UNIVERSITY OF BERGEN
DEPARTMENT OF INFORMATICS

Polar Codes and LDPC Codes in 5G New Radio

Author: Kristian Wøhlk Jensen
Supervisor: Chunlei Li

June 30, 2022

Abstract

The channel coding theorem by Shannon affirms the existence of digital communication systems that can achieve error-free communication over a noisy channel. However, channel coding increases the reliability of communication by adding structured redundancy and introducing extra computational costs. The prime objectives for the 5th generation mobile network (5G) are increased reliability, less redundancy, and lower latency. LDPC codes and polar codes are two promising communication systems for meeting this objective. This thesis is a survey on the encoding/decoding process and the reliability of communication in the communication system utilized in the 5G developed by 3rd Generation Partnership Project (3GPP). We will evaluate the encoding/decoding process over three of the most studied communication channels, namely: the Binary Symmetric Channel (BSC), the Binary Erasure Channel (BEC), and the Additive White Gaussian Noise (AWGN) channel.

Acknowledgements

Firstly I would like to thank my supervisor Chunlei Li. He offered great advice and encouragement with a perfect blend of insight and humor, and is what made this work possible. I would like to thank Andrea Tenti and Nicolay Stoyanov Kaleyski who both taught courses during my master's programme, and this thesis builds on work done in those courses. They taught their courses in an intelligible way, and were always open for questions, and I would like their work not to go unnoticed. I would also like to thank my fellow students who have assisted me with both motivational words, and many laughs.
Finally, I would like to express my gratitude to my parents for their unconditional trust, timely encouragement, and endless patience.

Contents

List of Abbreviation

3GPP 3rd Generation Partnership Project. i, v, 1, 45, 49–51

5G 5th generation mobile network. i, v, ix, 1, 2, 24, 37, 40, 41, 43, 56, 59, 63

AMS Adapted Min-Sum. v, x, 57, 58, 73, 77–80

AWGN Additive White Gaussian Noise. i, v, viii, x, 12, 13, 18, 21, 60, 61, 68, 70, 73, 78

B DMC Binary Discrete Memoryless Channel. v, 10, 12, 14, 15

BEC Binary Erasure Channel. i, v, viii, x, 12, 15–18, 60, 67, 69, 72, 76, 77, 80

BER Bit Error Rate. v, 59, 60, 64, 65, 73, 76

BG Base Graph. iv, v, 48–52, 54

BLER Block Error Rate. v, 59, 60, 64, 65, 68, 69, 73, 76

BPSK Binary-Phase Key Shifting. v, 12, 13

BSC Binary Symmetric Channel. i, v, viii–x, 12, 14, 15, 17, 18, 20, 60, 66, 71, 72, 79

CA SCL Cyclic Redundancy Check Aided Successive Cancellation List. v, ix, x, 46, 64–69, 81, 82

CK SCL Check and Keep SCL. v

CN Check Node. v, 35, 37, 38, 58

CR SCL Check and Remove SCL. v, 46, 65, 68, 82

CRC Cyclic-Redundancy Check. v, 23, 24, 40–43, 46, 47, 49, 53, 59, 62, 64, 68, 73, 77, 81

GF(2) Galios Field of size two. v, 4, 6, 10, 12

IOMS Improved Offset Min-Sum. v, x, 57, 58, 73, 77–81

LDPC Low-Density Parity-Check. v, 1, 2, 26, 34–37, 48, 51, 53, 56, 59, 63, 77, 81

LLR Log-Likelihood Ratio. v, 18, 19, 38, 56, 73

ML Maximum Likelihood. v, 19

OMS Offset Min-Sum. v

PCM Parity-Check Matrix. v, 22, 23, 34–37, 47, 48, 52, 56, 58

QC LDPC Quasi-Cyclic LDPC. v, 48, 49, 58, 63, 81

SC Successive Cancellation. v, ix, x, 32, 40, 45, 64–69, 81

SCL Successive Cancellation List. v, ix, x, 41, 45, 64–69, 73, 81

SNR Signal-To-Noise Ratio. v, 58–61, 72, 73

VN Variable Node. v, 35, 37–39, 56–58

List of Figures

List of Tables

Chapter 1

Introduction

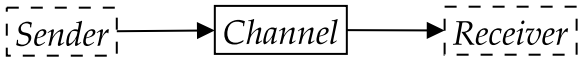

Figure 1.1: Digital communication is the act of transmitting packets from sender to receiver over a channel. Sender adds structured information to the transmitted information so that the receiver can better correct faults induced by the channel to the received packet.

The 5G structure of LDPC codes and polar codes specified in the standard developed by 3GPP specifies the intermediate steps the sender must follow to achieve communication, called encoding. While the intermediate steps the receiver must follow, called decoding, are not mentioned in the standard.

The encoding detailed in the standard assumes much background knowledge from the reader. At the same time, the decoding is much left to the individual developer to research in order to piece together a coherent decoder. Developing a well-functioning decoder is not easy, as many individual studies propose different solutions to achieve both efficient and reliable communication.

This thesis will research proposed decoding algorithms and develop a coherent communication system to profile their contribution to reliable communication with the transmission of smaller packets. The size of the packet to be transmitted can, in some cases, be too large for the encoder/decoder to handle as one unit. The packet is then segmented into a collection of smaller-sized packets. Ideally, our project should include the segmentation of incoming packets. However, this thesis will focus on the performance of LDPC codes and polar codes in 5G of input packet sizes appropriate for the encoder/decoder.

Chapter 2

Information Theory

Information theory is the theoretical study of achieving an arbitrarily small probability of wrong estimation of the sent information sequence by only reading the received information sequence. In digital communication, an arbitrarily small chance of incorrectly determining the transmitted information is referred to as error-free communication.

If we were to use a perfect channel, the sender would be guaranteed that the content arriving at the receiver is identical to the sent information. However, in practice, there is no such thing as a perfect channel; non-perfect channels are referred to as noisy channels. Noise over a channel is defined by the probability of each symbol in the information sequence arriving as a value different from its corresponding value at the receiver. All channels have varying amounts of noise; the noise over each channel is never assumed to be constant. If a received symbol is not identical to the transmitted symbol, it is said to be erroneous. A fundamental concept of information theory is that information is the resolution of uncertainty regarding an outcome.

In digital communication, the information sequence requested by the receiver from the sender is a given sequence of binary numbers a of length A. The elements of a are one of two values and are drawn from the Galios Field of size two (GF(2)), which is the set containing the elements 0 and 1. The information sequence a is given as $a = \{a_0, a_1, \ldots, a_{A-1}\}$, where $A = |a|$, $a_i \in GF(2)$, and $GF(2) = \{0, 1\}$

By introducing an encoder, error-free communication can be achieved by appending parity bits to a. The purpose of the encoder is to add the parity bits as structured additional information to a through intermediate steps, providing error detection and error correction capabilities at the receiver. The collection of these intermediate steps at the sender is called encoding. Through these intermediate steps, the codeword is assigned a different variable notation and a corresponding variable for the codeword length to easier distinguish each step in the encoding process. The final codeword to be transmitted is always denoted as x and is of length E.

The transmission rate R is given as a ratio of information bits A to the total number of transmitted bits E, on the form $R = \frac{A}{E}$. We want the information rate to be as close to 1 as possible, but the noise property of the channel forces the rate to be lower. The higher the noise, the lower the rate.

Information theory heavily builds on the work of Claude Shannon, *A Mathematical Theory of Communication* [14], which laid the groundwork for the mathematical explanation and proof of error-free communication while achieving a non-zero rate.

2.1 Shannon's Channel Coding Theorem

The work of [14], showed and proved how to calculate the maximum rate of a channel given its noise level while achieving error-free communica-

tion. This maximum rate is called the channel capacity C, often referred to as the Shannon limit, and is defined as:

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

Where X is the probability distribution of an input ensemble, and Y is the probability distribution of an output ensemble.
The following definitions explain the Shannon limit.

Entropy
The Shannon information content of an outcome x in X is defined as:

$$h(x) = -log_2 p(x)$$

The entropy of a single outcome $h(x)$ is the amount of uncertainty resolved by learning the outcome of an event x, occurring with probability $p(x)$. The entropy of an ensemble X is the average information content of all single outcomes $x \in X$ expressed as:

$$H(X) = \sum_{i=1}^{n} p(x)\, h(x)$$

Note that the entropy $H(X)$ depends upon the probabilities of an outcome x and not the specific values of x.

Conditional Entropy
Conditional entropy $H(X \mid Y)$ is the measure of information needed to fully determine the outcome of a probability ensemble X when observing the outcome of a probability ensemble Y. This gives us the equation:

$$H(X \mid Y) = -\sum_{x \in \mathcal{X}} H(Y \mid x)$$

Joint Entropy
Suppose we map a value x of the ensemble X to a value y of an ensemble

Y, and we want to measure the dependencies between ensembles X and Y. The uncertainty of a joint variable ensemble $H(X, Y)$ is the uncertainty in the ensemble Y plus the uncertainty remaining in the ensemble X when Y is observed:

$$H(X,Y) = H(X) + H(Y \mid X) \overset{\text{symmetry}}{=} H(Y) + H(X \mid Y)$$

Mutual Information
The mutual information $I(X; Y)$ of two ensembles X and Y is the difference between the average uncertainty in X and the uncertainty remaining in X after observing Y. The mutual information $I(X; Y)$ is defined by:

$$I(X;Y) = H(X) - H(X \mid Y) = H(X) - H(Y) + H(X,Y) \qquad (2.1)$$

To maximize equation (2.1), we need to maximize $H(X)$ and minimize $H(X \mid Y)$. $H(X)$ is maximum when the probability ensemble X is uniformly distributed. When the probability ensemble X is uniformly distributed and $x \in$ GF(2), the entropy $H(X) = 1$. $H(Y)$ is 0 as there is no uncertainty of the outcome of the received values at the receiver. $H(X, Y)$ depends on the channel utilized and will be further discussed as their respective channels are explained in chapter 3.2.

Channel capacity is the difference between the information gained from learning X and the uncertainty remaining about X once its joint variable Y is observed.

Shannon's limit [10] states the following.

1. The channel used for digital communication has the property that for any chance of bit error $p_b \geq 0$, and rate $R \leq C$, then for a large enough codeword length N there exists a code of length N and rate $\geq R$ accompanied with a decoding algorithm such that the maximum probability of decoding to an erroneous codeword is less than p_b.

2. If a probability of bit error p_b is acceptable, rates up to R are achievable, where
$$R = 1 - C$$

3. For any probability of bit error p_b rates greater than R are not achievable.

The channel capacity is the maximum rate R which gives us the minimum amount of parity bits needed to ensure an arbitrarily low probability of wrong recovery of the information bits a, given the received codeword y.

Chapter 3

Channel Coding

Channel coding is the field of applying information theory to an encoder/decoder system to achieve communication with an acceptable rate of erroneous bits while maintaining low complexity of the encoder/decoder. The primary function of channel coding is to give the receiver the extra information and the necessary tools to detect and correct the erroneous symbols induced by the transmission channel.

Many channel coding systems have been proposed since Shannon's channel coding theorem. As the channel coding systems became more advanced, information theory became more integrated into the implementation of channel coding systems instead of being used as a tool to evaluate its performance. Figure 3.1 shows the encoder and decoder's role in a communication model to achieve reliable communication better.

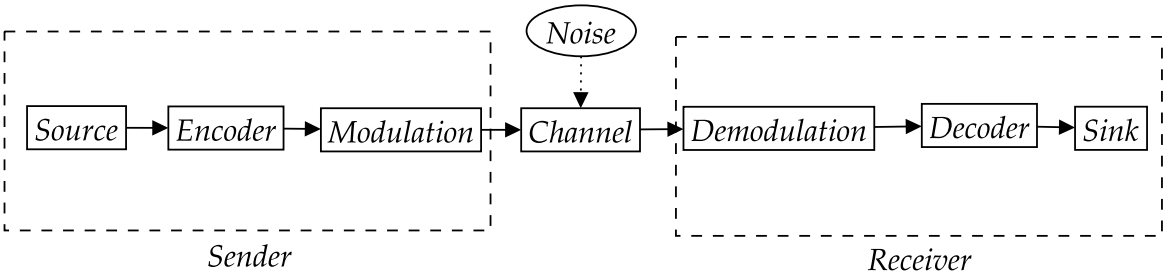

Figure 3.1: The intermediate steps in the coding system for communication

- The *source* is the resting place of information in the form of binary digits, which is requested by the sink.
- The *encoder* segments the information into chunks if necessary and adds redundancy to each chunk to form a codeword.
- The *modulation* converts each bit to a signal appropriate to the channel.
- The *channel* is the medium by which the codeword is transmitted.
- The *noise* is an uncontrollable factor that has the property of altering the symbols of a codeword transmitted over the channel.
- The *demodulation* determines the reliability of each incoming bit. Higher levels of noise output less reliable bits at the receiver.
- The *decoder* uses the redundant information introduced by the encoder to detect and correct erroneous bits.
- The *sink* is the final destination of the data requested from the source.

Communication systems are run for different rates over varying noise levels to better profile the error detection and correction performance. This is done as the noise is difficult to accurately estimate in practice [10].

3.1 Binary Discrete Memoryless Channels

This chapter will discuss the three communication channels utilized in this thesis. In doing so, we will first review the shared characteristics of the channels, and later discuss how they differ. All three channels are said to be Binary Discrete Memoryless Channel (B DMC). The definition of a B DMC is:

- *Binary:* The values of the codeword x_i transmitted over the channel can take one of two values, on the form $x_i \in \text{GF}(2)$.
- *Discrete:* Communication is discrete when the transmitted codeword of length E uses a finite number of transmissions to send its codeword. For a codeword of length E, the channel is used E times.
- *Memoryless:* A channel is memoryless in that learning the value of one incoming symbol y_i gives us no information about previous incoming symbols y_{i-1} or following incoming symbols y_{i+1}. Therefore the value of the corresponding noise symbol n_i gives no information about neither the next incoming symbol y_{i+1} nor the next noise symbol n_{i+1}. As each symbol in both x_i and n_i are statistically independent, it is sufficient to model each channel on the transmission of a single symbol x_i.

Noise

The noise is a random quantity with the expected value of 0, where each drawn quantity is independent of one another, and are added to each transmitted symbol. By the central limit theorem, a significant sample size of many similar and independent random effects added together produces a normal distribution. It is therefore normal to model the noise as variables randomly drawn from the normal distribution $N_0(0, \sigma^2)$, also called

Gaussian distribution, with the expected value of the noise being 0, and the variance σ^2 is the predicted channel noise . The received symbol y_i is the channel noise n_i added to the transmitted codeword symbol x_i on the form $y_i = x_i + n_i$.

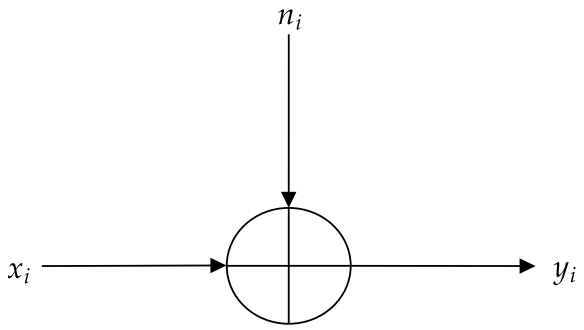

Figure 3.2: The received symbol y_i is the noise n_i added to the sent symbol x_i by addition $\oplus$.

The difference between the three channels is the mapping $n_i : x_i \rightarrow y_i$, and how much uncertainty of x_i can be resolved by y_i. The correlation between the sent symbol, noise, and the received signal can be modeled by a transition matrix T.

Let SX and SY be the set from which the sent symbol x_i, and the received symbol y_i can take their values, respectively. A transition matrix T shows the mapping of the channel input variable x_i to the channel output variable y_i.

$$T = \begin{bmatrix} p(x_i = SX_0 \,|\, y_i = SY_0) & \dots & p(x_i = SX_{|SX|-1} \,|\, y_i = SY_0) \\ \vdots & \vdots & \vdots \\ p(x_i = SX_0 \,|\, y_i = SY_{|SY|-1}) & \dots & p(x_i = SX_{|SX|-1} \,|\, y_i = SY_{|SY|-1}) \end{bmatrix} \tag{3.1}$$

Each column is assigned to a specific input symbol x_i, and each row is assigned to a particular output symbol y_i, where each element in the row is the probability of the symbol x_i being sent, given that we observe y_i.

A channel is said to be symmetric if both the rows are permutations of one another, and the columns are permutations of one another. A channel is said to be weakly symmetric if it is symmetric, and the rows of the transition matrix add up to the same value [10]. For a weakly symmetric channel, the entropy of a row in the transition matrix T does not depend on the choice of its row. Meaning that the input symbol to the channel does not affect the probability of a faulty transmission.

For a weakly symmetric channel we can write that the mutual information $I(X;Y)$ is:

$$I(X;Y) = H(Y) - H(Y \mid X) = H(Y) - H(T_r) = log_2|X| - H(T_r)$$

Where T_r is a row of the transition matrix T. With equality if and only if X is uniformly distributed. X being uniformly distributed implies that Y is also uniformly distributed. Over a B DMC with uniform input distribution $P(x = 0) = P(x = 1) = \frac{1}{2}$, where $x_i \in GF(2)$, the channel capacity of a given symmetric channel equals the mutual information $I(X;Y)$ between its input ensemble X and its output ensemble Y.

3.2 The three channels

The three channels covered in the thesis are the Binary Symmetric Channel (BSC), Binary Erasure Channel (BEC), and the Additive White Gaussian Noise (AWGN) channel with Binary-Phase Key Shifting (BPSK) modulation. They can be defined by the set of values the input x_i, and the output y_i take. The three channels all share the characteristic that the channel input x_i can be one of two values drawn from GF(2), while the values from which the channel output y_i is drawn differ in all three channels.

3.2.1 Additive White Gaussian Noise Channel

The AWGN channel is a channel that sends and receives continuous signals, where the receiver collects the continuous signals into chucks through sampling. We omit the details of sampling and view the AWGN as a discrete-time channel, where each sample is fed into the demodulator. The channel is called Additive White Gaussian Noise since the instances of noise added to the signal are uncorrelated and drawn from the Gaussian distribution.

There are different ways to modulate each incoming bit to an appropriate signal for the channel. Here we will review the Binary-Phase Key Shifting (BPSK). In BPSK, each bit x_i is modulated to $\tilde{x}_i$ according to equation (3.2).

$$\tilde{x}_i = \begin{cases} \sqrt{E_b} & \text{if } x_i = 0 \\ -\sqrt{E_b} & \text{if } x_i = 1 \end{cases} \tag{3.2}$$

E_b is the quantity of energy per information bit. After encoding, the energy put into each bit in the transmission codeword is $E_c = \sqrt{E_b} \times R$. For simplicity under simulation we set $E_b = 1$, and get $E_c = E_b \times R$.

The noise maps $n_i : x_i \to y_i$ by $y_i = \tilde{x}_i + n_i$, where n_i is a random variable drawn from the Gaussian distribution σ^2, $N_0(0, \sigma^2)$.

The channel capacity C is:

$$C = W \, log_2 \left(1 + \frac{2RE_b}{N_0}\right) \tag{3.3}$$

where W is the bandwidth and N_0 is the normal distribution [17].

3.2.2 Binary Symmetric Channel

The BSC is a B DMC where each received symbol y_i is identical to the transmitted symbol x_i by probability $1 - p$, and different by probability p. Let $x_i \in GF(2)$ be the symbol from the transmitted codeword x, and $y_i \in GF(2)$ be the symbol from the corresponding channel output of x_i.

The noise maps $n_i : x_i \to y_i$ by $y_i = x_i \oplus n_i$, where n_i is a random variable drawn from the uniform distribution with $P(n_i = 1) = p$ and $P(n_i = 0) = 1 - p$, with summation $\oplus$ being the remainder of sum operation modulo 2. Therefore, the received symbol y_i in BSC is:

$$y_i = \begin{cases} x_i \oplus 1 & \text{if } n_i \leq p \\ x_i & \text{otherwise} \end{cases}$$

By equation (3.1) its transition matrix T is:

$$T = \begin{bmatrix} p(x_i = 0 \,|\, y_i = 0) & p(x_i = 1 \,|\, y_i = 0) \\ p(x_i = 0 \,|\, y_i = 1) & p(x_i = 1 \,|\, y_i = 1) \end{bmatrix} = \begin{bmatrix} 1 - p & p \\ p & 1 - p \end{bmatrix}$$

The transition matrix T is visualized in Figure 3.3.

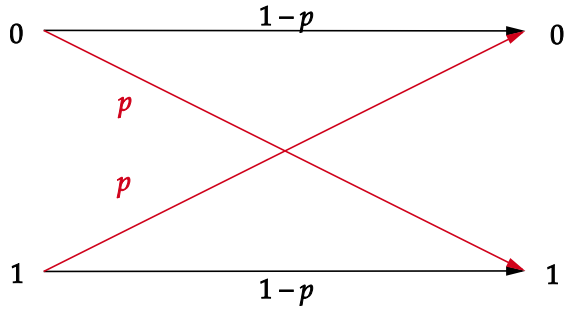

Figure 3.3: Representation of the transition matrix T for the BSC channel.

The BSC is a weakly symmetric channel as the rows of the transition matrix T are permutations of one another, the columns are permutations of

one another, and the sum of every row are equal. Therefore, its channel capacity C is expressed by any row T_r.

$$C = log|R| - H(T_r) = 1 - H(x) \tag{3.4}$$

A plot of the channel capacity for the BSC is shown in Figure 3.4.

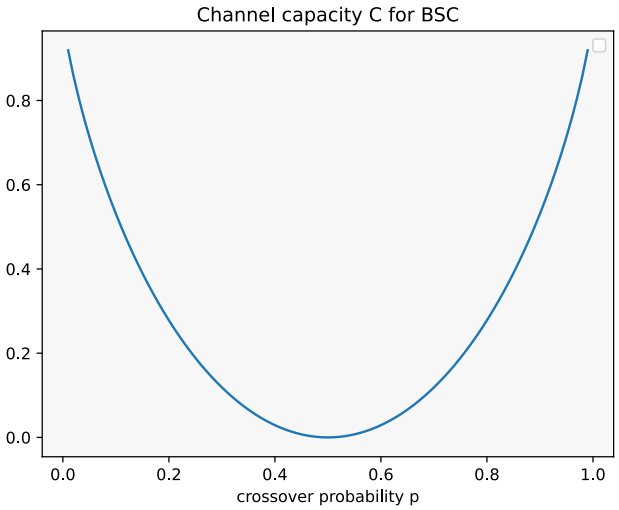

Figure 3.4: Channel capacity for the BSC with crossover probability p on the x-axis and channel capacity C on the y-axis.

3.2.3 Binary Erasure Channel

The BEC is a B DMC where each bit may be lost under transmission with probability α or successfully transmitted by probability $1 - \alpha$, where the loss of a symbol is given by the character e. Let $x_i \in GF(2)$ be the symbol from the transmitted codeword x, and $y_i \in \{0, e, 1\}$ be the symbol from the corresponding channel output of x_i.

The noise maps $n_i : x_i \to y_i$ by $y_i = x_i \oplus n_i$, with some abuse of notation, where n_i is a variable drawn for the uniform distribution. Therefore, the received symbol y_i in BEC is:

$$y_i = \begin{cases} e & \text{if } n_i \leq \alpha \\ x_i & otherwise \end{cases}$$

By equation (3.1) its transition matrix T is:

$$T = \begin{bmatrix} p(x_i = 0 \,|\, y_i = 0) & p(x_i = 1 \,|\, y_i = 0) \\ p(x_i = 0 \,|\, y_i = e) & p(x_i = 1 \,|\, y_i = e) \\ p(x_i = 0 \,|\, y_i = 1) & p(x_i = 1 \,|\, y_i = 1) \end{bmatrix} = \begin{bmatrix} 1-\alpha & 0 \\ \alpha & \alpha \\ 0 & 1-\alpha \end{bmatrix}$$

The transition matrix T is visualized in Figure 3.5.

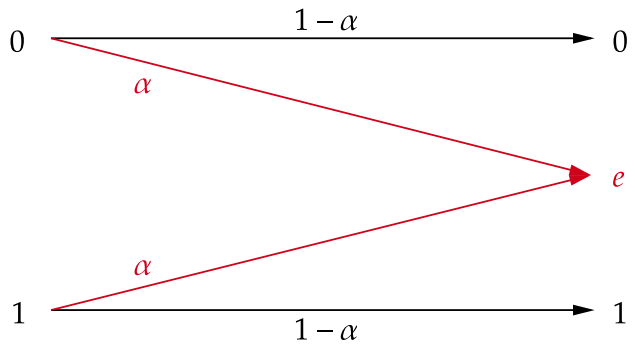

Figure 3.5: Representation of BEC channel

The channel capacity C of the BEC channel is:

$$C = 1 - \alpha \tag{3.5}$$

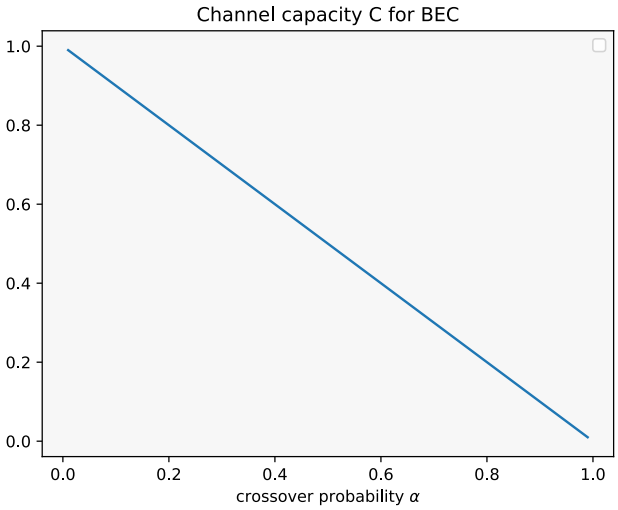

Figure 3.6: Channel capacity for the BEC with crossover probability α on the x-axis and channel capacity C on the y-axis.

3.3 Log-Likelihood Ratio

One might be able to see that while reading the codeword $y = \vec{0}$ over two different channels over the BSC, one with crossover probability $p_1 = 0.0$, and the other $p_2 = 0.5$, that y_2 received from channel with p_2 should be treated with more precaution than y_1 received from channel with p_1 as each bit in y_2 is totally random. Yet, the two codewords are treated with equal certainty.

Instead of passing the output from the channel directly to the decoder, one can pass the probability of each received bit being correct, called the *soft information*. Since $x_i \in GF(2)$ the soft information of y_i can be expressed by (3.6).

$$LR(y_i) = \frac{P(x_i = 0 \,|\, y_i)}{P(x_i = 1 \,|\, y_i)} \, y_i \tag{3.6}$$

called the posteriori probability. One usually passes the log of equation (3.6) to the decoder to avoid underflow representation, called the Log-Likelihood Ratio (LLR). The shorthand notation for LLR of equation (3.6) is:

$$\Lambda(y_i) := \log \frac{p_0}{p_1} y_i \tag{3.7}$$

The LLR instances for the AWGN, BSC, and BEC are given by equation (3.8), (3.9), and (3.10), respectively.

$$\Lambda(y_i)^{AWGN} = \frac{2\sqrt{E_c}}{\sigma^2} y_i \tag{3.8}$$

$$\Lambda(y_i)^{BSC} = \begin{cases} log(\frac{1-p}{p}) & \text{if } y_i = 0 \\ log(\frac{p}{1-p}) & \text{if } y_i = 1 \end{cases} \tag{3.9}$$

which can be simplified to $\Lambda(y_i)^{BSC} = (-1)^{y_i} log(\frac{1-p}{p})$.

$$\Lambda(y_i)^{BEC} = \begin{cases} log(\frac{1}{0}) = \infty & \text{if } y_i = 0 \\ log(\frac{0}{1}) = -\infty & \text{if } y_i = 1 \\ log(\frac{1/2}{1/2}) = 0 & \text{if } y_i = e \end{cases} \tag{3.10}$$

It is normal to let the decoder operate on the soft values as they contain more information about the correctness of each bit, increasing the error correction performance of the decoder. However, when the decoder terminates it outputs the hard decision of its soft information. The hard decision $\hat{x}_i$ from the received symbol y_i is given by equation (3.11).

$$\hat{x}_i = \begin{cases} 0 & \text{if } sign(y_i) \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.11}$$

3.4 Repetition Codes

Through repetition codes, we will discuss that obtaining a communication model which performs better than uncoded transmission, is not as trivial as it may seem.

The (N, K) repetition code produce a codeword d of length N by repeating $K = 1$ input bits N times. An error is detected at the receiver if the incoming codeword y is neither a sequence of N 1-bits nor 0-bits. Decoding is then performed by choosing the candidate codeword $\hat{d}$ that maximizes the probability of $\hat{d}$ being equal to d given the received codeword y. Under equal input distribution in the channel coder, ML decoding is:

$$\hat{d} = \arg\max_{d} p(y \mid d = \hat{d})$$

When y is described in terms of Log-Likelihood Ratio (LLR)s, the ML decisions can be seen as $\hat{d}_i = sign(\sum_i^{i+K} y_i)$. If N were to be an even number this could lead to $\hat{d}_i = 0$, so the decoder might not output a candidate codeword $\hat{d}_i$ on $\sum_i^{i+K} y_i$, this is why N should be an odd number.

Repetition codes' correction capability is $t = \lceil (n-1)/2 \rceil$. The probability of exactly i erroneous bits out of N bits is:

$$P_b^i = \begin{pmatrix} N \\ i \end{pmatrix} p^i (1-p)^{N-i}$$

The probability of decoding error P_b^N is given by:

$$P_b^N = \sum_{i=t+1}^{N} \begin{pmatrix} N \\ i \end{pmatrix} p^i (1-p)^{N-i}$$

Figure 3.7 illustrates that the decoding error probability decrease as N increases. It is then possible to obtain an arbitrarily small probability of error, but at the cost of a very low rate.

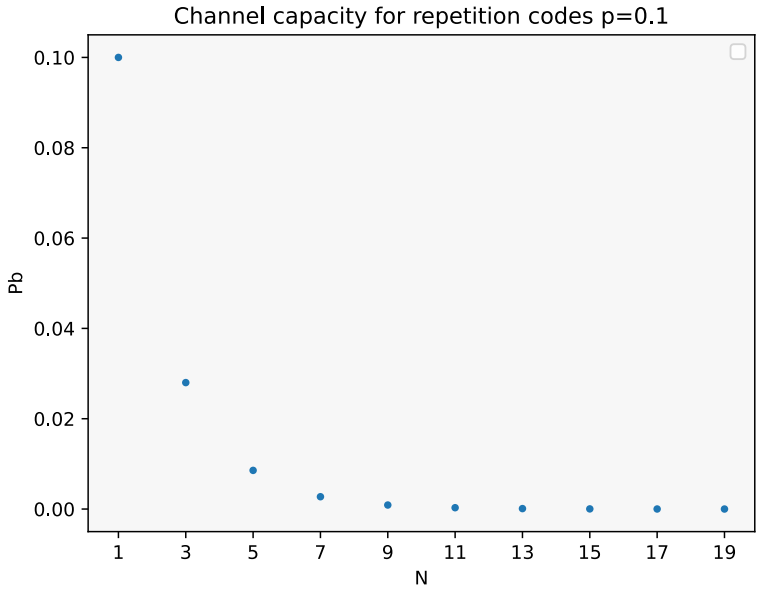

Figure 3.7: Probability of wrongful decoding of repetition codes in BSC with crossover probability $p = 0.1$.

Repetition Codes over AWGN

Suppose the transmitter has $E_b = 1$ Watt of energy available for each bit of information. Under the $(N, 1)$ repetition code the energy per coded bit E_c is $E_c = E_b \times R$. With less energy available per bit, the probability of error becomes:

$$p = Q\left(\sqrt{2E_c/N_0}\right) = Q\left(\sqrt{2E_b/N \times N_0}\right)$$

The crossover probability p in the (N, K) repetition code is higher than in the uncoded transmission due to the coding scheme [11].

The decoder makes the hard decision $\hat{d}_i$ by:

$$\hat{d}_i = \begin{cases} 0 & \text{if } \sum_{i=1}^{i+K} y_{i+K} \geq 0 \\ 1 & \text{if } \sum_{i=1}^{i+K} y_{i+K} < 0 \end{cases}$$

since each $\sqrt{E_c} = \sqrt{E_b} \times R$, $\hat{d} = \sum y_i = \sqrt{E_b}$. This implies that the probability of error for the $(E, 1)$ repetition code over AWGN is:

$$P_b = Q\left(\sqrt{2E_b/N_0}\right)$$

which is the same as for uncoded transmission [11]. Figure 3.8 shows the probability of error with repetition codes of $(1, 1)$, $(3, 1)$, and $(5, 1)$ over the AWGN channel are shown in Figure 3.8. From Figure 3.8 we can conclude that repetition codes does not increase the chance of error-free communication compared to uncoded transmission.

No further encoding steps are applied to the (N, K) repetition code, making the transmission codeword $x = d$, of length $E = N$.

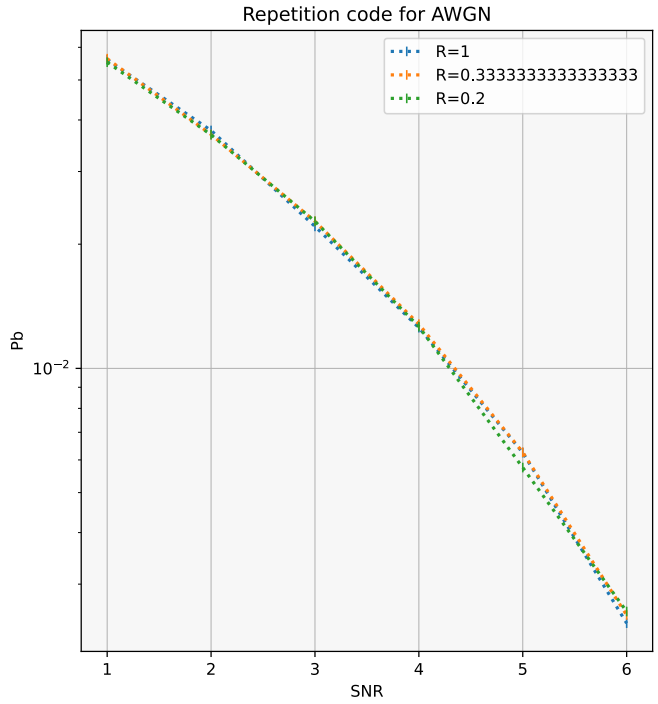

Figure 3.8: Probability of wrongful decoding of repetition codes in AWGN channel.

3.5 Hamming Codes

Through Hamming codes, we will discuss encoding of an (N, K) linear block code by use of a generator matrix G, and a method to check the correctness of the incoming codeword y by a Parity-Check Matrix (PCM) H associated with G.

A channel coding system is said to be an (N, K) linear block code if it takes K input bits and produces N output bits by a generator matrix G. The codeword d can be generated by matrix-matrix multiplication $d = a \times G$.

Hamming codes invented by R. W. Hamming in his work titled *Error detecting and error correcting codes* [8], is one of the first non-trivial codes as they perform better than uncoded transmission. A $(7, 4)$ hamming code is a linear block code that takes $K = 4$ information bits and produces a codeword of length $N = 7$. Since hamming codes are linear block codes, they can be expressed by a systematic generator matrix $G = (I_k \mid C^T)$, and the PCM $H = (C \mid I_{n-k})$. For a matrix H to be a PCM of G, it must satisfy the conditions $H \times d^T = \vec{0}$ and $G \times H^T = \vec{0}$.

The set of all codewords $\mathcal{D}$ are obtained through linear combinations of the rows in G. The codeword space is $|\mathcal{D}| = 2^4 = 16$. It can be verified that the minimum distance $d_{min} = 3$ for the $(7, 4)$ Hamming code by generating $\mathcal{D}$, and noting the minimum distance of any two codewords $d \in \mathcal{D}$ is three. The code is capable of detecting $d_{min} - 1 = 2$ errors and correcting $t = \lfloor (d_{min} - 1)/2 \rfloor = 1$ error.

No further encoding steps are applied to the (N, K) Hamming code, making the transmission codeword $x = d$, of length $E = N$.

3.5.1 Syndrome Decoding

Let $\hat{d}$ be the hard decision of d by the received codeword y, and H be the given PCM. The syndrome s is:

$$s = H \times \hat{d}$$

The procedure of syndrome decoding involves finding the syndrome s by as few linear combinations of the columns from H [10].

let $a = [0, 1, 0, 1]$, and $n = [0, 1, 0, 0, 0, 0, 0]$. Additionally, let generator matrix $G = (I_k \mid C^T)$ and PCM $H = (C \mid I_{n-k})$ be:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \quad H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

d is then $d = a \times G = [0, 1, 1, 1, 0, 0, 1]$, making $y = d \oplus n = [1, 1, 1, 0, 1, 0, 1]$, and our syndrome $s = H \times y^T = [1, 0, 1]$.
Since p $\leq 1/2$, we know that a bit flip is less likely than successful transmission. We then look for as few linear combinations of the columns in H that match the syndrome s. Indeed, the second column of H equals s. Therefore, we estimate that the error occurred in the 2nd position of y.

3.6 Cyclic-Redundancy Check

Cyclic-Redundancy Check (CRC) is used to detect transmission errors in a codeword $\hat{c}$ [16]. Before transmission, the codeword a is appended with a check value derived from the remainder of polynomial long division over

$GF(2)$, of the incoming codeword and a pre-specified generator polynomial $g_P(x)$ of degree P. The A incoming information bits of a are used to compute the P checksum bits appended to a. When the CRC-bits are directly input to the encoder, the CRC vector is denoted by c, otherwise it will be denoted by c'. In both cases, the CRC vector is of length $K = A + P$.

The CRC computation implemented in this thesis is based on the work of [13]. The CRC generator matrix $\mathrm{C}_{A\times P}^{CRC}$ can be constructed recursively through the generator polynomial $g_P(x)$. The last row of C^{CRC} is given by the coefficients of $g_P(x)$ in decreasing order, excluding the highest term. The last row of C^{CRC} is computed by $C^{CRC}(A, i-1) = g_{P-i}$ with $i \in \{1, \ldots, P\}$. Previous rows are computed by equation (3.12). Vector c of length K is computed as $\mathrm{c} = [a \mid a \times C^{CRC}]$.

$$\begin{aligned} \mathbf{C}^{CRC}(k,i) &= \mathbf{C}^{CRC}(k+1,i+1) \oplus \left(\mathbf{C}^{CRC}(k+1,1) \times \mathrm{g}_{P-i}\right) \\ \mathbf{C}^{CRC}(k,P) &= \mathbf{C}^{CRC}(k+1,1) \times \mathrm{g}_0 \end{aligned} \tag{3.12}$$

The available generator polynomials $g_P(x)$ in 5G are given under Subclause 5.1 in [1] and are as follows:

$$\begin{aligned} \mathrm{g}_6(x) &= x^6 + x^5 + 1 \\ \mathrm{g}_{11}(x) &= x^{11} + x^{10} + x^9 + x^5 + 1 \\ \mathrm{g}_{24}(x) &= x^{24} + x^{23} + x^{21} + x^{20} + x^{17} + x^{15} + x^{13} + x^{12} + x^8 + x^4 + x^2 + x + 1 \end{aligned}$$

At the receiver, the bit-polynomial division is again performed with the same generator polynomial $g_P(x)$. When the remainder at the receiver is $\vec{0}$, the codeword is deemed valid and forwarded to the application layer. If not, the receiver can discard the codeword and request it once more from the sender.

Let $A = 12$, and $g_6(x)$ be the encoding polynomial. By equation (3.12) we

get the encoding matrix $C^{CRC}_{A\times P}$:

$$C^{CRC} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The last row of C^{CRC} is the binary representation of $g_6(x) = 1x^6 + 1x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$, with decreasing values of the exponents and leaving out the highest degree term x^6. Computation of second-to-last row of C^{CRC} is shown in Figure 3.9. The colored arrows shows the computation of the first equation of (3.12), and the last grey arrow shows the second equation of (3.12).

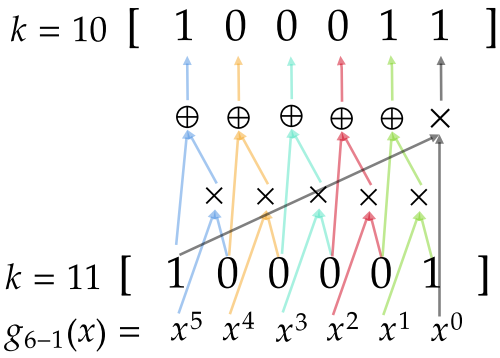

Figure 3.9: Visual representation of computation of row k=10 of C^{CRC} according to (3.12).

3.7 Rate Matching

If the discrete channel in use for transmission has E pre-determined finite amount of channel uses, it might be that the length of codeword N is greater than or shorter than E. The core function of rate matching is to match the length of the encoded codeword N to $E = N - U$, by either leaving out or introducing U more bits to our codeword d. Both puncturing and shortening reduce the length of the encoded codeword while repetition extends its length.

The rate matching implementation differs from polar codes to LDPC codes. Still, the goal is to satisfy the equation $E = N - U$, where E is the number of available bits to be transmitted. Puncturing leave out U codeword bits in the beginning of the codeword, while shortening leave out U codeword bits at the end of the codeword. Rate Matching is often visualized as a circular buffer seen in the Figure 3.10.

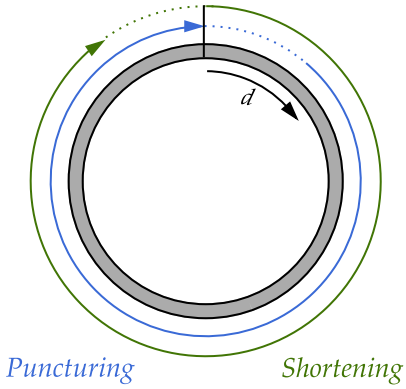

Figure 3.10: Rate Matching in terms of a circular buffer, where the dotted lines represent the U bits not to be transmitted in order to match N to E.

Chapter 4

Polar Codes

Polar codes introduced by Erdal Arikan in his work titled *Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels* [2], break the notion that a codeword needs to be sent over a channel where each codeword bit is equally affected by the noise. Rather, in polar codes, each bit position in the codeword can be seen as being transmitted over its own *synthetic* channel. Polar codes use the basis of mutual information to derive the channel capacity of the synthetic channels W from the original channel. The channels are called synthetic as the codeword is sent over the same physical channel, but each bit position has a different certainty of being decoded correctly, given by the capacity C of their corresponding synthetic channel W.

4.1 Preliminaries for Polar Codes

Polar codes differ from previous capacity-approaching codes in that they are mathematically proven. However, they are proven as the codeword

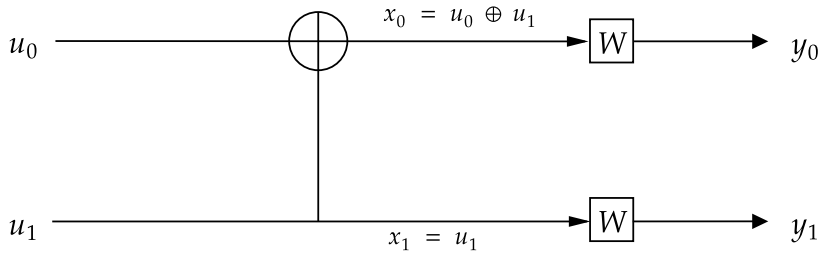

Figure 4.1: Figure for polar encoding of codeword x of length $N = 2$ by input vector u. Each symbol x_i is transmitted over their own synthetic channel W, where the received symbol for x_i is y_i.

length N approaches $+\infty$, which is not feasible as a transmitted block length. However, polar codes are still attractive as encoding and decoding can be implemented in $O(n \log n)$.

4.2 Polarization

One channel is split into several independent synthetic channels. Each channel sends its bit with different reliability measures, meaning every bit has a different probability of being decoded correctly. Increasing N tends the capacity of some synthetic channels toward 1 (noiseless channel), while other channels tend toward 0 (fully noisy channel). This concept is called polarization and is where the name *polar codes* comes from.
The foundation of polar codes lies in the polarization effect on the base matrix $G_2 = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$, often referred to as the *channel transformation matrix* or *kernel matrix* [5].

We will discuss the concept of polarization over $N = 2$.

We want to show that we keep the global property of the mutual information even though we derive new channels from the original.

$$I((X_0, X_1); (Y_0, Y_1)) = 2I(X_0, Y_0)$$

$$
\begin{aligned}
I((X_0, X_1); (Y_0, Y_1)) &= H(X_0, X_1) - H(X_0, X_1 \mid Y_0, Y_1) \\
&= H(X_0) + H(X_1 \mid X_0) - \big(H(X_0 \mid Y_0, Y_1) + H(X_1 \mid X_0, Y_0, Y_1)\big) \\
&= H(X_0) + H(X_1) - \big(H(X_0 \mid Y_0, Y_1) + H(X_1 \mid Y_0, Y_1)\big) \\
&= H(X_0) + H(X_1) - H(X_0 \mid Y_0) + H(X_1 \mid Y_1)
\end{aligned}
$$

Ultimately, we want to extrapolate the values of u and the mapping $(U_0, U_1) \to (X_0, X_1)$ is invertible, we can substitute U_i for X_i in the statements, giving us:

$$
I((U_0, U_1); (Y_0, Y_1)) = I((X_0, X_1); (Y_0, Y_1))
$$

And by applying the chain rule, we get:

$$
I(U_0, U_1; Y_0, Y_1) = I(U_0; Y_0, Y_1) + I(U_1; Y_0, Y_1 \mid U_0) \tag{4.1}
$$

The second term in equation (4.1) can be simplified,

$$
\begin{aligned}
I(U_1; Y_0, Y_1 \mid U_0) &= H(U_1 \mid U_0) - H(U_1 \mid Y_0, Y_1, U_0) \\
&= H(U_1) - H(U_1 \mid Y_0, Y_1, U_0) \\
&= I(U_1; Y_0, Y_1, U_0)
\end{aligned}
$$

Giving us:

$$
\begin{aligned}
I(U_0, U_1; Y_0, Y_1) &= I(U_0; Y_0, Y_1) + I(U_1; Y_0, Y_1, U_0) \\
&= I(W_2^{(0)}) + I(W_2^{(1)})
\end{aligned}
$$

We now want to show that these two synthetic channels keep the global property of mutual information, and that one of the channels has better mutual information property:

$$
I(W_2^{(0)}) + I(W_2^{(1)}) = 2I(W), \quad I(W_2^{(0)} \leq I(W_2^{(1)})
$$

$$\begin{aligned} 2I(W) &= 2I(X_0; Y_0) = I(U_0, U_1; Y_0, Y_1) \\ &= I(U_0; Y_0, Y_1) + I(U_1; Y_0, Y_1, U_0) \\ &= I(W_2^{(0)}) + I(W_2^{(1)}) \end{aligned}$$

Hence, the first statement is proven.

$$\begin{aligned} I(W_2^{(1)}) &= I(U_1; Y_0, Y_1, U_0) \\ &= I(U_1; Y_1) + I(U_1; Y_0, U_1 \mid Y_1) \\ &= I(W) + I(U_1; Y_0, U_1 \mid Y_1) \end{aligned}$$

Since mutual information is always non-negative, then $I(W_2^{(1)}) \geq I(W)$. But since $I(W_2^{(0)}) + I(W_2^{(1)}) = 2I(W)$, then

$$I(W_2^{(0)}) \leq I(W) \leq I(W_2^{(1)})$$

4.3 Code design

Polar codes are based on the n-fold Kronecker product $\otimes n$ of the base matrix G_2, which recursively calculates the transformation matrix $G_N = G_2^{\otimes n}$. By construction, N is limited to take integers on the form $N = 2^n$, while the number of information bits A can take an arbitrary value. Code design of an (N, K) polar code tackles the challenge of identifying the K best synthetic channels that have the highest reliability and using these channels to transmit the K bits of the input codeword. The index set of the K most reliable channels constitutes the *information set* $\mathcal{I}$. The index set of the remaining $N - K$ least reliable channels form the *frozen set* $\mathcal{F}$, which can be seen as the complementary set $\mathcal{F} = \mathcal{I}^{\mathcal{C}}$. The bit positions of $\mathcal{F}$ carry no information as they are likely to be decoded incorrectly and are therefore always set to 0. Setting the frozen bits to a predetermined value removes any uncertainty of their values in the decoder. The decoder of polar codes is therefore often referred to as a *genie aided* decoder.

4.4 Encoding

An (N, K) polar code is defined by its channel transformation $G_N = G_2^{\otimes n}$ and its information set $\mathcal{I}$.

To generate a polar encoded codeword $d = [\mathrm{d}_0, \mathrm{d}_1, ..., \mathrm{d}_{N-1}]$, the auxiliary vector $u = [\mathrm{u}_0, \mathrm{u}_1, ..., \mathrm{u}_{N-1}]$ is introduced. u is generated by assigning $\mathrm{u}_i = 0$ if $i \in \mathcal{F}$, and placing the bits from c in the remaining entries. The output of the encoding process is calculated by $\mathrm{d} = u \times G_N$.

Let $c = (1, 1, 0, 1)$ be the information vector of length K, and the reliability sequence $R_N = (0, 1, 2, 4, 3, 5, 6, 7)$. $\mathcal{I} = (3, 5, 6, 7)$ is the set containing the last K elements of R_N. $\mathcal{F}$ is the complementary set $\mathcal{I}^C$, $\mathcal{F} = (0, 1, 2, 4)$. From $\mathcal{I}$ and $\mathcal{F}$ we create the auxiliary vector $u = (0, 0, 0, 1, 0, 1, 0, 1)$. G_N is generated by the 3-fold Kronecker product of G_2 on the form:

$$G_8 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{\otimes 2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{\otimes 1}$$

The polar-encoded codeword $d = u \times G_8 = (0, 0, 0, 0, 1, 1, 1, 1)$

4.5 Decoding

The original proposal for decoding was by Successive Cancellation (SC). The SC decoder is fast and straightforward but may suffer from error propagation during the decoding process. The decoder can be visualized as a depth-first binary tree traversal prioritizing the left branch. The leaf nodes are the N bits to be estimated, and soft information on the received codeword y is the value of the root node.

A node will first send its soft beliefs α to the left child through the update function $f(\alpha_1, \alpha_2)$, where α_1 is the first half of its soft beliefs and α_2 is the second half of its soft beliefs. The node remains idle until the left child node returns its hard decision estimate β_l. Then it will take β_l, and send soft information to its right child by the update function $g(\alpha_1, \alpha_2, \beta_l)$. The node will then stay idle until the right child returns its hard decision β_r Lastly, the node will send its hard decision β to its parent by update function $\beta(\beta_l, \beta_r) = [\beta_l \oplus \beta_r \,|\, \beta_r]$. This is done for all nodes except for the leaf nodes. When a leaf node is reached, β can be determined by one of two update functions. If the node belongs to the frozen set $\mathcal{F}$, β is always set to 0. Otherwise, the node belongs to the information set $\mathcal{I}$, and β is the return value of the sign function of its incoming belief $\beta = sign(\alpha)$. The decoder terminates when all leaf nodes have made a hard decision on their incoming beliefs. This chain of events is shown in Figure 4.2. Note that the decoding process in the example terminates after β_{10} is executed.

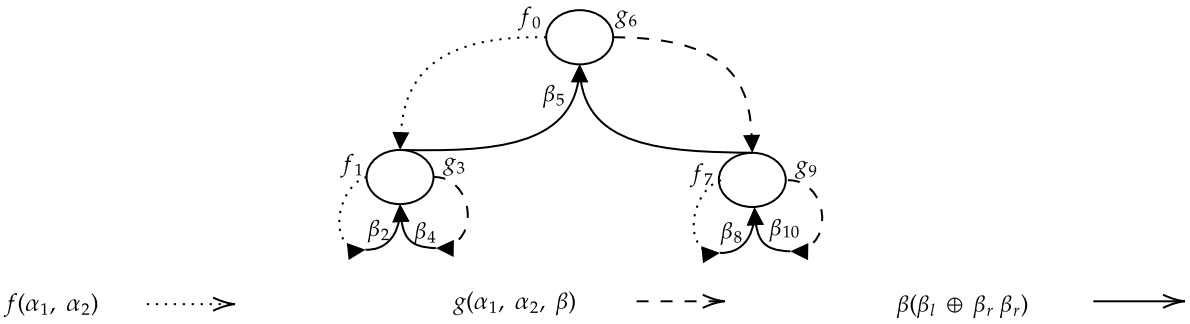

Figure 4.2: The decoding process for polar codes over a binary tree. Each function is assigned a number to indicate the traversal of the tree.

update functions $f()$ and $g()$ are defined by the channel, and are the following in the different channels:

BEC

$$f(\alpha_1, \alpha_2) = \alpha_1 \oplus \alpha_2$$

$$g(\alpha_1, \alpha_2) = \alpha_2 \vee (\alpha_1 \oplus \beta)$$

where $e \oplus \bullet = e$ and $e \vee \bullet = \bullet$.

BSC and AWGN

$$f(\alpha_1, \alpha_2) = (sign(\alpha_1, \alpha_2) \times min(|\alpha_1|, |\alpha_2|)$$

$$g(\alpha_1, \alpha_2, \beta) = \alpha_2 + \alpha_1 \times (1 - 2 \times \beta)$$

Chapter 5

LDPC Codes

5.1 Preliminaries for LDPC Codes

Low-Density Parity-Check (LDPC) codes introduced by R. G. Gallager in 1962 are linear block codes defined by a PCM H containing mostly 0's and few 1's [7]. Each check equation contains few variables, and each variable occurs in several check equations. The idea for decoding is to have codeword bits occurring in several check equations as a variable. Each equation must follow the constraint that the sum of its variables modulo 2 equals 0. With the PCM being sparse, the decoding can be of low complexity as few computations are executed. Each row of the PCM describes a single check equation. Since the PCM describes each check equation, they are often called *parity check equations*.

Let us look at a decoding example of incoming codeword y with a single erroneous bit. By successively evaluating each parity check equation, we can rule out the codeword bits which are not erroneous and build up a *belief* of which bit to be erroneous.

Let PCM H_{MxN} with $M = 3$, $N = 6$, codeword $d = (1,1,0,1,1,0)$, and the noise vector $n = (1,0,0,0,0)$. y is then $y = d \oplus n = (0,1,0,1,1,0)$. In addition, let H and its parity check equations α:

$$H = \begin{array}{cccccc} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{array} \qquad \begin{aligned} \alpha_{n,0}: &\quad x_1 \oplus x_2 \oplus x_3 = 0 \\ \alpha_{n,1}: &\quad x_0 \oplus x_2 \oplus x_4 = 0 \\ \alpha_{n,2}: &\quad x_0 \oplus x_1 \oplus x_5 = 0 \end{aligned}$$

We see that equations $\alpha_{n,1} : 0 \oplus 0 \oplus 1$ and $\alpha_{n,2} : 0 \oplus 1 \oplus 0$ does not satisfy the sum of variables equal 0, and that they share the variable x_0. Hence, we estimate the erroneous bit to be x_0 and get a valid codeword as $H \times d^T = 0$.

5.2 Factor Graphs

LDPC codes were mostly ignored up until the 1990s [17]. A key development in this resurrection was the introduction of factor graphs [17], with message passing between Check Node (CN) and Variable Node (VN).

The factor graph of $H_{M\times N}$ is a bipartite graph with N VNs and M CNs, with and edge between VN_n and CN_m if $H_{m,n} \neq 0$, in which they are said to be *neighbors*. The only edges allowed in a factor graph are those connecting VNs and CNs. The set of neighbors for a CN m is written as $\mathcal{N}(m)$ and of a VN written as $\mathcal{N}(n)$. The number of neighbors $\mathbf{d_v} = |\mathcal{N}(.)|$ of a node is called its degree, where (.) can be n or m.

An LDPC code is said to be *regular* if all the rows r of H have the same weight w_r and all the columns c of H have the same weight w_c. Resulting in all VNs having the same degree $d_v = w_r$, and all CNs having the same degree $d_c = w_c$. Otherwise, The LDPC code is said to be *irregular*.

A significant advantage of sparseness is that when storing the positions of the non-zero elements of the matrix H, the memory required to store H is much more efficient than if one were to store the entire matrix. An essential characteristic of the PCM in decoding of LDPC codes is the ability to segment the PCM, and process the rows of the matrix in a parallel manner.

An example of an irregular PCM H of dimension $M \times N = 8 \times 10$ is given in matrix (5.1), and its corresponding factor graph is in Figure 5.1.

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.1}$$

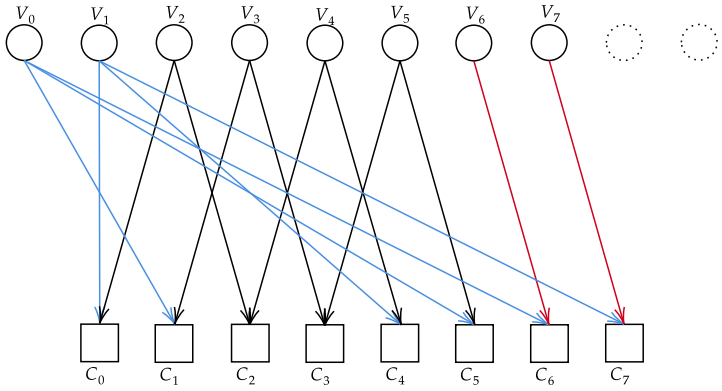

Figure 5.1: Factor Graph for PCM (5.1). Circle nodes represent the variable nodes (VN), and square nodes represent check nodes (CN).

5.3 Encoding

There are two distinct ways to perform encoding of LDPC codes. One approach is to generate the codeword d from the generator matrix G, while the other generates d from the PCM H. We will now look at encoding performed by $d = c \times G$ and later look at encoding by $d = H \times c^T$ as this is how it is performed in 5G.

To construct G, one can do the following: Let H_p be the invertible $(n-k)$ by $(n-k)$ matrix such that $H_p H = (I_{n-k}, H_2)$ for some H_2, a $n-k$ by k matrix. Then $G = (H_2^T, I_k)$ is a generator matrix. $c \times G$ generates the codeword d, and H is a PCM as its satisfies the constraint $H \times d^T = 0$ and $G \times H^T = 0$.

The drawback of storing G is that the entries of the generator matrix generally scale by n^2 as the matrix grows. If one were to store the non-zero elements in the matrix, it would require much more memory than storing the non-zero elements of H, which generally scales by n.

5.4 Decoding

The decoding algorithm for LDPC codes can be viewed as a message-passing algorithm over the factor graph, iteratively passing beliefs between VNs and CNs. The belief sent from VN_n to its neighbor CN_m is denoted by $\alpha_{n,m}$, and the beliefs sent from CN_m to VN_n is denoted as $\beta_{m,n}$ [19]. After every iteration, the estimate at each variable node becomes more certain of their own beliefs. The decoding algorithm is continuously processed until the decoder hits a stopping criterion. This criterion can be that either the estimated codeword $\hat{d}$ satisfies $H \times \hat{d}^T = 0$, or that a

maximum number of iterations $Iter_{max}$ is reached. In the latter, a decoding failure is declared.

Each CN_m can be seen as a soft-output decoder that outputs a list of $|\mathcal{N}(m)|$ beliefs, one for each connected VN. Where the sum of all beliefs sent from CNs connected to VN_n generates the global belief L_n^{tot} that form the basis of the hard decision $\hat{d}_n = sign(L_n^{tot}) + L_n$, where L_n is the shorthand notation for the LLR value $\Lambda(y_n)$.

The total belief for every $_n$ is given by equation (5.2).

$$L_n^{\text{tot}} = L_n + \sum_{m \in \mathcal{N}(n)} \beta_{m,n} \tag{5.2}$$

In every iteration of the decoding algorithm, the belief for each VN_n is updated by equation (5.3).

$$\alpha_{n,m} = L_n + \sum_{n' \in \mathcal{N}(m) \setminus n} \beta_{m,n} \tag{5.3}$$

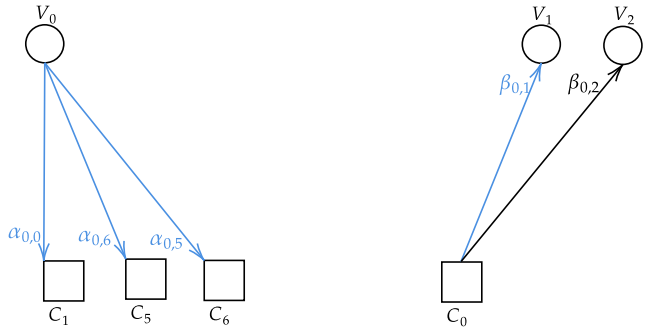

Figure 5.2: Message passing of $\alpha_{0,i}$ and $\beta_{0,i}$ respectively of selected nodes from Figure 5.1.

Computations at Check Nodes

The computations performed at the CNs are more complicated than those

performed at the VNs and are often performed using simple approximations [17]. The belief sent from CN_m to VN_n is:

$$\beta_{m,n} = (-1)^{|N(n)|} \, 2 \tanh^{-1} \left(\prod_{m' \in \mathcal{N}(n) \setminus m} \tanh \left(\alpha_{m',n} / 2 \right) \right) \tag{5.4}$$

also known as *Belief Propagation*. Belief Propagation is computationally expensive and can be replaced by a simple approximation called the *Min-Sum* approximation, given by equation (5.5).

$$\beta_{m,n} \approx (-1)^{|\mathcal{N}(n)|} \prod_{n' \in \mathcal{N}(n) \setminus m} \operatorname{sign} \left(\alpha_{n',m} \right) \min_{n' \in \mathcal{N}(n) \setminus m} \left(|\alpha_{n',m}| \right) \tag{5.5}$$

The Min-Sum algorithm is not as precise as belief propagation, but the benefit of easier computations outweighs its loss in preciseness.

The decoder is initialized with $\beta_{m,n} = 0 \; \forall \, m, \, n \in M, \, N$.
(5.2) is then $L_j^{\text{tot}} = \Lambda(y) \; \forall \, j \in N$.
Equation (5.3) becomes $\alpha_{n,m} = L_n \, \forall m, \, n \in M, \, N$ if $H_{n,m} = 1$. After initialization, the algorithm executes formulas (5.5), (5.2), then verifies if (5.2) produced a valid codeword by checking if $H \times \hat{d}^T = 0$ is satisfied, if not then formula (5.3) is executed and the algorithms starts over.

Chapter 6

Polar Codes in 5G

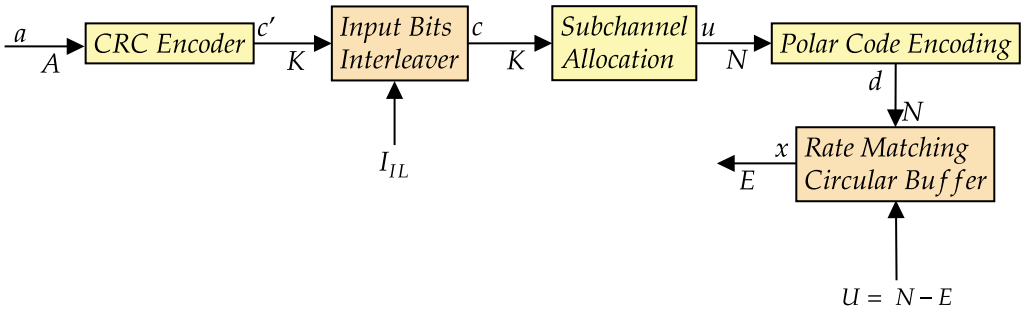

Figure 6.1: Implemented steps in encoding chain of polar codes in 5G, yellow, and orange blocks represent mandatory and optional steps, respectively.

6.1 CRC Encoder

The CRC structure in the 5G standard is used as an error detection of the output codeword under the Successive Cancellation (SC) decoder and used as a method to extract a final candidate codeword from a list of L

candidate codewords in the Successive Cancellation List (SCL) decoder. The selection of CRC generator polynomial $g_P(x)$ is specified as follows:

$$g_P(x) = \begin{cases} g_6(x) & \text{if } 12 \geq A \leq 19 \\ g_{11}(x) & \text{if } 20 \geq A \leq 1706 \\ g_{24}(x) & otherwise \end{cases}$$

6.2 Input Bits Interleaver

When CRC is used to select the final candidate codeword, the entire decoding process must be executed. In some cases, it is preferred to early terminate the decoding process if the yielding candidate codeword is bound to fail, rather than discarding the candidate codewords once the decoding terminates [9].

Early termination of the decoder for polar codes in 5G is achieved by distributing the CRC-bits within the codeword c in such a way that when encountering a CRC-bit, the decoder paths can be deemed as likely to fail or likely to succeed. A constraint of the CRC computation is that it must be able to be executed only with the information bits existing in the list of codeword candidates. Consequently, the decoding complexity can be reduced by early termination if all codeword candidates fail to pass the correctness check, as the decoder is terminated early [13].

To distribute the CRC-bits and create the interleaved codeword c from c', one pre-computes $h = |\Pi^{max}| - K$, then loops over the interleaving pattern Π^{max} given by Table 5.3.1.1-1 in the standard [1], and stores the elements in the interleaver vector Π satisfying $\pi \geq h$ as $\pi - h$, where $\pi \in \Pi^{max}$ [5]. Note that $|\Pi^{max}| = 164$, and the CRC-generator polynomial is always $g_{24}(x)$ when the interleaver flag I_{IL} is activated. I_{IL} is activate when $I_{IL} = 1$, and inactive when $I_{IL} = 0$.

Let $A = 12$, $K = 36$, and $I_{IL} = 1$. Then $h = 164 - 36 = 128$, and the interleaver vector $\Pi = [1, 4, 6, 10, 11, 12, 0, 2, 5, 7, 13, 3, 8, 14, 9, 15, \ldots, 35]$ where elements occurring after the last interleaved bit $\pi - h = 15$ are all occurring in incremental order. The interleaver vector Π is applied to c' and the interleaved codeword c is obtained by $c = [c_{\Pi_0}, c_{\Pi_1}, \ldots, c_{\Pi_{K-1}}]$ [13], shown in Figure 6.2.
The information bits a are in grey, and the CRC-bits are in green.

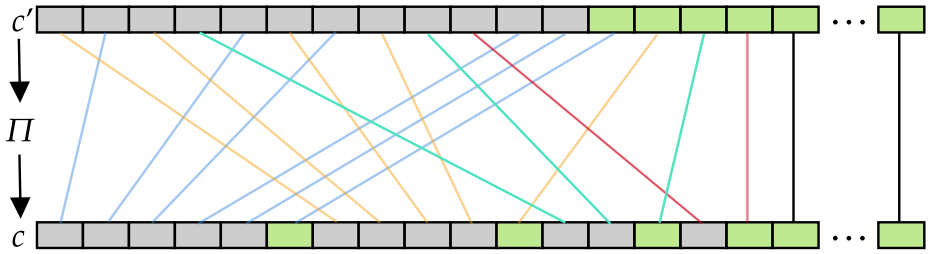

Figure 6.2: Creating the CRC interleaved vector c from input-vector c', when interleaver flag $I_{IL} = 1$.

If $I_{IL} = 0$, the output vector c of the interleaver is equal to the input vector c'. The input and output vectors are still denoted by c' and c, respectively.

6.3 Sub Channel Allocation

Initially, the information set $\mathcal{I}$ and its complementary frozen set $\mathcal{F}$ are computed based on the universal reliability order $R_{N_{Max}}$ and the matching set MS. $R_{N_{max}}$ is the list of all synthetic channels sorted by their reliability order in increasing order, and is given in Table 5.3.1.2-1 in the standard [1]. Note that $|R_{N_{Max}}| = 1024$. Afterwards, bits from codeword c are assigned to bit positions in u according to $\mathcal{I}$.

The first step in sub-channel allocation is to retrieve the reliability sequence R_N with N appropriate channels from $R_{N_{Max}}$. This is done by extracting the channels from $R_{N_{Max}}$ that are smaller than N, creating $R_N = \{R_0, ..., R_{N-1}\}$ where $R_i < N$.

From R_N, the matching set is determined. The first bit positions extracted from R_N are the U bit positions belonging to the matching set MS. These bits will ultimately not carry any information bits and are therefore added to $\mathcal{F}$. The information set $\mathcal{I}$ are the K most reliable positions in R_N not belonging to MS, the remaining entries in R_N are added to $\mathcal{F}$.

6.4 Polar Code Encoding

The core polar encoder $d = u \times G_N$ in 5G is executed much in the same way as explained in the preliminary. The difference is that the information bits a are appended with P CRC-bits beforehand, making the encoding input vector not a but c, of length $K = A + P$.

6.5 Rate Matching Circular Buffer

As previously mentioned, the codeword length N is rigorous as it is restricted to be a power of two, while the length of the information sequence A can be of any finite integer length. Hence, the ability to scale the number of bits transmitted to a desired amount is an important functionality of polar codes in 5G. The techniques used in this paper to modify the codeword d to achieve codeword e of desired length E, are through the classical rate matching schemes shortening and puncturing, is based on the work of [4].

The difference between puncturing and shortening lies in its effect on the code's behavior and how the decoder reconstructs the bits belonging to the matching set. Note that some information bit positions determined in the sub-channel allocation without rate matching may become unfit to carry information bits due to the rate matching scheme [3].

The matching pattern is determined by the reliability sequence R_N and the bit-reversal sequence of B_N. The bit-reversal sequence B_N is defined by the reverse of the binary representation of each element in R_N, then finding the index of the $U = N - E$ appropriate indices of B_N in R_N. Note that all binary representations must be of equal length. Therefore, some binary representations are pre-padded with 0 to reach the longest non-padded binary digit length.

6.5.1 Puncturing

In puncturing, the first U left-out bit positions are obtained by taking the first U integers in R_N and returning the index of these U integers in B_N. The punctured bits make up the matching set MS.
The decoder recovers the punctured bits by setting their $LLR = 0$, meaning the decoder cannot derive any information about the values of the punctured bits when initializing the decoder. Finally, $\mathcal{F}$ and $\mathcal{I}$ is selected according to the bit reliabilities of the mother polar code, while the matching set MS is generated from the bit-reversal permutation.

6.5.2 Shortening

In shortening, the last U left-out bit positions are obtained by taking the last U integers in R_N and returning the index of these U integers in B_N. The shortened bits make up the matching set MS.

The shortened bit positions in u are set so that linear combinations of the shortened bit positions are of 0-values only in their respective bit positions in d. The decoder then knows the shortened bits to take the zero value, setting their $LLRs = +\infty$.

6.6 Decoding

The Successive Cancellation List decoder proposed in the work of [15], improves the performance of the SC decoder by keeping L candidate codewords during the decoding. SCL provides better error correction performance at a higher complexity cost. The most attractive trade-off between better error correction and low complexity is set to $L = 8$ in the 3GPP standard [1]. The decoder is said to have L different decoders, where each decoder stores the hard decisions on the computed information set and the path metric PM.

The SCL decoder traverses the binary tree similarly to the SC decoder. The difference between the SC and SCL decoders is that the SCL evaluates the information bits to be both 0 and 1. When the decoder evaluates an information node, the path-metric PM of each candidate codeword is computed by equation (6.1).

$$PM = \begin{cases} PM + |\alpha|, & \text{if } sign(\alpha) \neq (-1)^{\hat{c}_i} \\ PM, & \text{otherwise} \end{cases} \tag{6.1}$$

where α is the incoming beliefs for the leaf node. After the estimation decision, the size of the list decoder is $2L$, and the decoder is then pruned to keep the $L = 8$ most likely codeword candidates.
There are no updates to the candidate codewords when a frozen node is encountered, and PM remains the same.

6.6.1 Decoding of interleaved Polar Codes

In the work of [13], little is mentioned about the implementation of the check-sum of the interleaved CRC-bits. We assume that their check-sum is performed similarly to the work of [6].

Here, the rows of the CRC encoding matrix G is permuted according to $G' = [G_{\Pi'_0}, G_{\Pi'_1}, \ldots, G_{\Pi'_{A-1}} \,|\, \Pi < A]$. When evaluating the correctness of candidate codeword $\hat{x}$ to the interleaved CRC-bit p_i, where p_i occurs in c' at index Π_{p_i}. The CRC-check is computed by $\hat{a} \times G'$, where $\hat{a}$ is $\hat{d}$ when all previous CRC-bits are excluded. If the product of $\hat{a} \times G'$ in position Π_{p_i} matches the hard decision β of the incoming belief α we determine $\hat{a}$, and equivalently $\hat{d}$, to be correct.

Three check methods for Cyclic Redundancy Check Aided Successive Cancellation List (CA SCL) decoding are suggested in the work of [13]. We will focus on the Check and Remove SCL (CR SCL) as it is said to have the better error correction performance of the three decoders discussed.

CR SCL
In the CR SCL, when estimating a CRC-bit, the only paths kept in the decoder are the paths passing the CRC check. Those that do not pass the CRC-check are discarded. The number of remaining paths in the decoder V after CRC-evaluation is then $V \leq L$, and the list of candidates grows only from the V codewords passing the CRC-check. The said paper mentions that the CR SCL decoder is expected to have better error-correction performance and therefore, fewer earlier terminations. Better error-correction performance can be credited to the removal of incorrect candidate codewords that may have lower path-metric than the correct codeword.

The inability to evaluate the incoming parity bit p_i as both one and zero poses a significant drawback of the CA SCL decoder. Ultimately, we would

like to generate $2L$ codewords from the L existing candidate codewords, one path for $p_i = 0$, the other for $p_i = 1$.

Here, we propose a decoding algorithm to evaluate the incoming parity bit p_i to be both zero and one, and append these to each candidate codeword $\hat{c}'$. We will call the candidate codewords $\hat{c}'_0$ and $\hat{c}'_1$ for 0-appending and 1-appending, respectively, and perform $H^{CRC} \times \hat{c}'_i$ of $i \in \{0, 1\}$.

First, we need to create PCM H^{CRC} from our CRC generator matrix G^{CRC}. In order to derive H^{CRC} from G^{CRC}, we represent G as a systematic matrix on the form $G = [I_{A\times A}\, C]$, where I is the identity matrix and C is computed by (3.12) with $G_{24}(x)$. H^{CRC} can then be represented as $H^{CRC} = [C^T\, I_{P\times P}]$. The interleaved PCM H'^{CRC} is created by permuting its columns by the interleaver vector Π on the form $H'^{CRC} = [H^{CRC}_{\Pi'_0}, H^{CRC}_{\Pi'_1}, \ldots, H^{CRC}_{\Pi'_{K-1}}]$. Then, let $\mathcal{K}$ be the indices of the interleaved CRC-bits in c', and $\mathcal{P}$ be the indices of each element in $\mathcal{K}$. The candidate codewords that are kept in the decoder are the ones where $H^{CRC} \times \hat{c}_i^T$ gives a 0-vector in the first positions up to $\mathcal{P}_{p_i}$.

Though the interleaver is designed to catch faulty candidate codewords, one wants as few early terminations as possible. Fewer early terminations can be attributed to a well-performing removal of unwanted candidate codewords.

Chapter 7

LDPC Codes in 5G

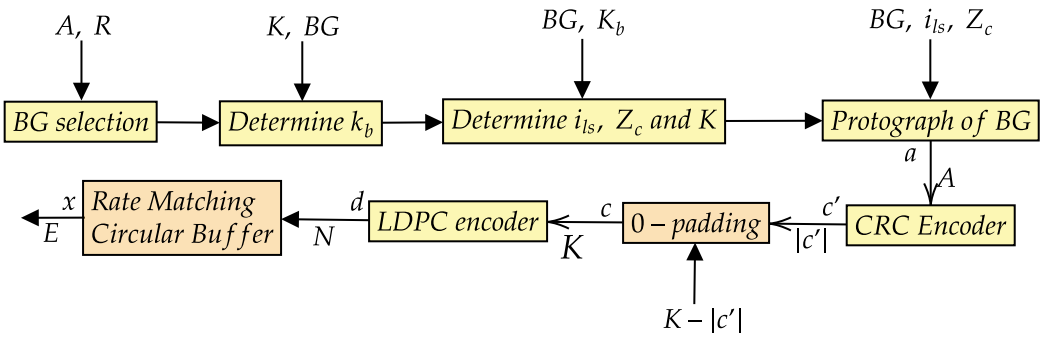

Figure 7.1: Implemented steps in encoding chain of LDPC in 5G, yellow and orange blocks represent mandatory and optional steps, respectively.

One big disadvantage for LDPC codes is the high complexity of the encoding process $d = c \times G$. A family of LDPC codes called Quasi-Cyclic LDPC (QC LDPC) codes aims at overcoming high encoding complexity by leaving out the generator matrix G and performing encoding with the PCM H by $d = H \times c^T$. QC LDPC codes can be defined by a Base Graph (BG) and a lifting size Z_c. The $\text{BG}_{m_b \times k_b}$ is a matrix consisting of integer entries $I \in \{-1, 0, 1, \ldots, Z_c - 1\}$. Each integer I is expanded to an identity

matrix of size Z_c and then circularly shifted to the right I times, except for $I = -1$, which is a $Z_c \times Z_c$ zero matrix.

7.1 Code Design

Code design of an (N, K) QC LDPC code finds an appropriate Base Graph (BG) and a lifting size Z_c for the CRC encoded codeword c and the rate R, creating the codeword d of length N.

7.1.1 Determine $g_P(x)$

The CRC structure in the 3GPP standard is used as error-detection for the codeword candidate $\hat{c}$, where $\hat{c}$ are the K first bits in $\hat{x}$ output from the Min-Sum decoder.
The selection of CRC generator polynomial $g_P(x)$ is as follows:

$$g_P(x) = \begin{cases} g_{11} & \text{if } A \leq 3824 \\ g_{24} & \text{otherwise} \end{cases}$$

and the output of the CRC generator is codeword $c = [c \,|\, c \times G]$ of length $K = A + P$.

7.1.2 Determine BG

There are many variations to the Base Graph (BG), but they can all be sectioned in to Base Graph 1 and Base Graph 2, $BG1$ and $BG2$ for short, respectively. BG2 is of dimension 42×52 supporting shorter codewords

over higher rates, and BG1 is of dimension 46×68 supporting longer codewords over lower rates.

The selection of the BG is specified in the 3GPP standard under Clause 6.2.2 *LDPC base graph selection* [1], and can be summarized as follows:

- If $K \leq 292$, then BG2 is selected.
- If $K \leq 3824$ and $R \leq 0.67$, then BG2 is selected.
- If $R \leq 0.25$, then BG2 is selected.
- Else BG1 is selected.

7.1.3 Determine K_b

K_b denotes the number of information bit groups in the BG, where each information bit group contains Z_c information bits.
As specified in the standard under Clause 5.2.2 [1].
For BG1 $K_b = 22$.
For BG2:

- If $K > 640$ then $K_b = 10$.
- If $560 < K \leq 640$ then $K_b = 9$.
- If $192 < K \leq 560$ then $K_b = 8$.
- If $K \leq 192$ then $K_b = 6$

7.1.4 Determine lifting size Z_c

The *lifting size* Z_c, also called *expansion factor*, is computed by selecting the minimum value in the *sets of LDPC lifting sizes* Z specified in the 3GPP standard in Table 5.3.2-1 [1], such that $K_b \times Z_c \geq K$. While also storing the set index i_{LS}. The sets of lifting sizes can be seen in Table 7.1.

Set index (i_{LS})	**Set of lifting sizes (Z)**
0	{2, 4, 8, 16, 32, 64, 128, 256}
1	{3, 6, 12, 24, 48, 96, 192, 384}
2	{5, 10, 20, 40, 80, 160, 320}
3	{7, 14, 28, 56, 112, 224}
4	{9, 18, 36, 72, 144, 288}
5	{11, 22, 44, 88, 176, 352}
6	{13, 26, 52, 104, 208}
7	{15, 30, 60, 120, 240}

Table 7.1: sets of LDPC lifting size Z.

Note that Z is the set of lifting sizes, while Z_c is the minimum value in Z satisfying $K_b \times Z_c \geq K$.

7.1.5 Create BG

Each row element m and column element n in BG1 are gathered from Table 5.3.2-2, and in BG2 they are gathered from Table 5.3.2-3 in the 3GPP standard [1]. The elements gathered from these tables are denoted as $V_{m,n}$, and are stored in $BG_{m,n}$ as $BG_{m,n} = V_{m,n} \mod Z_c$. Only the non-zero elements are stored in each table, if one tries to gather an element which is not in the table $BG_{m,n}$ then $BG_{m,n} = -1$.

7.1.6 Create PCM H

By previous steps we have BG, i_{LS} and Z_c. To create the PCM H each entry of $BG_{m,n}$ is expanded to a $Z_c \times Z_c$ submatrix. Elements of $BG_{m,n}$ which are -1 is a $Z_c \times Z_c$ 0-submatrix in H, while for elements $BG_{m,n} \geq 0$ is the identity matrix circularly shifted $BG_{m,n}$ times to the right.

The overall structure of H, and subsequently BG, is:

$$\mathrm{H} = \begin{bmatrix} \mathrm{A} & \mathrm{B} & 0 \\ \mathrm{C} & \mathrm{D} & \mathrm{I} \end{bmatrix}$$

Submatrix $\begin{bmatrix} \mathrm{A} & \mathrm{B} \end{bmatrix}$ of dimension $4 \times K_b$ are said to be the core part of H, while the rest are said to be extensions parts of H. $\begin{bmatrix} 0 \end{bmatrix}$ is a zero submatrix, and $\begin{bmatrix} \mathrm{I} \end{bmatrix}$ is an identity submatrix.

To give a short example of the expansion of the BG, we omit the details of parameter selection and let $Z_c = 2$ and BG be:

$$BG = \begin{bmatrix} 1 & 0 & -1 & -1 & -1 \\ -1 & 0 & 0 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 \\ 0 & -1 & -1 & 0 & -1 \end{bmatrix}$$

With $Z_c = 2$, we get the following unique submatrices:

$$BG_{0,0} = 1 \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad BG_{0,1} = 0 \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad BG_{0,2} = -1 \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Giving us a PCM equal to that of (5.1), and the factor graph from Figure 5.1.

7.1.7 CRC Encoder

The notation for the codeword after CRC-calculation is slightly different for LDPC as K denotes the size of the codeword after 0-appending. The length of the CRC-encoder output will just be given as $|c'|$. We say we obtain the codeword c' of length $|c'| = A + P$ after CRC computation in LDPC. c' might then be appended with 0's to achieve the length $K = k_b \times Z_c$.

For LDPC codes, the length of the CRC output vector might not be of correct dimension K. Sp by appending $K' = K - |c|$ 0-bits to c', we ensure the input vector of the LDPC encoder c to be of correct length.
Note that LDPC does not use any interleaving of the CRC-bits.

7.2 QC LDPC Encoding

The goal of encoding is to solve the following system of equations:

$$\mathrm{Hd}^{\mathrm{T}} = 0 \Rightarrow \begin{bmatrix} \mathrm{A} & \mathrm{B} & 0 \\ \mathrm{C} & \mathrm{D} & \mathrm{I} \end{bmatrix} \begin{bmatrix} \mathrm{c}^{\mathrm{T}} \\ \mathrm{p}_{\mathrm{c}}^{\mathrm{T}} \\ \mathrm{p}_{\mathrm{a}}^{\mathrm{T}} \end{bmatrix} = 0 \tag{7.1}$$

where d is a codeword of length N and can be represented in systematic form as:

$$\mathrm{d} = \begin{bmatrix} \mathrm{c} & \mathrm{p_c} & \mathrm{p_a} \end{bmatrix} \tag{7.2}$$

p_c is a vector of the core parity bits containing four vectors each of size Z_c, p_a is a vector of additional parity bits, and c is the information bits vector appended with both CRC-bits and $K' = K - B$ 0-bits.

Structure of H

The core parity bits can be calculated from the codeword c and core submatrix $[\,\mathrm{A}\ \mathrm{B}\,]$. Note that submatrix $[\,0\,]$ does not affect the encoding computations and can therefore be left out of the encoding process. The additional parity bits can be calculated from information and core parity bits using the extension submatrix $[\,\mathrm{C}\ \mathrm{D}\,]$ [12]. $[\,\mathrm{I}\,]$ and $[\,0\,]$ represent the identity-matrix and the zero-matrix, respectively.

A key difference between the two BGs is the structure of submatrix $[\,\mathrm{B}\,]$, which directly affects the computations of $p_c = \{p_{c_0},\ p_{c_1},\ p_{c_2},\ p_{c_3}\}$. We have observed the following structures of submatrix B:
For BG2 there are:

$$\mathrm{B0}_{\mathrm{BG2}} = \begin{bmatrix} 0 & 0 & - & - \\ - & 0 & 0 & - \\ 1 & - & 0 & 0 \\ 0 & - & - & 0 \end{bmatrix} \quad \mathrm{B1}_{\mathrm{BG2}} = \begin{bmatrix} 1 & 0 & - & - \\ - & 0 & 0 & - \\ 1 & - & 0 & 0 \\ 0 & - & - & 0 \end{bmatrix} \tag{7.3}$$

While for BG1 there is one general B1 structure on the form:

$$\mathrm{B1}_{\mathrm{BG1}} = \begin{bmatrix} 1 & 0 & - & - \\ S & 0 & 0 & - \\ - & - & 0 & 0 \\ 1 & - & - & 0 \end{bmatrix} \tag{7.4}$$

Where 0 represents the identity matrix, 1 Is the identity matrix circularly shifted by one, $-$ is the zero matrix, and S is an identity matrix circularly shifted S times. The value of S is dependent on Z_c and iLS.

To obtain $p_c = \{p_{c_0},\ p_{c_1},\ p_{c_2},\ p_{c_3}\}$ one tries to satisfy the following equation:

$$\begin{bmatrix} \mathrm{A} & \mathrm{B} \end{bmatrix} \times \begin{bmatrix} c^T \\ p_c^T \end{bmatrix} = [\,A\,] \times C^T + [\,B\,] \times p_c^T = 0 \tag{7.5}$$

Denoting the intermediate result $[\,A\,] \times c^T$ as $\gamma^T = \{\gamma_0^T, \gamma_1^T, \gamma_2^T, \gamma_3^T\}$. Let $[\,\mathrm{B}\,] = \mathrm{B1_{BG2}}$, from $\gamma^T + [\,B\,] \times p_c^T$ we get the following set of linear equations:

$$\begin{aligned} \gamma_0 + 1p_{c_0} + p_{c_1} &= 0 \\ \gamma_1 + p_{c_1} + p_{c_2} &= 0 \\ \gamma_2 + p_{c_0} + p_{c_2} + p_{c_3} &= 0 \\ \gamma_3 + 1p_{c_0} + p_{c_3} &= 0 \end{aligned}$$

By solving the system of linear equations for $\mathrm{B1_{BG2}}$ we get the results:

$$p_{c_0} = \sum_0^4 \gamma_i \quad p_{c_1} = \gamma_0 + 1p_{c_0} \quad p_{c_2} = \gamma_1 + p_{c_1} \quad p_{c_3} = \gamma_3 + 1p_{c_0}$$

For $\mathrm{B0_{BG2}}$ we get:

$$1p_{c_0} = \sum_0^4 \gamma_i \quad p_{c_1} = \gamma_0 + p_{c_0} \quad p_{c_2} = \gamma_1 + p_{c_1} \quad p_{c_3} = \gamma_3 + p_{c_0}$$

And for $\mathrm{B1_{BG1}}$ p_c we get:

$$p_{c_0} = \sum_0^4 \gamma_i \quad p_{c_1} = \gamma_0 + Sp_{c_0} \quad p_{c_3} = \gamma_3 + Sp_{c_0} \quad p_{c_2} = \gamma_2 + p_{c_3}$$

The additional parity bits p_a can be calculated as:

$$\mathrm{p_a^T} = \begin{bmatrix} \mathrm{C} & \mathrm{D} \end{bmatrix} \times \begin{bmatrix} \mathrm{c^T} \\ \mathrm{p_c^T} \end{bmatrix} \tag{7.6}$$

We have now solved the equation (7.1), and the codeword is in the systematic form of (7.2).

7.3 Rate Matching

The codeword-bits corresponding to the first $2 \times Z_c$ bits are said to always be punctured in 5G. This is due to their VNs having a high out-going degree d_v, making them highly likely to be recovered correctly. In the decoder, their respective LLR values are set to 0, as there is no certainty of what their values were before puncturing. $\lfloor \frac{K-K'}{Z_c} \rfloor \times Z_c$ of the $K - K'$ padded 0-bits are also punctured, and the LLR values of these punctured bits are set to $-\infty$, as their values are known to be 0. The corresponding column positions of the punctured bits are kept in H, as the first $2 \times Z_c$ columns correspond to information bits in a, and the 0-padding is part of the core submatrix of H.

The length of the rate-matched codeword x is $E = \lceil \frac{K}{R Z_c} \rceil \times Z_c$. Let E' be the number of bits from c after puncturing. The length of the codeword input to the LDPC decoder is then of length $E + E'$, and the PCM is of dimension $(E - E') \times K + (E - E')$.

As seen in Figure 7.2, the section of the matrix colored in green is used in the decoder, while the parts colored in grey are discarded. The sketched parts of the green submatrix are the punctured bits, which need to be recovered at the receiver. The puncturing of bits enables a bigger part of the PCM is utilized in the decoder as E bits are pre-determined to be transmitted, and the sender can include more parity bits in the transmitted codeword.

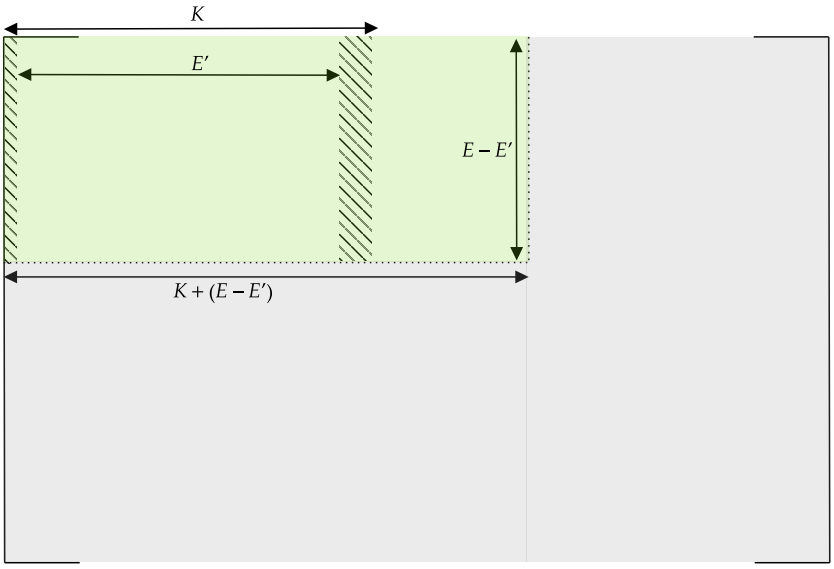

Figure 7.2: Figure representation of rate matching of codeword and scaling of H.

7.4 Decoding

In our work, we will analyze the Min-Sum algorithm, a variation of the Adapted Min-Sum (AMS) decoder, and the Improved Offset Min-Sum (IOMS) decoder. By the definition of H there is a significant amount of degree-1 VNs, which are difficult to correctly estimate in the Min-Sum decoder, leading to a significant deterioration of decoding performance. Both the IOMS and the AMS decoders suggest two different modifications to equation (5.5) in the Min-Sum decoder to mitigate the erroneous effect of the degree-1 VNs.

7.4.1 Adapted Min-Sum

The work of [19] presents an adaption of the Min-Sum decoder for the QC LDPC structure to improve the correction performance of the Min-Sum decoder called the Adapted Min-Sum (AMS) decoder. The core idea is to apply a different offset factor to the core and extension parts of the PCM when executing equation (5.5).

The beliefs sent from the CNs in IOMS are updated by equation (7.7).

$$\beta_{m,n} \approx (-1)^{|\mathcal{N}(n)|} \prod_{n' \in \mathcal{N}(n) \setminus m} \operatorname{sgn}(\alpha_{n',m}) \max\left(\min_{n' \in \mathcal{N}(n) \setminus m} (|\alpha_{n',m}|) - \eta^{(l)}, 0 \right) \tag{7.7}$$

where $\eta^{(l)}$ is said to be the offset factor. For Min-Sum decoding the offset factor is always $\eta^{(l)} = 0$, while for AMS $\eta^{(l)} = 0$ if $l \leq 4 \times Z_c$ else $\eta^{(l)} = 0$. $l \leq 4 \times Z_c$ comes from the fact that the extensions part of H starts from row $l > 4 \times Z_c$ by definition of H.

By Figure 5 in the work of [19], The Offset Min-Sum with offset factor $\eta = 0.2$ outperforms the AMS over most instances of SNR. We will in our implementation of the AMS decoder consider the offset factor to be $\eta^l = 0.2$ if $l < 4 \times Z_c$, else $\eta^l = 0$.

7.4.2 Improved Offset Min-Sum

To overcome the overestimation of degree-1 VNs, the IOMS normalize the beliefs sent from CN to VN by offsetting the product of a pre-specified normalizing factor μ and the minimum value $|\alpha_{n',m}|$ by μ. The beliefs sent from the CNs in IOMS are done by equation (7.8).

$$\beta_{m,n} \approx (-1)^{|\mathcal{N}(n)|} \prod_{n' \in \mathcal{N}(n) \setminus m} \operatorname{sgn}(\alpha_{n',m}) \max\left(\left(\mu \times \min_{n' \in \mathcal{N}(n) \setminus m} (|\alpha_{n',m}|) \right) - \eta, 0 \right) \tag{7.8}$$

Chapter 8

Performance of Polar Codes and LDPC Codes

8.1 Intro

In this chapter, we will detail the steps involved in assessing the Bit Error Rate (BER) and Block Error Rate (BLER) performance of polar codes and LDPC codes over different SNR values and transmission codeword lengths E. Through our implementation in *SageMath* of the encoding/decoding modules in 5G, we simulate the encoding of binary codewords, the behavior of noise appropriately for the different channels, and different decoding algorithms.

There are some limitations to the simulation compared to the actual employment as the segmentation of incoming packets is not implemented. Instead, we focus on CRC calculation, encoding of the incoming codeword without segmentation, rate matching, and simulation of noise to the transmitted codeword.

Monte Carlo
Simulation of the program is done with Monte Carlo simulation. We have run instances of different information lengths A and rates R over the noise corresponding to the channel capacity and lower. We stress that a rate higher than the channel capacity C is not feasible by Shannons limit. Therefore, when simulating the SNR values, the highest rate will always be equal or lower than the channel capacity. The number of iterations for each combination of A, R, and noise is $runs = 40000$.

Bit Error Rate
Bit Error Rate (BER) is measured by the total Hamming distance between every transmitted information sequence a and the output codeword $\hat{a}$ divided by the total amount of transmitted information bits.

Block Error Rate
Block Error Rate (BLER) is the number of times the output $\hat{a}$ did not match a divided by the total number of transmission $runs$.

The motivation for implementing all modules of the encoder and decoder in a simulation model is to extrapolate various data and produce a profile of the communication system. With high SNR values, we expect the error correction capabilities to be low, and we will get a high rate of erroneous codewords with high BER.

Error-Free Communication
We recall that each channel given an SNR value, has a theoretical maximum rate R at which information bits K occur in ratio to the total amount of transmitted bits E, on the form $R = \frac{K}{E}$, to achieve error free communication. The channel capacity is given in (3.3), (3.4) and (3.5) for the AWGN channel, BSC and BEC, respectively. Since we are testing for the rates $R = \frac{1}{2}$, we want our SNRs over the channels to reflect our choices of R.

For the BSC the maximum value p for $R = \frac{1}{2}$ is $p = 0.1$, as $C^{BSC} = 1 - H(0.9, 0.1) \approx 0.53$. For the BEC the maximum value for α for $R = \frac{1}{2}$ is

$\alpha = 0.5$, as $C^{BEC} = 1 - \alpha = 0.5$. When simulating for the AWGN channel, the convention is to run the system over SNR values starting for $SNR = 1$ and higher.

8.2 Components

When simulating for a specified rate, we include the encoding/decoding for different instances of a. The focus is on selecting the different parameters to achieve the desired rate R as well as possible.

8.2.1 Components of Polar Code

´

I_{IL}	0		1
	$A \leq 12$	$A \geq 20$	$1 \leq A \geq 140$
P	6	11	24
A_{min}	12		1
A_{max}	501		140
N_{min}	64		
N_{max}	1024		

Table 8.1: Bounds and parameters for polar codes when segmenting incoming packets is not utilized.

Again, the length N of the polar code codeword d is a power of two on the form $N = n^2$. When choosing the lowest value of N while satisfying $K = \frac{A+P}{R} \geq A_{min}$, we get $N_{min} = 64$ for both $I_{IL} = 0$ and $I_{IL} = 1$. As

$R_{N_{max}} = 1024$, the highest value n can take is $n_{max} = 10$ as $2^{10} = 1024$. We will need to choose A to achieve the desired rate R and getting the transmitted codeword length E as close to $N = 2^n$ as possible. While keeping the codeword length $N \leq R_{max}$ where $N = 2^{min(\lceil log_2 E \rceil, 10)}$.

When defining the rate, we will consider it to be $R = \frac{K}{E}$. Although the convention is not to view the CRC-bits as part of the information bits, they are part of the information set $\mathcal{I}$ of an (N, K) polar code, and in the case of $I_{IL} = 0$ they occupy the $K - A$ most reliable synthetic channels. As E is the transmitted codeword length, our goal is to minimize the difference of $N - E$ when the rate matching scheme is excluded. To achieve the practical rate $R = \frac{K}{E}$ as close to the rate of the polar code output $R = \frac{K}{N}$ as possible, we set A to be 21, giving us $R = \frac{K}{E} = \frac{32}{64} = \frac{1}{2}$.

Polar Codes with $I_{IL} = 1$
When activating the interleaver flag $I_{IL} = 1$ the generator polynomial $g_{24}(x)$ is always used. To maintain $N = 64$ and $R = \frac{K}{E} = \frac{1}{2}$, A was set to 8, giving us $K = 32$.

Polar Codes with Rate Matching
When profiling the effect of rate matching on the decoding performance, we evaluate the codeword over a constant $N = 128$ and vary the amount of rate-matched bits $U = [0, 8, 24, 40, 56, 64]$, while keeping the ratio $R = \frac{K}{E}$ constant by having $K = [64, 60, 52, 44, 36, 32]$ as U increase, all with a list size of $L = 8$. E was chosen by $N - U$, and $K = E \times R$.

U	(K, E)
0	(64, 128)
8	(60, 120)
24	(52, 104)
40	(44, 88)
56	(36, 72)
64	(32, 64)

Table 8.2: The table shows how the relation of K and E was kept constant while U increased for rate-matched polar codes.

8.2.2 Components of LDPC Codes

LDPC codes in 5G are used for transmission of longer information sequences. To compare polar codes and QC LDPC codes fairly, we will mimic the information lengths A with rate R of polar codes in QC LDPC, and then simulate for larger values of A. We define the rate for LDPC code to be $R = \frac{K}{E}$.

Validity Check for $\hat{d}$

In our work, we propose a slight modification to the validity check $H \times \hat{d} = \vec{0}$ of the hard decision estimate $\hat{d}$. By equation (7.2) the QC LDPC encoded codeword d is in systematic form $\mathrm{d} = \begin{bmatrix} \mathrm{c} & \mathrm{p_c} & \mathrm{p_a} \end{bmatrix}$. If $\hat{d}$ were to contain erroneous bits in $\begin{bmatrix} \mathrm{p_a} \end{bmatrix}$ only, the validity check $H \times \hat{d}^T = \vec{0}$ would not hold. In the worst case, the erroneous bits of $\hat{\mathrm{p}}_\mathrm{a}$ will cause the bits in $\hat{\mathrm{d}} = \begin{bmatrix} \hat{\mathrm{c}} & \hat{\mathrm{p}}_\mathrm{c} \end{bmatrix}$ to become erroneous as well, making the decoder declare a decoding failure when number of iterations equals $Iter_{max}$.

We propose to perform validity check of $\hat{d}$ by equation (8.1).

$$H_{core} \times \hat{d}_{core} = \begin{bmatrix} \mathrm{A} & \mathrm{B} \end{bmatrix} \begin{bmatrix} \hat{\mathrm{c}}^\mathrm{T} \\ \hat{\mathrm{p}}_\mathrm{c}^\mathrm{T} \end{bmatrix} = \vec{0} \tag{8.1}$$

In order for the decoder to terminate, we also impose that $\hat{c}$ needs to generate a valid CRC-check.

8.3 Performance of Polar Codes

In this chapter, we will first compare the performance of the SC decoder, SCL decoder, and the CA SCL decoder over the three channels. Later, we will review the effect of shortening over polar codes. All simulations were performed with list size $L = 8$.

Decoding Failures
When each leaf node is assigned a hard decision, the decoder terminates with L-candidate codewords sorted from the lowest path-metric to the highest path-metric. CRC-check is then performed sequentially on the candidate codewords. The first codeword satisfying the check-sum $H_{CRC} \times \hat{d}^T = \vec{0}$ is the final candidate codeword $\hat{a}$ for the information-vector a, where $H_{CRC} = \left[C^T I_{P\times P}\right]$. When none of the codewords in the list of candidates passes the CRC-check, the codeword of the lowest path-metric is selected to compute the BER and BLER, even though another candidate codeword could potentially have a lower hamming distance to a. In the case of an early termination in the CA SCL decoder, the total amount of erroneous bits are updated by taking the additive inverse of a.

8.3.1 SC, SCL, and CA SCL

Figures 8.6, 8.2, and 8.3 all show that the CA SCL decoder has the better performance for most instances of noise over all three channels. The high BER values for the CA SCL decoder is attributed to our method of

penalization by updating the total number of erroneous bits to be the additive inverse of a when the decoder terminates early. For this reason it is most fair to compare the CR SCL to the CA SCL decoder, and the SC decoder to the SCL decoder when reading the plot for BER performance. When reading the plot for the BLER performance it is fair to compare all decoders, since $\hat{a}$ is either accpeted or rejected at the receiver in practical applications.

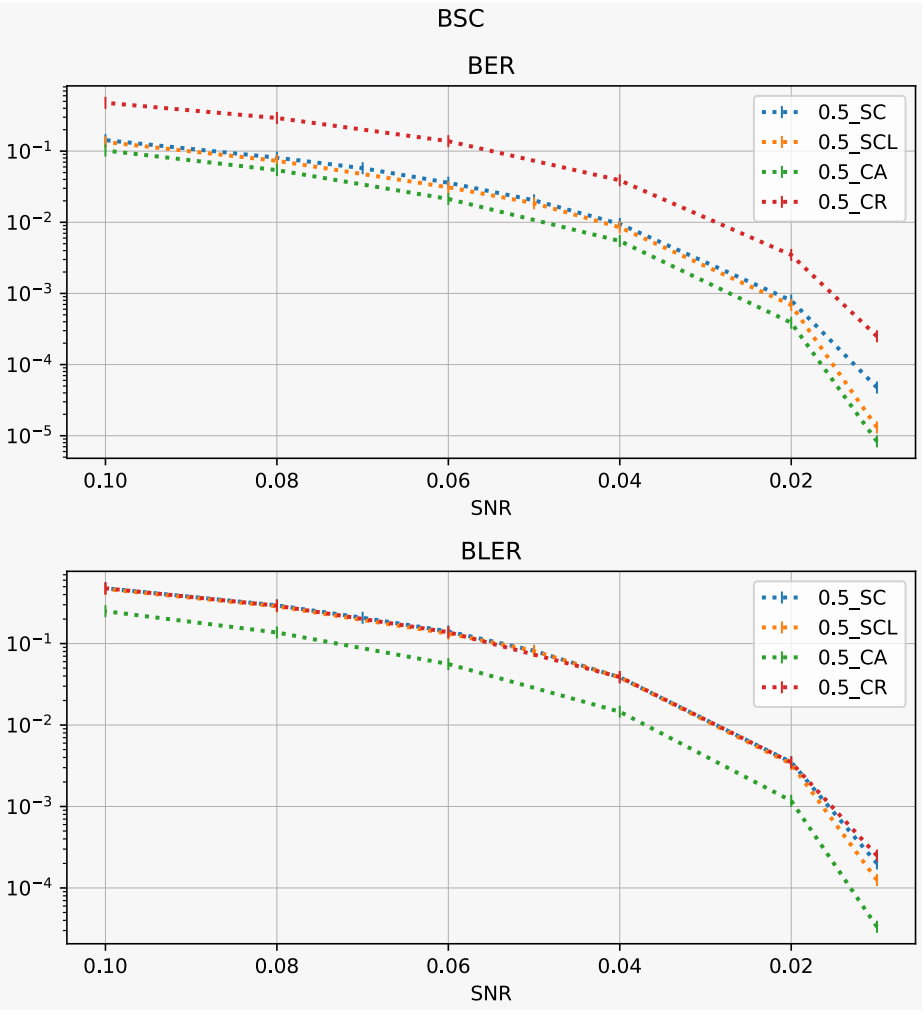

Figure 8.1: Results for the SC decoder, SCL decoder, and CA SCL decoder over the BSC channel.

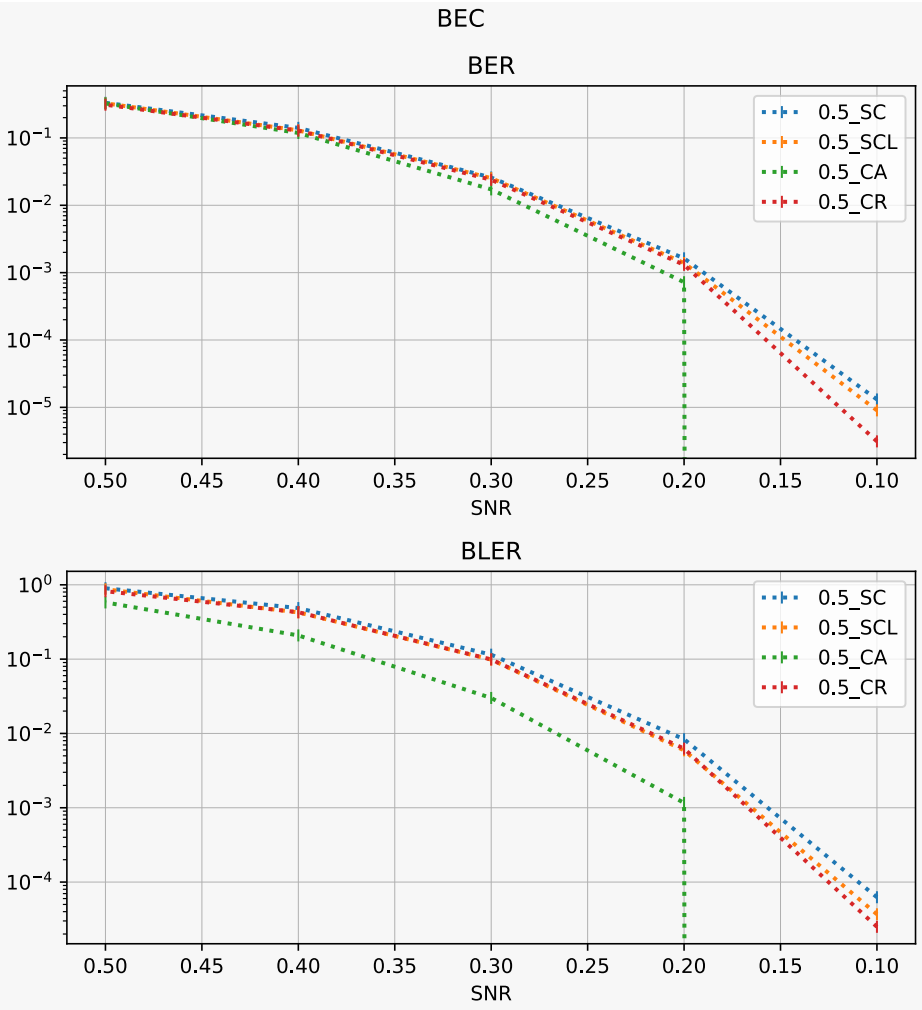

Figure 8.2: Results for SC, SCL, and CA SCL decoders over the BEC channel.

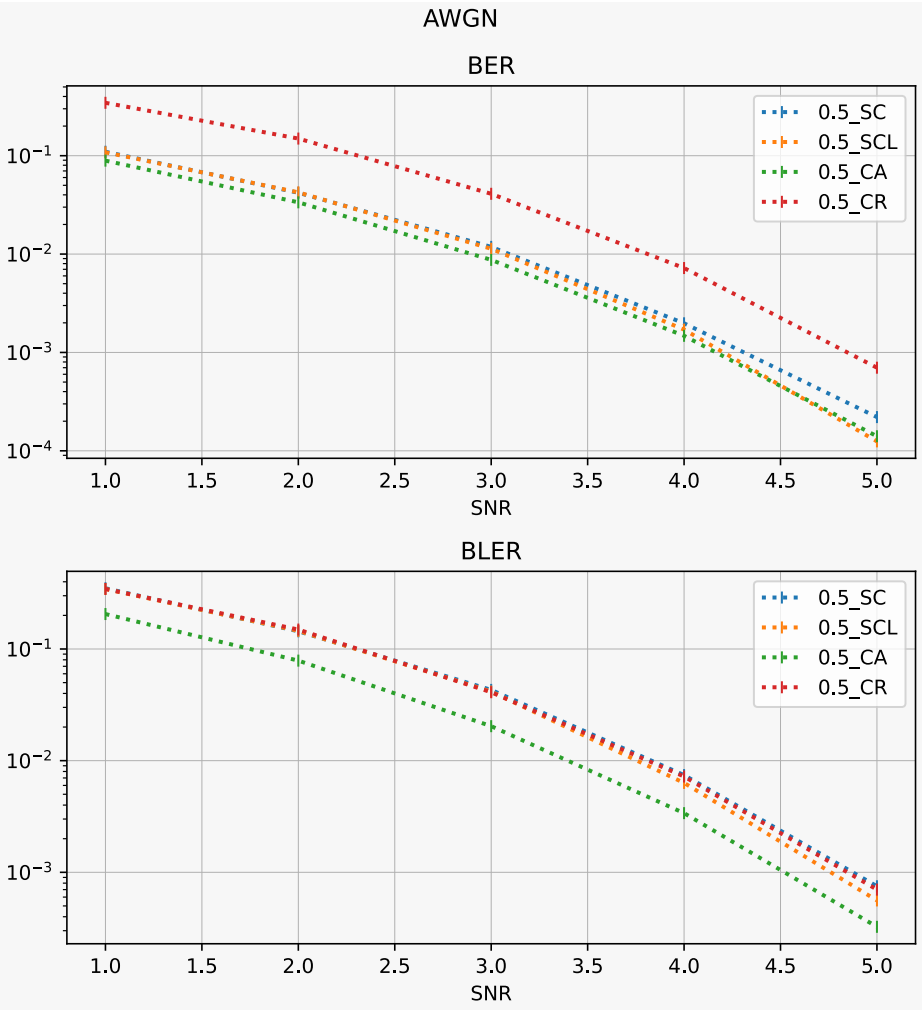

Figure 8.3: Results for SC, SCL, and CA SCL decoders over the AWGN channel.

From the displayed graphs, the CA SCL performs better in terms of BLER than the other decoders for polar codes. Both the CA SCL and the CR SCL decoder have higher computational complexity as the correctness for every CRC-bit is evaluated by vector-matrix multiplication of $H' \times \hat{c}^T$ and $\hat{c} \times G'$, respectively. For the CR SCL decoder L vector-matrix checks are

performed when evaluating the correctness of its candidate codewords, while the CA SCL decoder performs $2L$ vector-matrix checks. With the extra computational complexity, the BLER performance should ideally be noticeably better than its SC and SCL counterparts to offset the higher computational complexity. The CA SCL decoder, overall have better performance in terms of BLER. Most notable is its achievement of error-free communication for the BEC when $\alpha < 0.2$.

The performance of the SCL decoder does not clearly stand out from the SC decoder, if one wants to increase the error correction performance of the SCL decoder one could increase the list-size to $L > 8$.

8.3.2 Rate Matched Decoder

Originally, the values of U were chosen as to have six instances of evenly distributed integers between $U_{min} = 0$ and $U_{max} = 64$, on the form $U = [0, 8, 24, 40, 56, 64]$. After simulating these values of U, a trend where $|U|$ were to be a power of two seemed to have better performance than values were $|U|$ was not a power of two. We then included simulations were $U = 2^5 = 32$. The figures below show simulation results for instances of (R, U).

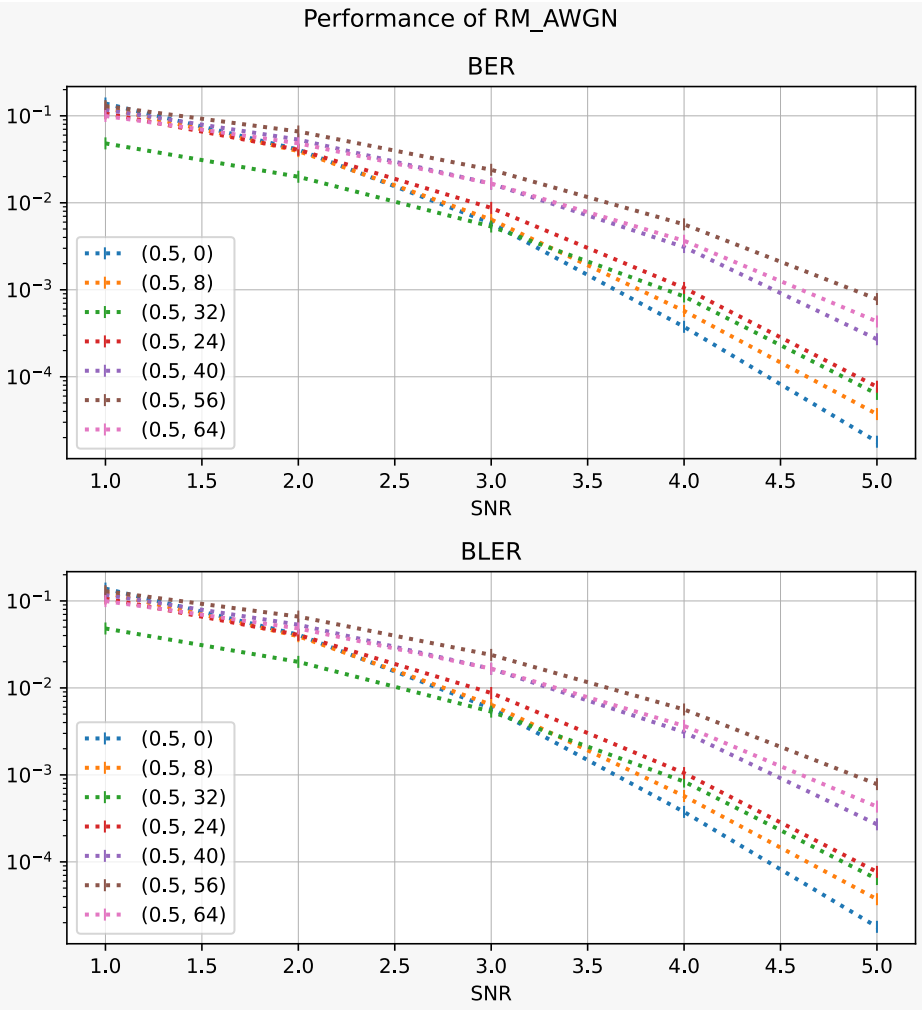

Figure 8.4: Results of rate matching of polar codes over the AWGN channel.

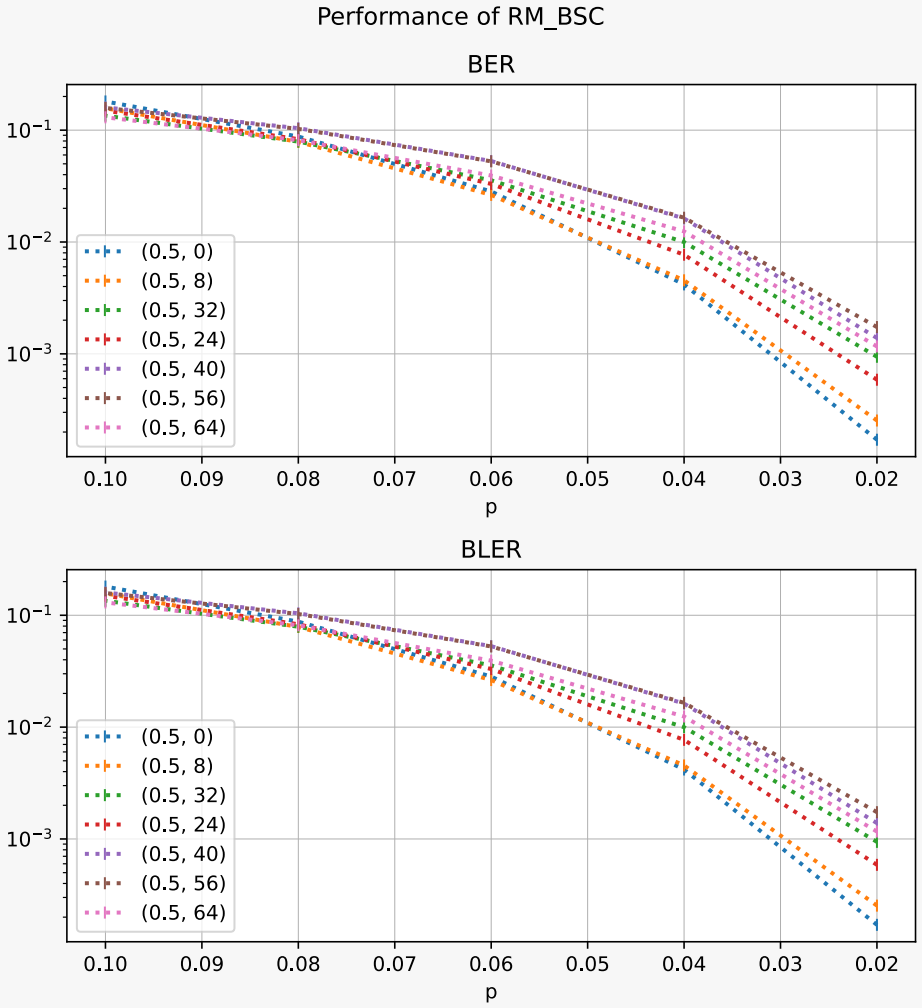

Figure 8.5: Results of rate matching of polar codes over the BSC.

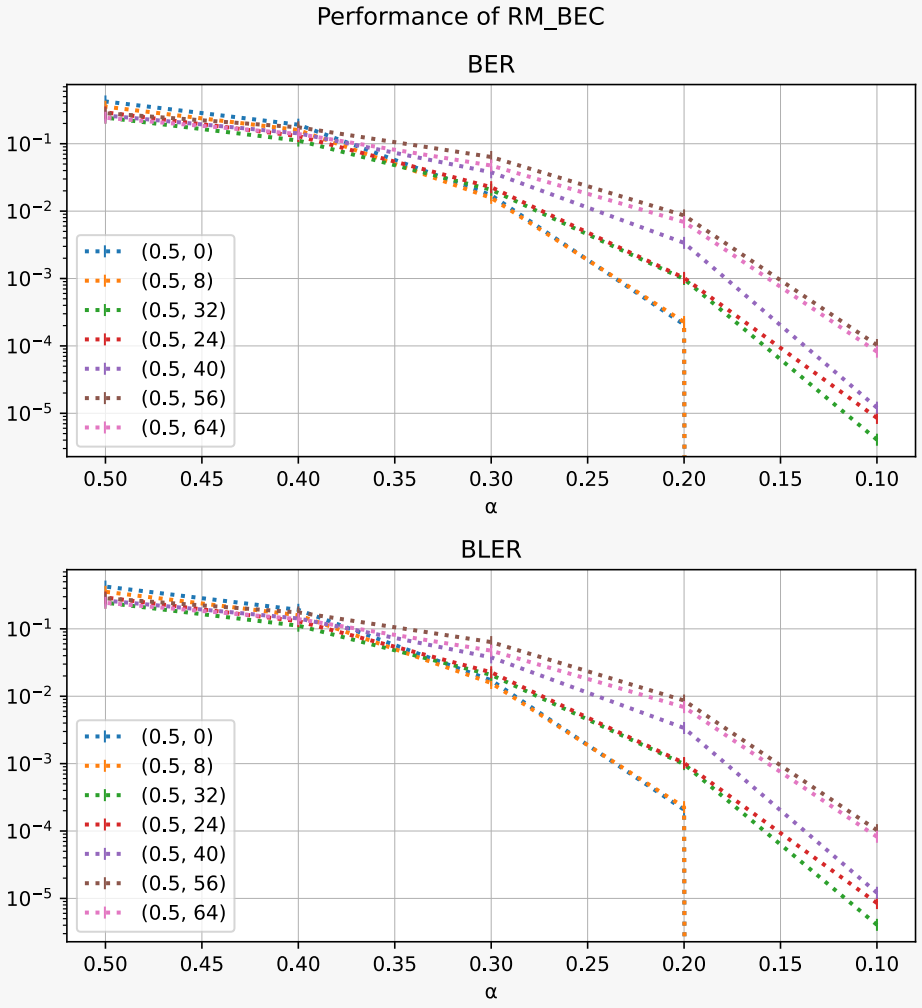

Figure 8.6: Results of rate matching of polar codes over the BEC.

From our results, the rate matching scheme slightly favors higher values of U that are powers of two over every channel with higher values of SNR. Then, as the SNR decrease the rate matching scheme tend to favor the channels with lower values of U. For the BEC we see that it no longer achieves error-free communication for higher values of U for $\alpha < 0.20$. The BSC is the only channel in which $U = 24$ performs better than $U = 32$

over higher values of SNR. Over the AWGN channel, the performance of $U = 32$ has a significant gain over lower values of SNR compared to the other values of U, which can be attributed to a good balance between inserting LLR values of $-\infty$ and not using the information set $\mathcal{I}$ over synthetic channels with lower reliability.

In general, the insertion of LLR values of $-\infty$ seem to counteract wrongful belief propagation for the SCL decoder. Especially for higher levels of noise, when each information leaf-node are more likely to be incorrect. As the noise over each channel decreases, each information leaf-node have a higher belief of their own correctness. In which, the insertion of $-\infty$ values does not have as strong counteracting of wrongful belief propagation, and the fact that the bit indices for the information nodes are transmitted over synthetic channels of lower reliabilities are exposed in their performance.

8.4 Performance of LDPC Codes

In this chapter, we will first compare the performance of validity check when performing $H_{core} \times \hat{d}_{core}$ with CRC-check, and when performing $H \times \hat{d}$. Later, we will compare the results of the Min-Sum decoder, AMS decoder, and the IOMS decoder over the three channels.

Figure 8.7 shows the BER and BLER performance of H_e and H_c. The performance of H_c does not clearly show until $SNR = 5$, our interpretation of the results is that when the overall probability of error is low, the probability that all erroneous bits occurring only in $\left[\ \mathrm{p_a} \ \right]$ is greater.
Table 8.3 shows the average amount of iterations needed for each decoder when the decoder outputs the correct codeword.

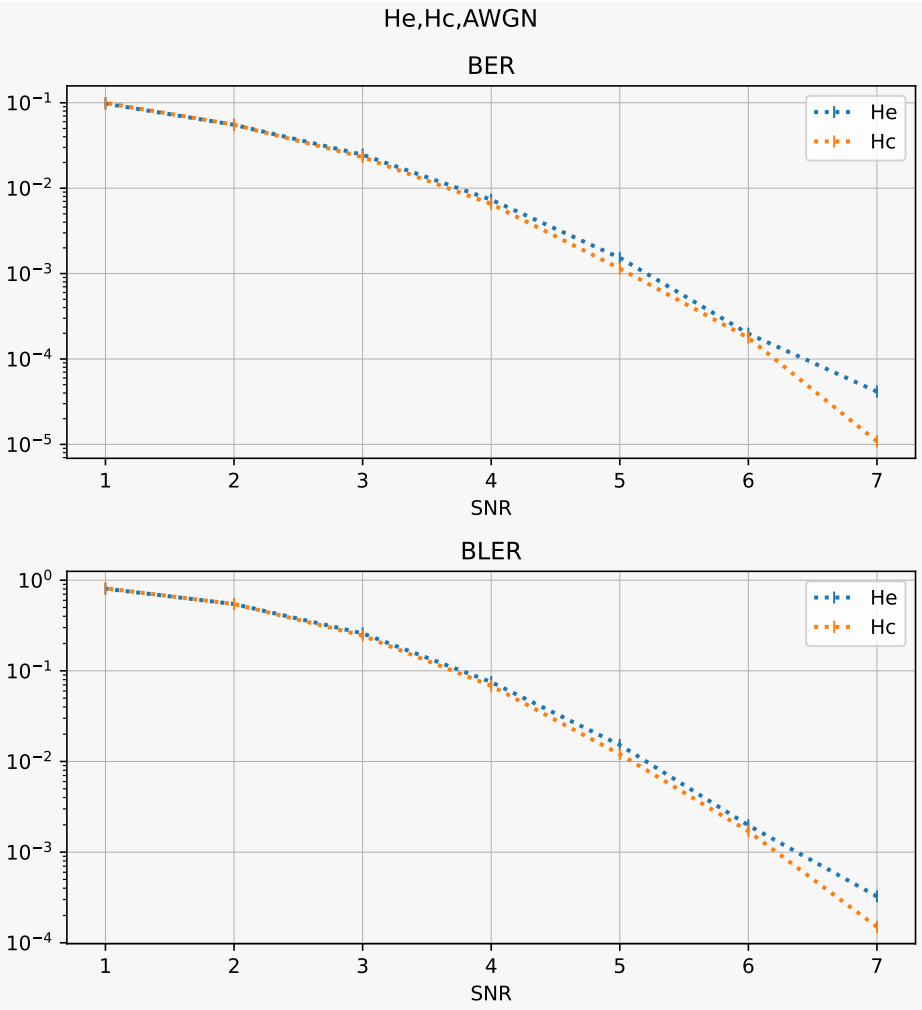

Figure 8.7: Results comparing $H_{core} \times \hat{d}_{core}$, H_c for short, with CRC-check versus $H \times \hat{d}$, H_e for short, over the AWGN channel.

SNR	7	6	5	4	3	2	1
H_e	1.14	1.44	2.16	3.34	4.93	6.63	8.66
H_c	1.06	1.28	1.86	2.96	4.50	6.18	7.6

Table 8.3: The average amount of iterations when the decoder terminated with a valid codeword over the AWGN channel.

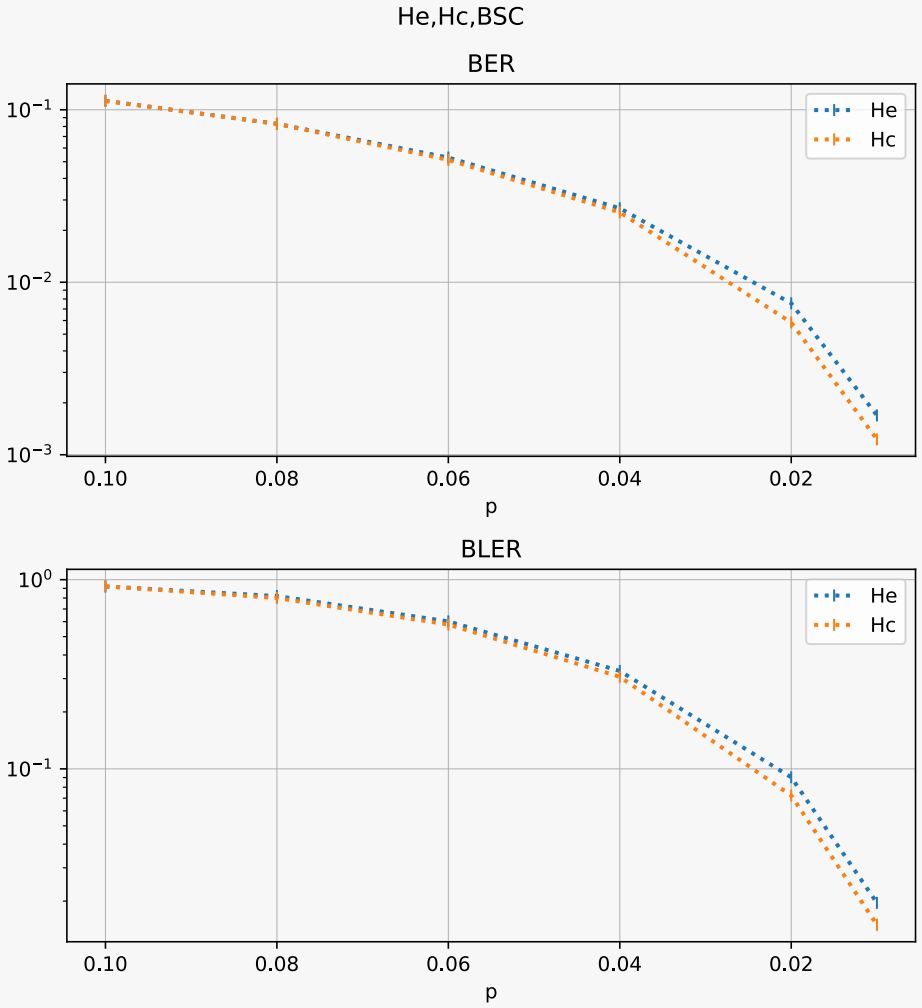

Figure 8.8: Results comparing $H_{core} \times \hat{d}_{core}$ with CRC-check versus $H \times \hat{d}$ over the BSC.

SNR	0.01	0.02	0.04	0.06	0.08	0.1
H_e	1.65	2.79	5.23	7.12	8.76	9.45
H_c	1.33	2.21	4.58	6.54	8.20	8.92

Table 8.4: The average amount of iterations when the decoder terminated with a valid codeword over the BSC.

Figure 8.9 compares the BER and the BLER performance for H_c and H_e over the BEC channel. Their overall performance match until $\alpha = 0.25$. The drop for graph H_c after $\alpha = 0.25$ represents that there were no more erroneous outputs from the decoder for values lower than $\alpha = 0.25$, while for graph H_e there were no more erroneous outputs from the decoder for values lower than $\alpha = 0.2$.

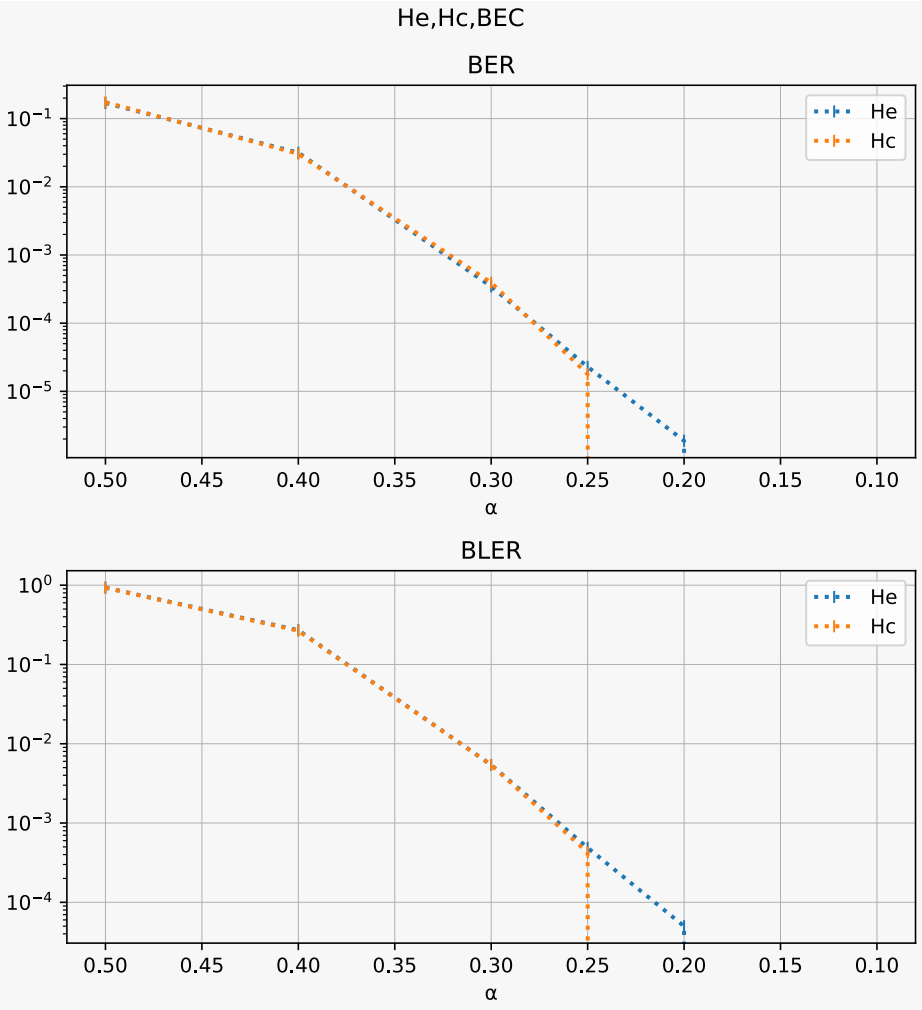

Figure 8.9: Results comparing $H_{core} \times \hat{d}_{core}$ with CRC-check versus $H \times \hat{d}$ over the BEC.

From the results of Figures 8.7, 8.8, and 8.9 we will run the Min-Sum, IOMS, and the AMS decoders and perform $H_{core} \times \hat{d}_{core}$ with CRC-check when checking the validity of $\hat{d}$ in the LDPC decoder.

Graphs from Figure 8.10, 8.11, and 8.12 displays the performance of the Min-Sum, IOMS and the AMS decoders. We only ran the tanh update function (5.4) over the BEC as the variables nodes are either correct, or contain no information. Applying an offset to variable nodes which are certain to be correct could deteriorate the performance of the decoder.

The offset factor μ and the normalized factor η for the AMS decoder and the IOMS decoder are gathered from the work of [18], and are set to 0.4 and 0.95, respectively. The offset factor μ and the normalized factor η for the Min-Sum decoder are set to 0 and 1, respectively.

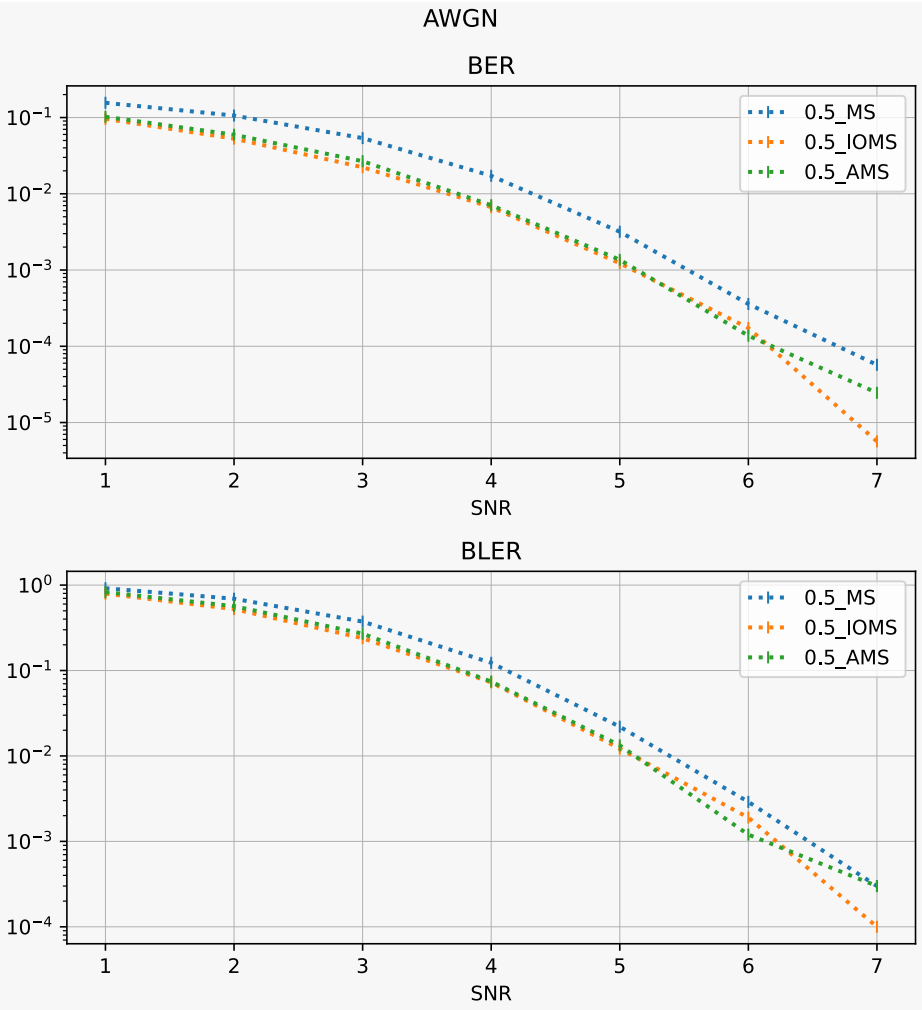

Figure 8.10: Results for the Min-Sum, AMS, and the IOMS decoders over the AWGN channel.

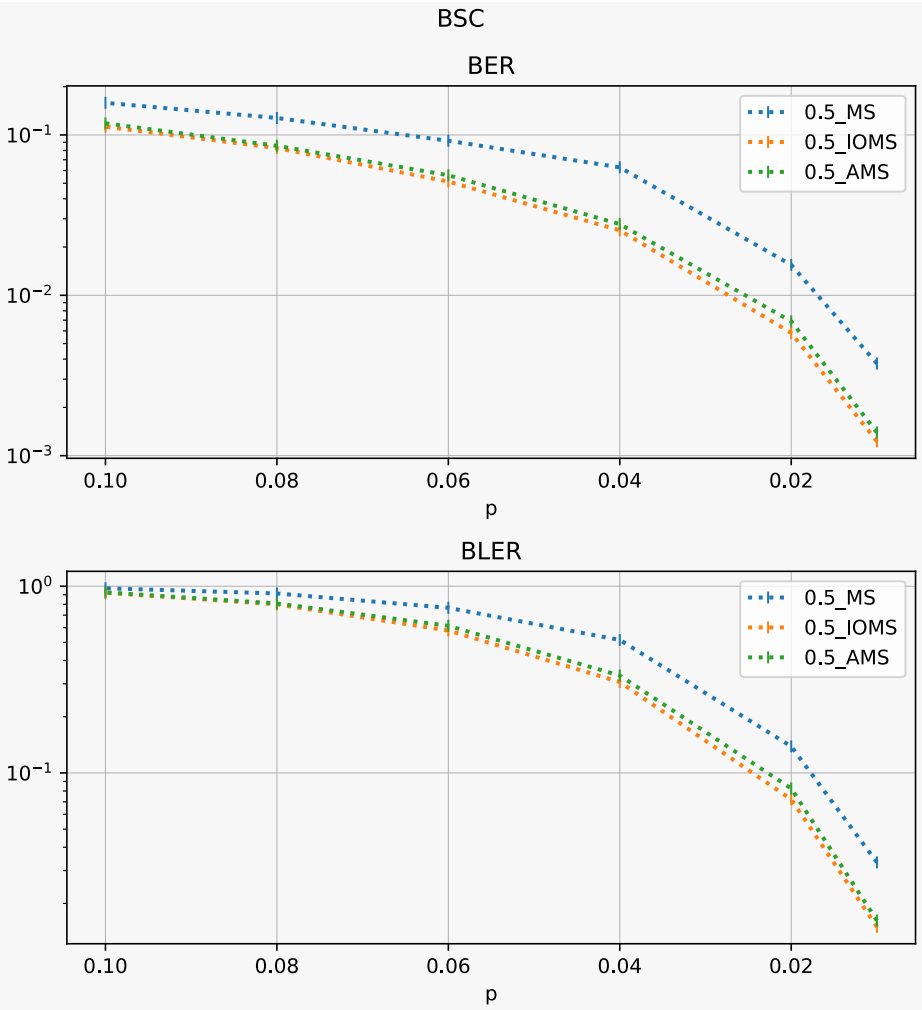

Figure 8.11: Results for the Min-Sum, AMS, and the IOMS decoders over the BSC channel.

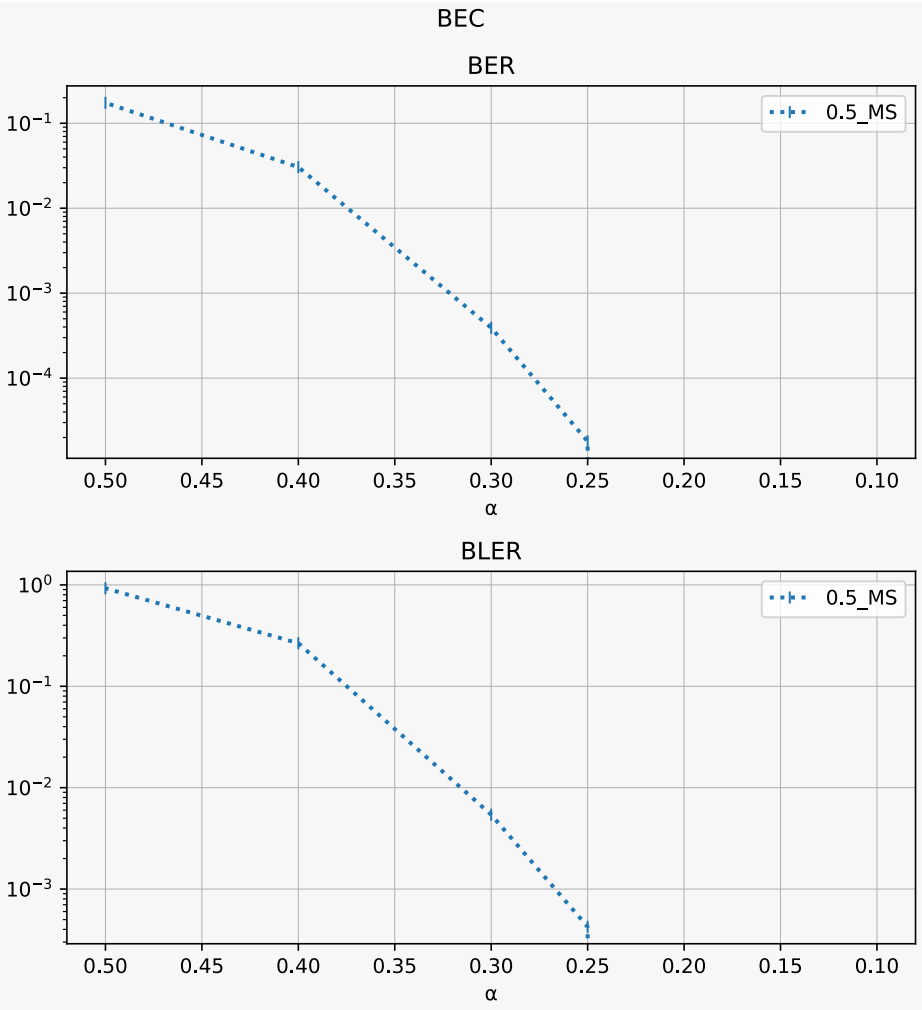

Figure 8.12: Results for the Min-Sum BEC channel.

From the displayed graphs, both the IOMS and the AMS decoders shown an improvement to the Min-Sum decoder, where the IOMS decoder has a slight advantage over the AMS decoder.

Chapter 9

Conclusions

In this thesis, we discussed the development of LDPC codes and polar codes and detailed the CRC calculation, the mother encoder, and the rate matching scheme for both structures, with the aim of explaining the material in such a way that is more inviting to new readers in the field of reliable digital communication.

Through our research of independent papers, we implemented and profiled the originally proposed approximation for the belief propagation function, called the Min-Sum, to adapted versions of the Min-Sum decoder to achieve better error correction performance of the QC LDPC code. The adapted versions we included in this thesis were the IOMS decoder and acrshortams decoder. We suggested a modification for the validity check of $\hat{d}$, which utilized the core parts of both H and $\hat{d}$ accompanied with a CRC-check before terminating the decoding algorithm.

For polar codes, we implemented and profiled the performance of the SC decoder, the SCL decoder, and two versions of a CA SCL decoder. The main objective for the CA SCL decoder is to interleave the CRC-bits to allow for early termination of the SCL decoder. One variant of the CA SCL

decoder, named CR SCL in the work of [13], was based on our interpretation by the work of [13] and [6]. The other variant of the CA SCL decoder was by our suggestion, and we simply kept the CA SCL name. Though the CA SCL decoder is designed to early terminate the decoder, if none of the candidate codewords will yield the correct codeword, one wants as few early terminations as possible. Fewer early terminations are attributed to a decoder which can better keep the correct candidate codeword throughout the decoding process.

Bibliography

[1] 3GPP. Etsi ts 138 212 v15.12.0 (2021-10), 2021. URL https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.12.00_60/ts_138212v151200p.pdf.

[2] Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. 2009. doi: 10.1109/TIT.2009.2021379.

[3] Jung Hyun Bae, Ahmed Abotabl, Hsien-Ping Lin, Kee-Bong Song, and Jungwon Lee. An overview of channel coding for 5G NR cellular communications. 2019. doi: https://doi.org/10.1017/ATSIP.2019.10.

[4] Valerio Bigolio, Frédéric Gabry, and Ingmar Land. Low-complexity puncturing and shortening of polar codes. 2017. doi: 10.1109/WCNCW.2017.7919040.

[5] Valerio Bioglio, Carlo Condo, and Ingmar Land. Design of polar codes in 5g new radio. 2020. doi: 10.1109/COMST.2020.2967127.

[6] Zeynep B. Kaykac Egilmez, Luping Xiang, and Robert G. Maunder. The development, operation and performance of the 5g polar codes. 2020. doi: 10.1109/COMST.2019.2960746.

[7] R. G. Gallager. Low-density parity-check codes. 1962. doi: 10.1109/TIT.1962.1057683.

[8] R. W. Hamming. Error detecting and error correcting codes. 1950. doi: 10.1002/j.1538-7305.1950.tb00463.x.

[9] Huang-Chang Lee, Yu-Shen Pao, Cheng-Yi Chi, Hsin-Yu Lee, and Yeong-Luh Ueng. An early termination scheme for successive cancellation list decoding of polar codes. 2020. doi: 10.1109/ICASSP40776.2020.9053566.

[10] David J.C MacKay. *Information Theory, Inference, and Learning Algorithms*. 2003.

[11] Todd K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. 2020.

[12] Vladimit L. Petrovic, Dragomir M. El Mezeni, and Andreja Radosevic. Flexible 5g new radio ldpc encoder optimized for high hardware usage efficiency. 2021. doi: https://doi.org/10.3390/electronics10091106.

[13] Charles Pillet, Valerio Bioglio, and Carlo Condo. On list decoding of 5g-nr polar codes. 2020. doi: 10.1109/WCNC45663.2020.9120686.

[14] C. E. SHANNON. A mathematical theory of communication. 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[15] Ido Tal and Alexander Vardy. List decoding of polar codes. 2015. doi: 10.1109/TIT.2015.2410251.

[16] Techopedia. Cyclic redundancy check (crc), 2020. URL `https://www.techopedia.com/definition/1793/cyclic-redundancy-check-crc`.

[17] Andrea Tenti. Lecture notes for inf244, graph-based inference, networks and coding theory, 2021. URL `https://git.app.uib.no/Andrea.Tenti/inf244-2021-material`.

[18] Bich Ngoc Tran-Thi, Thien Truong Nguyen-Ly, Hoa Nguyen Hong, and Trang Hoang. An improved offset min-sum ldpc decoding algorithm for 5g new radio. 2021. doi: 10.1109/ISEE51682.2021.9418782.

[19] Khoa Le Trung, Fakhreddine Ghaffari, and David Declercq. An adaptation of min-sum decoder for 5g low-density parity-check codes. doi: 10.1109/ISCAS.2019.8702344.