Proceedings of 1995 IEEE International Symposium on Information Theory*, 1995, pp. 468–.

[14] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[15] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[16] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[17] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory.*, vol. 47, pp. 585–598, Feb 2001.

[18] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory.*, vol. 47, pp. 599–618, Feb 2001.

[19] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[20] S.-Y. Chung, G. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, 2001.

[21] A. Jimenez Felstrom and K. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, 1999.

[22] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative Decoding Threshold Analysis for LDPC Convolutional Codes," *IEEE Trans. Inf. Theory.*, vol. 56, pp. 5274–5289, Oct. 2010.

[23] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 803–834, 2011.

[24] S. Kudekar, T. Richardson, and R. L. Urbanke, "Spatially Coupled Ensembles Universally Achieve Capacity Under Belief Propagation," *IEEE Trans. Inf. Theory.*, vol. 59, pp. 7761–7813, Dec. 2013.

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

[26] M. El-Khamy, H. Lin, and J. Lee, "Binary polar codes are optimised codes for bitwise multistage decoding," *Electronics Letters*, vol. 52, no. 13, pp. 1130–1132, 2016. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/el.2016.0837

[27] N. Stolte, "Recursive codes with the Plotkin construction and its decoding," Ph.D. dissertation, Technical University "a t, Darmstadt, January 2002. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/183/

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

[29] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, 2012.

[30] T. Wang, D. Qu, and T. Jiang, "Parity-check-concatenated polar codes," *IEEE Communications Letters*, vol. 20, no. 12, pp. 2342–2345, 2016.

[31] P. Trifonov and V. Miloslavskaya, "Polar subcodes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 254–266, Feb 2016.

[32] P. Trifonov and G. Trofimiuk, "A randomized construction of polar subcodes," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 1863–1867.

[33] P. Trifonov, "Randomized polar subcodes with optimized error coefficient," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6714–6722, 2020.

[34] E. Arikan, "From sequential decoding to channel polarization and back again," *CoRR*, vol. abs/1908.09594, 2019. [Online]. Available: http://arxiv.org/abs/1908.09594

[35] 3GPP, "NR; Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.212, 01 2019, version 15.4.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214

[36] M. Kamenev, Y. Kameneva, O. Kurmaev, and A. Maevskiy, "Permutation decoding of polar codes," *CoRR*, vol. abs/1901.05459, 2019. [Online]. Available: http://arxiv.org/abs/1901.05459

[37] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "CRC-aided belief propagation list decoding of polar codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 395–400.

[38] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief Propagation List Decoding of Polar Codes," *ArXiv e-prints*, Jun. 2018.

[39] ——, "Belief Propagation Decoding of Polar Codes on Permuted Factor Graphs," *ArXiv e-prints*, Jan. 2018.

[40] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *2009 IEEE International Symposium on Information Theory*, June 2009, pp. 1488–1492.

[41] S. A. Hashemi, N. Doan, M. Mondelli, and W. J. Gross, "Decoding Reed-Muller and Polar Codes by Successive Factor Graph Permutations," *ArXiv e-prints*, Jul. 2018.

[42] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "On the Decoding of Polar Codes on Permuted Factor Graphs," *ArXiv e-prints*, Jun. 2018.

[43] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "On the Automorphism Group of Polar Codes," *arXiv:2101.09679 [cs, math]*, Jan. 2021, arXiv: 2101.09679. [Online]. Available: http://arxiv.org/abs/2101.09679

[44] C. Pillet, V. Bioglio, and I. Land, "Polar Codes for Automorphism Ensemble Decoding," *arXiv:2102.08250 [cs, math]*, 2021. [Online]. Available: http://arxiv.org/abs/2102.08250

[45] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Şaşoğlu, and R. Urbanke, "Reed-Muller codes achieve capacity on erasure channels," in *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16. New York, NY, USA: ACM, 2016, pp. 658–669. [Online]. Available: http://doi.acm.org/10.1145/2897518.2897584

[46] G. Reeves and H. D. Pfister, "Reed-Muller codes achieve capacity on BMS channels," *arXiv:2110.14631 [cs, math]*, 2021, arXiv: 2110.14631. [Online]. Available: http://arxiv.org/abs/2110.14631

[47] E. Abbe, A. Shpilka, and A. Wigderson, "Reed–Muller Codes for Random Erasures and Errors," *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5229–5252, 2015.

[48] E. Abbe and M. Ye, "Reed-Muller Codes Polarize," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7311–7332, 2020.

[49] J. Hązła, A. Samorodnitsky, and O. Sberlo, "On Codes Decoding a Constant Fraction of Errors on the BSC," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 1479–1488. [Online]. Available: https://doi.org/10.1145/3406325.3451015

[50] V. M. Sidel'nikov and A. S. Pershakov, "Decoding Reed-Muller codes with a large number of errors," *Problemy Peredachi Informatsii*, vol. 28, no. 3, pp. 80–94, 1992.

[51] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed-Muller codes: recursive lists," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1260–1266, March 2006.

[52] M. Ye and E. Abbe, "Recursive projection-aggregation decoding of Reed-Muller codes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 2064–2068.

[53] M. Kamenev, "On decoding of Reed-Muller codes using a local graph search," in *2020 IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–5.

[54] C. Carlet, *Boolean Functions for Cryptography and Error-Correcting Codes*, Y. Crama and P. L. Hammer, Eds. Cambridge: Cambridge University Press, 2010. [Online]. Available: https://www.cambridge.org/core/product/identifier/CBO9780511780448A022/type/book_part

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

[56] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, 3rd ed. North-holland Publishing Company, 1981.

[57] M. Legeay, "Permutation decoding: Towards an approach using algebraic properties of the σ-subcode," in *WCC 2011 - Workshop on coding and cryptography*, Paris, France, Apr. 2011, pp. 193–202. [Online]. Available: https://hal.inria.fr/inria-00608107

[58] A. Soro, J. Lacan, V. Roca, V. Savin, and M. Cunche, "Enhanced recursive Reed-Muller erasure decoding," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 1760–1763.

[59] K. Ivanov and R. Urbanke, "Permutation-based decoding of Reed-Muller codes in binary erasure channel," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 21–25.

[60] M. Kamenev, Y. Kameneva, O. Kurmaev, and A. Maevskiy, "A new permutation decoding method for Reed-Muller codes," *CoRR*, vol. abs/1901.04433, 2019. [Online]. Available: http://arxiv.org/abs/1901.04433

[61] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "Decoding Reed-Muller codes with successive factor-graph permutations," *CoRR*, vol. abs/2109.02122, 2021.

[62] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Automorphism Ensemble Decoding of Reed-Muller Codes," *arXiv:2012.07635 [cs, math]*, Dec. 2020, arXiv: 2012.07635. [Online]. Available: http://arxiv.org/abs/2012.07635

[63] C. Pillet, C. Condo, and V. Bioglio, "SCAN list decoding of polar codes," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[64] B. Sakkour and P. Loidreau, "Modified version of Sidel'nikov-Pershakov decoding algorithm for binary second order Reed-Muller codes." Ninth International Workshop on Algebraic and Combinatorial Coding Theory, ACCT'2004, 06 2004.

[65] K. Ivanov and R. Urbanke, "Improved decoding of second-order Reed-Muller codes," in *2019 IEEE Information Theory Workshop (ITW)*, 2019, pp. 1–5.

[66] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes," in *2008 IEEE International Symposium on Information Theory*, July 2008, pp. 1173–1177.

[67] M. Kamenev, "An efficient block error probability estimation of Reed-Muller codes under permutation decoding," in *2020 IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–5.

[68] R. Green, "A serial orthogonal decoder," *JPL Space Programs Summary*, vol. 37, pp. 247–253, 1966.

[69] B. Sakkour, "Decoding of second order Reed-Muller codes with a large number of errors," in *Proceedings of the IEEE ITSOC Information Theory Workshop 2005 on Coding and Complexity, ITW 2005, Rotorua, New Zealand, August 29 - September 1, 2005*, 2005, pp. 176–178. [Online]. Available: https://doi.org/10.1109/ITW.2005.1531882

[70] A. Balatsoukas-Stimming, M. Bastani Parizi, and A. Burg, "Llr-based successive cancellation list decoding of polar codes," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 3903–3907.

[71] B. Sakkour, "Etude du décodage des codes de Reed-Muller et application à la cryptographie. (Decoding of Reed-Muller codes and applications to cryptography)," Ph.D. dissertation, École Polytechnique, Palaiseau, France, 2007. [Online]. Available: https://tel.archives-ouvertes.fr/pastel-00002412

[72] D. Fathollahi, N. Farsad, S. A. Hashemi, and M. Mondelli, "Sparse multi-decoder recursive projection aggregation for Reed-Muller codes," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1082–1087.

[73] P. Trifonov, "A score function for sequential decoding of polar codes," in *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. IEEE, 2018, pp. 1470–1474. [Online]. Available: https://doi.org/10.1109/ISIT.2018.8437559

[74] K. Ivanov and R. L. Urbanke, "On the dependency between the code symmetries and the decoding efficiency," in *International Symposium on Information Theory and Its Applications, ISITA 2020, Kapolei, HI, USA, October 24-27, 2020*. IEEE, 2020, pp. 195–199. [Online]. Available: https://ieeexplore.ieee.org/document/9366151

[75] ——, "On the efficiency of polar-like decoding for symmetric codes," *IEEE Transactions on Communications*, vol. 70, no. 1, pp. 163–170, 2022.

[76] ——, "On the efficiency of polar-like decoding for symmetric codes," *CoRR*, vol. abs/2104.06084, 2021. [Online]. Available: https://arxiv.org/abs/2104.06084

[77] C. E. Shannon, R. G. Gallager, and E. R. Berlekamp, "Lower bounds to error probability for coding on discrete memoryless channels. I," *Information and Control*, vol. 10, no. 1, pp. 65–103, 1967. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0019995867900526

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

[79] G. Li, M. Ye, and S. Hu, "A Dynamic Programming Method to Construct Polar Codes with Improved Performance," *arXiv:2111.02851 [cs, math]*, Nov. 2021, arXiv: 2111.02851. [Online]. Available: http://arxiv.org/abs/2111.02851

[80] Y. Li, H. Zhang, R. Li, J. Wang, W. Tong, G. Yan, and Z. Ma, "The complete affine automorphism group of polar codes," *arXiv:2103.14215 [cs, math]*, 2021. [Online]. Available: http://arxiv.org/abs/2103.14215

[81] C. Pillet, V. Bioglio, and I. Land, "Classification of Automorphisms for the Decoding of Polar Codes," *arXiv:2110.14438 [cs, math]*, 2021. [Online]. Available: http://arxiv.org/abs/2110.14438

[82] S. H. Hassani, K. Alishahi, and R. L. Urbanke, "Finite-length scaling for polar codes," *IEEE Transactions on Information Theory*, vol. 60, no. 10, pp. 5875–5898, 2014.

[83] D. P. Nadlinger, P. Drmota, B. C. Nichol, G. Araneda, D. Main, R. Srinivas, D. M. Lucas, C. J. Ballance, K. Ivanov, E. Y.-Z. Tan, P. Sekatski, R. L. Urbanke, R. Renner, N. Sangouard, and J.-D. Bancal, "Device-independent quantum key distribution," *CoRR*, 2021. [Online]. Available: https://arxiv.org/abs/2109.14600

[84] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, pp. 145–195, Mar 2002. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.74.145

[85] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, "The security of practical quantum key distribution," *Rev. Mod. Phys.*, vol. 81, pp. 1301–1350, Sep 2009. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.81.1301

[86] H.-K. Lo, M. Curty, and K. Tamaki, "Secure quantum key distribution," *Nat. Photonics*, vol. 8, p. 595–604, 2014. [Online]. Available: https://doi.org/10.1038/nphoton.2014.149

[87] A. Acín, N. Gisin, and L. Masanes, "From Bell's Theorem to Secure Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 97, p. 120405, Sep 2006. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.97.120405

[88] Y. Zhao, C.-H. F. Fung, B. Qi, C. Chen, and H.-K. Lo, "Quantum hacking: Experimental demonstration of time-shift attack against practical quantum-key-distribution systems," *Phys. Rev. A*, vol. 78, p. 042333, Oct 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.78.042333

[89] L. Lydersen, C. Wiechers, C. Wittmann, D. Elser, J. Skaar, and V. Makarov, "Hacking commercial quantum cryptography systems by tailored bright illumination," *Nat. Photonics*, vol. 4, pp. 686–689, 2010. [Online]. Available: https://doi.org/10.1038/nphoton.2010.214

[90] I. Gerhardt, Q. Liu, A. Lamas-Linares, J. Skaar, C. Kurtsiefer, and V. Makarov, "Full-field implementation of a perfect eavesdropper on a quantum cryptography system," *Nat. Commun.*, vol. 2, no. 1, Jun 2011. [Online]. Available: http://dx.doi.org/10.1038/ncomms1348

[91] H. Weier, H. Krauss, M. Rau, M. Fürst, S. Nauerth, and H. Weinfurter, "Quantum eavesdropping without interception: an attack exploiting the dead time of single-photon detectors," *New J. Phys.*, vol. 13, no. 7, p. 073024, 2011.

[92] J. C. Garcia-Escartin, S. Sajeed, and V. Makarov, "Attacking quantum key distribution by light injection via ventilation openings," *PLoS ONE*, vol. 15, no. 8, p. e0236630, Aug. 2020.

[93] S. L. Braunstein and S. Pirandola, "Side-Channel-Free Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 108, p. 130502, Mar 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.108.130502

[94] H.-K. Lo, M. Curty, and B. Qi, "Measurement-Device-Independent Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 108, p. 130503, Mar 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.108.130503

[95] A. Rubenok, J. A. Slater, P. Chan, I. Lucio-Martinez, and W. Tittel, "Real-World Two-Photon Interference and Proof-of-Principle Quantum Key Distribution Immune to Detector Attacks," *Phys. Rev. Lett.*, vol. 111, p. 130501, Sep 2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.111.130501

[96] T. Ferreira da Silva, D. Vitoreti, G. B. Xavier, G. C. do Amaral, G. P. Temporão, and J. P. von der Weid, "Proof-of-principle demonstration of measurement-device-independent quantum key distribution using polarization qubits," *Phys. Rev. A*, vol. 88, p. 052303, Nov 2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.88.052303

[97] Y. Liu, T.-Y. Chen, L.-J. Wang, H. Liang, G.-L. Shentu, J. Wang, K. Cui, H.-L. Yin, N.-L. Liu, L. Li, X. Ma, J. S. Pelc, M. M. Fejer, C.-Z. Peng, Q. Zhang, and J.-W. Pan, "Experimental Measurement-Device-Independent Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 111, p. 130502, Sep 2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.111.130502

[98] D. Mayers and A. Yao, "Self Testing Quantum Apparatus," *Quantum Info. Comput.*, vol. 4, no. 4, p. 273–286, Jul. 2004.

[99] A. Acín, N. Brunner, N. Gisin, S. Massar, S. Pironio, and V. Scarani, "Device-Independent Security of Quantum Cryptography against Collective Attacks," *Phys. Rev. Lett.*, vol. 98, p. 230501, Jun 2007. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.98.230501

[100] U. Vazirani and T. Vidick, "Fully Device-Independent Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 113, p. 140501, Sep 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.113.140501

[101] R. Arnon-Friedman, F. Dupuis, O. Fawzi, R. Renner, and T. Vidick, "Practical device-independent quantum cryptography via entropy accumulation," *Nat. Commun.*, vol. 9, no. 1, p. 459, 2018.

[102] N. Brunner, D. Cavalcanti, S. Pironio, V. Scarani, and S. Wehner, "Bell nonlocality," *Rev. Mod. Phys.*, vol. 86, pp. 419–478, Apr 2014. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.86.419

[103] J. Barrett, L. Hardy, and A. Kent, "No Signaling and Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 95, p. 010503, Jun 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.95.010503

[104] L. Masanes, "Universally Composable Privacy Amplification from Causality Constraints," *Phys. Rev. Lett.*, vol. 102, p. 140501, Apr 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.102.140501

[105] B. W. Reichardt, F. Unger, and U. Vazirani, "Classical command of quantum systems," *Nature*, vol. 496, pp. 456–460, 2013.

[106] M. Ho, P. Sekatski, E. Y.-Z. Tan, R. Renner, J.-D. Bancal, and N. Sangouard, "Noisy Preprocessing Facilitates a Photonic Realization of Device-Independent Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 124, p. 230502, Jun 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.124.230502

[107] R. Schwonnek, K. T. Goh, I. W. Primaatmaja, E. Y.-Z. Tan, R. Wolf, V. Scarani, and C. C.-W. Lim, "Device-independent quantum key distribution with random key basis," *Nat. Commun.*, vol. 12, p. 2880, 2021. [Online]. Available: https://doi.org/10.1038/s41467-021-23147-3

[108] E. Woodhead, A. Acín, and S. Pironio, "Device-independent quantum key distribution with asymmetric CHSH inequalities," *Quantum*, vol. 5, p. 443, 2021. [Online]. Available: https://doi.org/10.22331/q-2021-04-26-443

[109] P. Sekatski, J.-D. Bancal, X. Valcarce, E.-Z. Tan, R. Renner, and N. Sangouard, "Device-independent quantum key distribution from generalized CHSH inequalities," *Quantum*, vol. 5, p. 444, 2021. [Online]. Available: https://doi.org/10.22331/q-2021-04-26-444

[110] P. Brown, H. Fawzi, and O. Fawzi, "Device-independent lower bounds on the conditional von Neumann entropy," *arXiv:2106.13692*, 2021.

[111] M. Masini, S. Pironio, and E. Woodhead, "Simple and practical DIQKD security analysis via BB84-type uncertainty relations and Pauli correlation constraints," *arXiv:2107.08894*, 2021.

[112] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Phys. Rev. Lett.*, vol. 67, no. 6, p. 661, 1991.

[113] S. Pironio, A. Acin, N. Brunner, N. Gisin, S. Massar, and V. Scarani, "Device-independent quantum key distribution secure against collective attacks," *New J. Phys.*, vol. 11, no. 4, p. 045021, 2009.

[114] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt, "Proposed experiment to test local hidden-variable theories," *Phys. Rev. Lett.*, vol. 23, no. 15, p. 880, 1969.

[115] G. Murta, S. B. van Dam, J. Ribeiro, R. Hanson, and S. Wehner, "Towards a realization of device-independent quantum key distribution," *Quantum Sci. Technol.*, vol. 4, p. 035011, 2019.

[116] E. Y.-Z. Tan, P. Sekatski, J.-D. Bancal, R. Schwonnek, R. Renner, N. Sangouard, and C. C.-W. Lim, "Improved DIQKD protocols with finite-size analysis," *arXiv:2012.08714*, 2020.

[117] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[118] S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, 2003.

[119] L. Wang and Y. Kim, "Linear code duality between channel coding and Slepian-Wolf coding," in *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA*, 2015, pp. 147–152.

[120] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel Coding Rate in the Finite Blocklength Regime," *IEEE Trans. Inf. Theory.*, vol. 56, p. 2307, 2010.

[121] S. H. Hassani, K. Alishahi, and R. L. Urbanke, "Finite-Length Scaling for Polar Codes," *IEEE Trans. Inf. Theory.*, vol. 60, pp. 5875–5898, Oct. 2014.

[122] I. Sason and R. Urbanke, "Parity-check density versus performance of binary linear block codes over memoryless symmetric channels," *IEEE Trans. Inf. Theory*, vol. 49, pp. 1611–1635, Jul. 2003.

[123] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory.*, vol. 47, pp. 619–637, Feb 2001.

[124] A. Shokrollahi, "New sequences of linear time erasure codes approaching channel capacity," *Proc. IEEE Int. Symp. Information Theory and its Applications, Honolulu, HI*, pp. 65–76, Nov. 1999.

[125] P. Oswald and A. Shokrollahi, "Cappacity-achieving sequences for the erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, p. 3017–3028, Dec. 2002.

[126] H. D. Pfister, I. Sason, and R. Urbanke, "Capacity-achieving ensembles for the binary erasure channel with bounded complexity," *IEEE Trans. Inf. Theory*, vol. 51, p. 2352–2379, July 2005.

[127] A. Amraoui, A. Montanari, T. Richardson, and R. Urbanke, "Finite-Length Scaling for Iteratively Decoded LDPC Ensembles," *IEEE Trans. Inf. Theory.*, vol. 55, pp. 473–498, Feb. 2009.

[128] M. Mondelli, S. H. Hassani, and R. L. Urbanke, "How to achieve the capacity of asymmetric channels," *IEEE Transactions on Information Theory*, vol. 64, no. 5, pp. 3371–3393, 2018.

[129] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "Spatially Coupled LDPC Codes Constructed From Protographs," *IEEE Trans. Inform. Theory*, vol. 61, p. 4866–4889, Sep. 2015.

[130] T. Richardson and R. Urbanke, *Modern Coding Theory.* Cambridge: Cambridge University Press, 2008.

[131] J. Chen, D. He, and A. Jagmohan, "Slepian-Wolf Code Design via Source-Channel Correspondence," *arXiv:cs/0607021*, Jul. 2006. [Online]. Available: http://arxiv.org/abs/cs/0607021

[132] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, pp. 209–213, Feb. 2005.

[133] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC code," *Proc. IEEE Workshop Signal processing and Systems (SIPS. 04), Austin, TX*, pp. 107–112, Oct. 2004.

[134] M. Lentmaier, M. M. Prenda, and G. P. Fettweis, "Efficient message passing scheduling for terminated LDPC convolutional codes," *Proc. IEEE Int. Symp. Inf. Theory, St. Petersburg, Russia*, p. 1826–1830, Jul./Aug. 2011.

[135] A. Iyengar, M. Papaleo, P. Siegel, J. Wolf, A. Vanelli-Coralli, and G. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inf. Theory*, vol. 58, p. 2303–2320, 2012.

[136] R. Arnon-Friedman, R. Renner, and T. Vidick, "Simple and Tight Device-Independent Security Proofs," *SIAM J. Comput.*, vol. 48, p. 181, 2019.

Kirill Ivanov

Email: kirill.ivanov@epfl.ch
LinkedIn: kirill-ivanov-46014685
GitHub: github.com/kir1994

Experience

École Polytechnique Fédérale de Lausanne — Lausanne, Switzerland
Doctoral assistant — Sep 2017 - Mar 2022

- Developed decoding algorithms for Reed-Muller codes using their automorphism group
- Demonstrated the limitations of polar-like decoding for symmetric codes, e.g., eBCH and Reed-Muller codes
- Proved the restrictions on the automorphism group of polar codes
- Designed a custom error-correction scheme based on SC-LDPC codes for device-independent quantum key distribution

Huawei Technologies — Moscow, Russia
Research intern — Oct 2018 - Dec 2018

- Investigated the application of Reed-Muller codes for shaping
- Worked on coding solutions for optical communications

Peter the Great St.Petersburg Polytechnic University — St.Petersburg, Russia
Research assistant — Feb 2014 - Jun 2017

- Developed fast decoding algorithms for polar codes
- Designed a blind decoding procedure for 5G control channel
- Implemented the solution for data storage with local error correction based on polar codes with Reed-Solomon kernel

Scholarships and Awards

- ISITA2020 Outstanding Early Career Researcher Paper Award — 2020
- EPFL EDIC PhD Fellowship — 2017
- Vladimir Potanin Fellowship for outstanding students — 2016
- Vladimir Potanin Fellowship for outstanding students — 2012

Education

École Polytechnique Fédérale de Lausanne — Lausanne, Switzerland
Ph.D. in Computer science, Advisor: Prof. Rüdiger Urbanke — 2017–2022

- Thesis: “Symmetry in design and decoding of polar-like codes”

Peter the Great St.Petersburg Polytechnic University — St.Petersburg, Russia
M.S. in Computer science, GPA: 5.0/5.0. Advisor: Dr. Peter Trifonov — 2015–2017

- Thesis: “A method of polar subcodes construction”

Peter the Great St.Petersburg Polytechnic University — St.Petersburg, Russia
B.S. in Computer science, GPA: 5.0/5.0. Advisor: Dr. Peter Trifonov — 2011–2015

- Thesis: “Coded modulation based on nonbinary polar codes”

Skills

- **Programming:** C++, Python
- **Coding theory**
- **Other:** LaTeX, Visual Studio, SVN/Git

Languages

- **Language: Russian** Native speaker
- **Language: English** Advanced
- **Language: French** Elementary

Publications

[1] D. P. Nadlinger, P. Drmota, B. C. Nichol, G. Araneda, D. Main, R. Srinivas, D. M. Lucas, C. J. Ballance, K. Ivanov, E. Y.-Z. Tan, P. Sekatski, R. L. Urbanke, R. Renner, N. Sangouard, and J.-D. Bancal, "Device-independent quantum key distribution", submitted to Nature. Preprint available online at https://arxiv.org/abs/2109.14600.

[2] **K. Ivanov** and R. L. Urbanke, "On the efficiency of polar-like decoding for symmetric codes", *IEEE Transactions on Communications*, vol. 70, no. 1, pp. 163–170, 2022.

[3] **K. Ivanov** and R. Urbanke, "On the dependency between the code symmetries and the decoding efficiency", in *2020 International Symposium on Information Theory and Its Applications (ISITA)*, 2020, pp. 195–199.

[4] G. Trofimiuk, N. Iakuba, S. Rets, K. Ivanov, and P. Trifonov, "Fast block sequential decoding of polar codes", *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 10 988–10 999, 2020.

[5] K. Ivanov and P. Trifonov, "Polar subcodes for encoding and blind decoding of variable-sized data blocks", in *SCC 2019; 12th International ITG Conference on Systems, Communications and Coding*, VDE, 2019, pp. 1–6.

[6] K. Ivanov and R. Urbanke, "Improved decoding of second-order Reed-Muller codes", in *2019 IEEE Information Theory Workshop (ITW)*, 2019, pp. 1–5.

[7] K. Ivanov and R. Urbanke, "Permutation-based decoding of Reed-Muller codes in binary erasure channel", *IEEE International Symposium on Information Theory - Proceedings*, vol. 2019-July, pp. 21–25, 2019.

[8] K. Ivanov and P. Trifonov, "Hybrid decoding of interlinked generalized concatenated codes", in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, IEEE, 2016, pp. 41–45.

Teaching

- **Teaching Assistant** at EPFL — Spring 2021, Spring 2019
  Learning theory (CS-526)
- **Teaching Assistant** at EPFL — Fall 2020
  Foundations of Data Science (COM-406)
- **Teaching Assistant** at EPFL — Spring 2020
  Quantum computation (CS-308)
- **Teaching Assistant** at EPFL — Fall 2019
  Quantum information processing (COM-309)
- **Teaching Assistant** at EPFL — Spring 2018
  Theory of computation (CS-251)
- **Teaching Assistant** at Peter the Great St.Petersburg Polytechnic University — Spring 2017
  Algorithms
- **Teaching Assistant** at Peter the Great St.Petersburg Polytechnic University — Spring 2017
  Programming in C++

Interests

- Sports: football, table tennis, hiking, social dances
- Board games