Polar Codes
Primary Concepts and Practical Decoding Algorithms

Member 1: Himanshu Sharma [1610110149]
Member 2: Dhruv Mehra [1610110119]
Member 3: Ram Charan [1610110093]

Under the guidance of Prof. Vijay K. Chakka

Dept. of Electrical Engineering

(Midterm Evaluation Report)

Shiv Nadar University
NH91, Tehsil Dadri, Greater Noida, Uttar Pradesh 201314

1 Introduction

Polar Codes are an emerging class of error correcting codes. Error correcting codes are used to detect the errors that arise from the noisy channel through which the data is transmitted and then appropriately removing its effect. Many such codes exist already in the field of *Information Theory*. Hamming codes are one of the popular classes of error correcting codes. Polar codes are new and emerging class of these types of codes.

There is something called channel coding which basically means that we code a stream of data at the transmitter end by some set of mathematical operations and then at the receiver side the reverse operations are done to get back the orignal data stream. Together these operations are called *encoding* and *decoding* respectively. The basic requirement when designing a new error correcting code is to make it less complex, because if the code complexity is increased, then the power consumption, memory consumption, computation power, etc increase at a fast rate. Polar codes have a complexity of $\mathcal{O}(n \log n)$.

There are two types of channel coding schemes, these are, *block coding* and *convolutional codes*. Block codes work on a block of data/bits with fixed size and apply the manipulation to this block at the transmitter and receiver. Reed-Solomon codes, commonly used on the hard disks of computers, are one example of this type of code. Convolutional codes, on the other hand, work on streams of data with more arbitrary numbers of data/bits. These codes apply a sliding window method that provides a substantial decoding benefit. Simple Viterbi codes are an example of this type of code.

Polar codes come under the category of block codes. Their approach is remarkably different. In many ways, polar codes represent techniques similar to that performed in the *fast Fourier transform*. There are two processes involved in this technique, first, *channel combining* and *channel splitting*. Channel combining is similar to what happens in OFDM. In channel combining, the encoder, combines group of bits and assigns that group to a particular channel. The channel splitting that follows performs an implicit transformation operation (analogous to frequency domain to time domain as performed by an IFFT operation), translating these bit/symbol combinations into decoder-ready, time domain vectors. The real magic in polar codes lies in the clever bit manipulations and mappings to the channels at the encoder. The original technique, in combination with channel splitting, and successive-cancellation decoding, is shown to essentially convert a block of bits, and their associated channels between the encoder and decoder, into a polarized bit stream at the receiver. That is a received bit and its associated channel ends up being either a "good channel" or "bad channel" pole/category. It has been mathematically proven that as the size of the bit block increases, the received bit stream polarizes in a way that the number of "good channels" approaches Shannon capacity. This phenomenon is what gives polar codes their name and makes them the first and only explicitly proven capacity-achieving practical channel codes ever conceived.

1.1 Shannon's Channel Coding Theorem

The Shannon's channel coding theorem states that any channel has certain capacity upto which bits can be transmitted through it without being affected too much by noise. The maximum capacity is called *Shannon's limit*. The capacity C in general, is given by

$$C = I(X;Y) = \log_2 \frac{P(y_j|x_i)}{P(y_j)} \tag{1}$$

where, X and Y denotes two random variables which acquire a value x_i and y_j at random and P denotes probability. Now, the entropy of the channel is given by

$$H = \sum_{i=1}^{m} P_i \log_2 \left(\frac{1}{P_i} \right) \tag{2}$$

Note that m denotes the number of symbols in a transmission. If an event is sure, then its entropy is 0, becuase for a sure event $P = 1$ and thus making $H = 0$. The maximum entropy occurs when all the symbols are equally probable, i.e., $P_i = \frac{1}{m}$.

$$H = \sum_{i=1}^{m} \frac{1}{m} \log_2(m) = \log_2(m) = H_{max}$$

1.2 Asymptotic Equipartition Property

This property states that in a long turn, random sequences which are *iid* will tend to have a uniform distribution. Consider a biased coin having *head* probability of 0.7 and *tail* probability of 0.3. In a trial of 10 tosses, it is likely that out of 10, only 5 coins come out to be head. But if the trials are 10,000, then it is obvious that heads will occur mostly 7000 times. Mathematically,

$$\lim_{N\to\infty} \frac{n(H)}{N} = p$$

where p is the probability of getting heads H and N is the total number of trials. To utilize the full channel capacity, many error correcting codes have been used like LDPC and turbo codes.

1.3 Channel Polarization

At first some independent channels are separated out by the order of their reliability. Separating channels based on their reliability is important because it determines which channel would be suitable for transmitting the information. Applying this algorithm recursively, which is used to separate them out on the basis of their reliability, we find that at the final stage all the "good" and "bad" channels are separated out. This process of separating out the channels as useless and useful is called *channel polarization.*

1.4 The Q Function

The Q function is something that will be widely used in this paper. Q function is defined as follows

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz \tag{3}$$

The graph of the Q function is shown below.

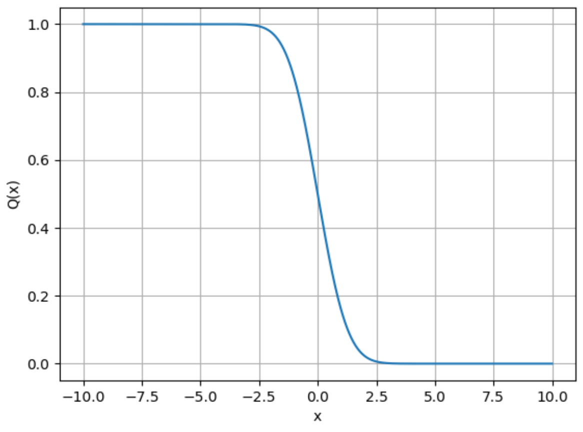


Figure 1: The Q Function

The Q function is also expressible as shown below.

$$Q(x) = \frac{1}{2}\text{erfc}\frac{x}{\sqrt{2}} \tag{4}$$

From equation (2), it is clear that for a random variable X with mean μ and standard deviation σ following a normal distribution, when standardized, will form a yet another normal distribution Z such that $Z = \dfrac{X-\mu}{\sigma}$. Q function can also be defined now in terms of probability of the random variable Z.

$$Q(x) = P(X > x) \tag{5}$$

An important property of Q function is $Q(x) = 1 - Q(-x)$.

1.5 Bit Error Rate

Whenever we transmit a signal over some communication channel, the original signal is modified by the noise. The bit error rate or BER for short, is the probability that a received symbol $\hat{s}$ is not equal to the original transmitted symbol s. Mathematically saying, $BER = P(s \neq \hat{s})$. Consider the case of BPSK. If the logic 1 is denoted by $\sqrt{E_b}$ and logic 0 is denoted by $-\sqrt{E_b}$ where E_b is the energy per bit of the symbol, then BER is given as

$$BER = Q\left(\sqrt{\frac{2RE_b}{N_0}}\right) \tag{6}$$

where, R is the rate of transmission. For uncoded case, $R = 1$. If we plot $\log(BER)$ vs E_b/N_o ratio (in dB), then the following curve is generated,

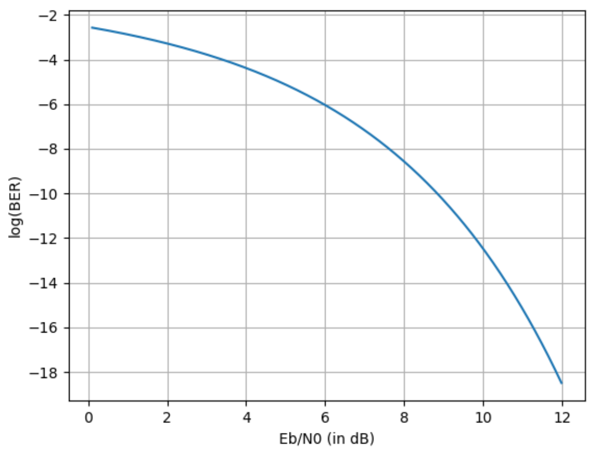
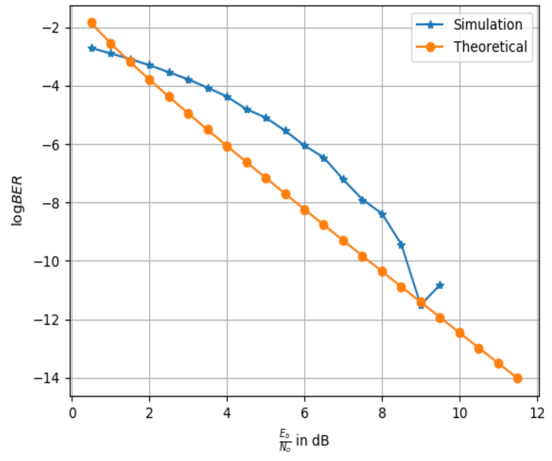


Figure 2: $\log(BER)$ vs. E_b/N_o (in dB)

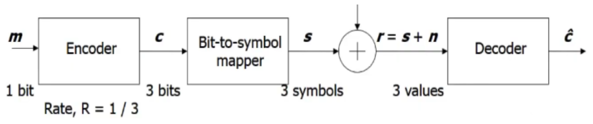
Below, a graph has been shown just to compare the theoretical and simulated BER results.



Later on, in this paper, it will be shown that the theoretical and the simulated BER are same.

1.6 $n = 3$ Repetition Codes

Using codes is beneficial in transmission because they reduce the bandwidth consumption. Consider the block diagram given below.



This block diagram corresponds to the binary transmission only, i.e., the input to the encoder is either 1 or 0. The encoder maps 1 to 111 and 0 to 000 and thus changing the rate to $R = 1/3$. Since the bits are repeated, it is called the repetition code. The bits to symbol mapper maps these symbols to $[1, 1, 1]$ for 000 and $[-1, -1, -1]$ for 111. The signals are then transmitted in an AWGN medium. The BER, in general for rate R is given by $BER = Q\left(\sqrt{\frac{1}{R\sigma^2}}\right)$. Hence, here, in this case, BER is $Q\left(\sqrt{\frac{3}{\sigma^2}}\right)$.

Generally, we always try to make E_b/N_o ratio to be same in both coded and uncoded case and hence we can do the following;

$$\frac{1}{2R\sigma^2_{coded}} = \frac{1}{2\sigma^2_{uncoded}}$$

Since, $R < 1$, we get, $\sigma_{coded} > \sigma_{uncoded}$. That is, noise is more probable in coded case. It is also evident from the curves shown below.

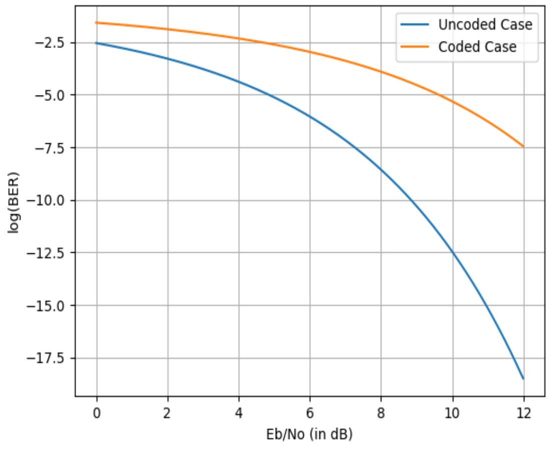


Figure 3: log BER vs E_b/N_o ratio for Uncoded and Coded case

It is evident from the curves that for the same E_b/N_o ratio, the coded case gives higer bit error rate. This is the disadvantage of the coding schemes. They introduce more BER but reduce bandwidth consumption.

Now after passing through AWGN, if the vector $[r_1, r_2, r_3]$ is received, then the symbol $\hat{c}$ can be found out by comparing the distance from the ideal symbols;

$$|r - [1, 1, 1]| < |r - [-1, -1, -1]| \implies \hat{c} = 000$$

On further simplification, it states that if

$$r_1 + r_2 + r_3 > 0 \implies \hat{c} = 000$$

1.7 $(7, 4)$ Hamming Code

A $(7, 4)$ hamming code is something similar to repetition code but not exactly like it. In this type of coding, 4 bits of symbol is coded as a 7 bits symbol. The extra 3 bits that are added were decided after extensive research. After the 7 bits are formed, the symbols are transmitted over AWGN channel and the vector r is received. At the side of the receiver, either a *hard decision* or a *soft decision* is made.

Hard Decision

In this form of decision making process, if $r_i > 0 \implies b_i = 0$ and vice versa. After the b vector is formed, the hamming distance is calculated from the original mapping. The original mapping which has the least hamming distance, that mapping is decided as the output.

Soft Decision or ML Decision

In this form of decision making process, the norm is taken, just as we did in the repetition code case, and then the final symbol is decided.

1.8 Linear Block Codes

Consider a message stream of k bits and an encoder which codes them into n bits. We define a generator matrix G such that it generates a codeword matrix C as follows;

$$C = MG \tag{7}$$

where, $G = [IP]$, P is the parity matrix and M is the matrix for message bits. Also, the inverse relation of this equations stands as a parity check.

$$[P^T I]C^T = [0] \tag{8}$$

Note that all the additions are modulo 2.

2 Polar Transform

Before we discuss the polar codes, its important that we know what is a polar transform. Its something *similar* to FFT and IFFT. We define a transform matrix or generator matrix G_2 for it, where,

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

If we consider a message vector

$$U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

, then the polar transform of this message vector will be $U^T G$.

$$U^T G_2 = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} u_1 + u_2 & u_2 \end{bmatrix}$$

This was only the case until we considered two message symbols. If we have N, in general, message symbols, then the G matrix is generated by the *Kornecker product.*

$$G_k = G_2 \otimes G_{k/2} \tag{9}$$

More appropriately,

$$G_k = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes G_{k/2} = \begin{bmatrix} G_{k/2} & 0 \\ G_{k/2} & G_{k/2} \end{bmatrix}$$

3 Channel Polarization

The concept of channel polarization is simple. The information bits are transformed into polar domain by the application of suitable polar transform matrix. The point of applying is polar transform is simple; if we don't apply a polar transform on the bits, the bits goes simply like they would have gone in the case of simple BPSK. Applying the polar transform on these bits, changes the channel in such a way that now each bit has its own bit channel.

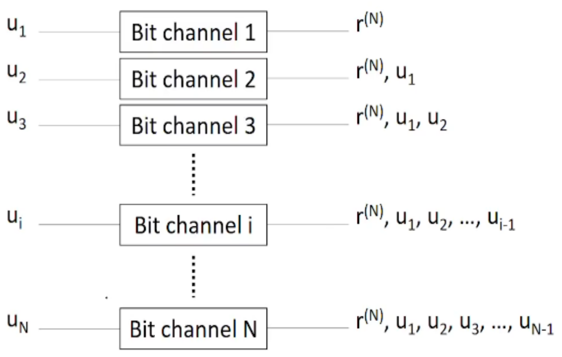


Figure 4: Transforming group of bits to each bit channel

The trick here is that, we will assume that the output vector $r^{(N)}$ does contain the information about the bit u_1. Similarly, for the i^{th} bit channel also, we will assume that the outputs are $r^{(N)}, u_1, u_2, ..., u_{i-1}$. For now, let us assume that we are getting the previous bits at the output of a bit channel from somewhere.

The properties of these bit channels is quite different. If we would have not applied the polar transform, the channel through which the bits would have gone would be same for all the bits. But now, each channel has a different property; why? Because the outputs of each bit channel is different. These properties are based upon the quality of the channels. The extremes of these bit channels are *good* and *bad*. There is nothing in between. That is why, this is called **channel polarization**.

3.1 Reliability Sequence

The reliability sequence is something that tells which bit channels are the good ones and which are the bad ones. For example, 1 2 3 5 4 6 7 8 is a reliability sequence. If we carefully look at it, we find that channel 5 is more bad as compared to channel 4. The current 5G standard uses 1024 length reliability sequence.

4 The (N, K) Polar Code

The (N, K) polar code is simple. Here we take $N = 2^n$ for some integer value of n. The value of $N > K$; this is a condition for the polar code. Now, we create a vector U of length N, such that

$$U = \Big[u_i | i \in [1, N]\Big]$$

Out of these N input bits, we then set the first $(N - K)$ bits to 0, because, according to the reliability sequence, the initial bits are worst. The remaining K bits in the reliability sequence are replaced by the message bits. So, the new U vector is

$$U = \Big[0, 0, 0, ..., 0_{(N-K)^{th}}, m_1, m_2, ..., m_K\Big] \quad (10)$$

After this, we apply the polar transform on this U vector to generate the codeword.

$$C = UG_N$$

Where, C is the codeword matrix and m_i denotes the i^{th} bit of the message. Let us illustrate this with the help of an example of $(8, 4)$ polar code. The reliability sequence is 1 2 3 5 4 6 7 8. We freeze the first $8 - 4 = 4$ bits by 0. The U vector becomes $U = [0, 0, 0, m_1, 0, m_2, m_3, m_4]$. The codeword matrix is given by,

$$C = UG_8$$

where, G_8 is

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

Therefore, the C matrix is

$$C = \begin{bmatrix} \sum_{i=1}^{4} m_i \\ m_1 + m_2 + m_4 \\ m_1 + m_3 + m_4 \\ m_1 + m_4 \\ m_2 + m_3 + m_4 \\ m_2 + m_4 \\ m_3 + m_4 \\ m_4 \end{bmatrix}^T$$

Where the additions correspond to the modulo 2 addition of the bits.

5 Successive Cancellation

In this part, we will see how to decode a given polar code. Although an intutive way could be to post multiply G_N^{-1} to the codeword matrix, i.e.;

$$U = CG_N^{-1}$$

But this method is computationally costly. However, for the simulation purposes, we have taken this approach only, because it is easy to code and we just wanted to analyse polar codes as a whole.

We will now see a more better option to decode.

First, let us consider the extra outputs that we were getting in the bit channels apart from the $r^{(N)}$ vector. Consider bit channel 2; in this the output was the r vector and the previous bit u_1. Now, if the bit u_1 was frozen, there is no need to do any calculation for the output bit of this kind. However, if it isn't, then the best we can do is, to estimate u_1 given the output vector r and the nature of the bit channel 2 (see figure 5).

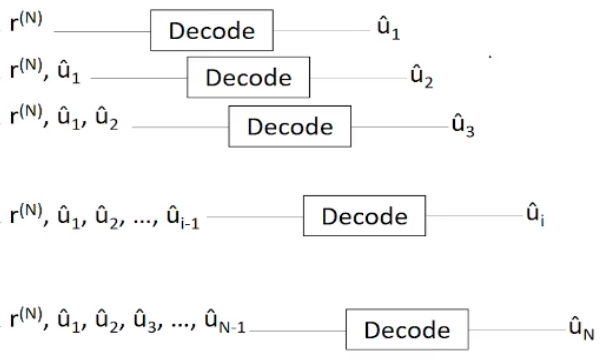


Figure 5: Decoding the Polar Codes

The idea behind decoding such outputs is very simple. For the first bit channel, we pass the r vector as it is to the decoder. According to figure 5, we will get an estimate of $u_1 = \hat{u_1}$. We then pass the same vector r to the next decoder alongwith $\hat{u_1}$ and decode u_2. Fortunately, by the very nature of the polar codes, the first $N - K$ bits will surely be zero, and therefore, there won't be much computations for the few initial decoders.

6 Results and Conclusions

We now discuss the results and conclusions of our experimentation. When we started, we first compared the BER vs E_b/N_o ratio plot of uncoded and the coded case. We found the following results based on our simulation.

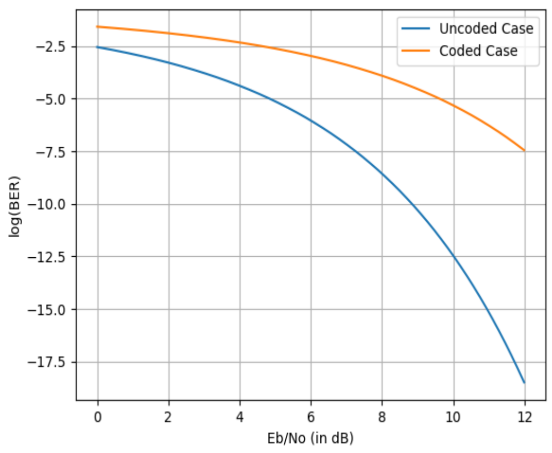


Figure 6: log BER vs E_b/N_o ratio for Uncoded and Coded case

Let us now understand why this happens. In the uncoded case $R = 1$ and therefore, the bit error rate was $Q\left(\sqrt{\frac{2E_b}{N_o}}\right)$. But when the rate increased to some value R such that $R \in (0, 1)$ then the BER becomes $Q\left(\sqrt{\frac{2RE_b}{N_o}}\right)$. Clearly,

$$Q\left(\sqrt{\frac{2E_b}{N_o}}\right) < Q\left(\sqrt{\frac{2RE_b}{N_o}}\right)$$

for the same E_b/N_o value. This is because $R < 1$ and therefore, the argument of RHS Q function is less than LHS Q function, which implies the current relation. For more information, please refer section 1.6.

Later on, we also saw the $(7, 4)$ Hamming code. The below shown graph shows that for higher values of E_b/N_o, the method of decision does not matter much. We can either use soft decision or hard decision.

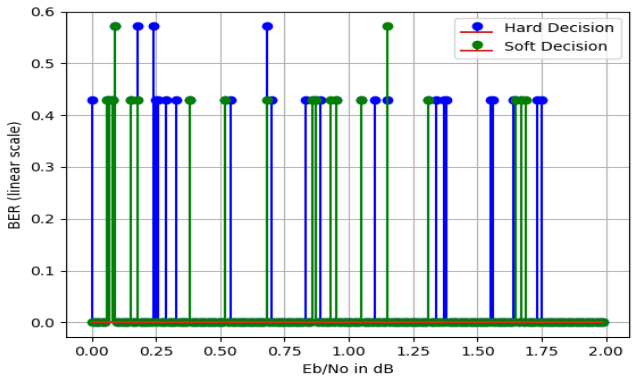


Figure 7: Soft decision vs. Hard decision

This graph is also the result of the simulation.

We now come to main simulation concerning the polar codes. We conducted two types of simulations here. In the first type of the simulation, we varied the message length (in bits) from 1 to 16 and for each length we calculated the average BER with E_b/N_o ratio varying from 0 to 12 in the steps of 0.01. On the successful simulation we found that for small message lengths the average BER is more than those with higher message lengths. The graph shown below, conveys the same thing.

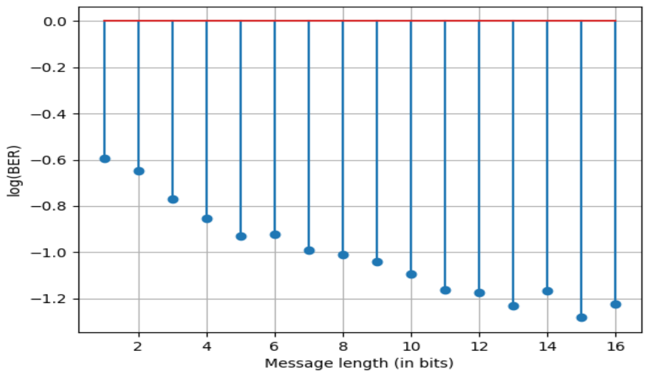


Figure 8: Average BER for diffent message bit length

This is happening because the logarithm is getting more and more negative. Therefore, based on our simulation, **we conclude that polar codes are good when $K \to N$.**

The second type of test that we conducted was by fixing the message length and varying the E_b/N_o ratio, just like we did with Hamming code and repitition code. The graph for this test is shown below.

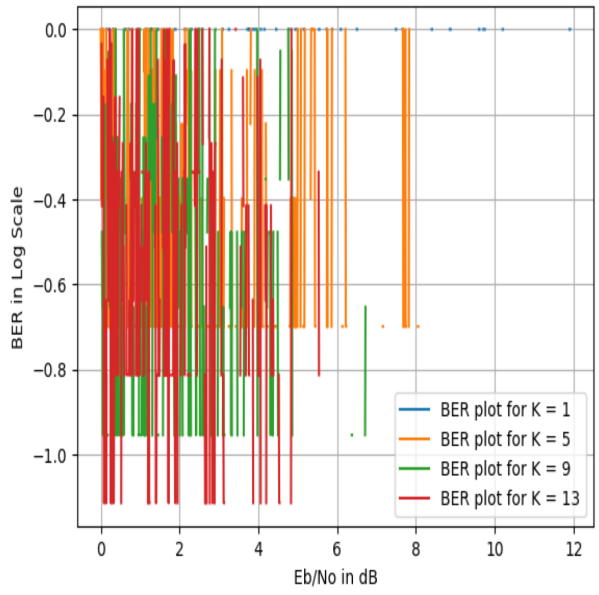


Figure 9: Average BER for differnt message bit length

Since this plot is also in the logarithmic scale, we need to be little cautious while reading it. The peaks having less height are more worse than those having high peaks. This is because the logarithm is negative, which means that the higher the negative log, the lesser is the linear scale value. Again, this graph shows the same results as were shown by figure 8 above. For larger message lengths, the polar code is favourable. Note that the tests conducted here were based on $(16, K)$ type polar codes where $K \leq 16$.

Apart from these simulations also, polar codes have various advantages. The most obvious one is the time complexity which is $\mathcal{O}(n \log n)$. On the other hand, repitition codes have poor error correcting capabilities. Polar codes have helped achieve high throughput in 5G wireless network when used as channel coding. During 5G field trials, Huawei has achieved 27 Gbps.

7 References

[1] Polar Codes: Primary Concepts and Practical Decoding Algorithms, Kai Niu, Kai Chen, Jiaru Lin, and Q. T. Zhang 0163-6804/14/$25.00 © 2014 IEEE

[2] https://nptel.ac.in/courses/108106137/26

[3] https://dsp.stackexchange.com