AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF COMPUTER SCIENCE, ELECTRONICS AND TELECOMMUNICATIONS

INSTITUTE OF TELECOMMUNICATIONS

Engineering thesis

Implementation of a Polar Code

Author:	*Tomasz Lejkowski*
Degree programme:	*Electronics & Telecommunications*
Supervisor:	*dr hab. inż. Piotr Chołda*

Kraków, 2022

I sincerely thank . . . my family for the support they gave me at each step and my friends for being there for me when I needed them the most.

Contents

1. Introduction

Wireless communication systems are in need of good signal coverage and data throughput. The signal sent between the transmitter and the receiver is under influence of interferences. This induces errors in the transmitted message. Mobile broadband technology uses a lot of different techniques to counteract that. Capabilities of the connection system can be improved, for example by: subcarriers, dense base station placement, multiple input multiple output (*MIMO*) antenna arrays, and error correction coding (*ECC*) — being the main focus of this thesis.

Error correction codes utilize the technique of sending redundant information and use it to check if the data was transmitted correctly. In some cases the algorithm can correct errors too. A few of the well known basic codes are repetition codes or Hamming codes, but those do not satisfy constantly growing demand for data transfer. Newer turbo-codes are utilized in the 3G and the 4G standards, but low density parity check (*LDPC*) algorithms proved to perform better, so they are used in 4G and 5G New Radio technologies. New mobile broadband systems are in constant need of better error correction codes. That is why this thesis focuses on a construction of a polar code, an error correction code used in 5th generation telecommunication system (*5G*).

Chapter 2 of this thesis provides a description of a communication chain framework in which a polar code works. Further, information about the mobile broadband signal modulation schemes is given and common electromagnetic interference model is presented. Moreover, this chapter also introduces the mathematical phenomenon of a polar code, encoding, and the type of decoding in focus is called simplified successive cancellation. Alongside that, we can find explanations of the techniques used to enhance the code performance, main one being the list decoding with cyclic redundancy check.

Practical tests are prepared within the Matlab environment and results with scripts descriptions are presented in Chapter 3. Dependently on transmitted signal clarity, characteristics of bit error rate (*BER*) and block error rate (*BLER*) are formed to showcase the error correction capabilities of the code. We also measure the working time of the calculation to assess the computation power necessary to perform decoding. The outcomes of the tests are analysed, and compared to expectations. This way it is possible to show the unique advantages and disadvantages of the decoding algorithm type.

2. Background

2.1. Communication chain

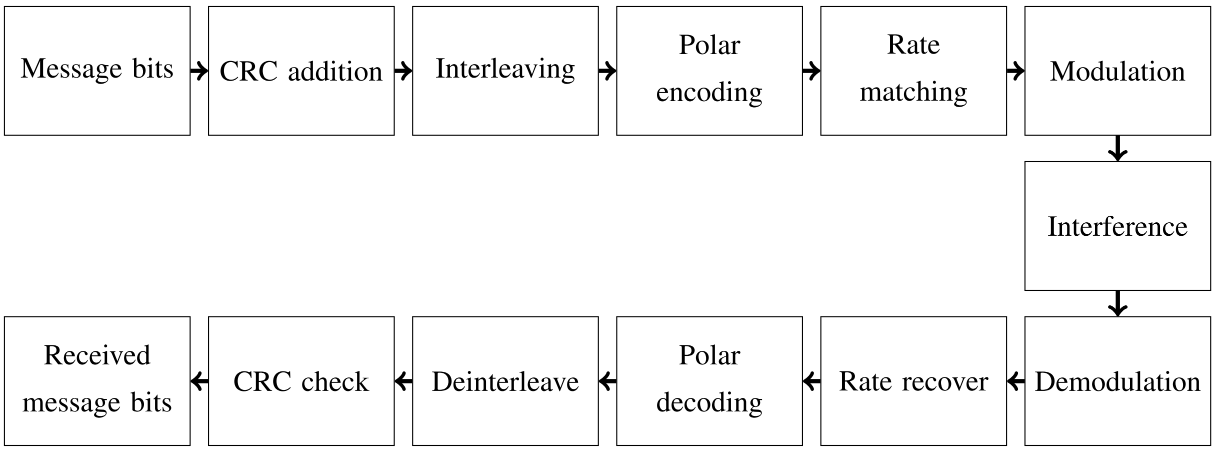

Figure 2.1. Communication chain block diagram

The scope of this thesis can be represented by a chain of data manipulation as shown in Figure 2.1. We can see a block representation of the communication chain, where the top row is the sending chain, and bottom row — the receiving one. This should not be confused with downlink and uplink operation. Downlink (*DL*) is when the mobile broadband provider transmits the data and user receives it, where on the contrary, uplink (*UL*) is the situation in which the user is sending the data to the provider.

In this thesis, for simplicity, we assume the transmitted message to be some pseudorandom input bit set. At first, cyclic redundancy check (*CRC*) bits are added. Those bits are used as an error detection at the receiver, which enhances the error correction performance of the list type decoder.

Interleaving is a stable way to scramble the data by a set pattern. This is needed because digital data structures often include long sequences of repeated zeros or ones. During processing, those bit bursts can divert the system performance. As it is highly beneficial to scramble the data, most of the transmissions schemes introduce interleaving. Technical specification for 5G systems [10] in downlink scenario introduces map interleaving to be just before the polar encoder input. Contrary to that, uplink interleaving is done after rate matching, just before modulation.

Rate matching is crucial in a communication system, it adapts the size of the payload given by a polar code to the size required by functions of higher order. At last, common modulation schemes are used to represent the symbols. To induce errors, generated signal undergoes interference, which is represented as simple mathematical model. When the procedure is reversed and message bits are decoded and the obtained output can be compared to the beginning input bit set.

2.2. Signal modulation

Wireless communication is based on electromagnetic (*EM*) wave modulation, which is a way of mapping data into changes of the wave displacement within symbol duration time. Where data symbol can be a single bit or a set of bits. There are a few fundamental types of digital modulation, such as: amplitude shift keying (*ASK*), frequency shift keying (*FSK*) and phase shift keying (*PSK*), which all come from their analog predecessors. The electromagnetic wave can be represented as a sine wave, such as given in Equation (2.1) below.

$$y(x,t) = A\sin\left(\frac{2\pi}{\lambda}x - \omega t + \phi\right) \tag{2.1}$$

where:

$y(x,t)$	– modulated waveform,
A	– amplitude of the signal,
λ	– wavelength of the carrier,
ω	– angular frequency,
ϕ	– phase shift,
x	– displacement,
t	– symbol duration time.

The most basic digital modulation is binary phase shift keying (*BPSK*). This is a technique that consists of chaining the phase shift of our carrier signal by $180°$ for each bit change, as seen on example in Figure 2.2. For clarity and ease of use, so called ‘constellation diagrams’ are used. Those present a two dimensional signal space which easily shows the mapping of our bits to wave properties used. Each data symbol is mapped on the constellation diagram as a point. Where the amplitude of the carrier wave is the distance of the point from the origin and the phase is the counterclockwise angle from the horizontal axis. Any constellation diagram can be represented as a complex plane, thus the points can be referred to as complex numbers. We can see that BPSK in 5G [11] can be mapped onto points A and C seen in Figure 2.3, assuming points B and D do not exist.

Contrary to BPSK, $\pi/2$-BPSK is not a strict mapping of bit sets onto waveform properties. Some systems benefit from low phase change during transmission, hence $\pi/2$-BPSK is based on shifting the carrier signal by $90°$, for every bit change. This would map onto every point on the constellation diagram

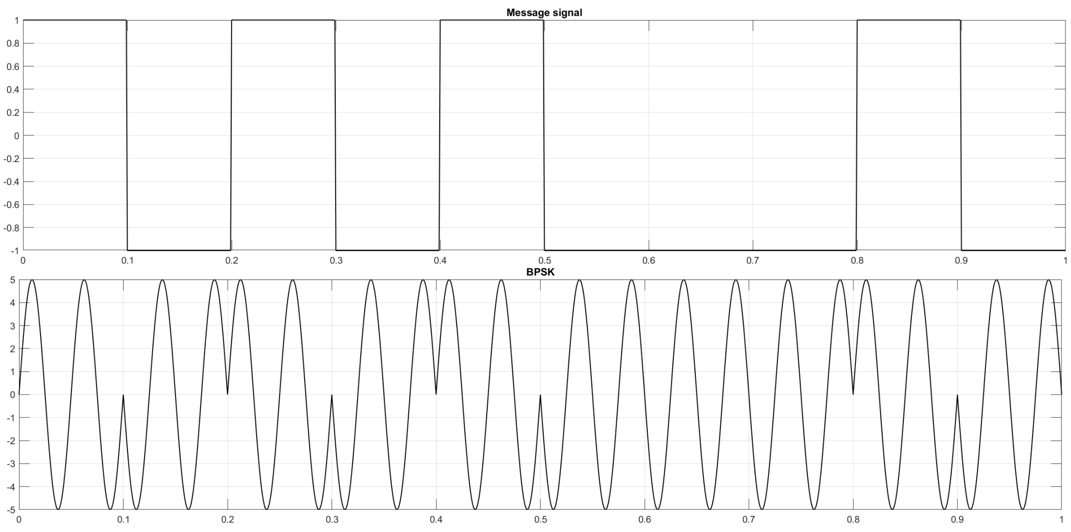

Figure 2.2. Example of a message signal (top) and BPSK modulated waveform (bottom)

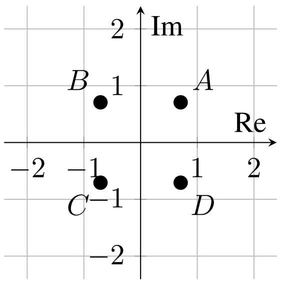

Figure 2.3. Constellation diagram for $\pi/2$-BPSK and QPSK modulation types

Table 2.1. QPSK 5G NR bits constellation mapping

Bit set	Point in Figure 2.3	Complex value representation
1 1	A	$\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
0 1	B	$-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
0 0	C	$-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$
1 0	D	$\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

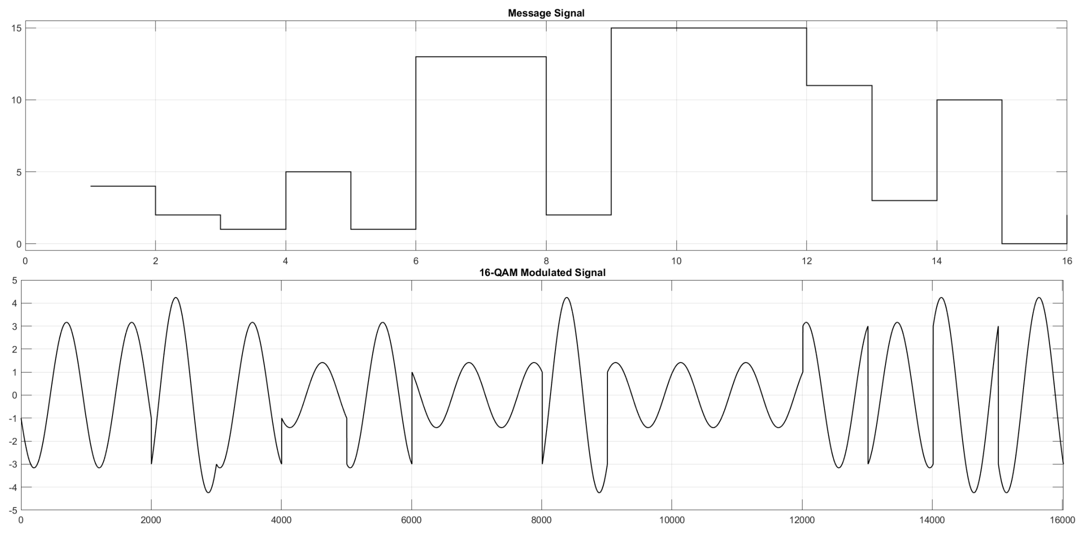

Figure 2.4. Example of message signal (top) and 16QAM modulated waveform (bottom)

shown in Figure 2.3. Another modulation type which would utilize all of the points is quadrature phase shift keying (*QPSK*). Here, Table 2.1 represents the concept of QPSK [11].

The quadrature phase shift keying can be enhanced with amplitude modulation. This creates a waveform similar to the one shown in Figure 2.4. This technique gives us the ability to send more information bits with one waveform change, which is clearly represented by the constellation diagram shown in Figure 2.5.

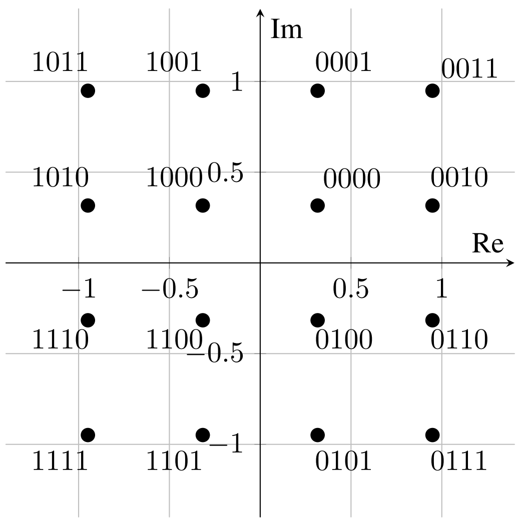

Figure 2.5. Constellation diagram for 16QAM

It is important to know that there exist higher order of quadrature amplitude modulation (*QAM*), such as 64QAM, 256QAM, which are derived similarly to the example stated. Because of their complex structure, they are more prone to interference.

2.3. Transmission — AWGN interference

Along EM wave propagation routes it is common that external interference influences our signal. This external interference can be mathematically modeled in a particular way. The simplest and most basic concept is the additive white Gaussian noise (*AWGN*) channel model. Here, with compliance to the classical textbook [13] Equation (2.2) is given. The transmitted signal $s(t)$, with some attenuation α, undergoes the linear addition of random noise $n(t)$. The noise has the uniform power distribution across the frequency band, and the Gaussian distribution of amplitude, as seen in the Equation (2.3).

$$r(t) = \alpha s(t) + n(t) \tag{2.2}$$

$$n \sim \mathcal{N}(\mu, \sigma^2) \tag{2.3}$$

where:

$r(t)$	–	received signal,
α	–	attenuation factor,
$s(t)$	–	transmitted signal,
$n(t)$	–	random noise,
μ	–	mean value of the Gaussian distribution,
σ^2	–	variance of the Gaussian distribution.

It is important to remember that there exists a lot of different types of interference models, where few are presented in [13]. Some of the types take into consideration different paths a signal can take, multi carrier communications, MIMO technology, or the movement of receiving/transmitting node.

2.4. Cyclic redundancy check

Cyclic redundancy check (*CRC*) is based on cyclic codes and widely used as an error detection technique in network transfer protocols and data storage. Cyclic redundancy check represents binary data as polynomials and generates parity bits which are appended to the data. The algorithm used here is polynomial long division within GF(2) with the remainder being our parity information. The concept is to divide our message polynomial by a divisor, ‘generator’ polynomial. As a result, we get some quotient, which is not important, because our reminder is a polynomial representing our parity bits. As an example we can see Equation (2.4) and (2.5) given by [8].

We can easily assure the amount of bits necessary to save the parity reminder information. We know that the remainder polynomial degree $\deg(r(x))$, will always be less then degree of the generator polynomial $\deg(d(x))$ (see Equation (2.5)). As each degree represents a bit state, we know that the amount of remainder parity bits will always be less then the degree of generator polynomial. That's why encoding procedure starts with appending $\deg(d(x))$ zeros into our bit message vector, or multiply message polynomial with $x^{\deg(d(x))}$. Further the division by generator polynomial yields us the reminder which is added to our message. This last operation states the decoding conditions, and keeps the message visible and unchanged within the created code word. When the whole encoded data is divided by the same generator polynomial, the reminder is equal to 0, which states no error was found within the set pattern.

$$p(x) = q(x)d(x) + r(x) \tag{2.4}$$

$$0 \leq \deg(r(x)) < \deg(d(x)) \tag{2.5}$$

where:

$p(x)$	– polynomial representing our data,
$q(x)$	– some quotient ,
$d(x)$	– the generator polynomial,
$r(x)$	– reminder representing CRC parity bits.

It is important not to trust our CRC check fully. This technique is vulnerable due to the patterns set by the generator polynomial. Source [8] describes details of how the parity bits are calculated and the relationships between generator polynomial and patterns detected by them. What is important to remember is that the Hamming distance (the theoretical amount of errors that can be detected minus one), of the CRC is set by the generator polynomial, and there is a set block length for generator polynomial to ensure the Hamming distance. Error bursts of length higher than the degree of the generator polynomial $(\deg d(x))$ have lower probability of being recognised. Finding a good new generator polynomial is not easy. This is why many standards provide information about which generator polynomial to use, and how parity bits should be saved.

2.5. Polar Code

In this section, we will discuss the basic structure of a polar code, that is: elements, construction and decoding with enhancement — 'aided' decoding. Polar coding is based on the random-coding method introduced by Shannon [15], where exact polar encoding and decoding methods were proposed by Erdal Arikan and published in 2009 [1]. Its appearance in the new technology, such as 5G NR [10], is understandable, as it stands out with low complexity and capacity-achieving error correction capabilities while working with supplementary codes.

2.5.1. Bit channel polarization

The polarization is a procedure in which we take N independent synthetic bit channels and manipulate them in a way that their mutual information changes. With sufficiently high N, the mutual information tends to 0 (very noisy) or 1 (noiseless) [1], [2]. This results in channels with extreme capacities, where we refer to capacity as the maximum rate at which the information can be reliably transferred [16].

The channel manipulation procedure for $N = 2$ is presented in Figure 2.6. Polarization transformation can be also represented as a kernel matrix written in Equations (2.7), (2.9). Polarization matrix G_2 allows encoding of input vector $u = [u_0, u_1]$ into a codeword $d = [d_0, d_1]$ as $d = uG_2$ where $d_0 = u_0 \oplus u_1$ and $d_1 = u_1$ (see Figure 2.6). Expansion is possible and is shown by the Kronecker (tensor) power in Equation (2.8) [1]. The higher order kernel representation can be seen in Figure 2.8.

$$N = 2^n, n \in (\mathbb{N} = \{1, 2, 3, \ldots\}) \tag{2.6}$$

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.7}$$

$$G_N = \begin{bmatrix} G_2 \end{bmatrix}^{\otimes n} \tag{2.8}$$

$$G_4 = \begin{bmatrix} G_2 \end{bmatrix}^{\otimes 2} = \begin{bmatrix} 1G_2 & 0G_2 \\ 1G_2 & 1G_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.9}$$

where:

N	–	input vector length,
G_2	–	polarization matrix for $N = 2$,
G_N	–	general equation for polarization matrix of size N,
G_4	–	polarization matrix for $N = 4$.

The polarization effect can be easily shown on the example of binary erasure channels (*BEC*), where the input bit is transmitted correctly, or lost with probability p (see Figure 2.7). If any of the code bits are not received, $u_0 = d_0 \oplus d_1$ cannot be recovered. That means losing bit d_0 or d_1 corrupts the output. This leads to addition of p probabilities of losing bit d_0 or d_1. By addition rule of probabilities we obtain $p_0 = p + p - p^2$. On the contrary, bit u_1 can be decoded as $u_1 = d_1$ or $u_1 = u_0 \oplus d_0$. This means that for u_1 bit to go extinct, both d_0 and d_1 bits must be lost, which equals $p_1 = p^2$. We can observe that probability of d_0 bit being lost increased $p_0 = (2 - p)p > p$, where bit d_1 decreased $p_1 = p^2 < p$ [1], [2].

These dependencies are more complicated with higher order kernel (see Figure 2.8). When BEC channel is denoted as W, using coupling showed in Figure 2.9, it is possible to assess the capacity $1 - p_N$ of each bit channel. This is done by source [1] and it shows that most of the bit indicies have the capacity close to 0 (unreliable) or 1 (very reliable). This is shown in Figure 2.10 provided by source [1].

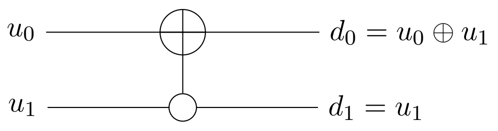

Figure 2.6. Representation of polarization kernel G_2

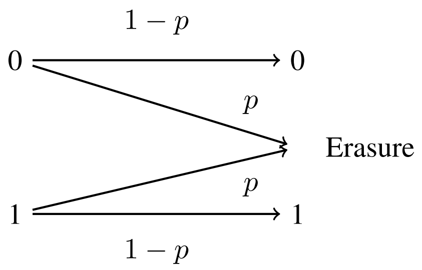

Figure 2.7. Binary erasure channel model

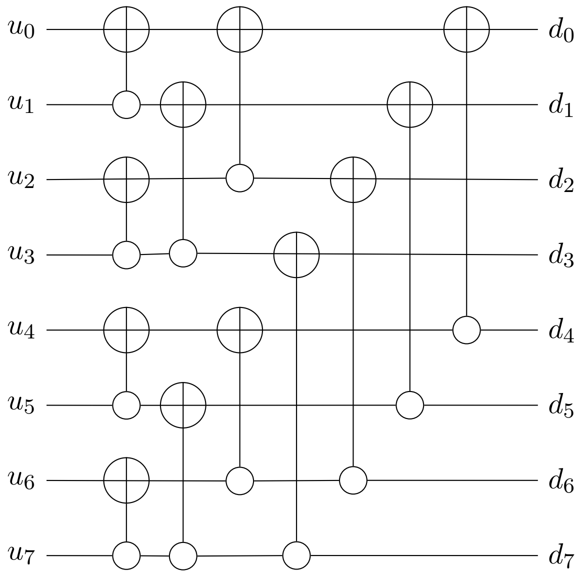

Figure 2.8. Representation of polarization kernel G_8

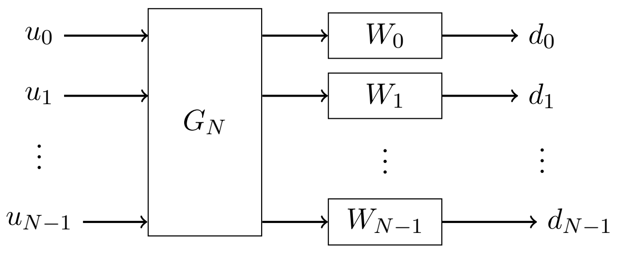

Figure 2.9. Polarisation of synthetic channels

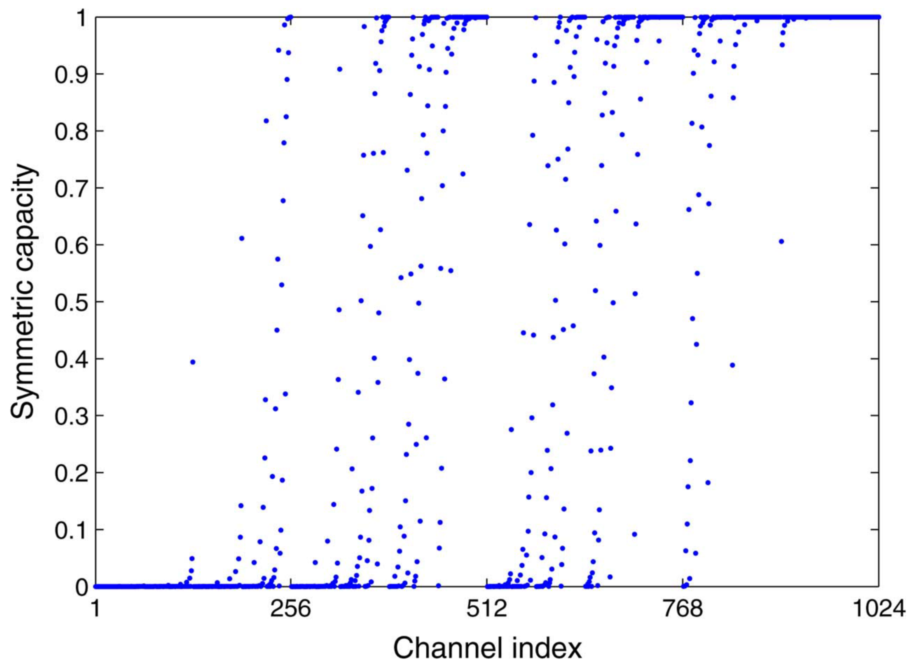

Figure 2.10. Bit channel capacity for BEC $\frac{1}{2}$ versus channel index N

2.5.2. Encoding

For the previously explained channel polarization phenomenon, (N, K) polar code is defined with N block length and K message bits, thus $N - K$ being the redundancy bits. Where $N - K$ least reliable bits of the polarization are forced to be 0, the so called 'frozen bits' or 'frozen positions'. This pattern, called 'reliability sequence' or 'frozen set', is known and set at the transmitter and receiver equally. By forcing low reliability bits to a set value we increase the chance of decoding the information bits correctly.

The frozen set influences the polar code error correction capabilities. The ideal set depends on the interference conditions, code length and techniques used to aid the performance of a polar code. Source [2] briefly covers and points out papers describing the techniques used to design the frozen set. Due to constantly changing conditions during transmission, an ideal constant reliability sequence does not exist. On-the-fly design of the set for AWGN channel was proposed, but it requires high computational power and induces too much latency. This is why, for simplicity, in the 5G standard [10] the reliability sequence is set constant and can be easily adapted. This sequence is created with rate matching and different types of decoding algorithm assists in mind, such as additional parity checks or CRC.

An example of the encoding procedure can be noted similarly to sources [2], [18]. The block is of length $N = 8$, message length is $K = 4$, and the frozen set for $N - K$ bits being $F_4 = \{0, 1, 2, 4\}$. Each number in F_4 represents the bit index of input vector $u = [u_0, u_1, ..., u_{N-1}]$. The message can be generalized as $m = [m_0, m_1, ..., m_{K-1}]$, for example $m = [1, 0, 1, 1]$. Message bits are put on the positions left behind by frozen set in the u vector of N length. Based on that we can encode, $u = [0, 0, 0, m_0, 0, m_1, m_2, m_3] = [0, 0, 0, 1, 0, 0, 1, 1]$. Vector u can be polarized as denoted earlier $d = u \cdot G_8$ creating the code word $d = [1, 0, 1, 0, 0, 1, 0, 1]$.

2.5.3. Simplified successive cancellation decoding

Arikan in his original article [1] proposed a successive cancellation (*SC*) decoder, it is based on the idea of decoding bits one by one. This is not very efficient. Thanks to the fact that the code has a recursive structure, such as the length N consists of two polar code structures of length $\frac{N}{2}$, some operations can be repeated in a certain manner. This strongly reduces complexity. To easily describe the SSC workflow, a binary tree representation can be used alongside depth-first type of algorithm to create simplified successive cancellation (*SSC*) [4], [12], [14], [18]. Let us take a look at an SSC decoder behaviour for the block length $N = 2$. Based on that example, a few concepts will be described. Then, we will generalize for arbitrarily big decoder.

Let us have a vector $u = [u_0, u_1]$ representing a message and frozen bits inside. This vector u was polarized, and sent through a channel with interferences. The received information can be noted as $r = [r_0, r_1]$, where we denote r_0, r_1 as received belief of a bit code word. What we refer to as the belief L is a 'soft information' which is a likelihood of a bit having particular value. Log-likelihood ratio (*LLR*) was proven to provide better stability and is more suitable for hardware application, thus it is widely used [2]. Log-likelihood is a logarithmic ratio of the likelihood of a bit having value 0 to a likelihood of a bit having value 1. A simple tree seen in Figure 2.11 can be drawn to showcase the decoding procedures. We start at the root node (most top one), at first, we perform the 'left step'.

As we can see in Figure 2.11, we are sending to the left node the likelihoods calculated by function f, presented in Equation (2.10) [12], [18]. Function f is soft decoding type of equation which originates from single parity check, it varies the bit belief based on the eccentric information. In Figure 2.11 we can see that the node we are sending soft decoded likelihood to is a so called 'leaf node', meaning it is the deepest one in the tree. If the leaf node is not at frozen position it performs 'hard decoding', as presented in Equation (2.11) [12], [18]. Hard decoding means that the belief is changed to a fixed bit value. If the leaf is at the frozen position, the belief is disregarded and the bit is forced to be equal to 0.

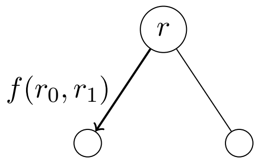

Figure 2.11. Example left step of successive cancellation for $N = 2$

What comes back from the bottom left node is a hard decoded bit denoted as $L(u_0) = \hat{u_0}$. Based on $\hat{u_0}$, the 'right step' is performed as seen in Figure 2.12. Using Equations (2.12) and (2.13) [12], [18] we can obtain $L(u_1)$ which can be hard decoded as in Equation (2.11) into $\hat{u_1}$, which gives us the proposed output.

$$L(u_0) = f(r_0, r_1) = \text{sgn}(r_0)\text{sgn}(r_1)\text{min}(|r_0|, |r_1|) \tag{2.10}$$

$$\hat{u_0} = 0 : L(u_0) \geq 0\,;\ \hat{u_0} = 1 : L(u_0) < 0 \tag{2.11}$$

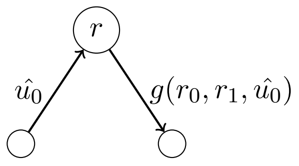

Figure 2.12. Example right step of successive cancellation for $N = 2$

$$g(r_0, r_1, \hat{u_0}) = r_1 + (1 - 2\hat{u_0})r_0 \tag{2.12}$$

$$L(u_1) = r_1 + r_0 : \hat{u_0} = 0\,;\; L(u_1) = r_1 - r_0 : \hat{u_0} = 1 \tag{2.13}$$

Generalizing the whole procedure in a big tree, we can declare a few repeating steps. We start at the top, at the root node. Each node performs the left step first, then waits (see Figure 2.13). As seen in Figure 2.16, after few steps, we stumble upon a leaf node, where hard decision is sent back up. If a node receives hard decision from the left child, the right step is performed and again the node waits (see Figure 2.14). When the node receives back the decisions from the right child, the 'up step' is performed and the procedure can continue as seen in Figure 2.17.

The main goal of the left step in the SSC decoding consists in sending the correct beliefs to the left child node. A parent node sends the incoming vector of beliefs L, which has some length $M = 2^m$, where $m \in (\mathbb{N} = \{1, 2, 3, \ldots\})$ and $M \leq N$. Incoming vector L is split in half, and function f is applied coordinatewise, as seen in Equation (2.14) [12], [18]. This operation lowers the amount of beliefs two times, that is why only root node performs left step where $M = N$, the next node should perform a left step where $M = \frac{N}{2}$.

$$f(a_{0:P}, b_{0:P}) = [f(a_0, b_0), f(a_1, b_1), ..., f(a_P, b_P)] \tag{2.14}$$

$$g(a_{0:P}, b_{0:P}, c_{0:P}) = [g(a_0, b_0, c_0), g(a_1, b_1, c_1), ..., g(a_P, b_P, c_P)] \tag{2.15}$$

When a node receives the decisions from its left child, the right step is performed, which can be generalised as seen in Figure 2.14. Let us set that what comes from the left node is a vector c_L containing hard decoded bits. We send the beliefs from the parent node and bits from the left node to the right node, applying function g coordinatewise, as seen in Equation (2.15) [12], [18]. What was not presented in the previous $N = 2$ example is the up step. This step combines the bits, as seen in Figure 2.15, and sends them up the tree to the parent node.

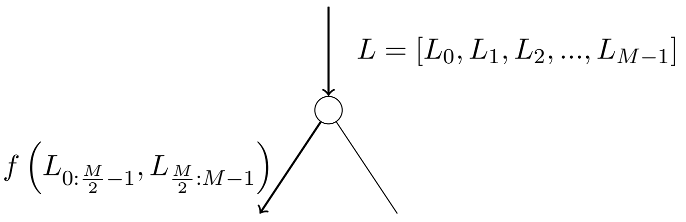
$L = [L_0, L_1, L_2, ..., L_{M-1}]$
$f\left(L_{0:\frac{M}{2}-1}, L_{\frac{M}{2}:M-1}\right)$

Figure 2.13. General left step in the simplified successive cancellation decoding tree

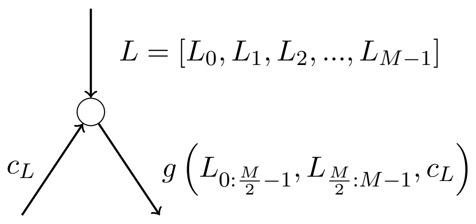
$L = [L_0, L_1, L_2, ..., L_{M-1}]$
c_L
$g\left(L_{0:\frac{M}{2}-1}, L_{\frac{M}{2}:M-1}, c_L\right)$

Figure 2.14. General right step in the simplified successive cancellation decoding tree

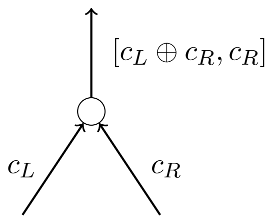

Figure 2.15. General up step in the simplified successive cancellation decoding tree

The leaf nodes behave like stated in the example $N = 2$, meaning that if the node is frozen, the decoded bit is equal to 0. Whereas if the node is not a leaf, the hard decision given in Equation (2.11) is applied, and the bit is sent back up. The decoded set of information can be found in the leafs of the tree, where each leaf node represents one bit (see Figure 2.16). This set contains the frozen bits and the message bits. By erasing the frozen bits, we find our message.

2.5.4. Successive cancellation list decoding

The successive cancellation decoder, at moderate block lengths, does not provide sufficiently good performance to be widely implemented. List decoding overcomes that by running a group of (N, K) decoders, each proposing a decoded message. In this way, we obtain a list of possible outcomes. From the list, we can choose, increasing the chance of decoding correctly. The decision which message to

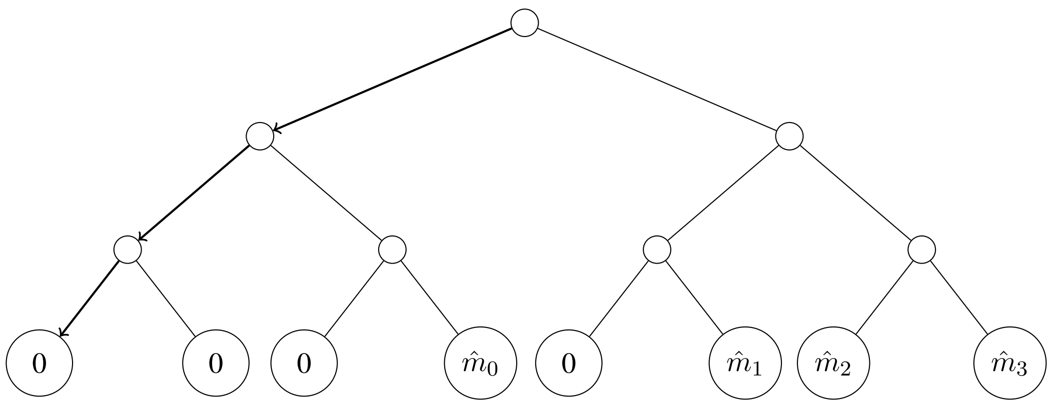

Figure 2.16. Example of the simplified successive cancellation decoding binary tree representation for $N = 8$

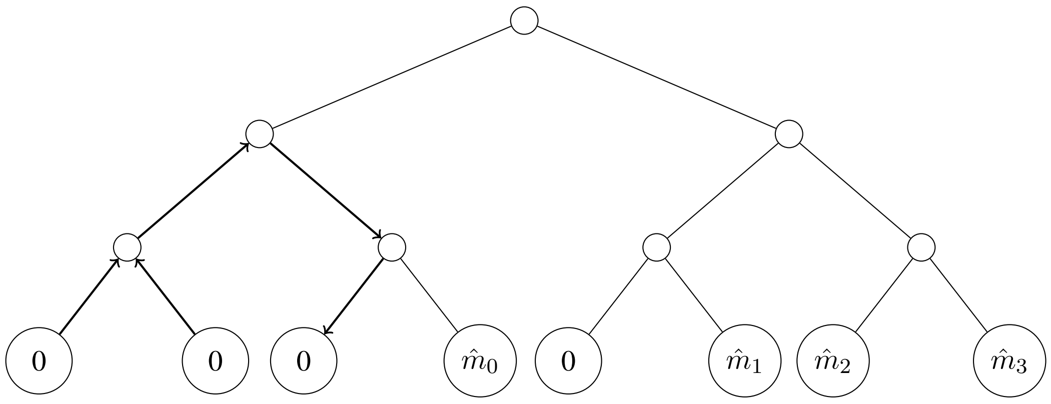

Figure 2.17. Example of the simplified successive cancellation decoding binary tree representation for $N = 8$

choose can be made based on the most probable outcome or it can be aided by the indication from any error detection technique which message has no errors — where commonly used is CRC. List decoding was proven to perform exceptionally well while used with any aiding technique [9], [17], [18], [19].

The key difference in list decoding versus SC type decoding is in the hard decision. Each hard decision is allowed to take both values — zero or one. This creates the so called 'paths', where each path saves the previously decoded bits and the information about probability of being correct.

When each hard decision is made, the 'decision metric' (*DM*) is calculated as in Equation (2.16), (2.17) and added to the 'path metric' (*PM*) (see Equation 2.18) [18]. The path metric is a scale showing if the decisions were made with or against the received likelihood. When the PM is closer to 0 it means the decisions were made according to received belief. High PM means there were many or strong decisions against the received belief. This implies the assumption that the probability of this path being correct is low. What is important to know is that frozen bits are always set to zero, but the DM for each frozen bit is still calculated and added to PM.

Let us have a list decoder with a list size of nL, that means we allow nL amount of (N, K) decoders to to run simultaneously, each creating a path. At each hard decision, the amount of paths doubles. As we can imagine, we quickly approach the maximum number of the paths nL. When that happens, we let the decoder calculate $2nL$ paths and based on the PM of each path, apply 'pruning' (erasing) of the least reliable paths. This achieves the maximum number of active paths nL, and progresses further.

$$L(u_i) \geq 0 : \hat{u}_i = 0 : DM_i = 0;\ \hat{u}_i = 1 : DM_i = |L(u_i)| \tag{2.16}$$

$$L(u_i) < 0 : \hat{u}_i = 1 : DM_i = 0;\ \hat{u}_i = 0 : DM_i = |L(u_i)| \tag{2.17}$$

$$PM = \sum_{i=1}^{N} DM_i \tag{2.18}$$

Having nL propositions of the decoded message, the final output must be chosen. For that, PM should be used, but main idea of list coding is for it to work with some error detection technique, for that CRC is often used. The CRC bits are appended to the message before encoder, this lowers the number of message bits that can fit, but it strongly enhances the error correction performance of a list decoder [9], [17]. It is said that A bits represent message, so the (N, K) polar code is described with K represented as A summed with the length of CRC. At the receiver, we check CRC for each path, and pick the one where the PM is low and CRC passes. When CRC fails, that is when none or many paths pass, the message should be picked by PM. At last, the CRC bits are disregarded and we are left with our decoded message.

2.5.5. Parity-check-concentrated decoding

As sources state, the polar code is weak with low block sizes [1], [2], but for list type decoder its error correction performance greatly improves when aided with other codes [9]. One of the aiding codes in 5G specification is a simple parity check. In 5G it applies in specific low block sizes [10]. Parity-check-concentrated (*PCC*) polar codes are not the main focus of this thesis, hence the topic is mentioned briefly.

Besides CRC, 5G specification provides criteria and algorithms used to compute and add the simple parity bits to our message. This is further described in depth and from the telecommunications industry point of view in [5]. List decoding with parity check and CRC perform exceptionally well. Combining the procedures into 'hybrid' algorithms can lower the complexity, while maintaining high error correction capabilities [3]. As a polar code aiding is a very broad and strongly growing branch of coding, many solutions based on the same concept but with better error correction performance are being researched [6].

2.6. Rate matching

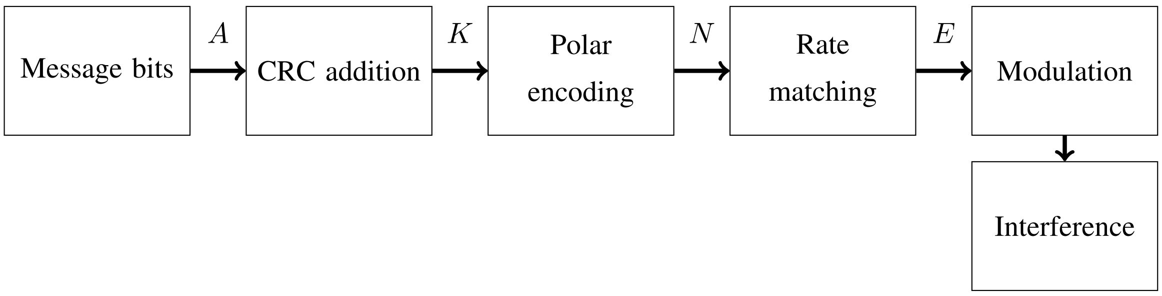

Figure 2.18. Chain of payload size change during processing

As stated previously, we have A message bits, with added CRC, it is K length polar code input (see Figure 2.18). The constraint of the polar code is that it procures N length output, which can be only the power of 2. This creates restrictions in a communication system. The main goal of the rate matching is to resize incoming N size block into the requested codeword of length E. The measure of this operation is called net rate $R = \frac{A}{E}$, where effective rate of the code is $R = \frac{K}{E}$.

To reduce the size, we can use 'puncturing' or 'shortening' techniques, they are used to disregard some of the frozen bits. The list of indices of the bits to disregard is called a 'matching pattern'. To create a matching pattern, ‘sub-block interleaving’ is applied. Sub-block interleaver divides frozen bits into 32 blocks of length $B = \frac{N}{32}$ [2]. The blocks are then shuffled, with regards to the sub-block interleaving pattern available in 5G standard [10]. Based on that the matching pattern is derived. The matching pattern in 5G specification is created based on the frozen set, rate and block sizes used [10]. With that, puncturing is used for lower rates and shortening for higher rates [2]. When puncturing is applied, the first $U = N-E$ bits from shuffled frozen set are not transmitted, whereas shortening disregards the last $U = N-E$ frozen bits from the sequence. Contrary to that, the enlargement of the input set is done with ‘repetition’, where the first $U = E - N$ bits of the encoded message are copied, appended at the end, and thus they are transmitted twice [2].

In some cases when passing further the rate matched output, ‘triangular bit interleaving’ is applied. This method was developed to improve higher order modulation coding, and is not always used. Let us imagine a triangular structure with rows and columns. We set its length to be T, where T is the smallest integer that passes condition (2.20) [2]. The incoming vector E can be described as $[e_0, e_1, e_2, ..., e_{E-1}]$, and used to fill the triangular structure, where each next row is one bit shorter, as seen in (2.19). To withdraw the interleaved output from the triangle structure, we read columns from the left, such as $[e_0, e_T, e_{2T-1}, ...]$. More in depth view can be found in source [2] and exact algorithms are provided in the 5G specification [10].

$$\begin{bmatrix} e_0 & e_1 & e_2 & \dots & e_{T-1} \\ e_T & \dots & e_{2T-2} & & \\ e_{2T-1} & \dots & e_{3T-3} & & \\ \vdots & & & & \emptyset \end{bmatrix} \tag{2.19}$$

$$\frac{T(T+1)}{2} \geq E \tag{2.20}$$

For rate recovery, we assume that K and N are known. If triangular interleaving is enabled, we deinterleave by writing into the triangle structure column-wise, and read row-wise. For puncturing, U amount of zero bits are put before the received message. When shortening is applied, U bits with value 1 are appended at the end. In the case of repetition, we simply crop the excess. The last step is to perform sub-block deinterleaving and pass the N amount of data to a polar decoder.

3. Implementation

The main focus of the implementation is to compare two decoder types in an environment based on 5G technical specification [10]. For that, CRC, interleaving, rate matching and different modulation schemes are taken into consideration. The created environment gives us the ability to estimate BER with different transmission parameters. We can check the error rate of the codes with different list length, block sizes, rates and different modulation. The scripts prepared for Matlab 2021 allow us to test the sending and receiving chains with different configurations under set noise variance of AWGN interference. This creates error rate characteristics graphs. The main concept is to compare the behaviour of polar coding technique based on SC list decoder provided in Matlab 5G Toolbox to a solution based on SSCL provided by source [18]. The prepared comparison scenarios are calculated and output characteristics are analysed. The calculation times are measured, which gives us an insight on the efficiency of the algorithm. The whole computation procedure is divided into separate script files, each having its own purpose. In Table 3.1 we can see the illustrative list of the prepared files. All of the prepared Matlab scripts can be found in `github.com\PLTomislaV\polar_code_thesis` and additionally used functions come from Matlab toolboxes which are listed below.

1. 5G Toolbox – is a main toolbox used. It provides functions for CRC addition, interleaving, rate matching, modulation and gives an implementation of a polar code with a decoder based on SCL.
2. Communications Toolbox – is needed for 5G Toolbox to work and it has an implementation of AWGN channel model.
3. Signal Processing Toolbox – is needed for 5G Toolbox to work.
4. DSP System Toolbox – is needed for 5G Toolbox to work.
5. Parallel Computing Toolbox – is used to accelerate the time of BER graph generation.

Table 3.1. Files list and their types

File name	File type	Main task
any `_testbench.m`	Run file	Set signal strength points, breaking conditions and the code settings for each point.
`transmit_chain.m`	Function file	Recreate the transmission chain by calling functions until break conditions are met.
`polar_encode_initialize.m`	Function file	Interleave input, decide the block size and prepare sequence with reference to rate matching.
`polar_encode_core.m`	Function file	Prepare proper size of kernel matrix and encode.
`polar_decode_initialize.m`	Function file	Decide the block size and prepare sequence with reference to rate matching.
`polar_decode_SC.m`	Function file	Decode using SSC.
`polar_decode_SCL.m`	Function file	Decode using SSCL.

3.1. Test bench

Any test bench file (`_testbench.m`) is used to set different settings of the calculations and to run the whole procedure. Each test bench was created to generate error rate characteristics graphs of different transmission settings on one figure. The settings we can set are: list length, UL or DL, encoder and decoder types, block size, rate matching and modulation. Additionally, times of calculation of different decoding types are measured.

For characteristic generation we must create a set of the test values of $\frac{E_b}{N_0}$ — the energy per bit to noise power spectral density represented in decibel. For each set value of $\frac{E_b}{N_0}$ we run `transmit_chain.m` function where some amount of message blocks are tested until breaking conditions are fulfilled. As the breaking conditions, we set the maximum amount of blocks to simulate per $\frac{E_b}{N_0}$ value and the maximum number of bit errors that can occur. If any of those two conditions arises, the measurement per $\frac{E_b}{N_0}$ value is done and the error rate is returned from `transmit_chain.m`.

The structure of calculating the characteristics is as follows. Using `for` function we call `transmit_chain.m` with fixed settings for each of the $\frac{E_b}{N_0}$ values. We can measure the time of calculation of each `transmit_chain.m` setting. This is done with `tic` and `toc` functions, representing the start and the end of the time measurement.

When calling `transmit_chain.m` function, besides $\frac{E_b}{N_0}$, maximum loops and maximum errors, we can specify: the message size A, rate matched output E, list size nL, encoder type, decoder type, uplink configuration, downlink configuration and modulation scheme. Details about `transmit_chain.m` are presented in Section 3.2 of this thesis.

Using `parfor` function from Parallel Computing Toolbox we can run each $\frac{E_b}{N_0}$ point calculation parallelly on different core of our central processing unit (CPU). This technique utilizes our hardware in a more efficient way, resulting in shorter time of characteristics generation while keeping the `transmit_chain.m` time measurement within the margin of error. From `transmit_chain.m` we obtain bit error rate (*BER*) and block error rate (*BLER*). For proper plotting of the characteristics we use base-10 logarithmic scale on the y-axis to represent the graph in uniform and perspicuous way.

3.2. Transmission chain

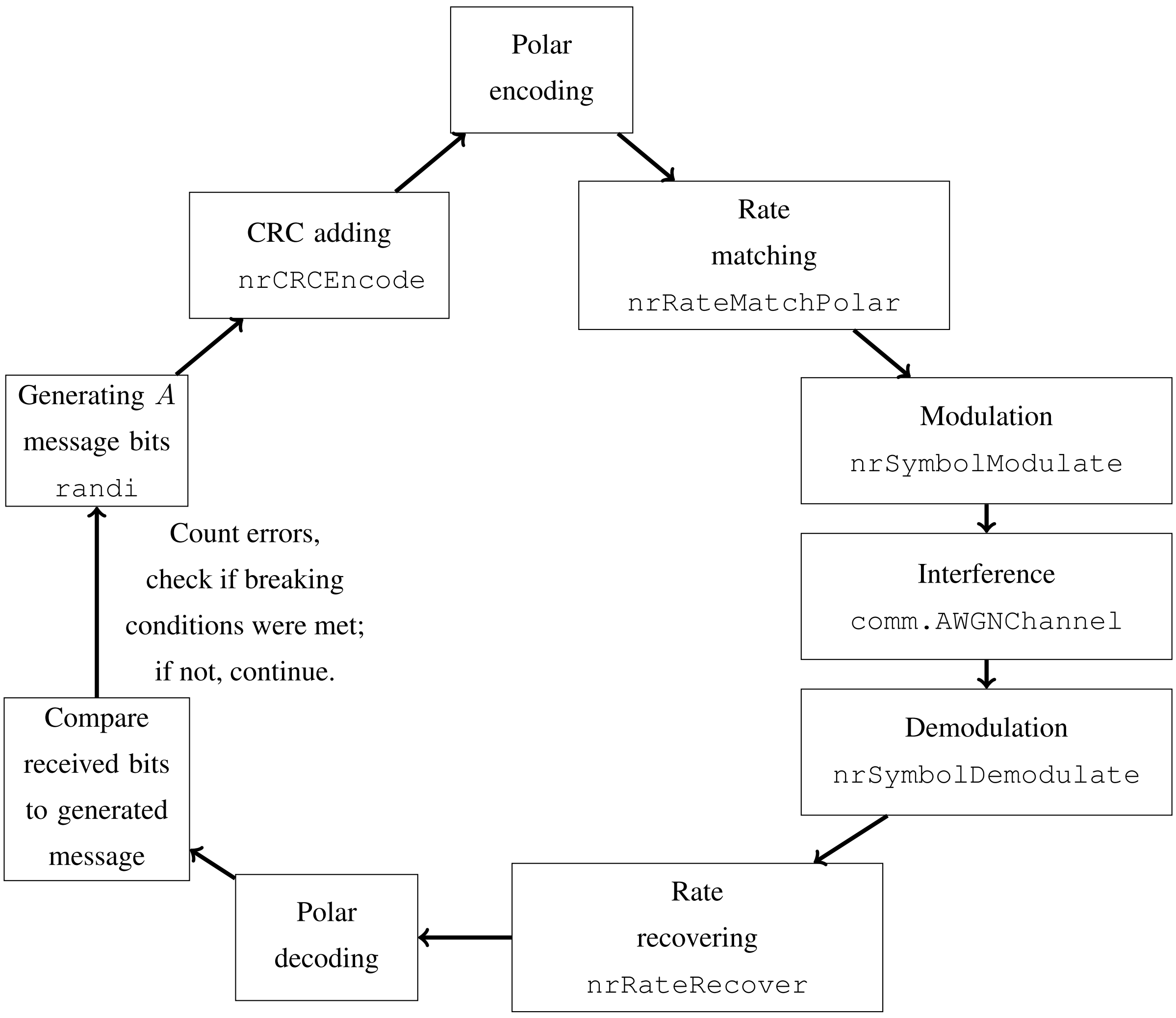
Polar encoding
Rate matching nrRateMatchPolar
Modulation nrSymbolModulate
Interference comm.AWGNChannel
Demodulation nrSymbolDemodulate
Rate recovering nrRateRecover
Polar decoding
Compare received bits to generated message
Count errors, check if breaking conditions were met; if not, continue.
Generating A message bits randi
CRC adding nrCRCEncode

Figure 3.1. Implemented transmission chain loop

The function `transmit_chain.m` processes blocks of message by properly calling functions of lower order (see Figures 3.1, 3.2, 3.3). This is similar to the concept presented in [7] and previously in Figure 2.1 in page 9 of this thesis. Here, Table 3.2 shows the inputs to the `transmit_chain.m`

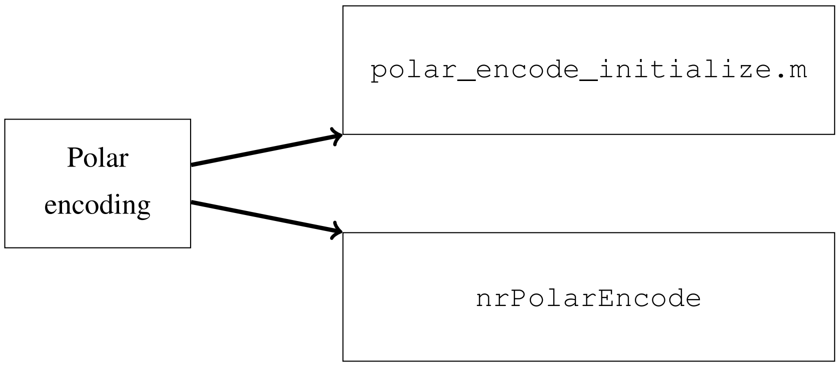

Figure 3.2. Different encoding functions

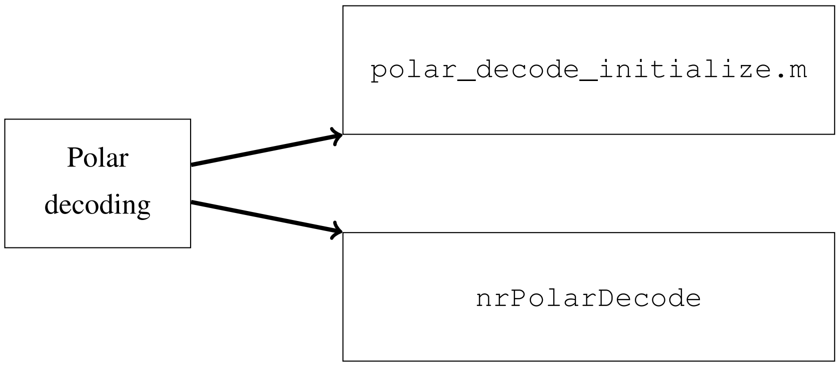

Figure 3.3. Different decoding functions

function in the proper order. The main task of `transmit_chain.m` is to: generate a block of A amount of message bits, process them through the sending chain, apply interferences, process through the receiving chain and compare with originally generated A input message. This procedure is repeated in a loop until the maximum amount of bit or block errors occurs. Then, BER and BLER are calculated.

Basic differentiation of the transmission is downlink or uplink. When variable `linkDir` is set to 'DL', the downlink settings are used, the uplink scenario is performed otherwise (see Table 3.3). Downlink is used for $164 \geq K \geq 36$ and uplink for $1023 \geq K \geq 30$, where $K = A + \texttt{crcLen}$ for (N, K) polar code. The options presented here alongside the contents of Table 3.3, comply with the 5G technical specification [10].

$$R = \frac{K}{E} \tag{3.1}$$

$$\frac{E_s}{N_0} = \frac{E_b}{N_0} + 10\log_{10}(\texttt{bps}) \tag{3.2}$$

$$\text{SNR} = \frac{E_b}{N_0} + 10\log_{10}(R) \tag{3.3}$$

$$\sigma^2 = \frac{1}{10^{\text{SNR}/10}} \tag{3.4}$$

As suggested by source [7], the noise variance σ^2 is calculated as presented in Equations (3.1), (3.2), (3.3), (3.4). The AWGN channel model named `chan` is set up using `comm.AWGNChannel` from

Table 3.2. `transmit_chain.m` input

Input name	**Action**	**Type**
`EbNo`	Set the bit energy to noise ratio in dB.	Double
A	Set the length of the message.	Integer
E	Set the length of the rate matched output.	Integer
`max_bit_errors`	Set the maximum amount of bit errors that can occur.	Integer
`max_blocks`	Set the maximum amount of block errors that can occur.	Integer
nL	Set the list size for list type decoder.	Integer
`encoder_type`	Choose the encoder type.	String
`decoder_type`	Choose the decoder type.	String
`linkDir`	Choose the link direction for other settings.	String
`modulation_type`	Choose the modulation scheme.	String

Table 3.3. Downlink and uplink settings

Setting name	**Setting description**	**Downlink**	**Uplink**
`crcLen`	Number of CRC bits.	24	11
`poly`	Polynomial name used for CRC calculation.	24C	11
`nMax`	Maximum value of n, where $N = 2^n$.	9	10
`iIL`	Interleave the input.	true	false
`iBIL`	Interleave coded bits.	false	true

Table 3.4. Modulation types and correlating amount of bits per symbol

Variable `modulation_type`	**Bits per symbol** `bps`
pi/2-BPSK	1
BPSK	1
QPSK	2
16QAM	4
64QAM	6
256QAM	8

Communications Toolbox, where the method of noise addition is set to be based on noise variance and σ^2 is passed as the argument.

We set up the main `while` loop where for each iteration the first action is to generate pseudorandomly A amount of bits representing our message. For this, function `randi` is used. Just after message generation, attachment of CRC bits is performed by function `nrCRCEncode` from 5G Toolbox.

At the input of the encoder `iIL` is checked if the input is interleaved. If the `encoder_type` variable is equal to 'polar_encode' as encoder, we run the function `polar_encode_initialize.m`, otherwise, the encoder from 5G Toolbox (`nrPolarEncode`) is used. Both encoder types have inputs as follows: vector of K amount of bits, E, `nMax` and `iIL`.

Rate matching is done by `nrRateMatchPolar` from 5G Toolbox, where by `iBIL` decision, coded bits may be interleaved. Further modulation is performed with `nrSymbolModulate` function from 5G Toolbox and modulated symbols are passed through previously set up `chan` AWGN channel from Communications Toolbox. Soft demodulation is done by `nrSymbolDemodulate` and bitwise approximation to the log-likelihood ratios of the demodulated bits are passed to `nrRateRecoverPolar` to properly resize.

For decoding, `decoder_type`='polar_encode' triggers `polar_encode_initialize.m` where different type decoders can execute. If other `decoder_type` is specified than SC or SCL, `nrPolarDecode` from 5G Toolbox is called.

When recovered message is received, A bits are compared with message that was generated at the beginning of the loop. Bit errors are counted as `bit_errors`, number of `Loops` increases by one each loop and if any bit error occurred, block error is counted as `block_errors`. When maximum number of bit errors or block errors is achieved, the `while` loop breaks and BER and BLER are calculated as shown in Equations (3.5), (3.6).

$$\text{BER} = \texttt{bit\_errors}/(A\,\texttt{Loops}) \tag{3.5}$$

$$\text{BLER} = \texttt{block\_errors/Loops} \tag{3.6}$$

3.3. Encoder

When an encoder is called, the first action is based on the previously chosen `iIL` value. For downlink scenario it is set to true, which implies we interleave the input by set pattern. Further, a technique provided in the 5G specification [10] lets us pick the N size of the polar encoder output. The minimum N is equal $N = 2^5$, but N is mainly picked with relation to rate and rate matched output size E. Further, the frozen set `F` is derived from the original sequence provided by [10]. Since the rate matching applies, the derivation procedure and the sequence are strongly related to each other. The just explained procedures are implemented in `polar_encode_initialize.m` exactly the same way as in `nrPolarDecode`. This complies with 5G specification [10], but the prepared script does not include procedures for parity-check-concentrated coding for short block length.

For encoding, the kernel G_2 is properly expanded with Kronecker power. Further, u vector of N size is created and filled with message and frozen bits. The encoded codeword is obtained by multiplication of prepared vector u and properly expanded kernel.

Script `polar_encode_initialize.m` does not encode the data; it calls `polar_encode_core.m` where the encoding happens. The main difference between `polar_encode_core.m` and `nrPolarDecode` is how the generator expansion is performed. The encoding itself is the same, but when expanding by Kronecker power, the prepared script is changing the size of the generator matrix with each step of the expansion (each next power). On the contrary, `nrPolarDecode` works in more optimal way, that means it creates the necessary matrices first and then performs the calculations on them.

3.4. Rate matching

The encoded codeword is passed to `nrRateMatchPolar`, where adequate bits are disregarded or repetition is performed. The bits chosen for puncturing and shortening correlate with the frozen set `F` created at the encoder. This ensures that no information bit is disregarded when rate matching is applied. Further, based on `iBIL`, triangular output interleaving may be applied.

For recovery, `nrRateRecoverPolar` is used to adapt the size of incoming payload for decoder processing. If the codeword triangular interleaving was applied, the first step for rate recovery is to deinterlive by the triangular scheme. For repetition, we crop off the excess, puncturing sets LLR values to 0. Shortening applies value representing bit 1, such as 1e20. At last, we apply sub-block deinterleaver and pass our rate matched payload further.

3.5. Decoder

For decoding, there are three main types available and are shown in Table 3.5. The main goal of `polar_decode_initialize.m` is to derive the frozen set adequately to code rate and and call `polar_decode_SC.m` or `polar_decode_SCL.m`, where the decoding is performed.

Table 3.5. Decoding functions

Variable `decoder_type`	**Function called**	**Decoding performed by**
SC	`polar_decode_initialize(`SC`)`	`polar_decode_SC.m`
SCL	`polar_decode_initialize(`SCL`)`	`polar_decode_SCL.m`
	`nrPolarDecode`	`nrPolarDecode`

Both SC and SCL solutions are decoder implementations given by [18] where they should be refereed to as SSC and SSCL because decoding is done by loops recreating the tree traversal described in Section

2.5.3 of this thesis. The 5G Toolbox function `nrPolarDecode` is based on the algorithms proposed by [17] and originates from original SC decoding [1].

When calling any solution from [18], we must adapt the frozen set generated by 5G procedures. The set `F` is containing zeros or ones in proper indices representing the frozen positions and information positions. This vector is rewritten as `F2` to contain the frozen positions indices. Additionally incoming codeword input is transposed to be properly utilized.

Every decoder, based on `iIL` may interleave the output by a map. For list type decoder it is done for each of proposed paths. In the script `polar_decode_SCL.m` we can find a procedure which picks correct message output based on CRC or PM. This is done the same way as in `nrPolarDecode` using `nrCRCDecode` for CRC check.

3.6. Results

There are a few of the `_testbench.m` type files, each generating different comparison. In particular focus of these scenarios was to generate BER characteristics using different settings for: list length, rate, link direction, modulation, and block size. Moreover, time measurements were taken to compare error correction capabilities of each code with time needed for computation.

The breaking conditions were mainly picked to showcase the bit error rate up to around the BER of value 10^{-4}, as proposed by source [2]. If the plot on a figure cutts off — it means that BER fell down to 0 on the next step. This happens because the condition of maximum loop is fulfilled. For this thesis, such an approximation is sufficient, as it showcases key characterises.

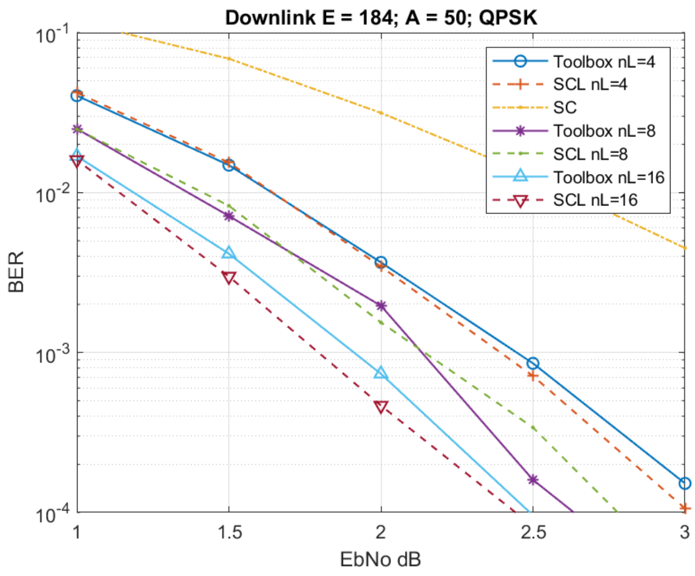

Figure 3.4. Bit error rate characteristics for different list lengths

Three most common list sizes were picked to showcase the advantage a polar code gains when the list size increases. In Figures 3.4, 3.5 we can see the comparison of different list sized codes, where SC decoder is also plotted to represent the list size of 1. The differences between the same list length

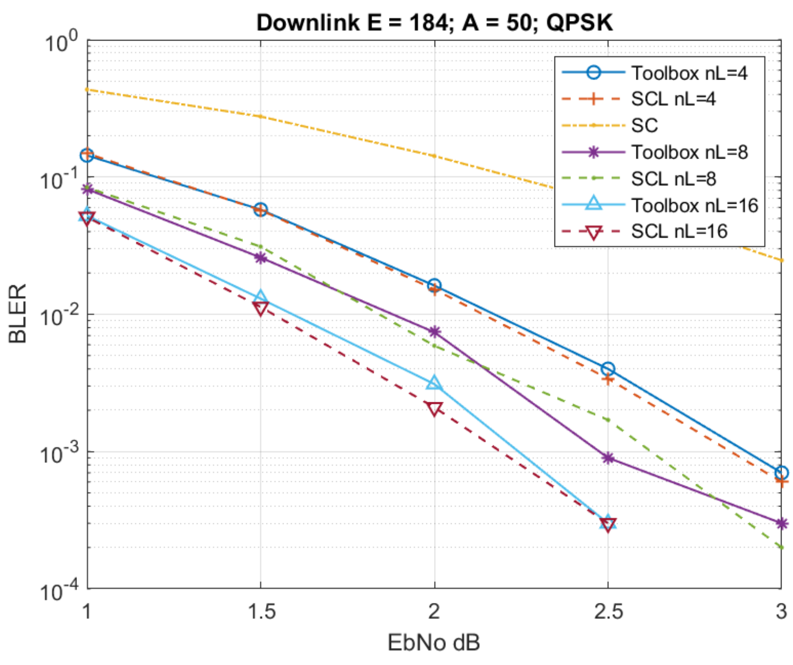

Figure 3.5. Block error rate characteristics for different list lengths

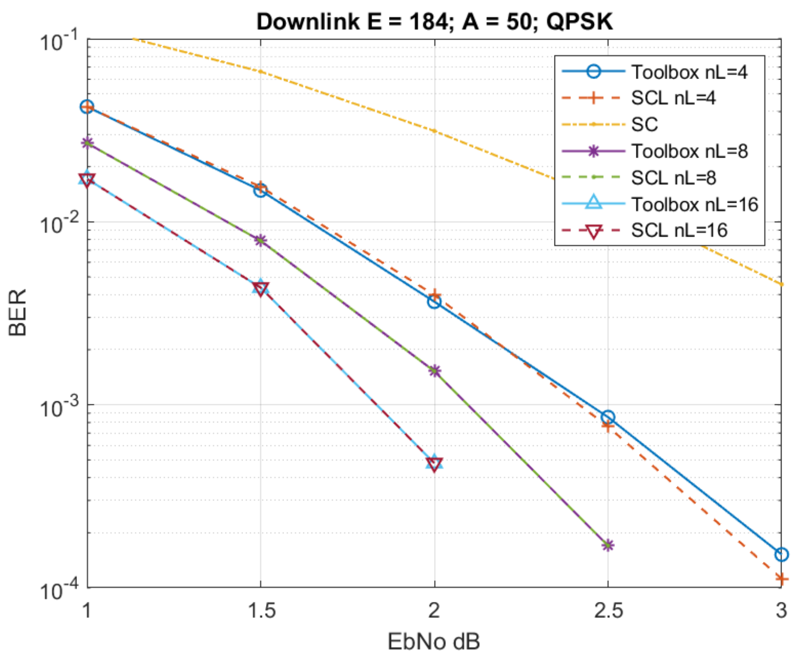

Figure 3.6. Bit error rate characteristics for different list lengths with repeatability

codes are negligible, as there is randomness induced by pseudorandom generator. Figure 3.6 is the only one in this thesis, that showcases the error rate of the code when tested with repeatability. Repeatability means, that for each point we reset the seed of pseudorandom generator, thanks to that, we obtain results with much lower margin of error. Further, this showcases that the Toolbox decoder and the simplified decoder have the same error correction capabilities. The computations for the length comparison were performed until 10^4 block errors or 10^6 bit errors occurred. The figures show that between the function `nrPolarDecode` and the simplified successive cancellation list decoder (here denoted as SCL), there is negligible difference in error correction performance, which is also seen on next examples. We can observe that increasing the list size improves the error correction of the code. This corresponds to results shown by source [17]. Longer list sizes need much more computational power, that is why, even though they bring additional error correction advantage, they are not commonly used.

Table 3.6. Times per each decoder configuration

Variable `decoder_type`	**Time** $[s]$
Toolbox	350
SCL	164

Sample time measurements are noted in Table 3.6, they show that the simplified type algorithm [18] (SCL) is superior to the SC based list decoding [17]. Simplified decoder uses less complex equations which are repeated in a recursive manner. This is more optimal than computing complicated dependencies for each bit, as is implemented in the 5G Toolbox [17]. Using simplified decoder reduces the delay induced by telecommunication system. This is not only desirable, but also allows the system to handle more users, as each calculation is performed faster.

The rate matching tests results are visible in Figure 3.7 and were calculated with constrains of 10^6 block errors or 10^6 bit errors. The values of rate and block sizes were picked to show different techniques of rate matching for as similar options as it was possible. Puncturing is used for higher block sizes, thus it a coding advantage is observed. In the figure it is clearly visible that rate matching has little to no influence on polar code error rate. This was expected as we manipulate single bits, which have little influence on the code performance. Thanks to that we do not gain much error rate when matching the system output to the modulation scheme.

As mentioned previously, uplink and downlink configurations use different type of the CRC generator polynomial and interleaving schemes. This difference is clearly visible in Figure 3.8, which were generated with maximum rate equal to 10^5 or maximum bit errors equal to 10^7. We can see that error rate for downlink configuration is higher than uplink. For downlink CRC takes 24 bits, where uplink configuration uses 11 bits for CRC. Because the different CRC polynomials are used, the effective rate $R = \frac{K}{N}$ differs for uplink and downlink. This influences the energy per bit which is the x-axis on the figure. Rate matching into the same output uses more of the frozen bits, as bits for the CRC. This is done to improve the uplink transmission.

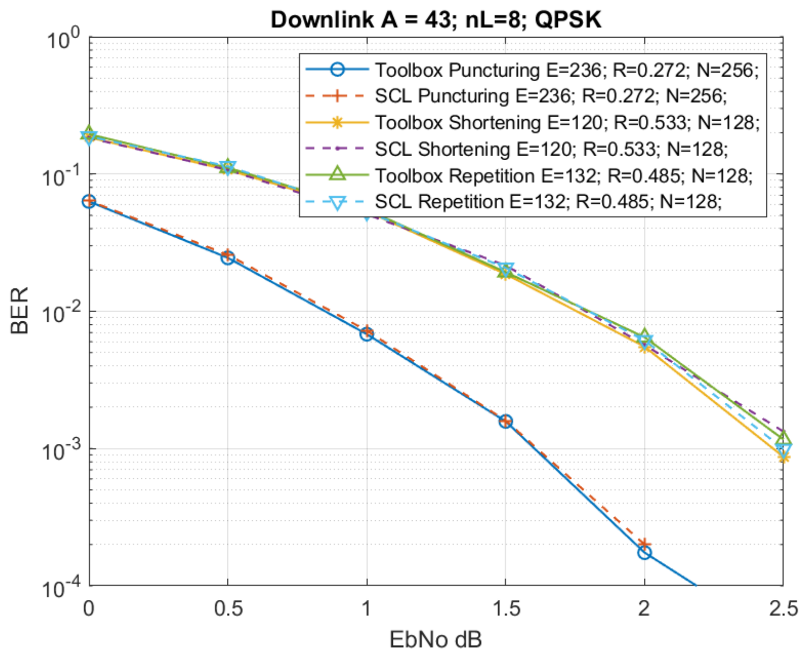

Figure 3.7. Bit error rate characteristics for different rates

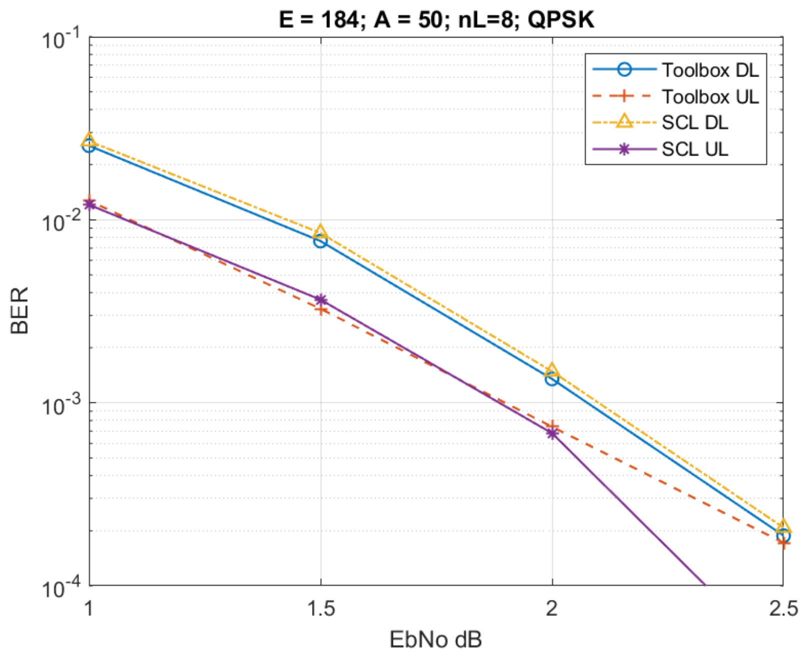

Figure 3.8. Bit error rate characteristics for different link direction

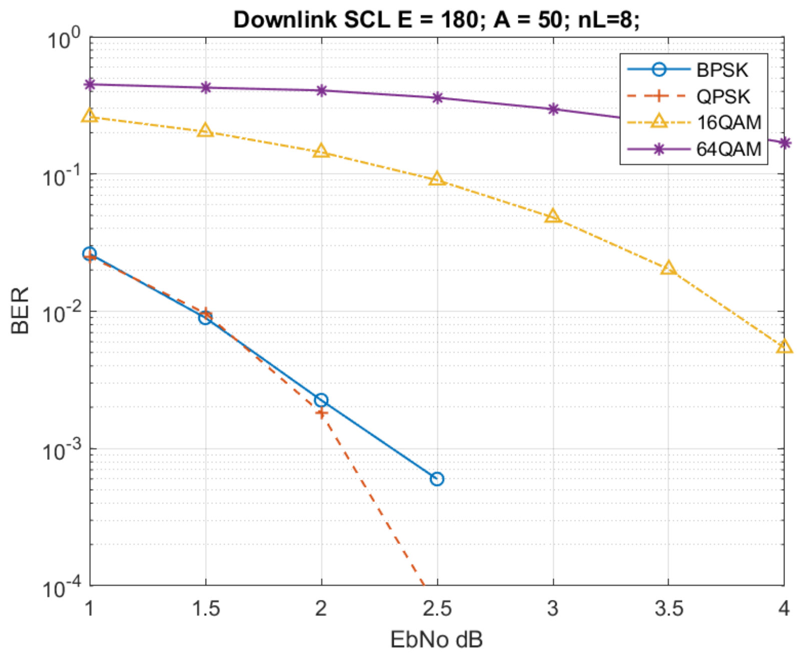

Figure 3.9. Bit error rate characteristics for different modulation schemes

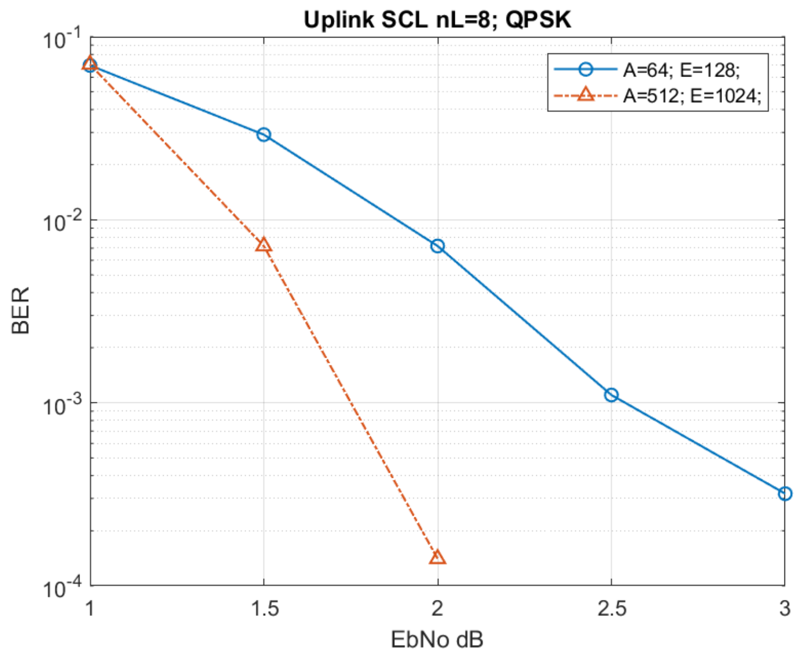

Figure 3.10. Bit error rate characteristics for different block sizes

Further, in Figure 3.9, we can see the code performance using different modulation schemes. We can clearly observe that higher order modulation types are more prone to interference. This does not exclude them from the common usage. Higher order modulation are mainly used to enhance the throughput when the signal is strong.

As last, basic block size difference is shown in Figure 3.10. The figure shows that increasing the block size improves the error correction capabilities of the code. This is apparent because the bigger block gives us more frozen positions, and the polarisation is stronger for higher indices, which further improves the error correction performance.

4. Summary and future work

The main goal of this thesis was to get acquainted with a polar code and its implementation. For that, necessary information was gathered and presented as a theoretical background. Moreover, Matlab scripts were prepared to assess the polar code error correction capabilities using the configuration proposed in the 5G specification. Essential elements of a telecommunication link, such as: rate matching, modulation and interference were taken into account. For implementation of those elements, functions from Matlab 5G Toolbox were used. In the toolbox, full implementation of the SCL polar decoder was found. The research done implied that there exists a more computationally efficient way of decoding, that is SSCL. The implementation of the SSCL decoder provided by source [18] was adapted to work with functions available in the toolbox. This gave us the ability to generate BER characteristics of the toolbox and the simplified decoders, within a very similar environment. The calculated output characteristics indicate that the error correction capabilities of both types of the decoders are very similar and differences can be assumed to be within the margin of error. The calculation times were measured and showed that simplified decoding can be more than twice faster. Moreover, the decoders were tested under different configurations, this showcased important characteristics. The results indicate that increasing the list length or the block size improves the error correction capabilities of the code. Furthermore it was shown that rate matching has low impact on the code performance and that modulation schemes of higher order require strong signal to transmit the data without corrupting it.

Polar code performance greatly improves when aided with other codes, this is why, much research is put into its development. List decoding with CRC was shown to gain a big coding advantage, where additional parity checks are known to improve the error correction capabilities even more. This would be a great branch to develop the topic further, as there exist many ways to implement parity checks into a polar code. The new 5G technology is planned to be mainly used in urban areas, where there are a lot of buildings. The main type of interference in urban areas comes from the constructive and destructive interferences of the signal that takes multiple paths to arrive at the receiver. This is why it would be preferable to check the polar code behaviour under a fading channel interference model. The 5th generation of mobile broadband was designed taking into consideration the concept of the internet of things (*IoT*), where many low power devices are connected to the network. Low complexity of a polar code gives us the ability to easily implement it in hardware. This type of implementation allows us to strongly optimize latency and power consumption, which are very important aspects of IoT, 5G and upcoming 6G mobile broadband technologies.

Bibliography

[1] Erdal Arikan. “Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels”. In: *IEEE Transactions on Information Theory* 55.7 (2009), pp. 3051–3073. DOI: *10.1109/TIT.2009.2021379*.

[2] Valerio Bioglio, Carlo Condo, and Ingmar Land. “Design of Polar Codes in 5G New Radio”. In: *IEEE Communications Surveys Tutorials* 23.1 (2021), pp. 29–40. DOI: *10.1109/COMST.2020.2967127*.

[3] Xinwei Cai et al. “Hybrid CRC and Parity-Check-Concatenated Polar Codes with Shared Encoder”. In: *2019 Computing, Communications and IoT Applications (ComComAp)*. 2019, pp. 288–292. DOI: *10.1109/ComComAp46287.2019.9018717*.

[4] Orhan Gazi. *Polar Codes A Non-Trivial Approach to Channel Coding*. Vol. 15. Springer, 2019.

[5] Jian Wang Shengchen Dai Gongzheng Zhang Ying Chen Hejia Luo Jun Wang Huazi Zhang Rong Li. “Parity-Check Polar Coding for 5G and Beyond”. In: 2018. DOI: *10.1109/ICC.2018.8422462*.

[6] Wanzhen Li and Zhongqiu He. “An Efficient CRC-Aided Parity-Check Concatenated Polar Coding”. In: *2021 IEEE Asia Conference on Information Engineering (ACIE)*. 2021, pp. 1–5. DOI: *10.1109/ACIE51979.2021.9381073*.

[7] *MathWorks Help Center 5G New Radio Polar Coding*. `https://www.mathworks.com/help/5g/gs/polar-coding.html` The MathWorks, Natick, MA, USA. 2021.

[8] Todd K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. John Wiley & Sons, 2005.

[9] Kai Niu and Kai Chen. “CRC-Aided Decoding of Polar Codes”. In: *IEEE Communications Letters* 16.10 (2012), pp. 1668–1671. DOI: *10.1109/LCOMM.2012.090312.121501*.

[10] *NR Multiplexing and channel coding*. TS 38.212. Release 15.2.0. 3GPP 3rd Generation Partnership Project. July 2018.

[11] *NR Physical channels and modulation*. TS 38.211. Release 15.2.0. 3GPP 3rd Generation Partnership Project. July 2018.

[12] Warren J. Gross Pascal Giard Claude Thibeault. *High-Speed Decoders for Polar Codes*. Springer, 2017.

[13] John G. Proakis. *Digital Communications.* McGraw-Hill Education; 4th edition, 2000.

[14] Tobias Rosenqvist and Joël Sloof. “Implementation and evaluation of Polar Codes in 5G”. Karlstad University, June 2019.

[15] C. E. Shannon. “A Mathematical Theory of Communication”. In: *The Bell System Technical Journal* 27 (1948), pp. 379–423.

[16] C.E. Shannon. “Communication in the Presence of Noise”. In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21. DOI: *10.1109/JRPROC.1949.232969*.

[17] Ido Tal and Alexander Vardy. “List Decoding of Polar Codes”. In: *IEEE Transactions on Information Theory* 61.5 (2015), pp. 2213–2226. DOI: *10.1109/TIT.2015.2410251*.

[18] Andrew Thangaraj. *LDPC and Polar Codes in 5G Standard.* `https://nptel.ac.in/courses/117/106/108106137/`. The National Programme on Technology Enhanced Learning (NPTEL). 2018.

[19] Tao Wang, Daiming Qu, and Tao Jiang. “Parity-Check-Concatenated Polar Codes”. In: *IEEE Communications Letters* 20.12 (2016), pp. 2342–2345. DOI: *10.1109/LCOMM.2016.2607169*.