ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA
ELETTRONICA E TELECOMUNICAZIONI PER L'ENERGIA

POLAR CODES FOR ERROR CORRECTION: ANALYSIS AND DECODING ALGORITHMS

Tesi in
Teoria dell'informazione LM

Relatore:
Chiar.mo Prof. **Marco Chiani**

Presentata da:
Davide Dosio

Correlatori:
Prof. Ing. **Enrico Paolini**
Dr. Ing. **Gianluigi Liva**
Dr. Ing. **Balazs Matuz**
M.Sc. **Thomas Jerkovits**

SECONDO APPELLO — SESSIONE III
ANNO ACCADEMICO 2014-2015

KEY WORDS

Polar codes
Channel polarization
Successive Cancellation decoding
Successive Cancellation List decoding
Channel selection
AWGN

Contents

Acronyms

APP a posteriori probability

AWGN additive white Gaussian noise

BEC binary erasure channel

B-DMC binary discrete memoryless channel

BPSK binary phase shift keying

CN check node

CRC cyclic redundancy check

EXIT extrinsic information transfer

FER frame error rate

LDPC low density parity check

LLR log-likelihood ratio

LL log-likelihood

LR likelihood ratio

SC successive cancellation

SCL successive cancellation list

SNR signal-to-noise ratio

VN variable node

Abstract

Polar codes are the first class of error-correcting codes that provably achieve the symmetric capacity of any memoryless channel thanks to a new method first introduced by Arikan, the *channel polarization*. In this thesis we describe in detail the encoding and decoding algorithms. In particular we show some comparisons between the performances of successive cancellation (SC) and successive cancellation list (SCL) decoding simulators and literature. Besides, we did a concatenation of the polar code with an outer code to improve the minimum distance and the performances. We propose a novel technique to analyze channel polarization on AWGN channels, which are the most common channels in the field of satellite and deep space communications. Furthermore, we investigate the importance of an accurate approximation of the polarization functions.

Chapter 1

Introduction

Digital communication systems are omnipresent in our daily lives. There exist many examples like cell phones, digital television, wireless internet connections, etc. These systems fit into a common structure as depicted in Figure 1.1. The main components of a communication systems are a *source encoder* for compressing data, a *channel encoder* which adds redundancy, a *communication channel* that is the channel where the message conveys and its probabilistic model depends on the application, a *channel decoder* for recovering the errors introduced by the noisy channel and a *source decoder* for decompressing data. This structure was first introduced by C. Shannon in his seminal paper in 1948 [12]. We will focus our attention only on the central blocks, namely the channel coding block shown in Figure 1.2. The role of the channel encoder is to preserve the message to be transmitted over a channel subject to noise, distortion and interference.

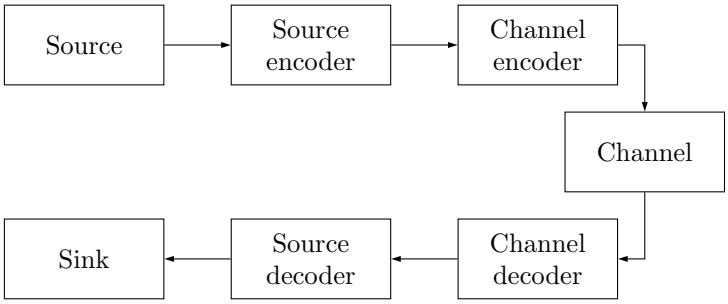

Figure 1.1: General block diagram of a communication system

Definition 1. *A Gaussian channel is a time-discrete channel with input X and output $Y = X + Z$, where Z models the noise and it is normal distributed with $Z \sim \mathcal{N}(0, \sigma^2)$. For the binary input additive white Gaussian noise (AWGN) the input alphabet $\mathcal{X} \in \{-1, +1\}$. The channel transition probability density function is*

$$P_{Y|X}(y|x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-x)^2}{2\sigma^2}}.$$

Usually we want to transmit *binary* messages composed by only two symbols namely "0" and "1". We want that if a 0 is sent, it is received as a 0 but it could happen that a 0 will be received as a 1, or viceversa. Thus, for each transmission the channel encoder adds redundancy and transforms the initial vector $\underline{u} = (u_1, u_2, \ldots, u_K)$ of length K in a vector $\underline{c} = (c_1, c_2, \ldots, c_N)$ of length N where the fraction $R = K/N$ is called the *code rate*. After that, $\underline{c}$ is mapped into modulation symbols $\underline{x}$. There is a probability that the channel introduces errors such that it changes a 0 into a 1 in our transmitted message so an error occurs. The channel decoder receives a vector $\underline{y} = (y_1, y_2, \ldots, y_N)$. The purpose of the decoder is to recover the input to the channel encoder from the channel output estimating $\underline{\hat{u}}$. The probability of error after the decoder is called *block error probability* P_B. Shannon showed that it is possible to transmit digital datas with high reliability, over noisy channels, by *encoding* the message with an error correction code. The channel encoder maps each vector of K bits into a vector of N bits called *codeword*. We need to introduce redundancy for achieving safe and reliable communication over unreliable channels. According to Shannon, every communication system can be characterized by a parameter C, called the *channel capacity*, which is a measure of how much *information* the channel can convey [8]. If $R < C$, there exists a code with length N such that the probability of error tends to zero if $N \to \infty$. Although his theorem shows the existence of a good channel code, the major objective after 1948 still is to find practical coding schemes that could approach channel capacity on different communication channels.

In the past decades, with the discovery of turbo codes [2] and low density parity check (LDPC) codes [6], the Shannon limit (the maximum rate at which data can be sent over a channel with zero error) has finally been

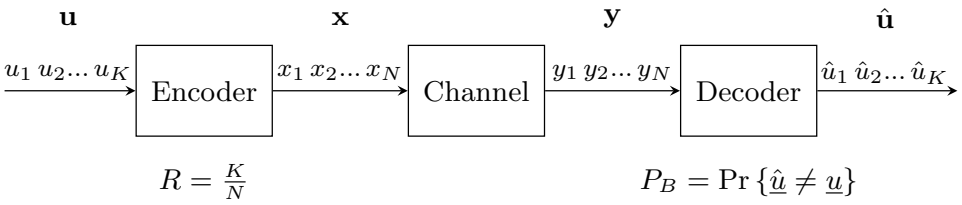

Figure 1.2: Channel coding block diagram

achieved but just in some particular cases with irregular LDPC codes [10] only for special binary erasure channel (BEC)s.
Polar codes were introduced by Arikan in 2009 in [1]. They are the first class of error correcting codes that provably achieve the capacity of any symmetric binary-input discrete memoryless channel (B-DMC) with a very efficient encoding and decoding algorithms using the technique of *channel polarization*. We will discuss different aspects of polar codes in the following chapters.
I developed my Master's Thesis at DLR (German Aerospace Center) situated in Oberpfaffenhofen, Munich (Germany), at the Institute of Communication and Navigation, Department of Satellite Networks, under the supervision of Dr. Ing. Gianluigi Liva, Dr. Ing. Balazs Matuz and M.Sc. Thomas Jerkovits. During this research activity at DLR I had the possibility to study and work on polar codes. In particular my main purposes were:

- the development of a Matlab/C++ simulator for the SC decoder.
- The development of a further Matlab simulator for an enhanced version of the SC decoder, the SCL decoder.
- The usage of a concatenation scheme of the polar code with an outer code to improve the minimum distance and the performances.
- The presentation of a new analysis tool for determining the frozen bit positions for an AWGN channel.
- Investigate the importance of using different approximations of the J-function to decide the best frozen bits positions for the transmission of data.

The thesis is organized as follows. Section 2.1 will give some general information about polar codes. In Section 2.2 we will speak on how the process of channel polarization could permit to achieve capacity. In Section 2.3 we will focus on polar coding algorithms while in Section 2.4 we will make an example of channel polarization for BEC. After that, in Section 3.1 and Section 3.2 we will explain two decoding algorithms for the simulations that have been put in place during my thesis: SC and SCL decoding algorithms. Section 3.3 will show a description of SCL implementation with an improvement using the concatenation of a cyclic redundancy check (CRC) and a polar code. Then, in Section 4.1 we will describe a novel analysis tool for computing and choosing the channels for transmitting information on the AWGN channel. In Section 4.2 we will demonstrate the importance of the J-function for choosing the channels, making some comparisons between two different approximations. Whereupon, Chapter 5 will show some comparisons with state of the art results. Finally, in Chapter 6 we will give some conclusions about our work on polar codes and provide some possible future works.

Chapter 2

Introduction to polar codes

2.1 Preliminaries and notations

Binary discrete memoryless channels (B-DMC) are an important class of channels studied in information theory and an important example of this kind of channels is the BEC, which will be considered for illustrative purposes in the next chapters. The main idea of polar codes is to construct from N independent copies of a (B-DMC) W, a new set of N channels $W_N^{(i)}$, with $1 \leq i \leq N$, using a linear transformation. The more N increases, the more these new channels $W_N^{(i)}$ are *polarized* (we will describe the concept of polarization in the Section 2.2). In this thesis we write $W : \mathcal{X} \to \mathcal{Y}$ to denote a generic binary discrete memoryless channel (B-DMC) with input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$ and transition probabilities $W(y|x)$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$. Considering a BEC, the input alphabet $\mathcal{X}$ will always be a binary input $\{0, 1\}$ while $\mathcal{Y}$ and the transition probabilities may be arbitrary. We write W^N to denote the channel corresponding to N independent uses of W; therefore, $W^N : \mathcal{X}^N \to \mathcal{Y}^N$ with $W^N\left(y_1^N|x_1^N\right) = \prod_{i=1}^N W(y_i|x_i)$. Let $y_1^N = (y_1, y_2, \ldots, y_N)$ be the observations of the outputs bits $x_1^N = (x_1, x_2, \ldots, x_N)$ through N copies of the channel W where the input bits are $u_1^N = (u_1, u_2, \ldots, u_N)$.
Given a B-DMC W, we can now define the mutual information.

Definition 2. *The mutual information of a B-DMC with input alphabet* $\mathcal{X} = \{0, 1\}$ *is defined as*

$$I(W) \triangleq I(X;Y) = \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} \frac{1}{2} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)}, \tag{2.1}$$

where X *and* Y *are two discrete random variables corresponding to input and output, respectively, and* $W(y|x)$ *is the channel transition probability for* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$.

The symmetric capacity *I(W)* equals the Shannon capacity when W is a symmetric channel, e.g. BEC.
In this thesis we will use $\log = \log_2$, $\oplus$ represents mod-2 sum, the binary field is GF(2), *I(W)* takes values in [0,1], n is a positive integer and $N = 2^n$. In the following chapters we will use the following notations.

Definition 3. *For matrices* $A = \{a_{i,j}\}$ *and* $B = \{b_{i,j}\}$ *,* $A \otimes B$ *represent the Kronecker product defined as*

$$A \otimes B \triangleq \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots \\ a_{2,1}B & a_{2,2}B & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.2}$$

Definition 4. *The* $N \times N$ *reverse shuffle permutation matrix* R_N *is given by*

$$(s_1, s_2, \ldots, s_N)R_N = (s_1, s_3, \ldots, s_{N-1}, s_2, s_4, \ldots, s_N) \tag{2.3}$$

Definition 5. *The* $N \times N$ *bit reversal permutation matrix* B_N *can be defined recursively with* R_N *as*

$$B_N = R_N \begin{bmatrix} B_{N/2} & 0 \\ 0 & B_{N/2} \end{bmatrix} = R_N(I_2 \otimes B_{N/2}), \tag{2.4}$$

with identity matrix I_2 and $B_2 = I_2$. We will also use the notation a_1^N for denoting a row vector $(a_1, \ldots, a_N)$. Given such a vector a_1^N, we write a_i^j, $1 \leq i, j \leq N$, to denote the subvector $(a_i, \ldots, a_j)$.

2.2 Channel polarization

Channel polarization is a new elegant effect observed by Arikan in [1]. It is an operation by which one yields, from N independent copies of a given B-DMC W^N, a second set of N channels $W_N^{(i)}$, with $1 \leq i \leq N$ that show a polarization effect. He shows that for any B-DMC W, the channels $\left\{W_N^{(i)}\right\}$ *polarize* in the sense that the fraction of channels such that $I(W_N^{(i)})$ tend to be 1 goes to $I(W)$ while the fraction of channels such that $I(W_N^{(i)})$ tend to be 0 goes to $1 - I(W)$, as $N \to \infty$.
Restrict now our attention on BEC where $x_1^N = c_1^N$. Initially, we have to combine copies of a given B-DMC W in a recursive manner to produce a vector channel W_N: $\mathcal{X}^N \longrightarrow \mathcal{Y}^N$, where $N = 2^n$, $n \geq 0$. At the 0th level of the recursion ($n = 0$), we have only one copy of W and we set $W_1 \triangleq W$. The first level ($n = 1$) of the recursion combines two copies of W_1 for obtaining the channel W_2: $\mathcal{X}^2 \longrightarrow \mathcal{Y}^2$, as shown in Figure 2.1 with transition probabilities

$$W_2(y_1^2|u_1^2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2). \tag{2.5}$$

In the second level ($n = 2$) we obtain W_4: $\mathcal{X}^4 \longrightarrow \mathcal{Y}^4$ from two copies of W_2 as shown in Figure 2.2. In this case the transition probabilities is

$$W_4(y_1^4|u_1^4) = W_2(y_1^2|u_1 \oplus u_2, u_3 \oplus u_4)W_2(y_3^4|u_2, u_4). \tag{2.6}$$

The mapping $u_1^4 \mapsto x_1^4$ from the input of W_4 to the input of W^4 can be written as $x_1^4 = u_1^4 G_4$ where G_N is the generator matrix. Hence, we obtain the relation $W_4(y_1^4|u_1^4) = W^4(y_1^4|u_1^4 G_4)$ between the two transition probabilities.

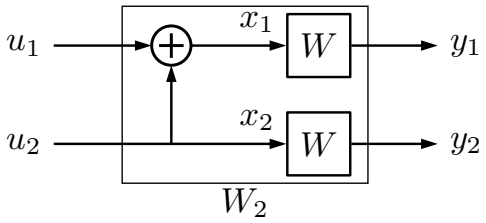

Figure 2.1: The channel W_2.

2.2 Channel polarization

The general mapping $u_1^N \mapsto x_1^N$ can be written by induction and it may be represented by G_N so that $x_1^N = u_1^N G_N$ and the transition probabilities of the two channels W_N and W^N are related by

$$W_N(y_1^N|u_1^N) = W^N(y_1^N|u_1^N G_N) \tag{2.7}$$

for all $y_1^N \in \mathcal{Y}^N$, $u_1^N \in \mathcal{X}^N$ where $W^N(y_1^N|u_1^N G_N)$ is the vector channel which contains the transformation.

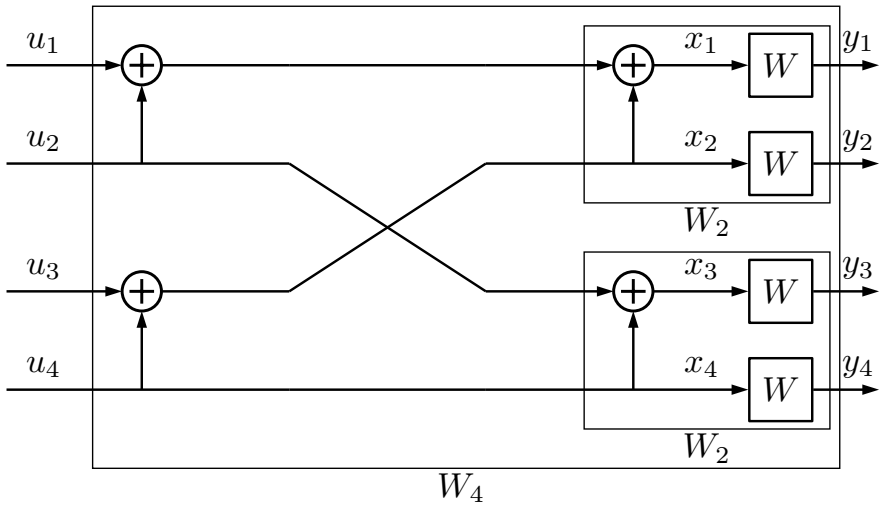

Figure 2.2: The channel W_4 and its relation to W_2 and W.

2.3 Encoding

An (N, K) polar code is a block code with K input bits and N output bits. The polar transform of size N is defined as

$$G_N \triangleq B_N \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n} = B_N G_2^{\otimes n}, \tag{2.8}$$

where

$$G_2 \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2.9}$$

G_N is called *generator matrix* of size $N \times N$. Its input is a row vector u_1^N denoted as $u_1^N = (u_1, u_2, \ldots, u_N)$, including the frozen bits and the output is a row vector x_1^N denoted as $x_1^N = (x_1, x_2, \ldots, x_N)$. The main idea of the transformation in the polar encoder is to create a set of channels with capacity $C \to 1$ for $N \to \infty$ and put the *information bits* into these ones since these channels are almost free of noise; the remaining $N - K$ *frozen bits* are transmitted in the noisy channels with $C \to 0$ and they are known at the transmitter and receiver (usually are fixed to 0's). Then the codeword $x_1^N = u_1^N G_N$ is transmitted over the channel W and we obtain $y_1^N = x_1^N + z_1^N$, where z is the noise introduced by the channel. The encoding of 2 bits is shown in Figure 2.3 where, using the notation introduced before, $x_1^2 = u_1^2 G_2$, i.e. $(x_1, x_2) = (u_1, u_2) \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. We have $x_1 = (u_1 \oplus u_2)$ and $x_2 = u_2$.

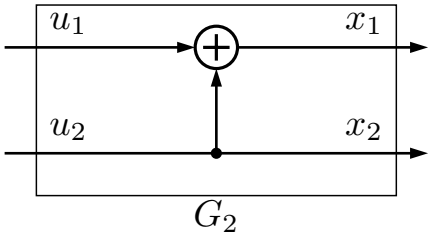

Figure 2.3: Graph of G_2

Example 1.1. For encoding $N = 4$ bits we need to use the matrix G_4. We obtain $x_1^4 = u_1^4 G_4$ where

$$G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

In this case we have $x_1 = (u_1 \oplus u_2 \oplus u_3 \oplus u_4)$, $x_2 = (u_3 \oplus u_4)$, $x_3 = (u_2 \oplus u_4)$ and $x_4 = u_4$. Figure 2.4 shows the main block of our polar encoding scheme; it has two different type of nodes: check node (CN)s and variable node (VN)s. These particular nodes are the same used for low density parity check (LDPC) codes. For the CN the output bit is XOR between the two input bits while for VN the output bit is the repetition of the input bit, as we can see from the examples above.

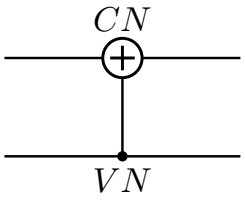

Figure 2.4: Main polar code block with a CN and a VN

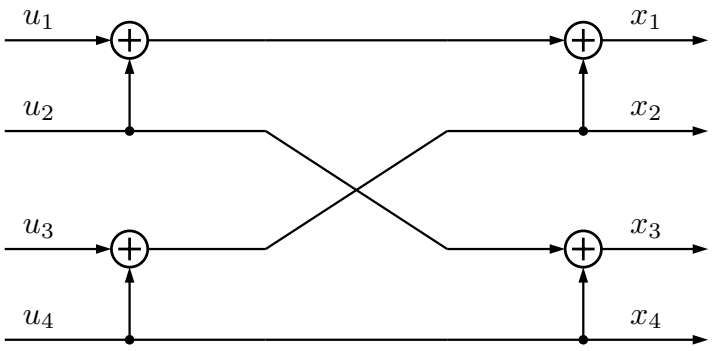

Figure 2.5: Encoding scheme for N=4

Example 1.2. For encoding $N = 8$ bits we use G_8 and we have $x_1^8 = u_1^8 G_8$ where

$$G_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Figure 2.5 and Figure 2.6 show the encoding scheme for $N = 4$ and $N = 8$, respectively.

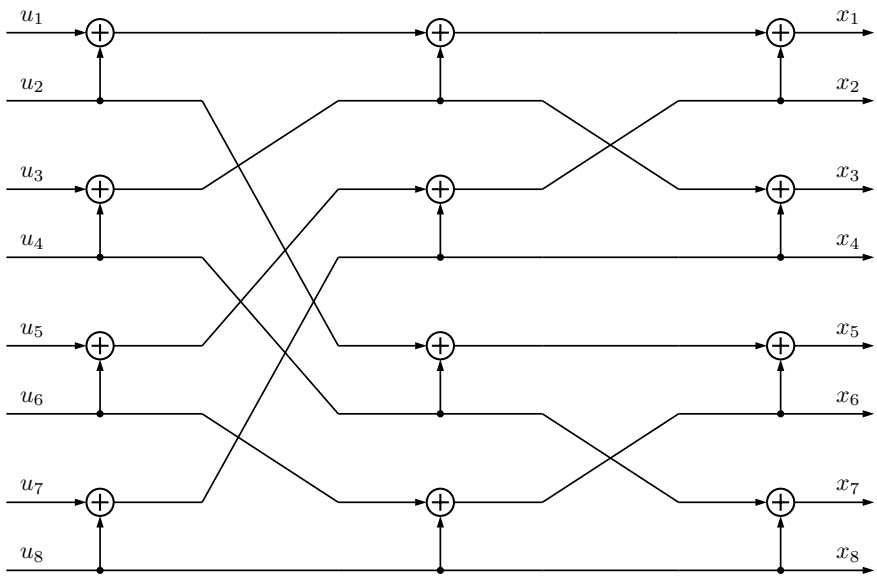

Figure 2.6: Encoding scheme for N=8

From [9] it can be demonstrated that various recursive definitions of G_N

could be useful for different representations. In our case of study we used

$$G_N = (I_{N/2} \otimes G_2) R_N (I_2 \otimes G_{N/2}). \tag{2.10}$$

If we use the recursive formula (2.10) we finally obtain the Figure 2.7, that is a generalization of Figure 2.5 and Figure 2.6 for N bits. It shows that computing $x_1^N = u_1^N G_N$ is equivalent to use a diagram with N/2 copies of G_2-butterfly section such in Figure 2.3 and two copies of $G_{N/2}$. The block R_N in Definition 2.3 is a reverse shuffle permutation matrix where the odd-numbered inputs are copied in the first N/2 outputs while the even-numbered inputs are copied to its last N/2 outputs. Let $\chi_E(N)$ the complexity of $u_1^N G_N$. If we assume the complexity of mod-2 sum as one unit and the complexity of R_N as N units, we see from Figure 2.7 that computation of $u_1^N G_N$ reduces to $N/2$ times the mod-2 sum, the complexity of R_N and two times computing $G_{N/2}$ thus $\chi_E(N) \leq N/2 + N + 2\chi_E(N/2)$. Starting from $\chi_E(2) = 2$, by induction we obtain that $\chi_E(N) \leq \frac{3}{2} N \log N$ for all $N = 2^n$. Thus, the encoding complexity is $\mathcal{O}(N \log N)$.

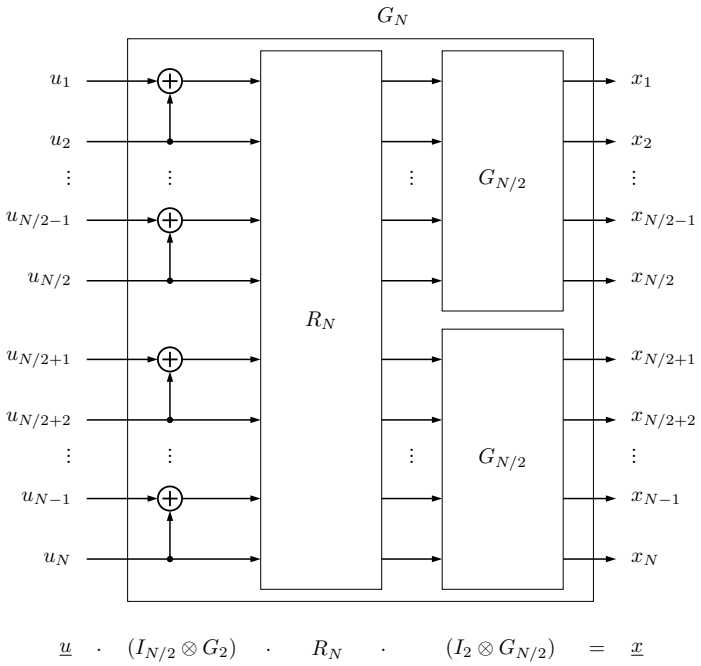

Figure 2.7: Recursive construction of G_N from two copies of $G_{N/2}$

2.4 BEC as a special case

The BEC is a symmetric channel illustrated in Figure 2.8 where ϵ is the *erasure probability*, which indicates the probability of erasure. The receiver either receives the bit or it receives a symbol "?", that represents that the bit was erased. For channels with input-output symmetry, the capacity is given by

$$C = \max_{p(x)} I(X;Y) \tag{2.11}$$

and, for the BEC channel $C = 1 - \epsilon$.

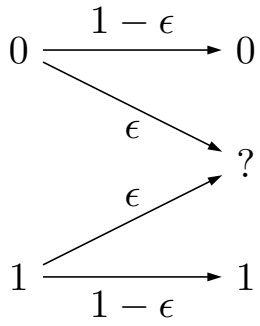

Figure 2.8: BEC

Arikan demonstrated that if W is a BEC, the numbers $\left\{I(W_N^{(i)})\right\}$ could be computed using the recursive relations

$$I(W_N^{(2i-1)}) = I(W_{N/2}^{(i)})^2 \tag{2.12}$$

$$I(W_N^{(2i)}) = 2I(W_{N/2}^{(i)}) - I(W_{N/2}^{(i)})^2 \tag{2.13}$$

where $I(W_1^{(1)}) = 1 - \epsilon$. This transformation means that two independent BECs are transformed into two polarized channels W^- and W^+, which we call the "bad" and the "good" channels respectively. Further, an important

property is the conservation of the mutual information, i.e. the sum capacities of two channels is unchanged, namely $I(W^-) + I(W^+) = 2I(W)$.
Example 1.3. Let's take $\epsilon = 0.5$. We obtain $C = 1-\epsilon = 1-0.5 = 0.5 = I(W)$. Since we have two recursive formulas (2.12) and (2.13), for $I(W) = 0.5$ we compute $I(W^+) = 2I(W) - I(W)^2 = 0.75$ and $I(W^-) = I(W)^2 = 0.25$ and we illustrated that, after the first step W is split into a "good" W^+ and a "bad" W^- and the sum of capacities is still preserved indeed $I(W^-)+I(W^+) = 2I(W) = 0.25+0.75 = 2 \cdot 0.5 = 1$. For the second step, we obtain $I(W^{--}) = I(W^-)^2 = 0.0625$, $I(W^{-+}) = 2I(W^-) - I(W^-)^2 = 0.4375$ and $I(W^{+-}) = I(W^+)^2 = 0.5625$, $I(W^{++}) = 2I(W^+) - I(W^+)^2 = 0.9375$ and so on for the following steps. This behavior can be seen in the Figure 2.9. By recursively applying such polarization transformation over the resulting channels, the result is that "the good ones get better and the bad ones get worse". This channel polarization evolution is shown in Figure 2.10 for $N = 2^{10}$. Observing the evolution of channels, we find that the capacities of most of the new channels tend to either 1, i.e. no noise on BEC, or 0. Similarly, the error probability of the bad channels or good channels go to 0 or 1. Therefore, the principal idea of polar coding is putting the *information bits* on those "good" channels whose capacity tend to be 1 and the *frozen bits* on the "bad" ones.

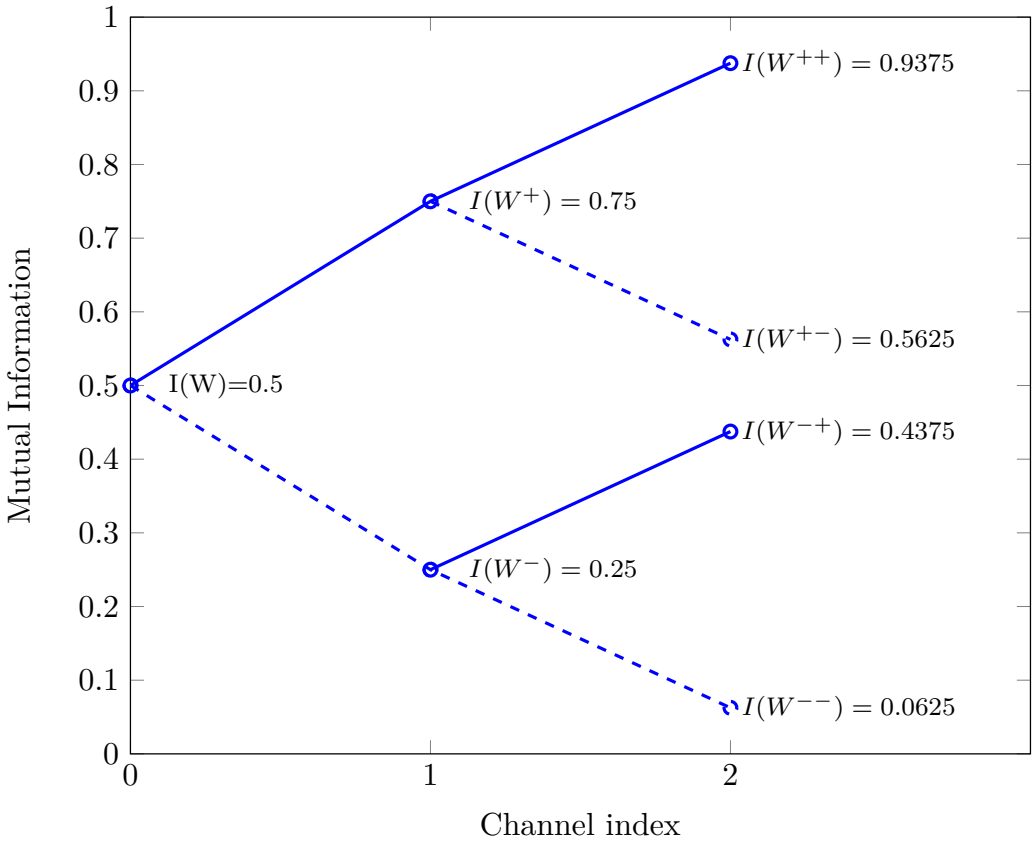

Figure 2.9: Evolution of channel polarization for N=4.

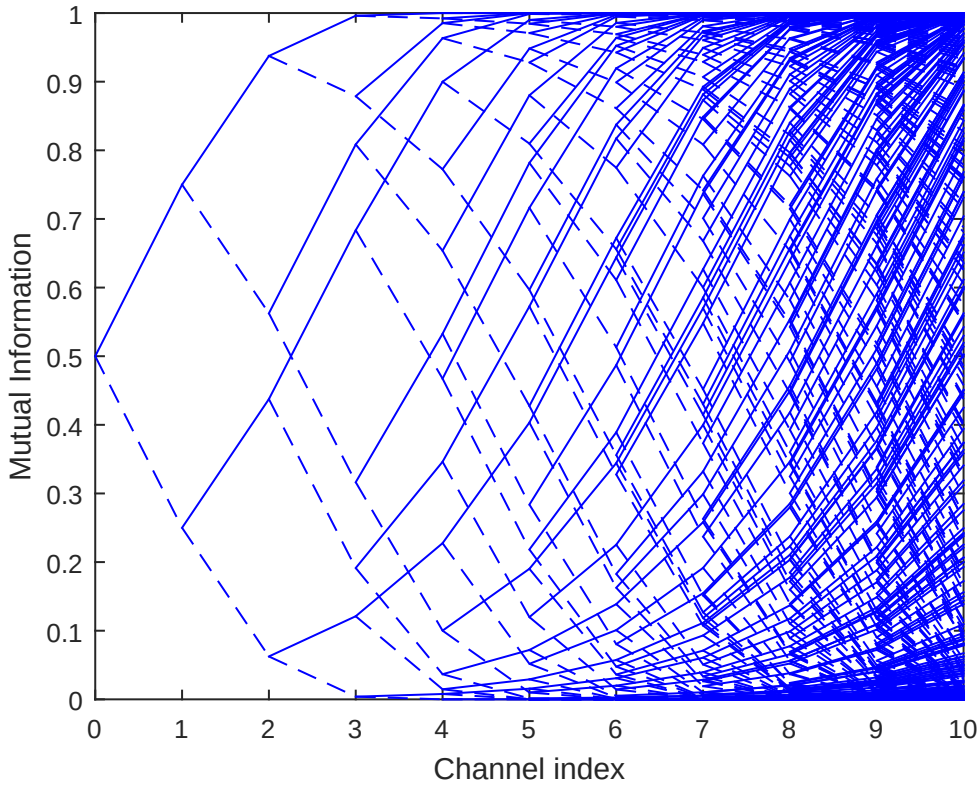

Figure 2.10: Evolution of channel polarization for N=1024.

Chapter 3

Polar decoding

3.1 Successive Cancellation decoder

Consider the recursive SC decoder for a polar transform of length N described in [1], with $N = 2^n$. The SC decoder generates an estimate $\hat{u}_i$ of u_i by observing the channel output y_1^N and using all past decisions $\hat{u}_1^{i-1}$. If u_i is a frozen bit, the decoder fixes $\hat{u}_i$ to its known value (usually 0). If u_i is an information bit, the decoder waits to estimate all the previous bits and then it may compute one of the three different types of metrics:

- log-likelihood ratio (LLR) where

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) = \ln\left(\frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 1)}\right); \tag{3.1}$$

- likelihood ratio (LR) where

$$\mathrm{LR}(y_1^N, \hat{u}_1^{i-1}) = \left(\frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 1)}\right); \tag{3.2}$$

- log-likelihood (LL) where

$$\mathrm{LL}(y_1^N, \hat{u}_1^{i-1}) = \left[\ln\left(W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 0)\right), \ln\left(W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 1)\right)\right]. \tag{3.3}$$

For reducing the numerical instability due to different type of computations, we used the LLR metric of (3.1) but we could obtain the same performances using either (3.2) or (3.3). For W being an binary AWGN channel, its transition probabilities are given in Definition 1. Hence, for any output y_i the corresponding channel LLR is

$$L_i \triangleq \ln\left[\frac{P(y_i|0)}{P(y_i|1)}\right] = \ln\left(\frac{\frac{1}{\sigma_n\sqrt{2\pi}}\exp(-\frac{(y-1)^2}{2\sigma_n^2}))}{\frac{1}{\sigma_n\sqrt{2\pi}}\exp(-\frac{(y+1)^2}{2\sigma_n^2})}\right) = \frac{2}{\sigma_n^2}y_i. \tag{3.4}$$

Since y_i is normally distributed with mean 1 and variance σ_n^2, then $L_N^{(i)} = \frac{2y}{\sigma_n^2}$ is a normally distributed variable with mean $2/\sigma_n^2$ and variance $4/\sigma_n^2$. In order to calculate each $L_N^{(i)}$, the LLRs are combined through the stages of a decoding graph containing nodes of two types, as we described in the previous sections. The SC decoding algorithm essentially follows the same encoder diagram of Figure 2.5 using decoding operations. The LLRs evolve in the reverse direction from right to left, using a pair of LLR transformation equations. Then the decisions of the bits are made at the left end of the circuit and transmitted to the rest of the circuit. In polar code, each node has two input LLRs which we denote by $L_1^{(1)}$ and $L_1^{(2)}$ as in Figure 3.1 and two output LLR which we denote by $L_2^{(1)}$ and $L_2^{(2)}$. From the context of LDPC decoding [11] adapted to our example of Figure 3.1, we can use the formulation

$$L_2^{(1)} = 2\tanh^{-1}\left(\tanh\left(\frac{1}{2}L_1^{(1)}\right)\cdot\tanh\left(\frac{1}{2}L_1^{(2)}\right)\right) \tag{3.5}$$

and

$$L_2^{(2)} = L_1^{(1)} + L_1^{(2)} \tag{3.6}$$

for computing the LLRs with inputs $L_1^{(1)}$ and $L_1^{(2)}$. (3.5) may be rewritten as

$$L_2^{(1)} = \alpha_1^{(1)}\alpha_1^{(2)}\cdot\phi\left(\phi(\beta_1^{(1)})\cdot\phi(\beta_1^{(2)})\right) \tag{3.7}$$

where

$$\alpha_i^{(j)} = \text{sign}(L_i^{(j)})$$
$$\beta_i^{(j)} = |L_i^{(j)}|$$

and

$$\phi(x) = -\ln[\tanh(x/2)] = \ln\left(\frac{e^x+1}{e^x-1}\right). \tag{3.8}$$

$\phi(x)$ is shown in Figure 3.2. For reducing the complexity of (3.7), it may be approximated with

$$L_2^{(1)} = \alpha_1^{(1)}\alpha_1^{(2)} \cdot \min[\beta_1^{(1)}, \beta_1^{(2)}]. \tag{3.9}$$

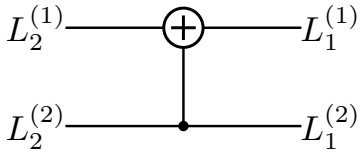

Figure 3.1: Example of CN and VN operations with N=2

All these formulas can be used in CN and VN of our polar SC decoder. In our SC decoding implementations, a LLR for the i-th bit is calculated recursively as

$$\begin{aligned} &L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) \\ &\quad = \text{sign}(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}))\text{sign}(L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}))\cdot \\ &\quad \phi\left[\phi\left(|L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2})|\right) \cdot \phi\left(|L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})|\right)\right] \end{aligned} \tag{3.10}$$

$$\begin{aligned} L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = (-1)^{\hat{u}_{2i-1}} L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}) \\ + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) \end{aligned} \tag{3.11}$$

where $\hat{u}_{1,e}^i$ and $\hat{u}_{1,o}^i$ are subvectors of $\hat{u}_1^i$ with even and odd indices, respectively and $L_1^{(i)}(y_i) = \ln\left[\frac{P(y_i|0)}{P(y_i|1)}\right]$.

Our Matlab and C++ implementations of the SC decoder used formulas (3.10) and (3.11). SC decoder computes stage by stage the necessary formulas and sequentially decides the estimation bit by bit. Thus, for $i = 1, 2, \ldots, N$,

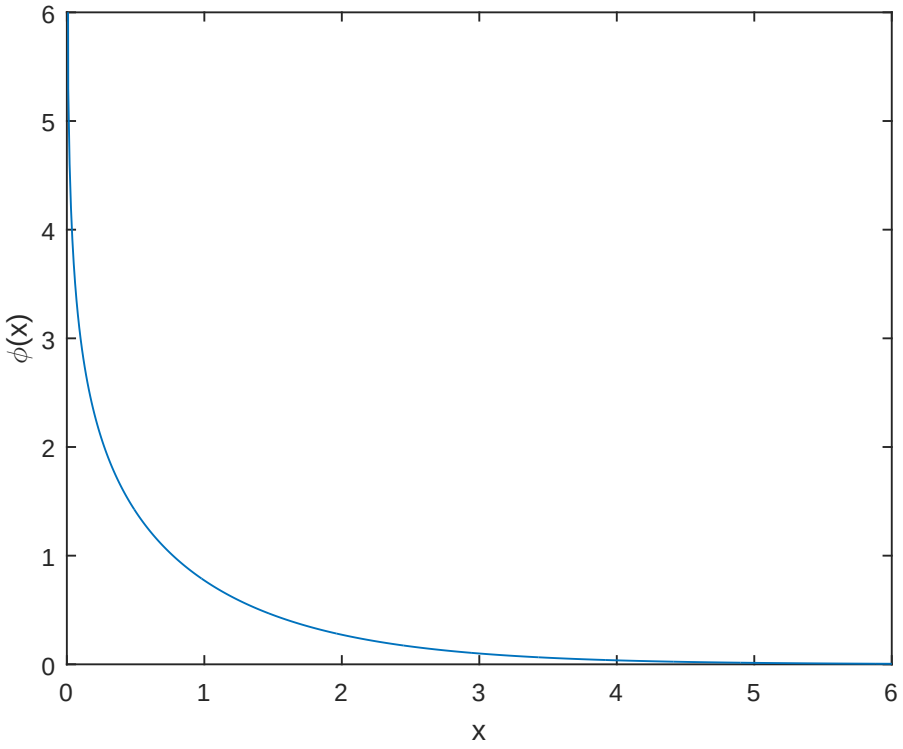

Figure 3.2: The ϕ function

if u_i is frozen, set $\hat{u_i} = 0$ else generates a decision as:

$$\hat{u_i} = \begin{cases} 0, & \text{if } L_N^{(i)} > 0 \\ 1, & \text{otherwise.} \end{cases} \tag{3.12}$$

To clarify, we describe the decoding procedure for $N = 4$ following the Figure 3.3. At first, the decoder activates node 1 to decide the value of $\hat{u}_1$; node 1 needs the LLRs at nodes 5 and 7 so it activates the two branches on its right. We call them $L_1^{(1)}$ and $L_3^{(1)}$ respectively using the notations $L_i^{(j)}$ where i indicates the *stage* and j the *level*, with $1 \leq i \leq N$ and $1 \leq j \leq n$. Then, nodes 5 and 7 need the values of L_1, L_2, L_3 and L_4 from the right branches, respectively. Therefore, LLRs at node 5 and 7 are computed using formula (3.10) and from them, LLR at node 1 ($L_1^{(2)}$) is calculated using the same formula since it is a CN. Now the decoder can finally decide the value of

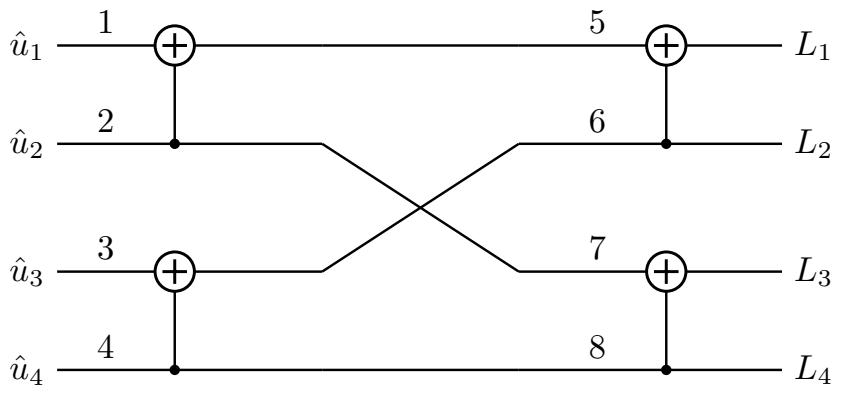

Figure 3.3: Polar decoder process for $N = 4$

$\hat{u}_1$ on the basis of (3.12). For computing LLR at node 2, this node activates nodes 5 and 7 and uses the formula (3.11) on the basis of the previous values already computed. For instance if $\hat{u}_1 = 0$ we obtain $L_2^{(2)} = L_3^{(1)} + L_1^{(1)}$ because LLR at node 1 becomes $+\infty$ according to hard decision in (3.1), i.e. $L = \ln(1/0) \to +\infty$; CN make the operation between it and LLR at node 5 and the result flows down to VN where we do the operation with LLR at node 7. Otherwise if $\hat{u}_1 = 1$ we have $L = \ln(0/1) \to -\infty$ and $L_2^{(2)} = L_3^{(1)} - L_1^{(1)}$ and it decides $\hat{u}_2$ using always (3.12). Therefore, we have decided $\hat{u}_2$ using the previous value of $\hat{u}_1$. Estimating $\hat{u}_1$ and $\hat{u}_2$ update the estimated bits at nodes 5 and 7. To estimate $\hat{u}_3$, we activate nodes 6 and 8 and the LLRs are computed using the values of L_1, L_2, L_3 and L_4 alongside the status of nodes 5 and 7, as previously described. In fact, node 5 determines whether the LLR L_2 should be combined with $+L_1$ or $-L_1$ for computing $L_2^{(1)}$; the same concept is used for $L_4^{(1)}$. Finally, from nodes 6 and 8 we compute $L_3^{(2)}$ at node 3 using the formula for the CN and we decide $\hat{u}_3$; then, $\hat{u}_4$ is estimating on the basis of $\hat{u}_3$ ($L_4^{(2)}$) and the decoding procedure is completed. Since the SC algorithm follows the same encoder diagram, the complexity of the SC decoder is the same as encoder namely $\mathcal{O}(N \log N)$. We will show some performances of SC decoding algorithms in terms of frame error rate (FER) vs E_b/N_0 compared with performances of literature in Chapter 5.

3.2 Successive Cancellation List decoder

In this section we want to explain a SCL decoder for polar codes, introduced in [13]. Polar codes have some beautiful properties such as a low encoding and decoding complexity ($\mathcal{O}(N \log N)$) and a recursive structure that makes them fit for hardware implementations [7]. Although polar codes have beautiful properties, studies indicate that for short and moderate block lengths, SC decoding of polar codes does not perform as well as turbo codes or LDPC codes. Tal and Vardy in [13] proposed an improvement to the SC decoder of [1]: a SCL decoder. The SCL decoder utilizes a parameter L, called the *list size*. As in [1], it decodes the input bits successively one-by-one; the idea of list decoding was already applied in [5] to Reed-Muller codes. A SCL decoder keeps track of several decoding results instead of just one, in fact for $L = 1$ we obtain the SC again. Instead of deciding to set the value of u_i, it takes both options. Since for each information bit it splits the decoding path into two new paths (one ends with "0" and the other ends in "1"), we must prune them and the maximum number of paths allowed is the list size L. In order to keep the best paths at each stage, the pruning criterion will be to keep the most likely paths.

Let us try now to make an example for $L = 4$ and following Figure 3.4. We assume $N = 4$ and all bits are unfrozen. In (a) the decoding algorithm starts and the first bit can be either 0 or 1. In the second step (b) the second bit assumes either 0 or 1 thus the the possible words are $\{00, 01, 10, 11\}$ but the number of paths is not greater than $L = 4$ so we don't need to prune, yet. (c) shows the all possible options for the first, second and third bit but now the paths are 8 so we must keep track of only the $L = 4$ most likely paths. (d) shows that we keep only the words $\{010, 011, 100, 111\}$. Decoding list algorithm continues for the fourth bit in (e); however, the paths are 8 which is too much so it prunes for the best $L = 4$ paths in (f). Finally, the decoding algorithm terminates and we obtain the codewords $\{0100, 0110, 0111, 1111\}$. The performance of SCL decoding is very close to that of maximum-likelihood decoding even for moderate values of the list size. The parameter L is always a power of two and [13] shows the list decoding performance from $L = 2$ to

$L = 256$ for various N sizes. Unfortunately, the performances of polar codes are in general worse than turbo and LDPC codes of similar length. However, they noted that the transmitted codeword is not the most likely codeword but it is on the list that the decoder generates. Thus, the performance could be gained using an additional CRC. Using both CRC and SCL decoder, they discovered that the performance are comparable to state of the art LDPC and turbo codes. They designed an efficient implementation of SCL decoding algorithm that runs in time $\mathcal{O}(LN\log N)$.

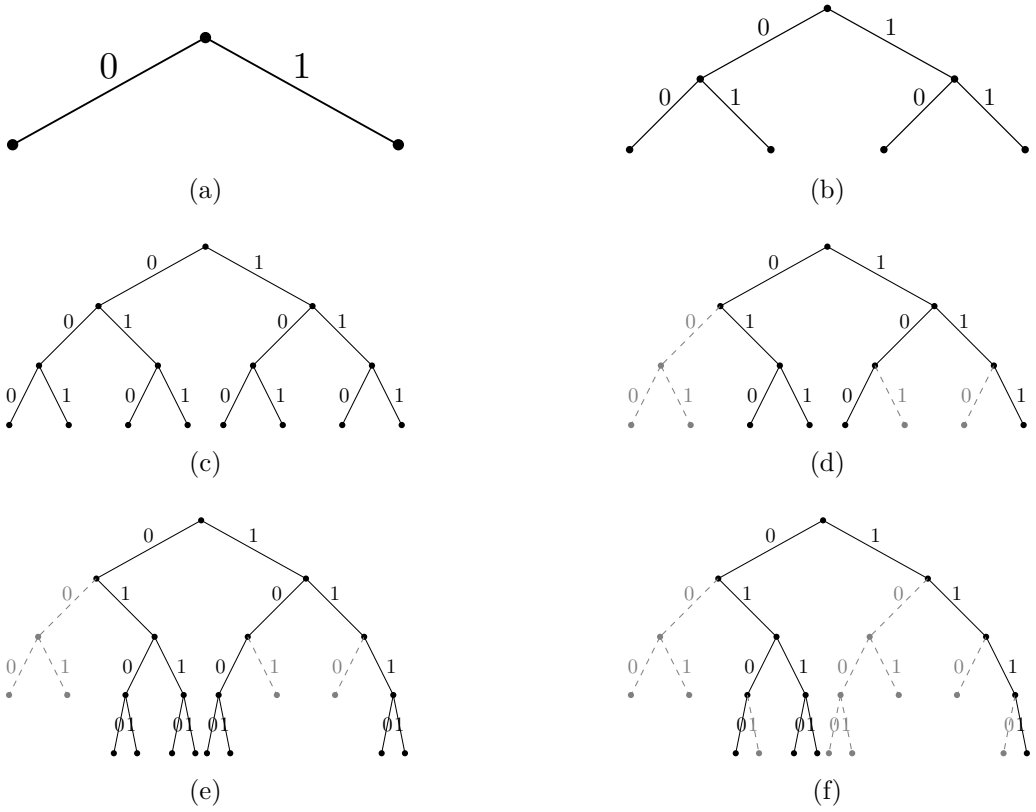

Figure 3.4: Evolution of decoding paths for $L = 4$

3.3 SCL implementation

In this section we want to describe our implementation of the SCL decoder. A high level description of the SCL decoding algorithm is given in Algorithm1. Let the polar code under consideration have length N with K information bits and $N-K$ frozen bits. The information bit vector u_1^N has length N and it includes the frozen bits. In words, for each bit $i = 1, 2, \ldots, N$, we must calculate the pair of log-likelihoods $\ln[W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)]$ and $\ln[W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)]$, which correspond to the same channel operations defined above in formula (3.3). Then, if u_i is an *information* bit we must double the path, as shown in Figure 3.4. After that, we apply the SC decoding algorithm for each path and we collect the different pair of probabilities corresponding to each path. Since in this case we have moved to the LL domain, the CN and VN operations in the SC algorithm changed.

Figure 3.5: Example of CN and VN

Let's take a and b two bits and p_a^0 and p_a^1 are the probability that bit a is 0 and 1, respectively. Furthermore we call $\tilde{p}_a^0 = \ln(p_a^0)$ and $\tilde{p}_a^1 = \ln(p_a^1)$ where $-\infty < \tilde{p}_a^0 \leq 0$. After these assumptions, we can define

$$\begin{aligned} \tilde{p}_c^0 &= \max^*(\tilde{p}_a^0 + \tilde{p}_b^0, \tilde{p}_a^1 + \tilde{p}_b^1) \\ \tilde{p}_c^1 &= \max^*(\tilde{p}_a^0 + \tilde{p}_b^1, \tilde{p}_a^1 + \tilde{p}_b^0) \end{aligned} \tag{3.13}$$

and

$$\begin{aligned} \tilde{p}_c^0 &= \tilde{p}_a^0 + \tilde{p}_b^0 \\ \tilde{p}_c^1 &= \tilde{p}_a^1 + \tilde{p}_b^1 \end{aligned} \tag{3.14}$$

for the CN and VN operations, respectively. The bit c is the output bit while a and b are the input bits as shown in Figure 3.5. The $\max^*$ operator is also known as the *Jacobian logarithm*. The operator is defined as follows

$$\begin{aligned} \max^*(x, y) &= \ln(e^x + e^y) \\ &= \max(x, y) + \ln\left(1 + e^{-|x-y|}\right). \end{aligned} \tag{3.15}$$

Following the Algorithm 1, we have to check the number of paths; if they are more than $2 \cdot L$, we must prune them and take only the L most likely paths as shown in Figure 3.4d. Namely we take only the higher probabilities and we continue to double the paths as depicted in Figure 3.4e. At the end of Algorithm 1, we have L possible information bits vector $\hat{u}$ and we decide for the most likely one. Unfortunately, our first straightforward implementation of the SCL decoder is not efficient as Tal Vardy one but as we will see in Chapter 5, the results are the same of literature.
Furthermore, we can do much better. We have observed in simulations that usually the transmitted vector u_1^N is on the list we generate, but unfortunately it is not the most likely word on the list so it is not selected from the decoder. We employed the concatenation of two codes: a CRC code as outer code for detecting the correct vector and for increasing the minimum distance of the concatenated code between codewords and a polar code as inner code. Recall that we have K information bits; instead of setting all of them, for some constant p, we set the first $K - p$ bits to information bits while the last p bits will be the parity bits of our CRC (we used an extended Hamming code(64,57)). Note that using this approach the code rate is lowered and is given by $(K - p)/N$. Therefore, we changed Algorithm 1 as follows: a path for which the corresponding codeword is not a possible codeword of the extended Hamming code can not correspond to the transmitted vector u_1^N. Thus, if at least one path correspond to a possible codeword, we remove from the list all the invalid paths and we choose the most likely one from the remained paths. Otherwise, we select the most likely path as in SCL

decoding algorithm. We will see the results of this improvement in Chapter 5.

Algorithm 1 High level description of the SCL decoder

Input: the received vector $\mathbf{y}$ and list size $\boldsymbol{L}$
Output: a decoded vector $\hat{\mathbf{u}}$

for $i = 1, 2, \ldots, N$ **do**
  calculate $\ln[W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)]$ and $\ln[W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)]$
  if u_i is an *information* bit **then**
    double each path
    for $j = 1 : Number\ of\ paths$ **do**
      apply SC decoder to each possible path
      and collect the probabilities
    end for
    if $Number\ of\ paths \geq 2 \cdot L$ **then**
      take only the L best paths i.e. paths with
      highest likelihoods and prune the other paths
    end if
  else
    set $\hat{u}_i$ to the frozen value of u_i
  end if
end for

Encoding and mapping each surviving path.
Take the most likely $\hat{\mathbf{u}}$.

Chapter 4

Channel polarization for AWGN

4.1 Analysis tool for AWGN channel polarization

In this section we propose and describe a novel analysis tool for computing the mutual information of channels through the transmission on an AWGN channel. This channel is the most popular channel in the field of communications and it is considered a suitable model for many applications such as satellite and deep space communications. The main idea was to use the extrinsic information transfer (EXIT) chart technique. EXIT charts are most used to evaluate performance of any iterative decoder and it can be used in any type of iterative decoding including LDPC codes, as described in [11]. We use the notation of [15]. An a posteriori probability (APP) decoder converts channel and *a priori* LLRs into *a posteriori* LLRs. The *a posteriori* LLRs minus the *a priori* LLRs are the *extrinsic* LLRs, which are passed on and interpreted as *a priori* information. An EXIT chart characterizes the operation of a decoder; we write I_A for the average mutual information between the bits on the decoder graph edges and the *a priori* LLRs. Likewise, we write I_E for the average mutual information between the bits on the

decoder graph edges and the *extrinsic* LLRs. Consider the AWGN channel with binary phase shift keying (BPSK) modulation and noise variance σ_n^2. We define the normalized signal-to-noise ratio (SNR) as $E_b/N_0 = 1/(2R\sigma_n^2)$. The channel LLR is $L_i = \log\left[P(y|x=+1)/P(y|x=-1)\right] = 2y/\sigma_n^2$ as defined before, where $P(y|x)$ is the channel conditional probability density function evaluated at the output y given the input x. The variance of L_i is $\sigma_{ch}^2 = \frac{4}{\sigma_n^2} = 8R\frac{E_b}{N_0}$. For the CNs and VNs the respective *extrinsic* mutual information can be computed using the recursive relations

$$I(W_N^{(2i-1)}) = 1 - J\left(\sqrt{2\cdot\left[J^{-1}\left(1 - I(W_{N/2}^{(i)})\right)\right]^2}\right) \tag{4.1}$$

$$I(W_N^{(2i)}) = J\left(\sqrt{2\cdot\left[J^{-1}\left(I(W_{N/2}^{(i)})\right)\right]^2}\right) \tag{4.2}$$

for the CNs and VNs, respectively, as in [15]. $I(W_1^{(1)}) = \sqrt{8R(E_b/N_0)}$. The functions $J(\,\cdot\,)$ and $J^{-1}(\,\cdot\,)$ will be discuss in the next section. Figure 4.1 shows a little example of using the recursive formulas (4.1) and (4.2). It is numerically verified that the property of sum capacity is still valid.

This evolution for $N = 1024$ is shown in Figure 4.2. Furthermore, we can see from the plots below the channel polarization of our new tool; for example we take a bit channel with $C = 0.7$ and we compute the polar transformation. Histogram in Figure 4.3 indicates that for $N = 2^{10}$ we have $\sim 20\%$ of the channels with capacity 0 and $\sim 58\%$ of the channels with capacity 1. If N increases, we can view from Figure 4.4 that for $N = 2^{20}$ we have $\sim 28\%$ of the channels with capacity 0 and $\sim 67\%$ of the channels with capacity 1. Thus, for $N \to \infty$, we will obtain $1 - C = 30\%$ of the channels with capacity 0 and $C = 70\%$ of the channels with capacity 1; it satisfies the *channel polarization* theorem of Arikan. The same results with $N = 2^{10}$ could be shown from another point of view in Figure 4.5.

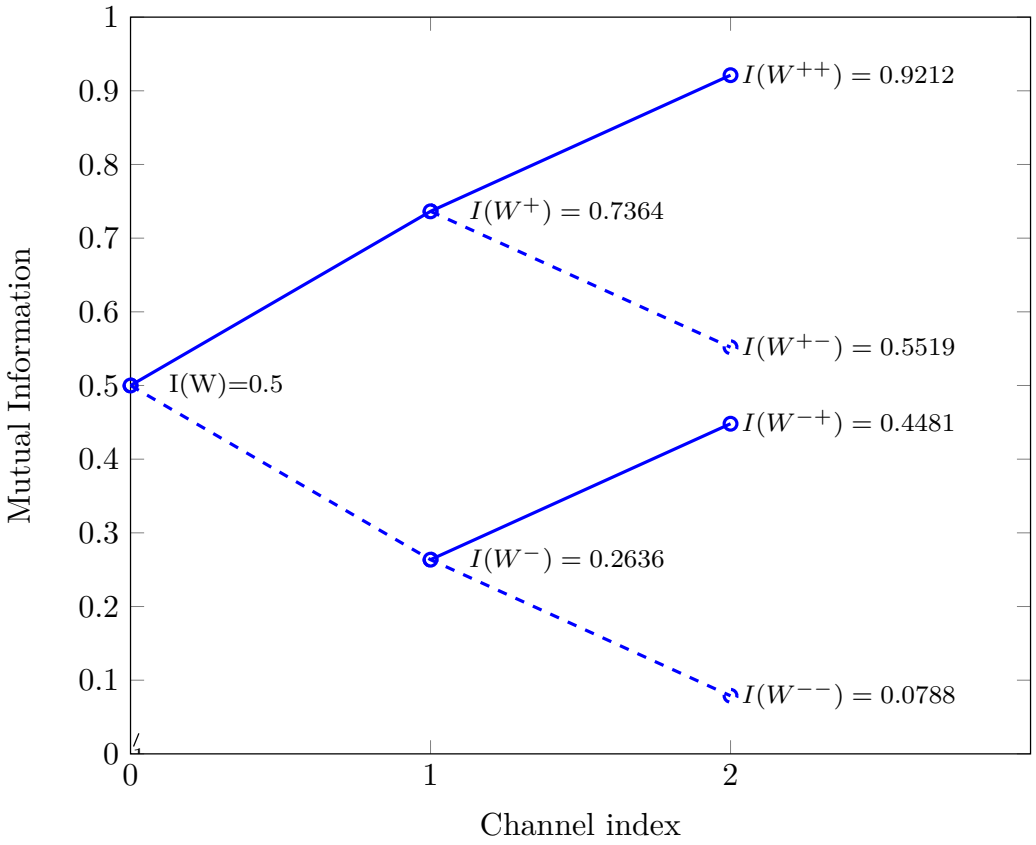

Figure 4.1: Evolution of channel polarization in an AWGN channel for N=4

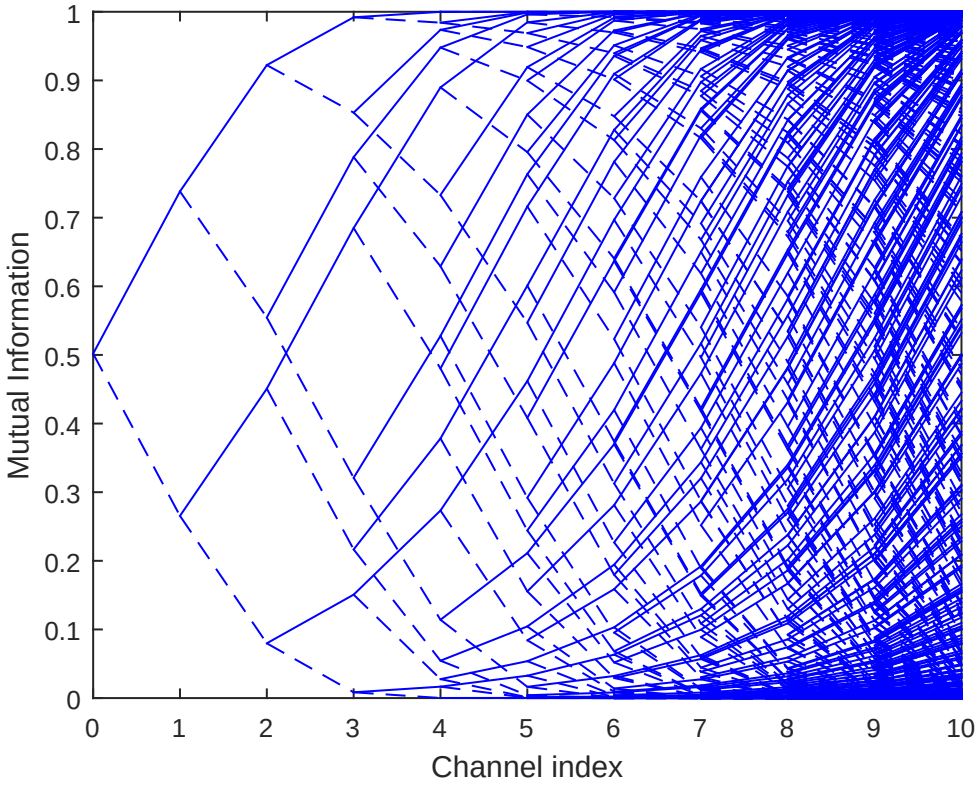

Figure 4.2: Evolution of channel polarization for N=1024 on an AWGN channel

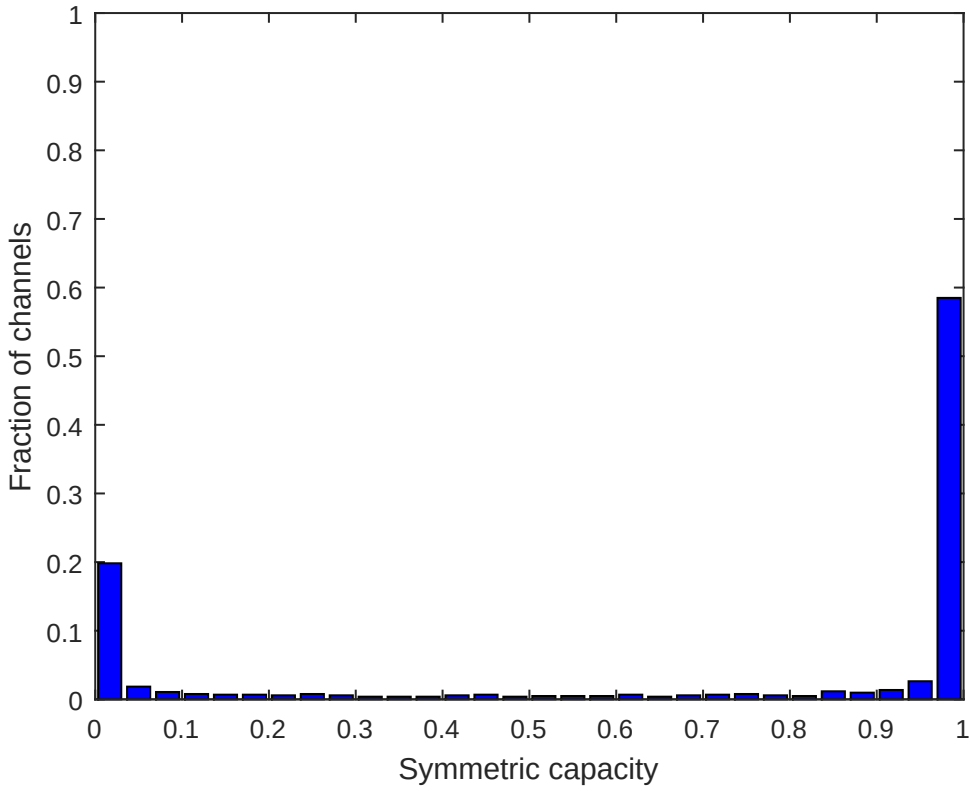

Figure 4.3: Capacity histogram for $N = 2^{10} = 1024$ channels

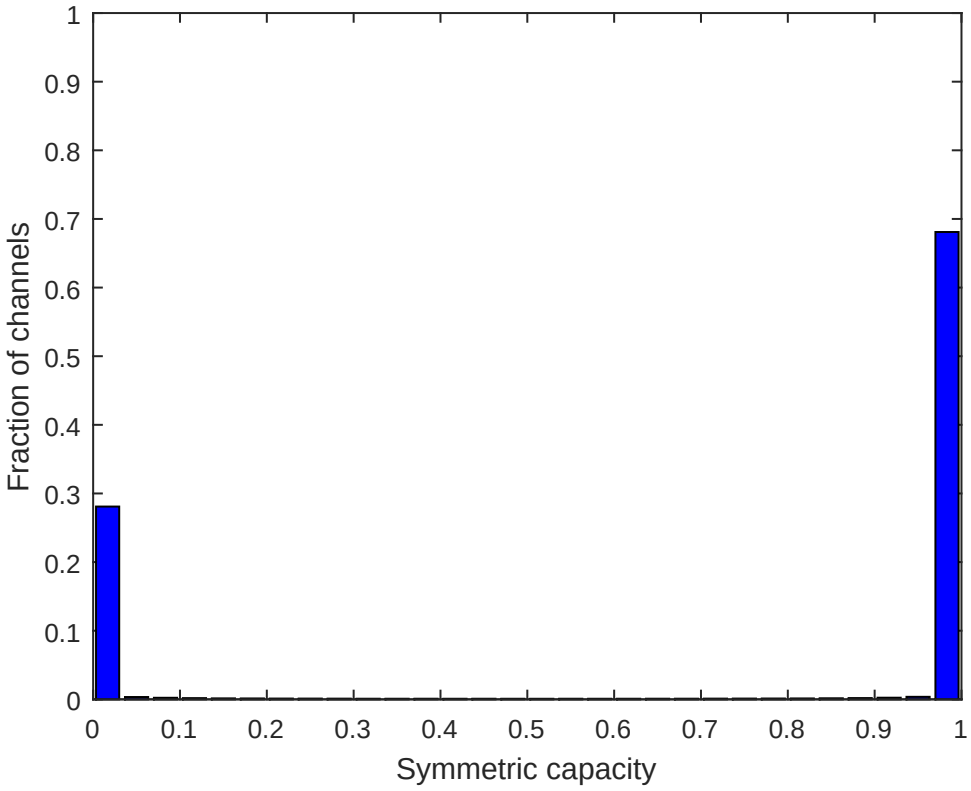

Figure 4.4: Capacity histogram for $N = 2^{20} = 1048576$ channels

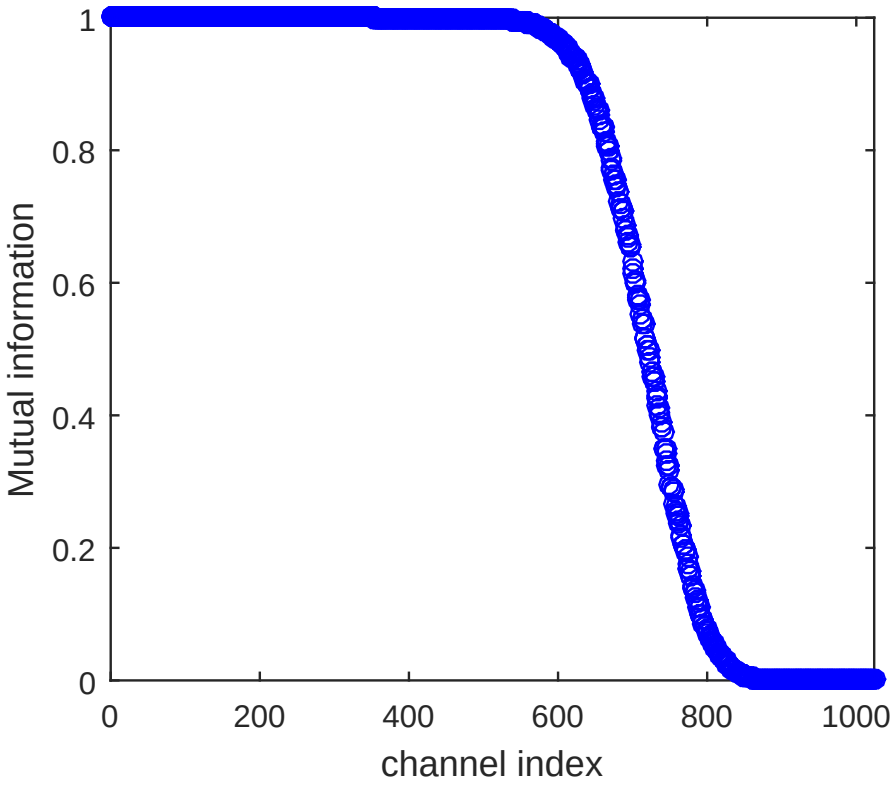

Figure 4.5: Mutual information evolution for $N = 2^{10} = 1024$ channels

4.2 Importance of J-function for channel selection

In this section we compare two approximations of the *J-function* and its inverse used in [3] and [15]. We illustrate that using dissimilar approximations involves a very different behaviour in the selection of channels used for the transmission of information bits. Besides, performance of polar code is sensitive to the choice of channels, as we will show in Table 4.1 and in Chapter 5. Consider $Y = X + Z$ is Gaussian with mean μX, $P(X = \pm 1) = 1/2$ and $Z \sim \mathcal{N}(0, \sigma_n^2)$. The L_i is a function of y and we write it as $L_i(y)$. Remember that $L_i(Y)$ conditioned on X is Gaussian with $L_i(Y) \sim \mathcal{N}(\mu_{ch}, \sigma_{ch}^2)$ where $\mu_{ch} = 2/\sigma_n^2$ and $\sigma_{ch}^2 = 4/\sigma_n^2$. Define $J(\sigma_{ch})$ be the mutual information $I(X; L_i(Y))$. We have

$$\begin{aligned} J(\sigma_{ch}) &\triangleq H(X) - H(X|L_i(Y)) \\ &= 1 - \frac{1}{\sqrt{2\pi\sigma_{ch}^2}} \int_{-\infty}^{\infty} e^{-\frac{(\xi - \sigma_{ch}^2/2)^2}{2\sigma_{ch}^2}} \log_2(1 + e^{-\xi})\, d\xi \end{aligned} \tag{4.3}$$

where $J(\,\cdot\,)$ is defined as [14]. In the formula above, $H(X)$ is the entropy of X and $H(X|L_i(Y))$ is the entropy of X conditioned on $L_i(Y)$. Note that $I(X; L_i(Y))$ is the same as $I(X; Y)$. The capacity of our $Y = X + N$ channel is, therefore, $J(\sigma_{ch}) = J(2/\sigma_n)$.
We compared two different closed forms for $J(\,\cdot\,)$ and its inverse from literature. From [15], the approximation of $J(\,\cdot\,)$ is split into two intervals with $\sigma^* = 1.6363$ and we obtain

$$J(\sigma) \approx \begin{cases} a_{J,1}\sigma^3 + b_{J,1}\sigma^2 + c_{J,1}\sigma, & 0 \le \sigma \le \sigma^* \\ 1 - \exp(a_{J,2}\sigma^3 + b_{J,2}\sigma^2 + c_{J,2}\sigma + d_{J,2}), & \sigma^* < \sigma < 10 \\ 1, & \sigma \ge 10 \end{cases} \tag{4.4}$$

where $a_{J,1} = -0.0421061$, $b_{J,1} = 0.209252$, $c_{J,1} = -0.00640081$, $a_{J,2} = 0.00181491$, $b_{J,2} = -0.142675$, $c_{J,2} = -0.0822054$, $d_{J,2} = 0.0549608$.

For the inverse $J(\cdot)$ we used two intervals with $I^* = 0.3646$ and the approximation is

$$J^{-1}(I) \approx \begin{cases} a_{\sigma,1} I^2 + b_{\sigma,1} I + c_{\sigma,1}\sqrt{I}, & 0 \leq I \leq I^* \\ -a_{\sigma,2} \ln[b_{\sigma,2}(1-I)] - c_{\sigma,2} I, & I^* < I < 1. \end{cases} \tag{4.5}$$

where $a_{\sigma,1} = 1.09542$, $b_{\sigma,1} = 0.214217$, $c_{\sigma,1} = 2.33727$, $a_{\sigma,2} = 0.706692$, $b_{\sigma,2} = 0.386013$, $c_{\sigma,2} = -1.75017$.
From [3] we obtain other two approximations:

$$J(\sigma) \approx \left(1 - 2^{-H_1 \sigma^{2H_2}}\right)^{H_3} \tag{4.6}$$

and

$$J^{-1}(I) \approx \left(-\frac{1}{H_2} \log_2 \left(1 - I^{\frac{1}{H_3}}\right)\right)^{\frac{1}{2H_2}} \tag{4.7}$$

with $H_1 = 0.3073$, $H_2 = 0.8935$ and $H_3 = 1.1064$.

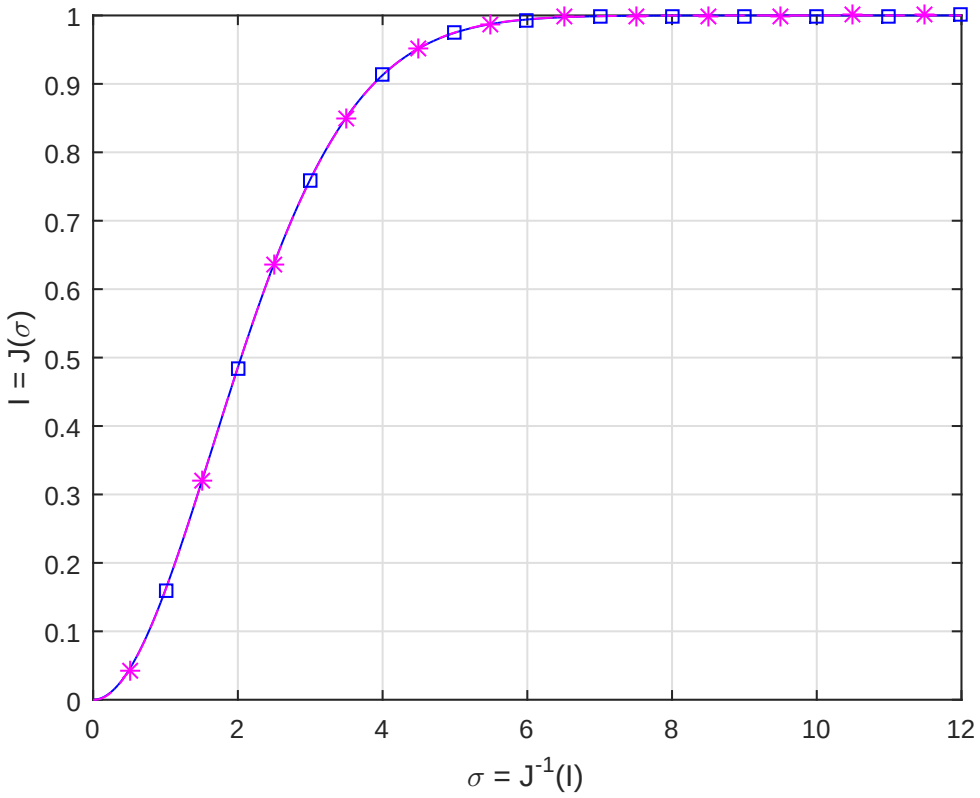

Figure 4.6: Plot of the $J(\cdot)$ functions (4.4) with magenta stars and (4.6) with blue square

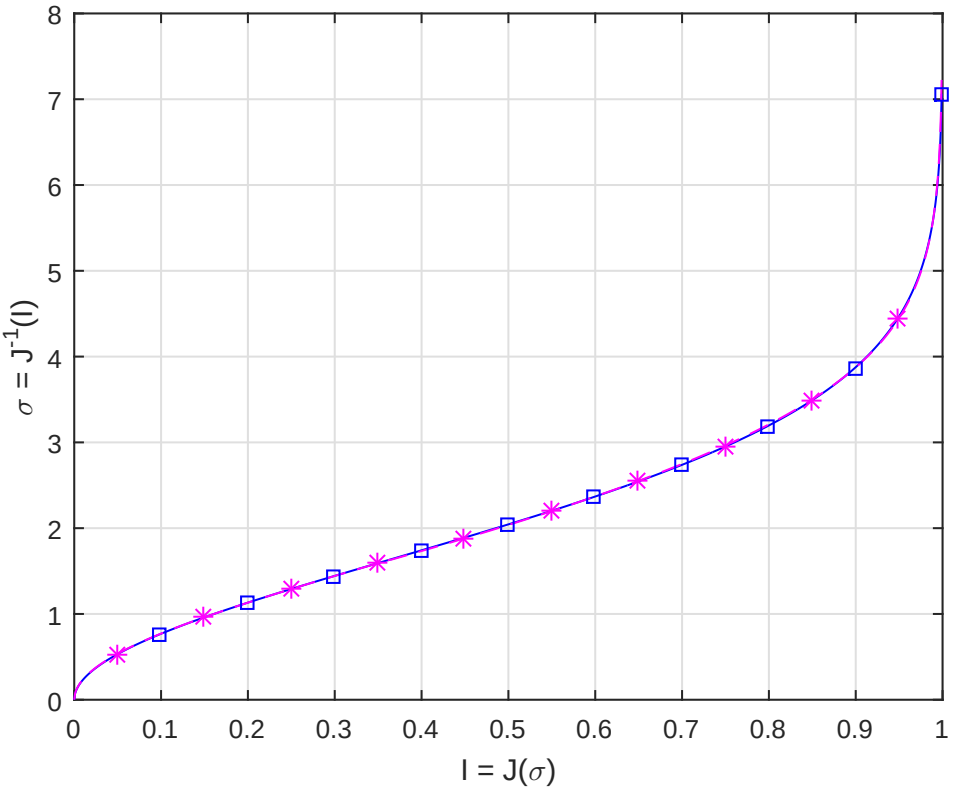

Figure 4.7: Plot of the inverse $J(\cdot)$ functions (4.5) with magenta stars and (4.7) with blue square

From Figure 4.6 and 4.7 we can see the both cases of $J(\cdot)$ and its inverse, respectively. They look very similar but if we plot the difference in magnitude from (4.4) and (4.6) (Figure 4.8) and from (4.5) and (4.7) (Figure 4.9) they are very different. They could be seen differences of $\simeq 10^{-4}$ for the $J(\cdot)$ and differences of $\simeq 10^{-1}$ for the inverse $J(\cdot)$. These dissimilarities produce different choices for the selection of the channels and different performances, accordingly. The losses are shown in Table 4.1 and a particular case is Figure 4.11.

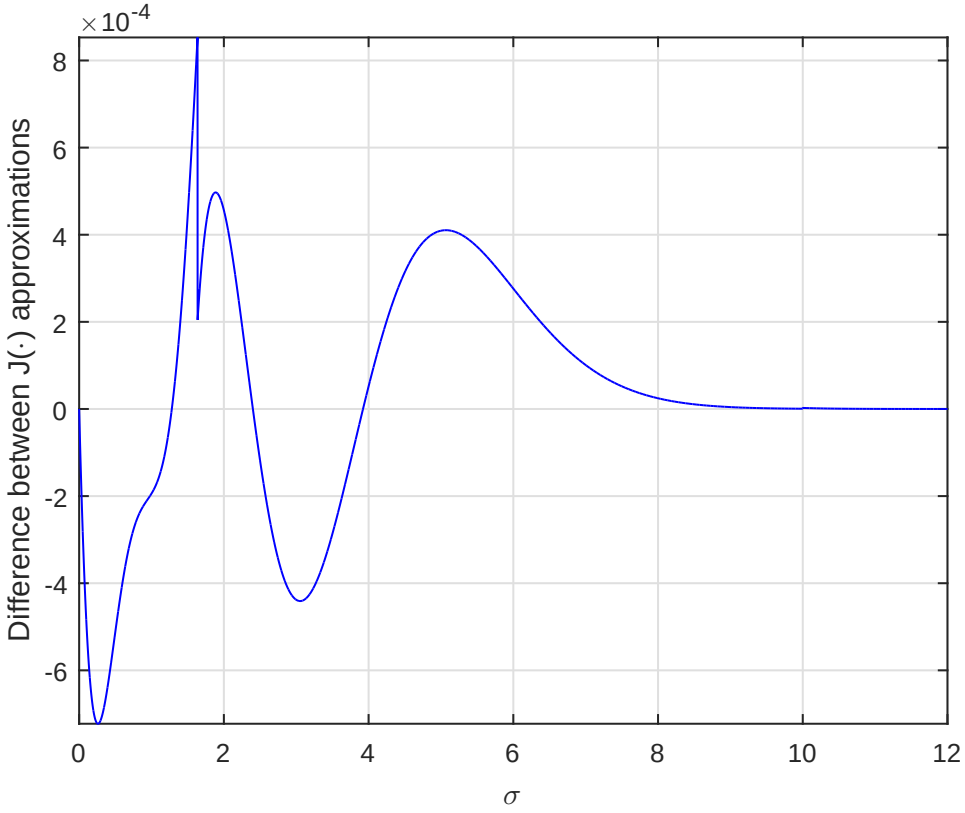

Figure 4.8: Difference between (4.4) and (4.6)

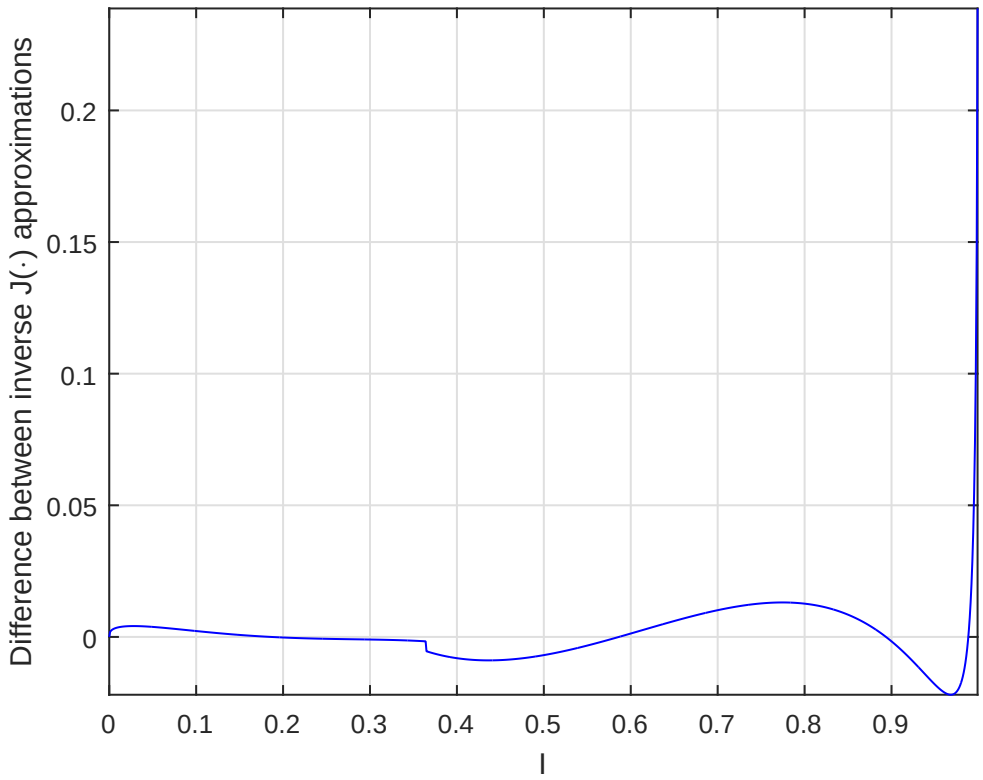

Figure 4.9: Difference between (4.5) and (4.7)

Let us try now to compute recursively formulas (4.1) and (4.2) for calculating the mutual information for a polar code of length $N = 2048$ and rate $R = 1/2$. We constructed the resulting vector of the mutual information using:

- first, $J(\cdot)$ and its inverse formulas (4.6) and (4.7);
- second, $J(\cdot)$ and its inverse formulas (4.4) and (4.5).

Then, we calculated the differences between the mutual information of the two vectors. For instance, Figure 4.10 shows a code constructed using the methods above designed for E_b/N_0 =3dB. From this figure we can see that the differences of mutual information are not negligible. After this step, we sorted the two mutual information vectors and we collected the K higher mutual information positions. Since the two vectors of K positions differ

in 134 positions, it implies that the code choose 134 different channels for transmitting the information bits and therefore we obtain different performances. Thus, if we compute the mutual information with the two different approximations of *J-function*, we obtain the results in Table 4.1. It reports the percentage of different channels whether we design the polar code for different values of E_b/N_0; besides, it shows the loss in dB for a FER of 10^{-3}.

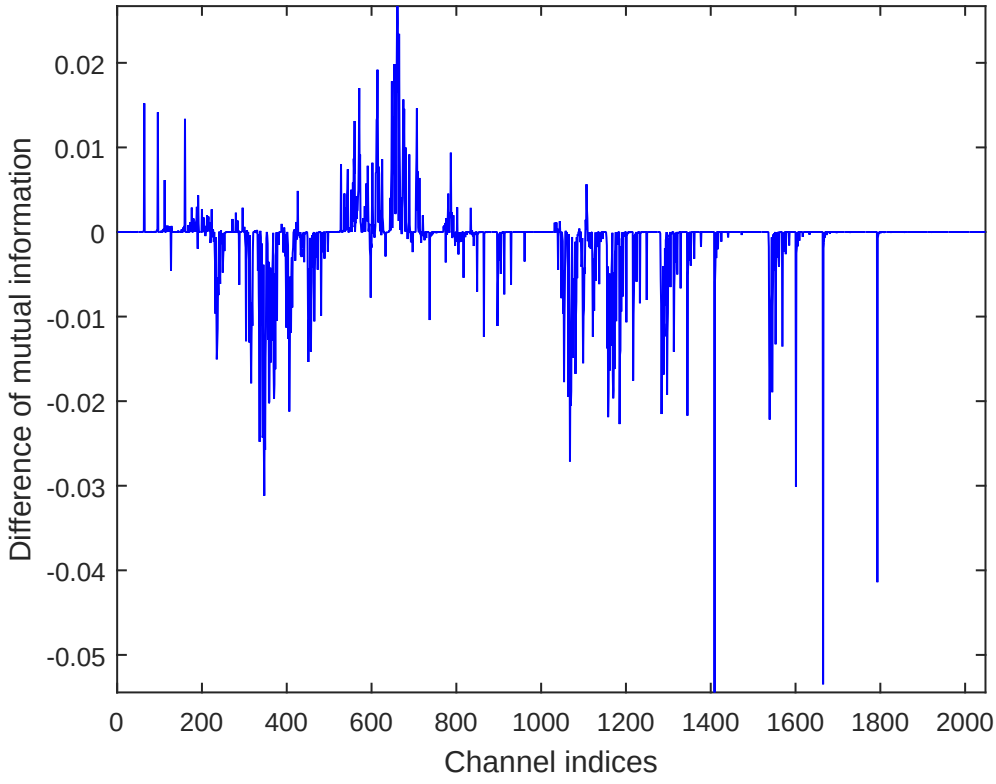

Figure 4.10: Difference of mutual information for a polar code of length $N = 2048$, rate $R = 1/2$ and designed for E_b/N_0 =3dB.

4.2 Importance of J-function for channel selection

design-$\frac{E_b}{N_0}$[dB]	% of different channels (No.)	$\frac{E_b}{N_0}$[dB] target	Loss [dB] at FER=10^{-3}
1	0.1953% (4)	2.9563	0.0605
1.5	0.3906% (8)	2.8380	0.2486
2	0.6836% (14)	2.7359	0.1667
2.5	1.8555% (38)	2.6673	0.5511
3	6.5430% (134)	2.7359	1.47
3.5	14.4531% (296)	2.7507	>1.5
4	23.0468% (472)	2.8837	>1.5

Table 4.1: Loss at FER=10^{-3} for a polar code of length $N = 2048$, rate $R = 1/2$, variable design-E_b/N_0 and using two different approximations.

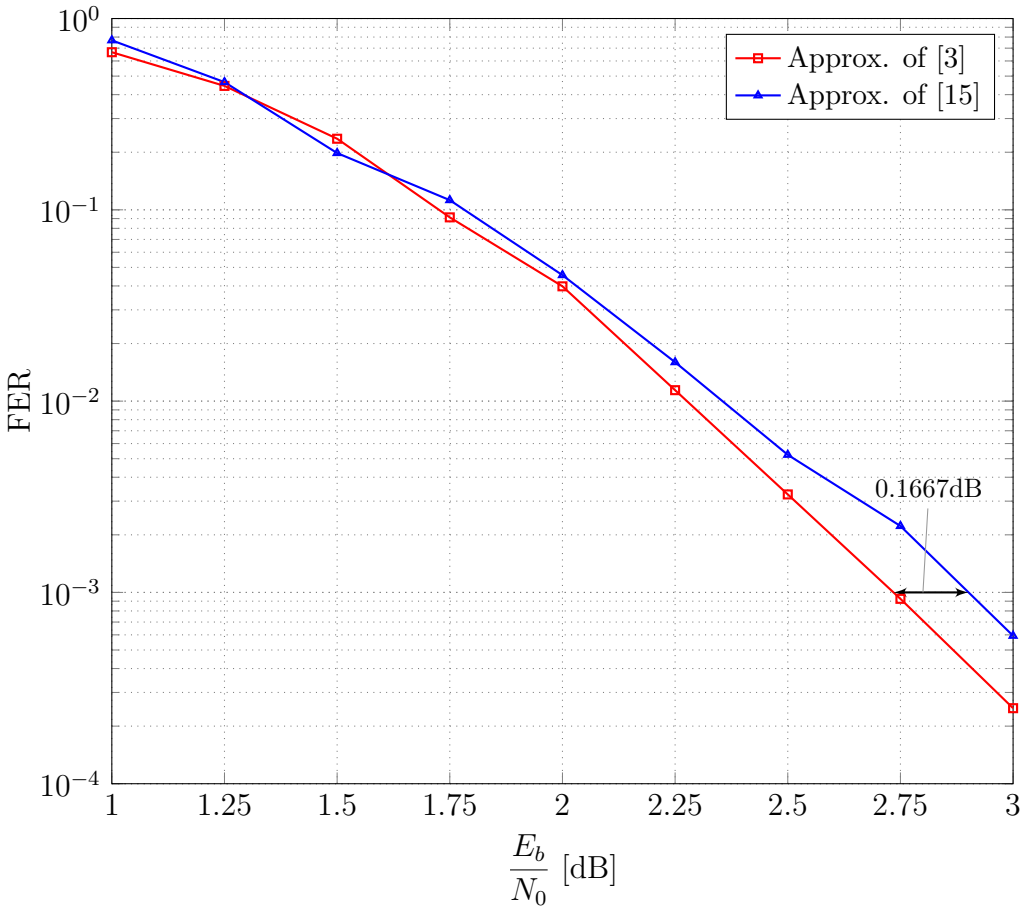

Figure 4.11: Performance of a polar code of length $N = 2048$, rate $R = 1/2$, designed for E_b/N_0 =2dB using approximations of [3] and [15].

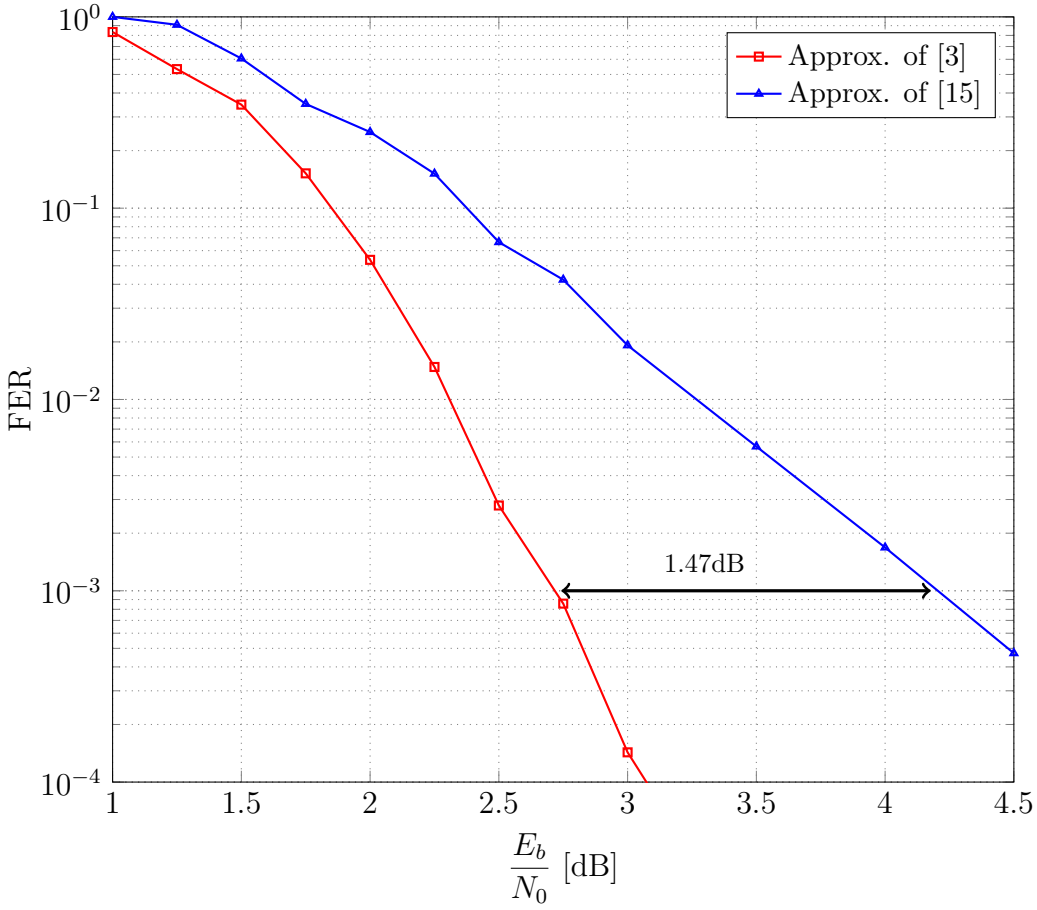

Figure 4.12: Performance of a polar code of length $N = 2048$, rate $R = 1/2$, designed for E_b/N_0 =3dB using different approximations of the $J(\cdot)$ functions.

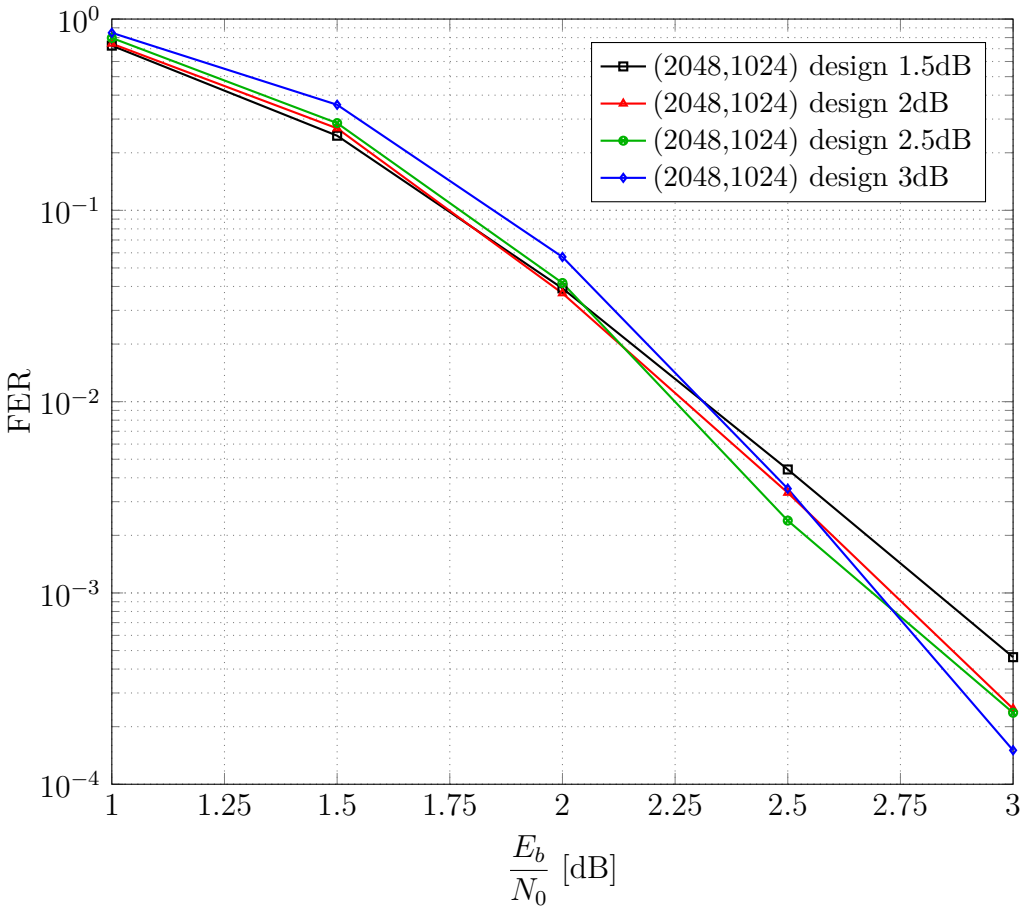

Figure 4.13: Performances comparison with different design-$\frac{E_b}{N_0}$[dB] using approximation of [3].

We report two examples: Figure 4.11 shows a performance loss of 0.1667dB for a FER=10^{-3} using the two different approximations explained above while Figure 4.12 shows a performance loss $\simeq$ 1.5dB for a FER=10^{-3}. Another indication of the exactness of our analysis tool is illustrated in Figure 4.13: it represents a comparison of SC decoding performances using different design-$\frac{E_b}{N_0}$ and the approximation [3]. Furthermore, it shows that the best performances are for the $\frac{E_b}{N_0}$ for which the code is designed. For instance for $\frac{E_b}{N_0}$ = 3dB, the lowest FER corresponds to the code with design-$\frac{E_b}{N_0}$ = 3dB.

Chapter 5

Results

First we compare our FERs results with references from literature to verify our implementations. For determining the performances, we have always used a Monte Carlo approach. Figure 5.1 shows a comparison between the result of list decoding with $L = 1$ from [13] (which corresponds to SC decoding) and our simulation for a polar code of length $N = 2048$, rate $R = 1/2$ and designed for E_b/N_0 =2dB. Figure 5.2 shows the performances of a polar code of length $N = 8192$ with the same characteristics as before. Furthermore, we found other interesting results from literature with different value of design-E_b/N_0. In particular [16] shows a curve for a code of length $N = 2048$, rate $R = 1/2$ but here it is designed for E_b/N_0 =3dB; we compared the performances in Figure 5.3. Finally, Figure 5.4 shows the performances of a polar code under SC decoding with variable lengths, rate $R = 1/2$ and designed for E_b/N_0 =2dB. From this summary picture we can also understand that the performances improve with $N \to \infty$.

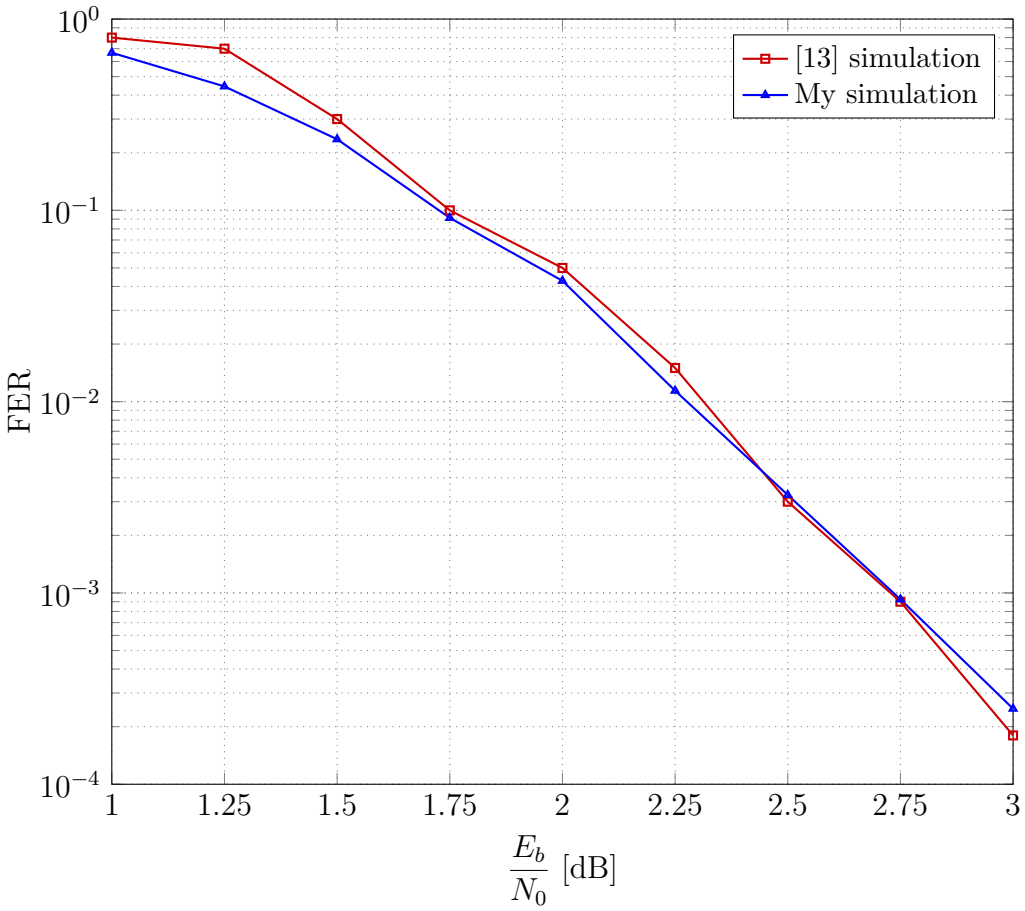

Figure 5.1: SC decoding performance of a polar code of length $N = 2048$, rate $R = 1/2$, designed for E_b/N_0 =2dB. The square markers curve is taken from [13] while the triangle markers one is the result of our simulation.

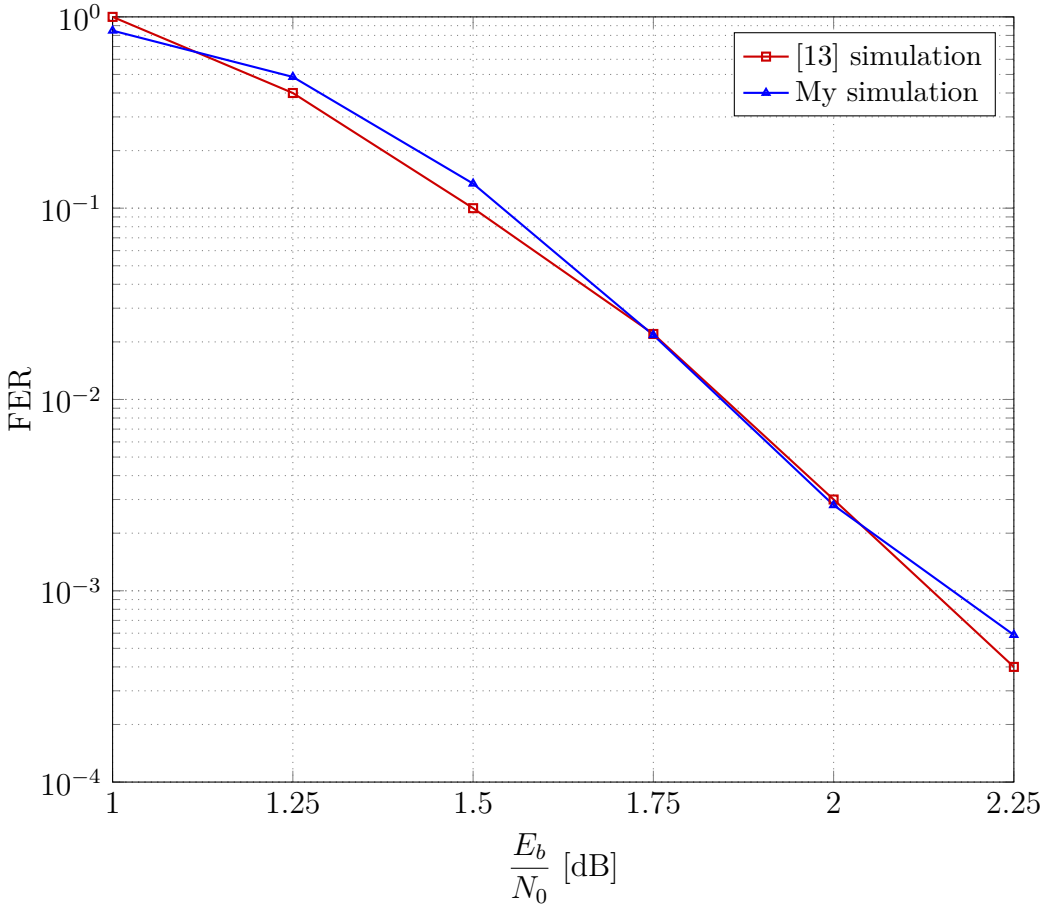

Figure 5.2: SC decoding performance of a polar code of length $N = 8192$, rate $R = 1/2$, designed for E_b/N_0 =2dB. The square markers curve is taken from [13] while the triangle markers one is the result of our simulation.

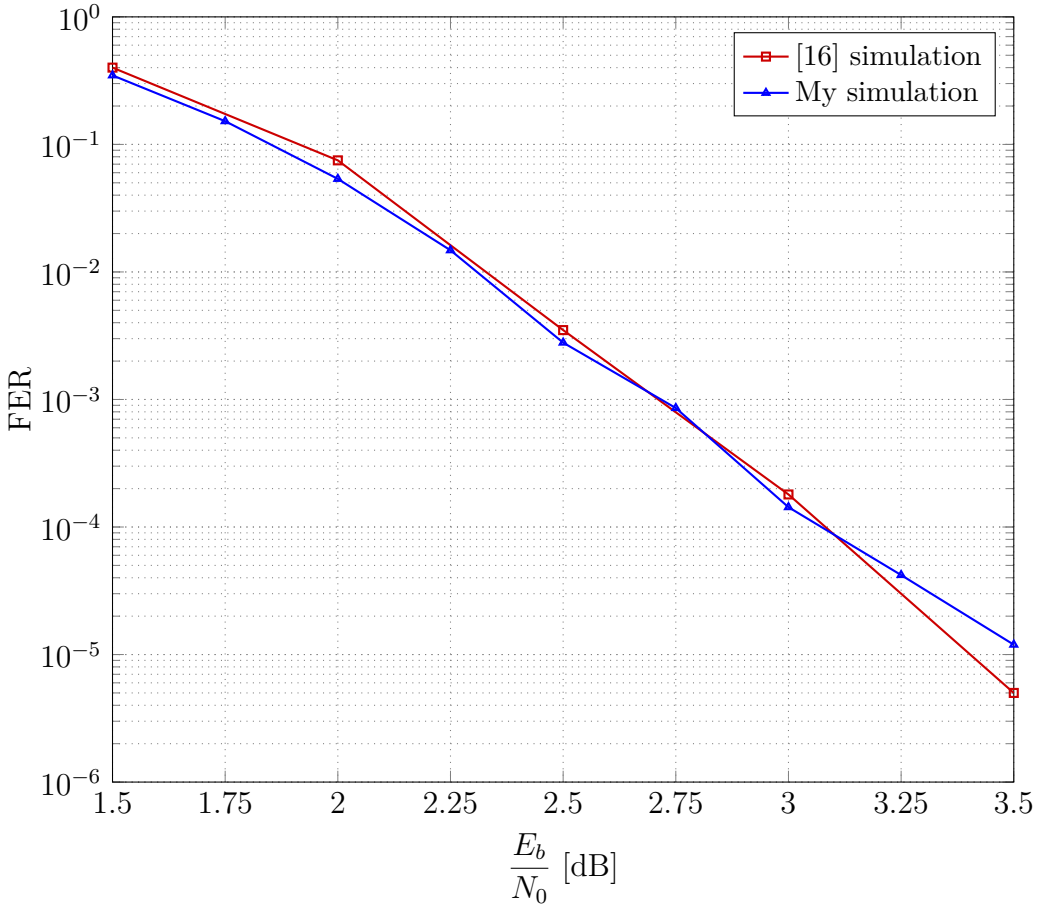

Figure 5.3: SC decoding performance of a polar code of length $N = 2048$, rate $R = 1/2$, designed for E_b/N_0 =3dB. The square markers curve is taken from [16] while the triangle markers one is the result of our simulation.

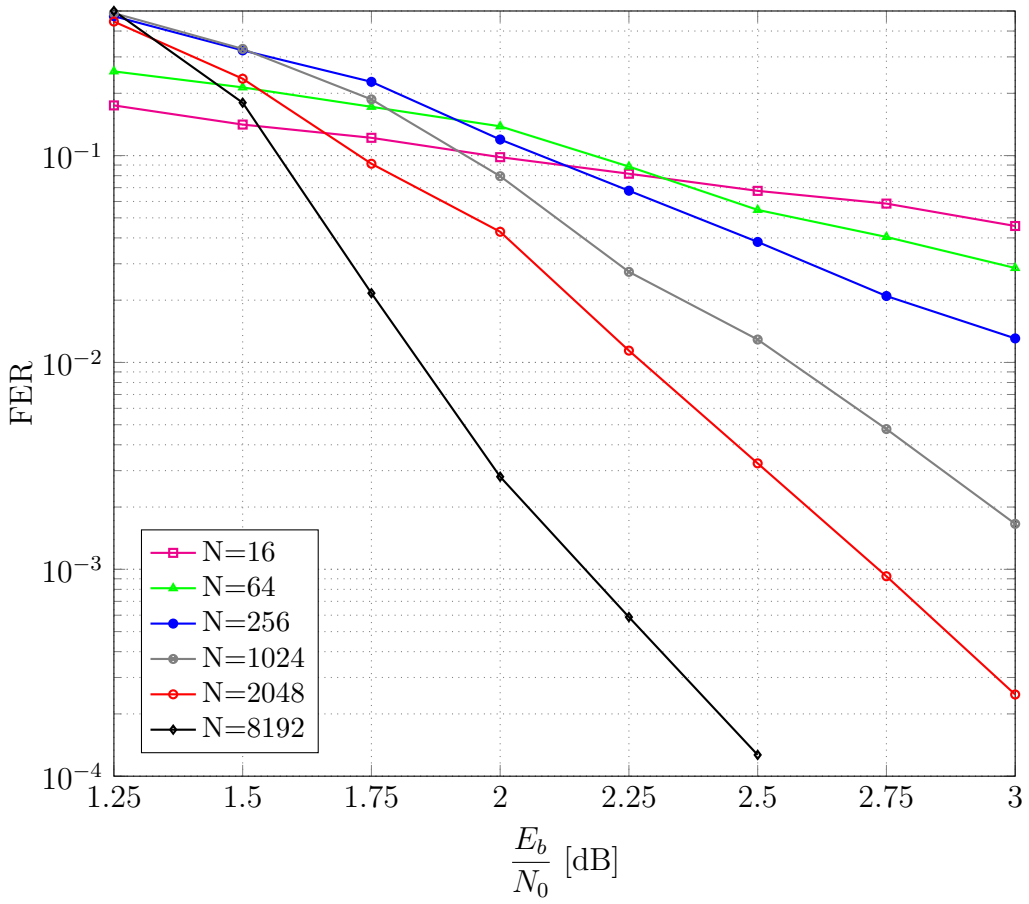

Figure 5.4: SC decoding performance of a polar code of rate $R = 1/2$ and variable lengths designed for E_b/N_0 =2dB.

Here we report some results about our SCL implementation and the concatenation with the CRC. Figure 5.5 and 5.6 show the performances of our SCL decoder simulator. In particular Figure 5.5 depicts the comparison between the CCSDS standard (128,64) binary protograph LDPC code used for deep space applications [4] and our (128,64) polar code for variable list sizes. The (128,64) polar code is worse than LDPC if we use only the SC decoding but, if we increase the list size, our performances improve. Furthermore, we can do much better with the concatenation of two codes as explained in Section 3.3. As can be seen from Figure 5.6, where we used an extended Hamming code (64,57) as a CRC, the resulting FER using an SCL inner code and CRC outer code outperforms both the only SCL decoding and the binary protograph LDPC code for applications that need FER$< 10^{-4}$. Moreover, we can see that the concatenation using a CRC implies a decrement of the error floor, namely it steepens the slope of the curve even if we loose something in terms of code rate. A summary of the gains introduced by the only SCL decoder and using both SCL and CRC concatenation compared to the performance of SC decoder for a FER=10^{-4} and for a (128,64) code is reported in Table 5.1. From this table we can also see that with this algorithm concatenation approach we can tune the tradeoff between complexity and performance depending on the requirements. For example for a FER requirement of 10^{-4}, we obtain similar results both for list size 2 and CRC and for list size 4 but we will probably choose the first option since the complexity is lower.

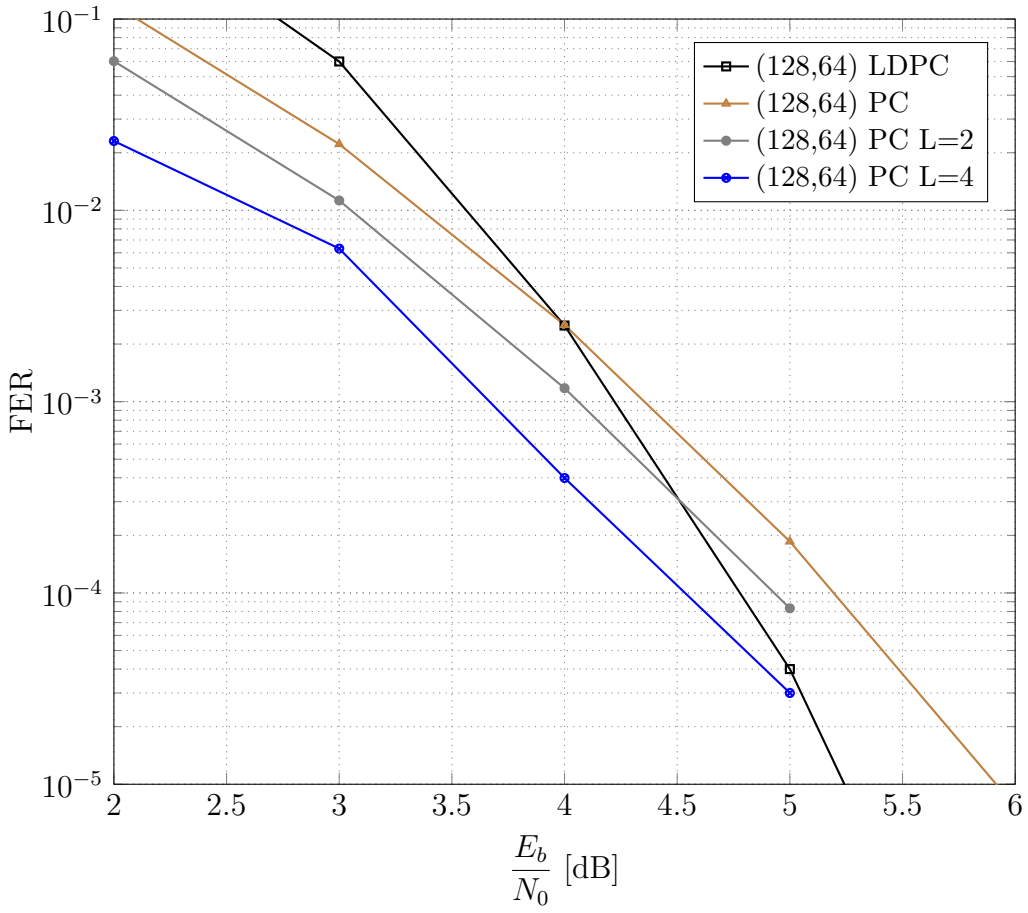

Figure 5.5: SCL decoding performances of a polar code of length $N = 128$, rate $R = 1/2$ optimized for E_b/N_0 =4.5dB for various list sizes compared with (128,64) binary protograph LDPC code.

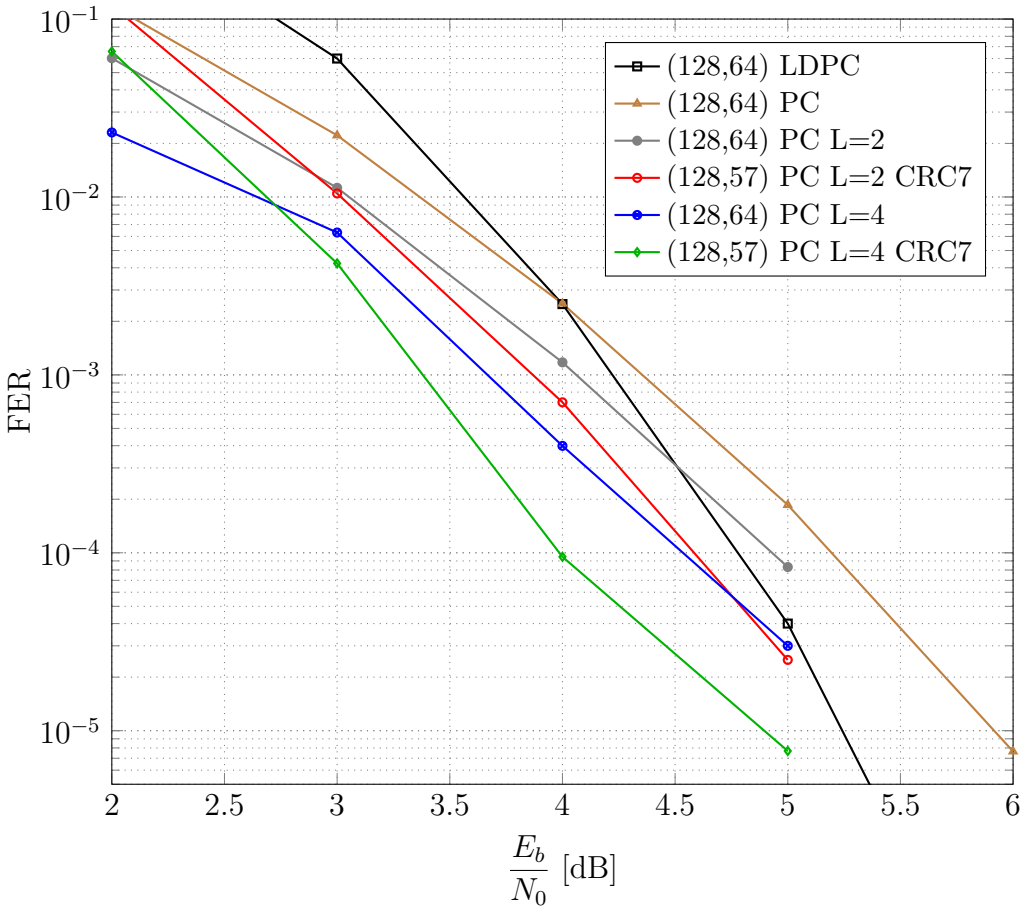

Figure 5.6: SCL decoding performance of a polar code of length $N = 128$, rate $R = 1/2$ optimized for E_b/N_0 =4.5dB for various list sizes using a further CRC outer code.

	Gain[dB] at FER=10^{-4}
List=2	0.243
List=2 + CRC7	0.582
List=4	0.630
List=4 + CRC7	1.191

Table 5.1: Gains at FER=10^{-4} using SCL decoder and concatenation for a (128,64) code compared to using only SC decoder.

Chapter 6

Conclusions

In this thesis the primary concepts of polar coding are reviewed, including channel polarization, encoding scheme and various decoding algorithms such as SC and SCL. Furthermore, we implemented a C++ simulator that reproduces SC decoding results from literature. We also developed a SCL decoder simulator that is an enhanced version of the previous one; it takes a list of possible paths and it decides for the most likely one. It has been further improved using a serial concatenation of a CRC and a SCL polar code, showing that it outperforms the (128,64) binary protograph LDPC code for applications that need FER$< 10^{-4}$. The concatenation of two codes involve a decrease of the error floor. Moreover, we introduced a novel analysis tool for the choice of the channels for an AWGN channel, demonstrating that channel polarization behaviour is keeping. We also investigated the importance of using different approximations of the J-function to decide the best channels for the transmission of data and we compared two approximations from literature, showing that polar codes are sensitive in terms of channel selections with different performances. The results of this thesis will be submitted for presentation at an international conference. Some possible future works could be a new polar transformation scheme that may improve the performance, a more complex decoding concatenation scheme where, after a fixed number of steps, we check if the list of codewords are partially correct during the SCL algorithm and we consequently prune the incorrect paths.

List of Figures

List of Tables

Bibliography

[1] Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *Information Theory, IEEE Transactions on*, 55(7):3051–3073, 2009.

[2] Claude Berrou and Alain Glavieux. Turbo codes. *Encyclopedia of Telecommunications*, 2003.

[3] Fredrik Brännström, Lars K Rasmussen, and Alex J Grant. Convergence analysis and optimal scheduling for multiple concatenated codes. *Information Theory, IEEE Transactions on*, 51(9):3354–3364, 2005.

[4] CCSDS. Short block length ldpc codes for tc synchronization and channel coding. *http://public.ccsds.org/publications/archive/231x1o1.pdf*, 2014.

[5] Ilya Dumer and Kirill Shabunov. Soft-decision decoding of reed-muller codes: recursive lists. *Information Theory, IEEE Transactions on*, 52(3):1260–1266, 2006.

[6] Robert G Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[7] Camille Leroux, Alexandre J Raymond, Gabi Sarkis, Ido Tal, Alexander Vardy, and Warren J Gross. Hardware implementation of successive-cancellation decoders for polar codes. *Journal of Signal Processing Systems*, 69(3):305–315, 2012.

[8] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[9] Henry Pfister. A brief introduction to polar codes. *http://pfister.ee.duke.edu/courses/ecen655/polar.pdf*, 2014.

[10] Thomas J Richardson, M Amin Shokrollahi, and Rüdiger L Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2):619–637, 2001.

[11] William Ryan and Shu Lin. *Channel codes: classical and modern*. Cambridge University Press, 2009.

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

[13] Ido Tal and Alexander Vardy. List decoding of polar codes. *CoRR*, abs/1206.0050, 2012.

[14] Stephan Ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *Communications, IEEE Transactions on*, 49(10):1727–1737, 2001.

[15] Stephan Ten Brink, Gerhard Kramer, and Alexei Ashikhmin. Design of low-density parity-check codes for modulation and detection. *Communications, IEEE Transactions on*, 52(4):670–678, 2004.

[16] Peter Trifonov. Efficient design and decoding of polar codes. *Communications, IEEE Transactions on*, 60(11):3221–3227, 2012.

Acknowledgements

First of all, I would like to express my gratitude to Gianluigi Liva, Balazs Matuz and Thomas Jerkovits, who helped me a lot and they were always available for every kind of question. The time spent with you was memorable and also very funny.
I am thankful to my professors Marco Chiani and Enrico Paolini for the wonderful opportunity to come in a fantastic and important place full of brilliant people as DLR in Munich. This experience was unforgettable.
Thank you to all the students and people I met during my experience, we spent a lot of funny and great time together. I hope we will meet again!
Un grazie va di sicuro anche alla mia famiglia, che mi ha supportato e motivato durante questi anni di studio, credendo sempre in me.
Vorrei ringraziare Martina, per aver sopportato ed accettato senza esitazione questi mesi in cui siamo stati distanti, capendo che sarebbe stata una indimenticabile ed indispensabile esperienza per il mio futuro. La lontananza non é stata facile, ma sono certo che abbia reso la nostra fantastica relazione ancora piú forte e la renderá sicuramente piú duratura.
Last but not the least, un grazie va anche a tutti i miei amici forlivesi ed ai compagni di universitá, per la fedele amicizia ed il continuo supporto in qualsiasi momento.
Spero di non aver dimenticato nessuno... nel dubbio grazie mille ancora a tutti quanti!! Thanks to all!!

Davide Dosio
"Polar Codes King"