# Introduction to Communication Systems
# (CT216)

## Project Report : Polar Codes
## Group-15

- **Mentor :** Bibin Baby John

- **Group Members :**
  **202301156** - RUSHI GADHIYA
  **202301157** – PARV KHETAWAT
  **202301158** – MANAV PATEL
  **202301159** – DIPKUMAR ZADAFIYA
  **202301160** – VISWA VIGNAN BADDULA
  **202301161** – DARSHAN RAMANI
  **202301162** – MITWA NINAMA
  **202301164** – ARYAN RATHOD
  **202301165** – SHREY SHAH
  **202301167** – RISHIT RAJ JAIN

***Report Includes :***

*Polar code explanation*
*( intro.,*
*encoding,*
*decoding system,  details)*
*Shannon's Channel Capacity proof*
*Graph analysis*
*Reference Result*
*{Bibliography}*

- ## Introduction:

Polar Codes are a breakthrough in coding theory, introduced by Erdal Arikan in 2008. These codes are the first provably capacity-achieving codes for a wide class of channels under low-complexity encoding and decoding algorithms. The term "polar" comes from the phenomenon of channel polarization, where channels become either perfectly reliable or completely noisy(most unreliable).

This process results in some channels becoming very reliable while others become highly unreliable. In simple terms, the channels get "polarized," which is how the codes got their name.

- ## Importance of Polar Codes in Modern Communication:

One of the most relevant and practical uses of polar codes today is in 5G communication systems. In 5G, polar codes are mainly used for control and channel encoding. They help ensure that data is transmitted accurately and efficiently over noisy wireless environments.

What makes polar codes special is that they are the first kind of codes that can reach the capacity limit of symmetric binary-input memoryless channels (B-DMCs) using relatively simple decoding algorithms like Successive Cancellation. This means they can provide extremely reliable communication close to theoretical limits.

Because of these strengths, polar codes help 5G systems deliver fast and reliable communication even in challenging wireless environments. As a group, we explored these capabilities and found polar codes to be not just powerful in theory, but also highly practical for modern communication challenges. This encouraged us to understand the inner workings of encoding and decoding in polar codes.

- ## Encoding of Polar Codes:

## Channels polarization:

To encode data using polar codes, we follow a few systematic steps that involve understanding channel reliability, constructing generator matrices, and applying matrix operations:

- **Understanding Reliable and Unreliable Channels:**

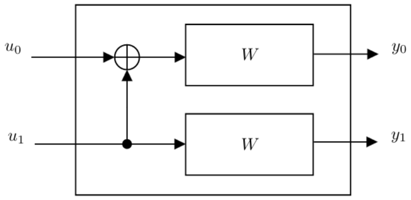
  In a polar code of length N, not all bit positions are equally reliable. To identify which bit positions should carry actual information, we use something called a reliability sequence. This sequence is usually predefined for a specific N (e.g., 5G uses one for N = 1024). The least reliable positions are “frozen” (set to zero), and only the most reliable positions are used for transmitting real information.

  To measure the reliability of a bit-channel, we use the Bhattacharyya Parameter (Z), defined as:

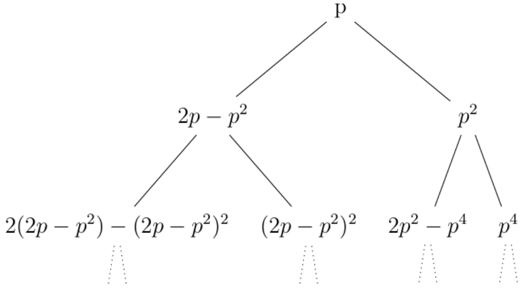
$$Z(W) = \sum_{y} \sqrt{W(y|0)W(y|1)}$$

  A lower value of Z means the channel is more reliable. This helps us decide where to put information bits and where to place frozen bits.

**Combining channels**



**Probability Diagram**



- **Generator Matrix Using Kronecker Product:**

  The core of encoding lies in using a generator matrix constructed recursively. The basic building block is:

  Generator matrix of size 2 × 2 :

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

  For $N = 2^n$ bits, the generator matrix G□ is formed using the Kronecker product of $G_2$ with itself n times. The Kronecker product takes a matrix and expands it systematically. This recursive process continues as N grows larger.

- **Final Encoding Step:**

  Let's say we have a vector *u* of N bits. This vector includes both the actual information bits (in reliable positions) and frozen bits (in unreliable ones, usually set to zero).

  For 2 bits input we can encode it by multiplying with 2 × 2 generator matrix :

$$[u1\ u2]G_2=[u1+u2\ u2]$$

  In general, to get the encoded message *x*, we multiply *u* with the generator matrix *G_N*:

$$x = u \cdot G\_N$$

## Modulation of Polar Codes:

Once we complete the encoding process where data is structured and encoded using the generator matrix the next important step is to prepare these bits for wireless transmission. This is done using a process called modulation, which converts digital bits into signal waveforms that can travel through a physical medium like air or cable.

For polar codes, we use BPSK (Binary Phase Shift Keying) modulation. In BPSK:

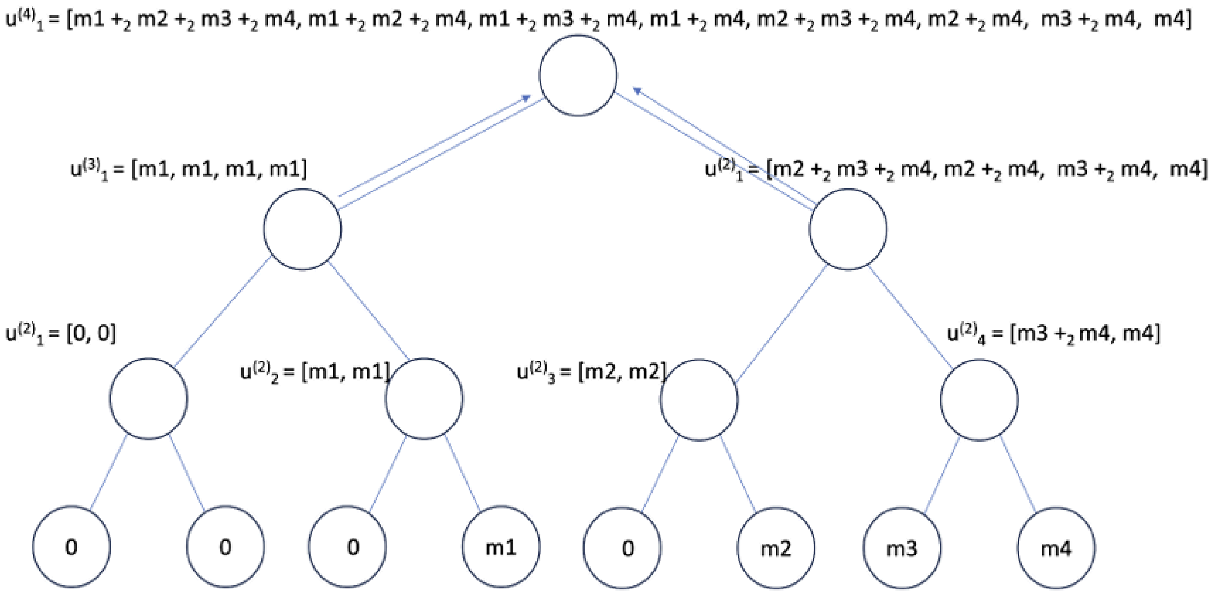
Bit **0** is mapped to signal **+1**
Bit **1** is mapped to signal **−1**

This simple mapping makes it easier to transmit data over wireless channels. However, wireless communication always comes with some level of noise due to interference, hardware imperfections, and natural disturbances.

To simulate this real-world condition, we add AWGN (Additive White Gaussian Noise) to our signal. This type of noise has a normal (Gaussian) distribution and helps us test how well polar codes can recover the original data despite interference.

By passing the modulated signal through a BPSK-AWGN channel, we are able to Test the robustness of polar codes against noise, Measure how efficiently bandwidth is being used Simulate realistic wireless environments, like what happens in 5G networks

This step is essential in evaluating the real-world performance of polar codes, helping us understand their error-correction ability under practical transmission conditions.



$u^{(4)}_1 = [m1 +_2 m2 +_2 m3 +_2 m4, m1 +_2 m2 +_2 m4, m1 +_2 m3 +_2 m4, m1 +_2 m4, m2 +_2 m3 +_2 m4, m2 +_2 m4, m3 +_2 m4, m4]$
$u^{(3)}_1 = [m1, m1, m1, m1]$
$u^{(2)}_1 = [m2 +_2 m3 +_2 m4, m2 +_2 m4, m3 +_2 m4, m4]$
$u^{(2)}_1 = [0, 0]$
$u^{(2)}_2 = [m1, m1]$
$u^{(2)}_3 = [m2, m2]$
$u^{(2)}_4 = [m3 +_2 m4, m4]$
0
0
0
m1
0
m2
m3
m4

- ## Decoding Methods of Polar Codes:

Decoding is a key step in the process of recovering original information from a polar coded message. There are two main decoding techniques used with polar codes:

1.Successive Cancellation (SC) Decoder

2.Successive Cancellation List (SCL) Decoder

Both methods rely on the recursive structure of polar codes but differ in their strategy for handling uncertainty during decoding.

- ### Successive Cancellation (SC) Decoder:

  This is the basic and original decoding method for polar codes. It works in a recursive manner and uses hard-decision decoding to estimate the original bits one by one. The steps are:

  1) Start by estimating the leftmost bit based on the received signal.
  2) Use that estimated bit to help in decoding the next bit.
  3) Continue this process recursively until all bits are decoded

  Even though SC decoding is simple and efficient, it may not always give the best performance especially when dealing with short block lengths or high noise.

- ### Successive Cancellation List (SCL) Decoding: Enhanced Polar Code Decoding

  While Successive Cancellation (SC) decoding provides a basic low-complexity method for decoding polar codes, it often fails to deliver good performance under noisy conditions or when using short block lengths. To overcome this limitation, the Successive Cancellation List (SCL) decoding algorithm was introduced, offering significantly improved reliability at the cost of additional complexity.

  **How SCL Decoding Works:**

The main idea behind SCL decoding is to maintain multiple decoding paths instead of committing to a single decision at each step. Each bit decision branches into two possibilities 0 and 1 and these branches grow into a list of candidate decoding paths.

To manage this process efficiently, SCL decoding uses:

Path Matrix: Keeps track of all active decoding paths. Each row in this matrix represents one candidate sequence of decoded bits up to the current step.

Decision Matrix / Metric Scores: Alongside the path matrix, a decision metric (also called a path metric or likelihood score) is maintained. This score reflects how likely a particular path is to be the correct decoding based on the received channel information.

**Path Pruning and List Management:**

As the decoding progresses bit by bit, the number of decoding paths could potentially double at every step. For example, starting with 1 path:

- After 1 bit: 2 paths
- After 2 bits: 4 paths
- ...
- After k bits: 2^k paths

This exponential growth is not practical, so to manage complexity, the decoder prunes the list at every step by:

1. Evaluating the path metrics for all candidate paths (i.e., how good each guess is).
2. Sorting the paths based on their metric scores (better list has lower score).
3. Keeping only the top L paths, where L is a fixed list size (e.g., 4, 8, 16).
4. Discarding the rest to control memory and processing.

This ensures the decoder always works with a manageable number of highly probable candidates.

At the final step, if a CRC (Cyclic Redundancy Check) is used, it is applied to all surviving paths. The path that passes the CRC check is chosen as the correct codeword. If multiple pass, the one with the best score is selected.

## CRC(Cyclic Redundancy Check):

The Cyclic Redundancy Check is a widely used technique for detecting errors in digital data during transmission or storage. It adds a small amount of redundancy to the original data, which can be used to detect whether any bits were altered or corrupted.

- **Working Principle:**

  CRC works by treating the original data as a long binary number (a polynomial) and dividing it by a predefined generator polynomial using modulo-2 division (similar to binary division but without carry/borrow). The remainder from this division is known as the CRC checksum.

  This checksum is appended to the data before transmission. At the receiver side:

  1. The received data (including the CRC checksum) is again divided by the same generator polynomial.
  2. If the remainder is zero, the data is considered error-free.
  3. If the remainder is non-zero, an error is detected.

  This process is efficient, fast, and can detect many types of common errors, including burst errors.

**Successive Cancellation Decoder (SC) details**

Because of the recursive structure of channel polarization, we can use a simple decoding scheme called Successive Cancellation (SC) for the decoding purpose.

Let us take an example of length 2 SC decoder to understand how it works.

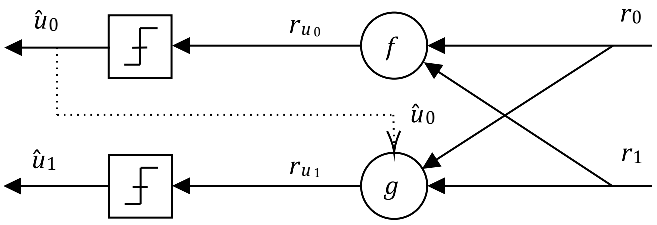


Figure 12: Length 2 SC decoder

As we have seen how to polarize a channel in previous section, we can say that decoding problem reduces to decoding a single parity check code (SPC) to make an estimate for $u_0$. After making decision for $u_0$, we can use it to decode $u_1$, which turns out to decode a repetition code.

As we are transmitting bits in AWGN channel with BPSK modulation, we can assume that we receive LLR (beleif) of transmitted bits.

Let us define LLRs for further calculations.

$$L_{r_0} = log\left(\frac{p\,(r_0 = 0)}{p\,(r_0 = 1)}\right) = log\left(\frac{p_{r_0}}{1 - p_{r_0}}\right) \tag{1}$$

Where $r_0$ and $r_1$ are received bits.

Let us define LLRs for $u_0$ and $u_1$ as well.

Now, problem of finding LLR for bit $u_0$ reduces to single parity check(SPC) decoding of $r_0$ and $r_1$.

Transmitted bit $u_0$ is 0 when both $r_1$ and $r_2$ are either 0 or 1. Thus we can write,

$$p_{u0} = p_{r0}p_{r1} + (1 - p_{r0})\,(1 - p_{r1}) \tag{3}$$

$$1 - p_{u0} = p_{r0}\,(1 - p_{r1}) + (1 - p_{r0})\,p_{r1} \tag{4}$$

Subtracting 4 from 3, we get,

$$\begin{aligned}
p_{u0} - (1 - p_{u0}) &= (p_{r0} - (1 - p_{r0}))\,(p_{r1} - (1 - p_{r1}))\\
\Rightarrow \frac{p_{u_0} - (1 - p_{u_0})}{p_{u_0} + (1 - p_{u_0})} &= \frac{(p_{r_0} - (1 - p_{r_0}))}{(p_{r_0} + (1 - p_{r_0}))}\,\frac{(p_{r_1} - (1 - p_{r_1}))}{(p_{r_1} + (1 - p_{r_1}))}\\
\Rightarrow \frac{1 - \frac{(1-p_{u_0})}{p_{u_0}}}{1 + \frac{(1-p_{u_0})}{p_{u_0}}} &= \text{Ñ}\frac{1 - \frac{(1-p_{r_0})}{p_{r_0}}}{1 + \frac{(1-p_{r_0})}{p_{r_0}}}\text{éÑ}\frac{1 - \frac{(1-p_{r_1})}{p_{r_1}}}{1 + \frac{(1-p_{r_1})}{p_{r_1}}}\text{é}\\
\Rightarrow \frac{1 - exp(-L_{u_0})}{1 + exp(-L_{u_0})} &= \text{Å}\frac{1 - exp(-L_{r_0})}{1 + exp(-L_{r_0})}\text{ãÅ}\frac{1 - exp(-L_{r_0})}{1 + exp(-L_{r_0})}\text{ã}
\end{aligned}$$

$$\Rightarrow L_{u_0} = 2\,tanh^{-1}\left(tanh\left(\frac{L_{r_0}}{2}\right)\, tanh\left(\frac{L_{r_1}}{2}\right)\right)$$

Above function can be approximated as,

$$\Rightarrow L_{u0} = sgn(L_{r0})\; sgn(L_{r1})\; min(|L_{r0}|,|L_{r1}|)$$

This function is called minsum function. Let us denote minsum function as

$$f(x_0,x_1) = sgn(x_0)\; sgn(x_1)\; min(|x_0|,|x_1|)$$

Thus, above equation can be rewritten as,

$$L_{u0} = f(L_{r0},L_{r1})$$

By calculating LLR value for $u_0$, we can make estimate for $u_0$ as follow:

®

$$\hat{u}_0 = \begin{matrix} 0, & \text{if } L_{u0} \geq 0 \\ 1, & \text{if } L_{u0} < 0 \end{matrix}$$

Now, using these estimated value of $u_0$, we can make estimate for $u_1$. Suppose $\hat{u}_0 = 0$, then $L_{r0}$ and $L_{r0}$ both becomes beleif for $u_1$. In other case, if $\hat{u}_0 = 1$, then $L_{r0}$ becomes beleif for and $\bar{u}_1$ and $L_{r0}$ becomes beleif for $u_1$.

Thus, problem of estimating bit $u_1$ reduces to make an estimate for repetition code.Assume that $\hat{u}_0 = 0$. For repetition code, we know that,

$$p_{u_1} = \frac{p_{r_0}\, p_{r_1}}{p_{r_0}\, p_{r_1} + (1-p_{r_0})\,(1-p_{r_1})} \tag{5}$$

$$1 - p_{u_1} = \frac{(1-p_{r_0})\,(1-p_{r_1})}{p_{r_0}\, p_{r_1} + (1-p_{r_0})\,(1-p_{r_1})} \tag{6}$$

Dividing 5 by 6, we get,

$$\frac{p_{u_1}}{1-p_{u_1}} = \text{Å}\frac{p_{r_0}}{1-p_{r_0}}\text{ãÅ}\frac{p_{r_1}}{1-p_{r_1}}\text{ã}$$

Taking log on both sides, we get,

$$log\;\text{Å}\frac{p_{u_1}}{1-p_{u_1}}\text{ã} = log\;\text{Å}\frac{p_{r0}}{1-p_{r0}}\text{ã} + log\;\text{Å}\frac{p_{r1}}{1-p_{r1}}\text{ã}$$

$$\Rightarrow L_{u1} = L_{r0} + L_{r1} \qquad \text{(From 1 and 2)}$$

Similarly, for $\hat{u}_0 = 1$, we get following expression using same steps,

$$L_{u1} = L_{r1} - L_{r0}$$

Combining both the equations above, we get:

$$L_{u1} = L_{r1} + (-1)^{\hat{u}_0} L_{r0}$$

Let us define function $g(x)$ as below:

$$g(x_0,x_1,y) = x_1 + (-1)^y x_0$$

Thus, above equation can be rewritten as,

$$L_{u1} = g(r_0,r_1,\hat{u}_0)$$

After calculating LLR for $u_1$, we can make estimate for $u_1$ as,

$$\hat{u}_1 = \begin{cases} 0, & \text{if } L_{u1} \geq 0 \\ 1, & \text{if } L_{u2} < 0 \end{cases}$$

Now, as we have some understanding how decoding works in sequential manner, we can fully understand functioning of SC decoder depicted in ??.

We first pass received LLRs through node f, then we make decision for bit $u_0$. Using this decision, we pass the values to node $g$ and make decision for bit $u_1$ and the decoding process is done.

As polar codes have recursive structure, we can apply this algorithm for decoding any length of codeword and understand its behaviour.

**Successive Cancellation List Decoder (SCL)**

This decoding scheme is similar to SC decoding, which we discussed in previous section. The only difference in Successive Cancellation List Decoder is that it maintains a list of possible codewords that are most likely to be transmitted.

Number of codewords that SCL decoder maintains is called the list size.

We have seen in the case of SC decoder that it makes decision about some nonfrozen bit $u_0$, and then uses this bit to make decision for other bit $u_1$.

Now, in the case of SCL decoder, we take both the possibilities that whether the bits are decoded correctly or not. So, when we make decision for each non-frozen bits, we take two possibilities of either the bit is 0 or 1 irrespective of its LLR. We get the following tree structure when we do this. Doing this for each bit is not practical, because number of branches of the tree grows exponentially. That leads to higher complexity decoder, which we want to avoid. Inorder to keep the complexity

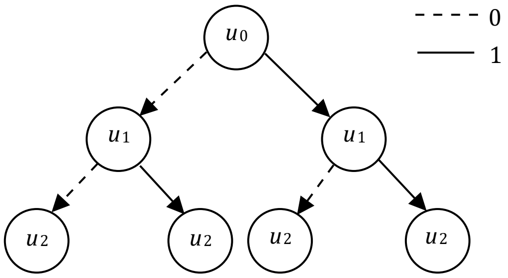


Figure 23: Tree representation for all possible decisions of non-frozen bits

reasonable, we keep only certain number of branches at any level of this tree. This number is called list size.

To decide which branch to keep and which to discard, we maintain a path metric based on LLRs of the bits for each path.

We continue this process of creation and termination of branches till last bit is decoded. At last, decoding path with lowest path metric is choosen as the decoded codeword. Let us compare the comined results for SC and SCL

Now, we can compute Bhattacharya Parameter $Z(W)$ as

$$Z(W) = \int_{-\infty}^{\infty} \sqrt{p(y|0)\; p(y|1)}\, dy$$

$$= \int_{-\infty}^{\infty} \sqrt{\frac{1}{\sqrt{2\,\pi\,\sigma^2}}\, exp\left(-\frac{(y-1)^2}{2\,\sigma^2}\right) \frac{1}{\sqrt{2\,\pi\,\sigma^2}}\, exp\left(-\frac{(y+1)^2}{2\,\sigma^2}\right.}$$

ã

$dy$

$$Z(W) = exp\left(-\frac{1}{2\,\sigma^2}\right.$$ ã

With the knowledge of mean of LLR for each channels $W_N^{(i)}$, Bhattacharya Parameter $Z_N^{(i)}$ can be calculated as,

Ñ é

$$Z_N^{(i)} = exp \quad -\frac{1}{2\left(\sigma_N^{(i)}\right)^2}$$

Using equation 7, we get,

$(i)$ $\Bigg($ $m_N^{(i)}$

å $Z_N$

$= exp -\underline{\quad\quad}$

4

We know that higher the Bhattacharya Parameter of channel, lesser the reliability of channel. After approximating $Z_N^{(i)}$ for each channels, we can sort them in increasing order and get channels in decreasing order of reliability.

After this process, We can select first $k$ channels to transmit information bits and we can send foreknown bits (usually we take them as 0) through remaining $N-k$ channels for $(N,k)$ Polar codes.

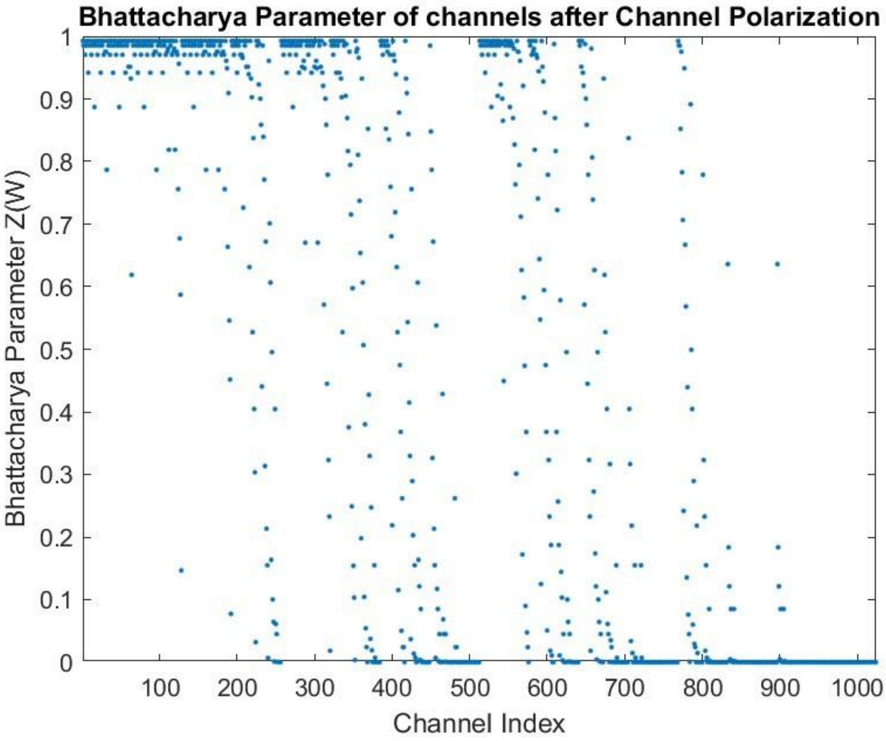


Figure 31: Bhattacharya Parameter for polarized channels using Gaussian Approximation method

After sorting these values, we get following graph.

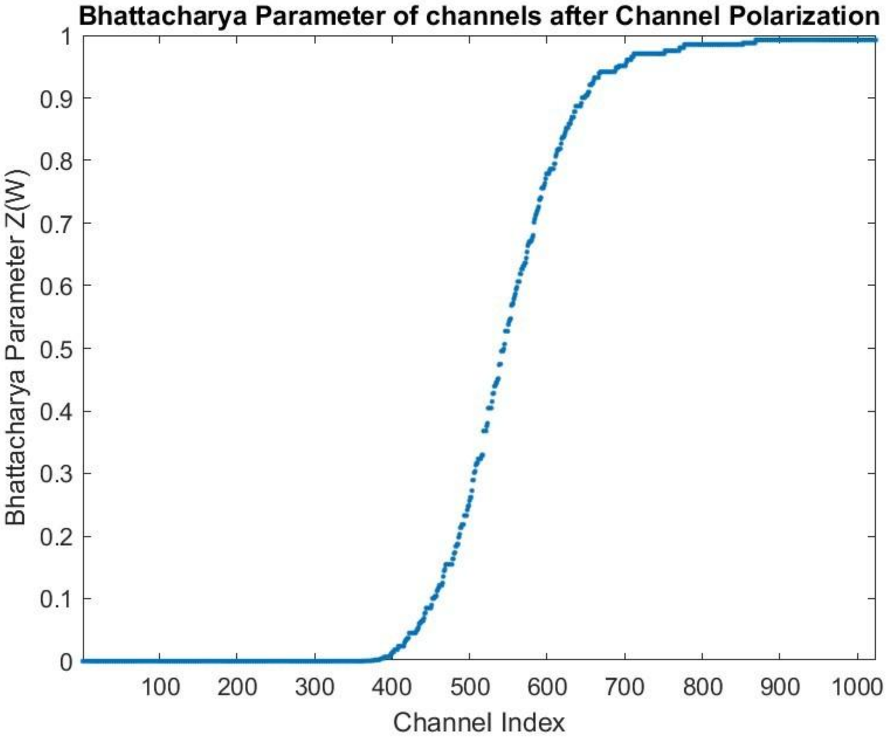


Figure 32: Sorted Values for Bhattacharya Parameter for polarized channels using Gaussian Approximation method

**Algorithm for obtaining the reliability sequence:**

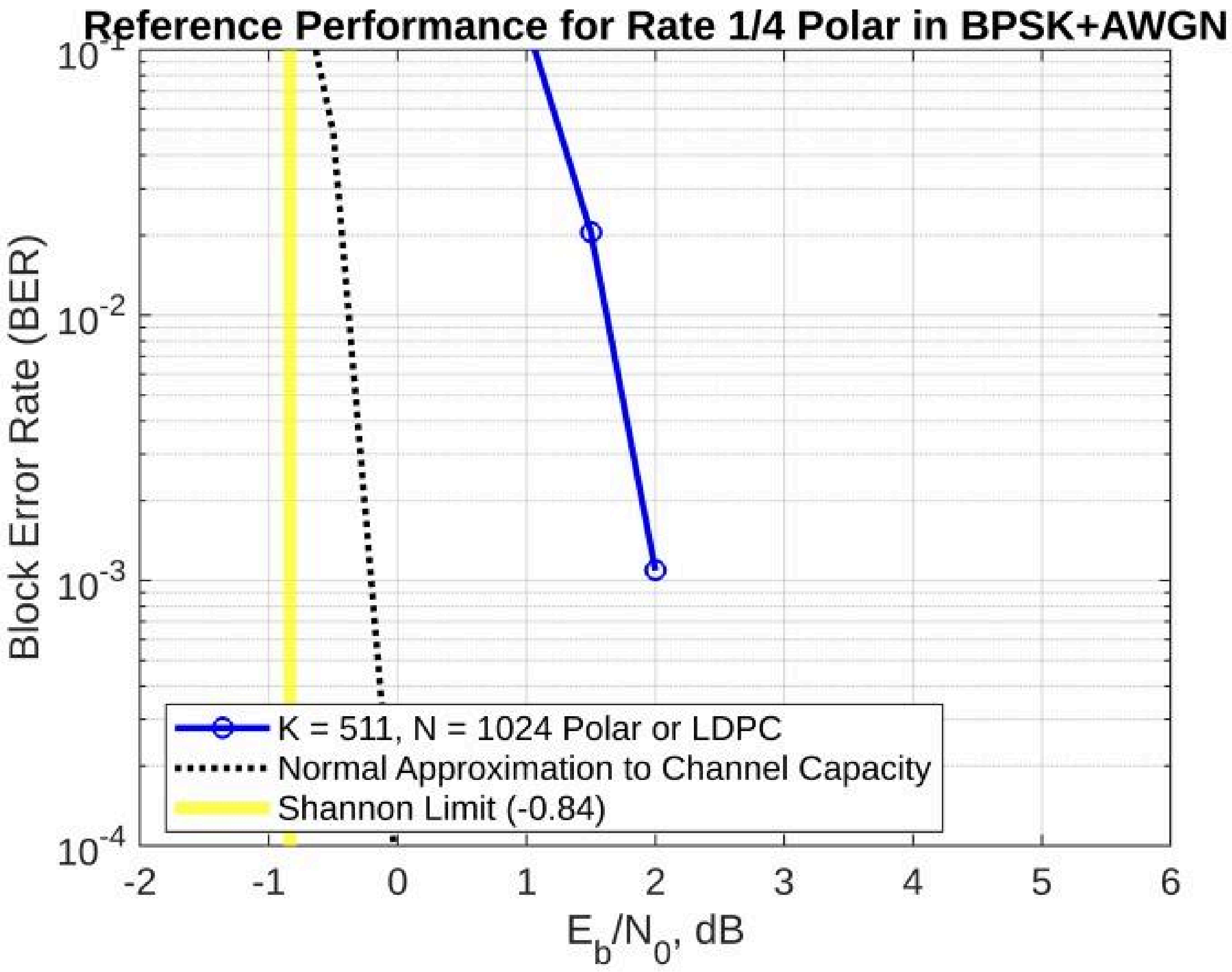
```
global var index = 1 define $N$,

map(double, int) $m$, i = 0

$n = \log_2 N$
$p = Q\left(\sqrt{\frac{\mathrm{SNR}}{2}}\right)$
split($m$,$p$,index,$i$)        reliability
sequence = $m$.values

define function split($m$,$p$,index,$i$) if (
    i == n ∨ $index$ == $N$) $m[p]$ =
    index index = index + 1
        return
split(m,$2p - p^2$,index,i+1)
split(m,$p^2$,index,i+1) return
```

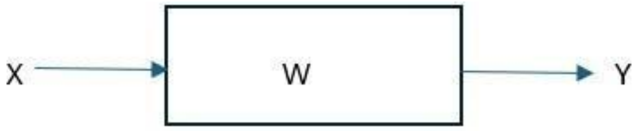
- **Reference Result**



# Theoritical Analysis – shannon's channel capacity proof

### Definition: Binary Discrete Memoryless Channel

A B-DMS $W$ is defined as a channel with input alphabets $X$, output alphabets $Y$ , and a probability distribution $w(y|x)$ that describes the probability of receiving $y$ given that $x$ was transmitted.



The entropy $H(W)$ and the mutual information $I(W)$ of a channel $W$ are related by the equation:

$$I(W) = 1 - H(W)$$

Assuming $X$ follows $Bernoulli(q = 0.5)$ The capacity of a BSC($p$) is given by: $I(W) = 1 - H_2(p)$

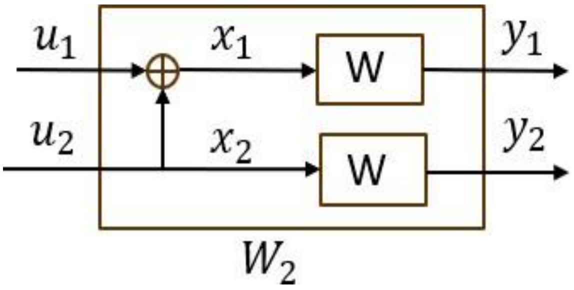
The capacity of a BEC($p$) is given by:

$$I(W) = 1 - p$$

### Channel Polarization

Now let's define two input-two output channel $W_2$ as below:

$$W_2(y_1, y_2|u_1, u_2) = W(y_1|u_1 \oplus u_2) \cdot W(y_2|u_2)$$



This is basically a $G_2$ Polar Transform where,

$$u = [u_1 u_2]$$

$$Y = [y_1 y_2] = uG_2$$

Two virtual channels $W^+$ and $W-$ are defined as:

$$W- : U_1 \longrightarrow (Y_1, Y_2)$$

$$W^+ : U_2 \longrightarrow (Y_1, Y_2, U_1)$$

If we consider BEC channel with erasure probability p then, the erasure probabilities for $W^+$ and $W-$ are given by:

Erasure Probability for $W^+ = p^2$

Erasure Probability for $W- = p(2 - p)$

Assuming $X$ follows $Bernoulli(q)$, then the information BEC is given by

$$I(W) = H_2(q)(1 - ErasureProbability)$$

So for $W^+$ and $W-$, the information will be as follows:-

$$I(W^+) = H_2(q)(1 - p^2)$$

$$I(W-) = H_2(q)(1 - 2p + p^2)$$

We can clearly see that the virtual channel W+ is performing better than the original channel W and the virtual channel W- is performing worse than the original channel W.

$$I(W-) \leq I(W) \leq I(W^+)$$

Similarly for W4 we can define 4-virtual channels, where some channels will perform well while others will get worse :-

$$W_4(y_1, y_2, y_3, y_4|u_1, u_2, u_3, u_4) = W_2(y_1, y_2|u_1 \oplus u_2, u_3 \oplus u_4)\ W_2(y_3, y_4|u_2, u_4)$$

For larger N, Recursively doing this Half of the channel becomes highly informative and half of the channel becomes information-less.

This relates with the Matthew Effect which is based on the concept that the rich gets richer and poor gets poorer. In our scenario, by performing polar coding, the channel is divided into two halves, which are ”very good channels” and ”very bad channels”.

For block length N we have N bit-channels and all Ui are i.i.d so the conservation of en- tropy/information property holds for information.

So the information for $W_1, W_2, \ldots, W_N$ is same

$$I(W_1) = I(W_2) = \ldots = I(W_N) = I(W)$$

$$\sum_{i=1}^{N} I(W_i) = N \cdot I(W)$$

For larger N fraction of bit-channels that are noise-less will be I(W) and the fraction of bit-channels that are highly-noisy will be $1 - I(W)$

So the Number of bit-channels that are noiseless is NI(W) and the number of bit-channels that are highly noisy are $N(1–I(W))$

As We are setting message positions which are highly informative so the number of message bits $K = NI(W)$

Rate is defined as Number of Message bits divided by Total Number bits

$$Rate = K/N$$

For large N,

$$\text{Rate} = \lim_{N \to \infty} \frac{\text{number of good channels}}{\text{Total Number of channels}}$$

$$= \lim_{N \to \infty} \frac{K}{N}$$

$$= \lim_{N \to \infty} \frac{N \cdot I(W)}{N}$$

$$= I(W)$$

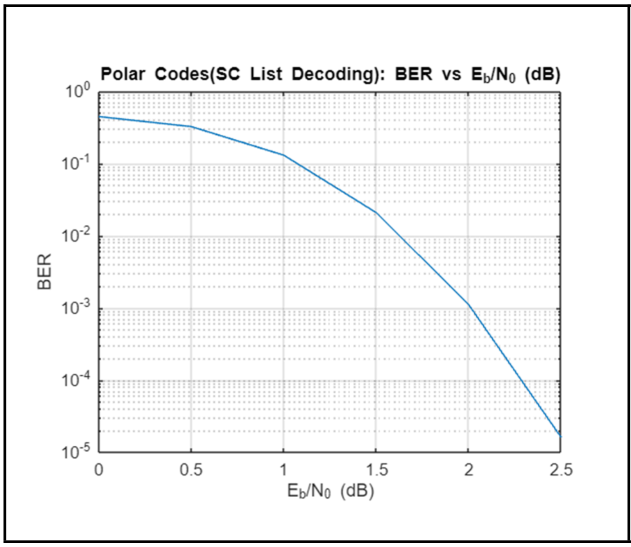
So for larger $N$, polar codes arbitrarily achieve Shannon’s Channel Capacity.

# ● Graph analysis

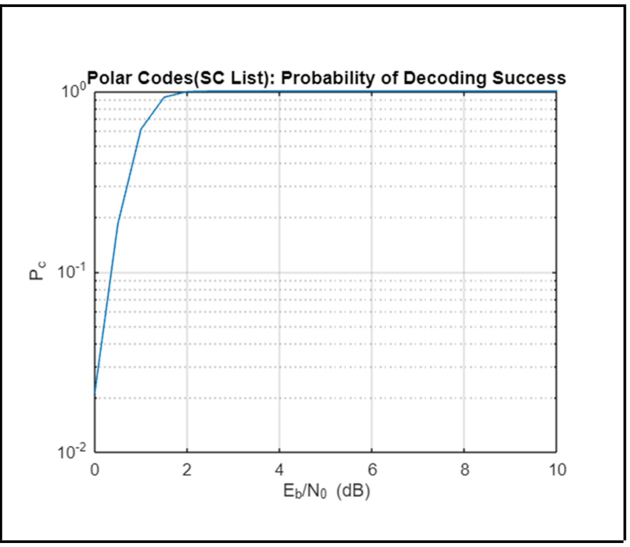
Graph for N = 1024 A = 500 (Plots for Nsim 10000 for Eb 0:0.5:10)

BER vs Eb/N0

Probability of encoding success



Graph 1



Graph 2

## Analysis & Explanation:

### Graph 1: BER vs Eb/N0

**Curve** : It starts high (near 0.5) at low Eb/N0, then **drops exponentially** as Eb/N0 increases.

**High BER at low SNR**
At very low Eb/$N_0$ (below ≈1 dB), the noise dominates the BPSK symbols and the SC decoder's hard decisions are effectively random, yielding a BER close to 0.5 (i.e., half the bits are in error).

As Eb/$N_0$ rises into the 2–4 dB range, the curve plunges sharply ,where a small SNR gain produces a large BER reduction.

Beyond about 6 dB, the BER continues to decrease nearly exponentially but begins to approach an “error floor” determined by the finite block length and the limitations of SC decoding.

### Graph 2: Pc vs Eb/$N_0$

Pc is essentially 0 at Eb/$N_0 \leq 1$ dB, since almost every 1024-bit frame contains at least one error under SC decoding

Around the same 3–5 dB region, Pc climbs rapidly from 0 toward 1. This is because once the BER falls below roughly $10^{-3}$, a large fraction of 1024-bit blocks become entirely error-free

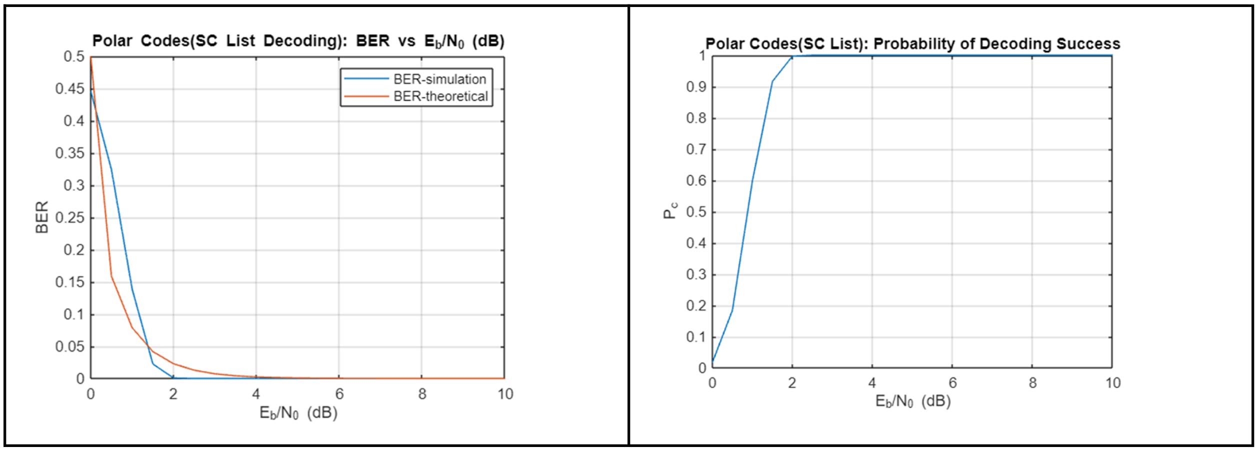
For Eb/$N_0$ beyond about 7 dB, Pc approaches unity, indicating that almost every transmitted block is decoded perfectly

**Now for change in rate (N/k) {k=A+l message+crc=total information bit} by changing N and K values we analyse what difference it makes in our output,**

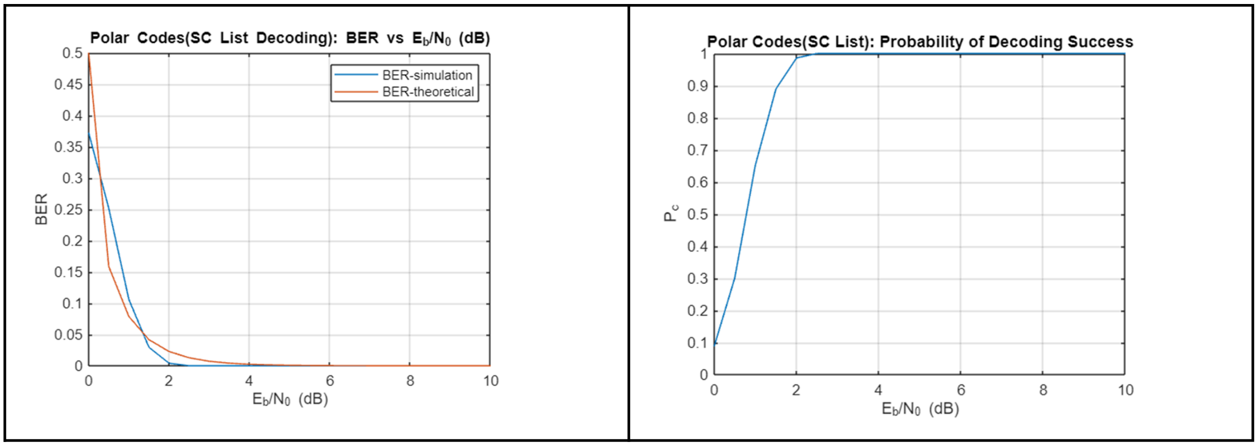
**Also we analyse how change in Ls (list size) affect output.**

**For this, next we have graphs for this conditions.**

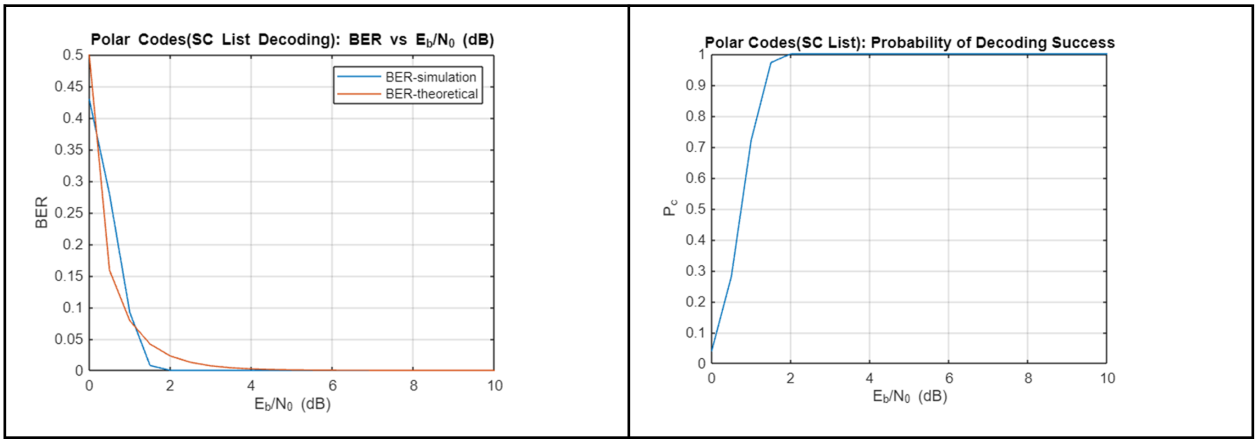
## Case 1 : N=1024(encoded bits) A=500(message) Ls=4(list size)



## Case 2 : N=512 A=245 Ls=4



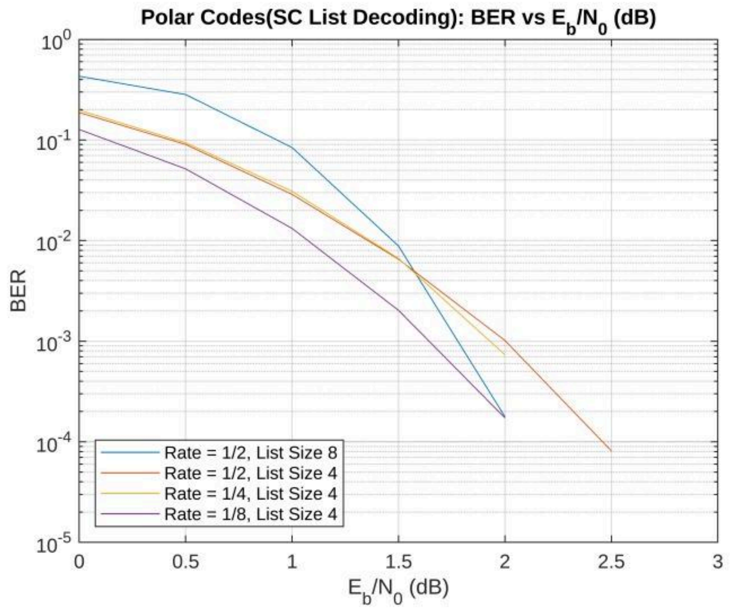
## Case 3: N=1024 A=500 Ls=8



# Graphs comparison observation & analysis :

Here , case 1 we analysed before. Now let us compare case 2 and 3 with it to see change by rate and change by list size.

## 1. Case 1 to case 2 : change in rate



- **Graph 1**

**Observation:**

We notice there is a shift in BER to Eb curve to left.overall by decrease in rate, BER decreases.

**Analysis:**

**Lower rate ⇒ more redundancy.** With A/N slightly smaller in Case 2, the decoder has more frozen bits for the same N, so it corrects errors more easily. Thus you see the BER curve for Case 2 drop earlier (leftward shift).

- **Graph 2**

**Observation:**

We notice there is a rise  in the Pc curve sooner and it becomes steeper , overall by decrease in rate, Pc increases sooner.

**Analysis:**

**More protection per bit** means entire blocks become error‑free at a lower Eb/N₀. The Pc curve for Case 2 rises sooner and more steeply.

## 2. Case 1 to case 3 : change in list size

- **Graph 1**

**Observation:**

We notice there is a very small  shift in BER to Eb curve to left.overall by increasing Ls, BER decreases slightly .

**Analysis:**

**Larger list ⇒ better path diversity.** Doubling the list size from 4 to 8 lets the decoder keep more candidate codewords, so the correct one survives more often—even under heavier noise.

- **Graph 2**

**Observation:**

We notice there is a slight rise in the Pc curve sooner and it becomes steeper , overall by increasing list size , Pc becomes steeper at point.

**Analysis:**

**Higher decoding accuracy per frame.** With more surviving paths, the chance of a completely correct 1024‑bit block jumps earlier in SNR.

## ❖ BIBLIOGRAPHY:

1. Nptel lectures of Andrew Thangaraj. Polar codes in the 5g standard, 2019.
2. EventHelix. Polar codes: Develop an intuitive understanding, 2019.
3. Session 4B - Arikan meets Shannon: Polar codes with near-optimal convergence to channel capacity by the Association of Computing Machinery.
4. Telatar Emre. The flesh of polar codes, 2017.
5. Venkatesan Guruswami, Patrick Xia - Polar Codes: Speed of polarization and polynomial gap to capacity.