G 8 - CT216

LDPC DECODING FOR 5G NR

Lab Group 2

Under Guidance of : Professor Yash Vasavada
Teaching Assistant Mentor: Vivek Patel

Honor Code

We declare that:

- The work that we are presenting is our own work.
- We have not copied the work (the code, the results, etc.) that someone else has done.
- Concepts, understanding and insights we will be describing are our own.
- We make this pledge truthfully.
- We know that violation of this solemn pledge can carry grave consequences.

1

Introduction to LDPC and 5G NR

2

Encoding

3

Hard Decision Decoding

4

Soft Decision Decoding

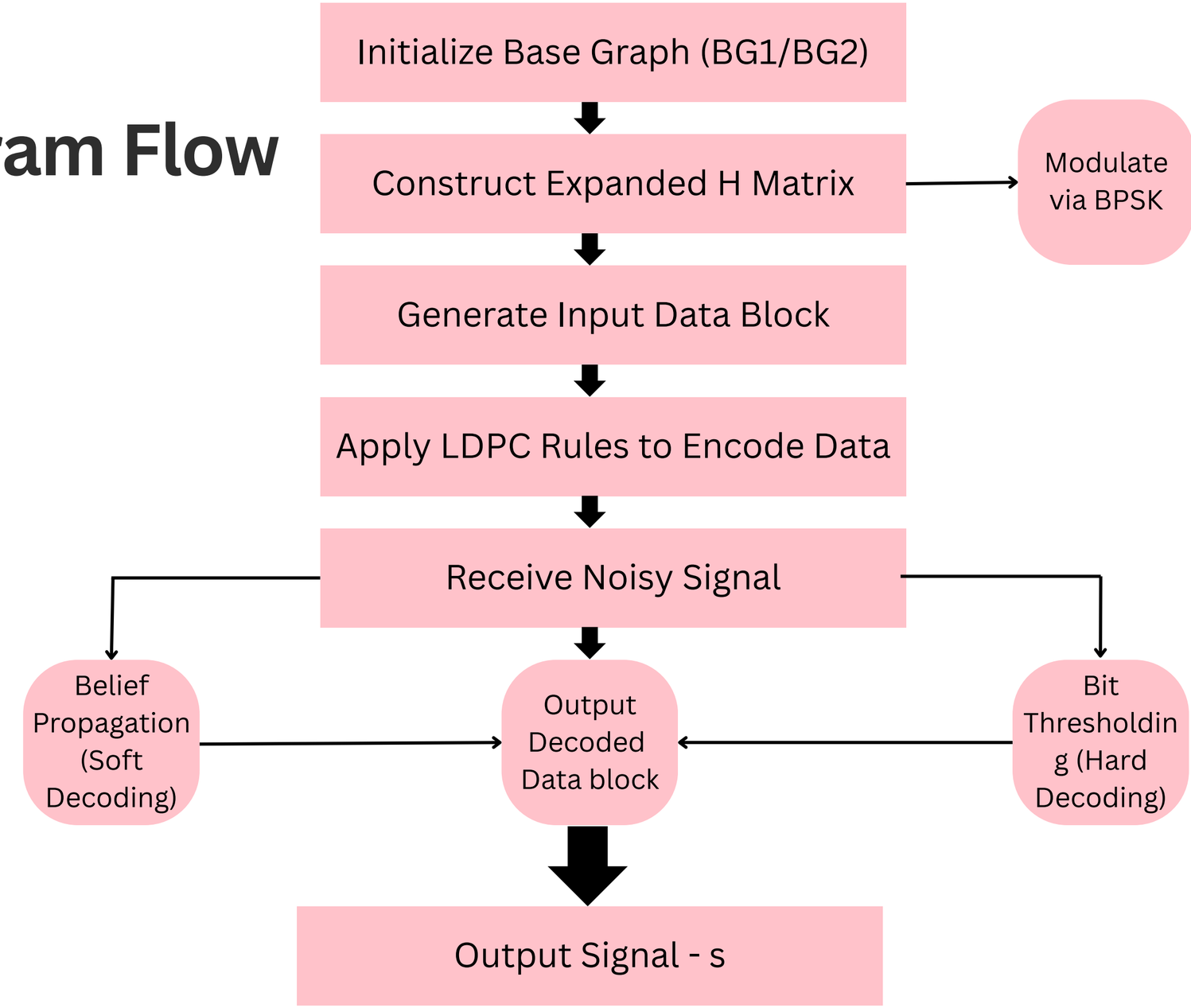
5

Results

6

Conclusion

Program Flow



What are LDPC Codes?

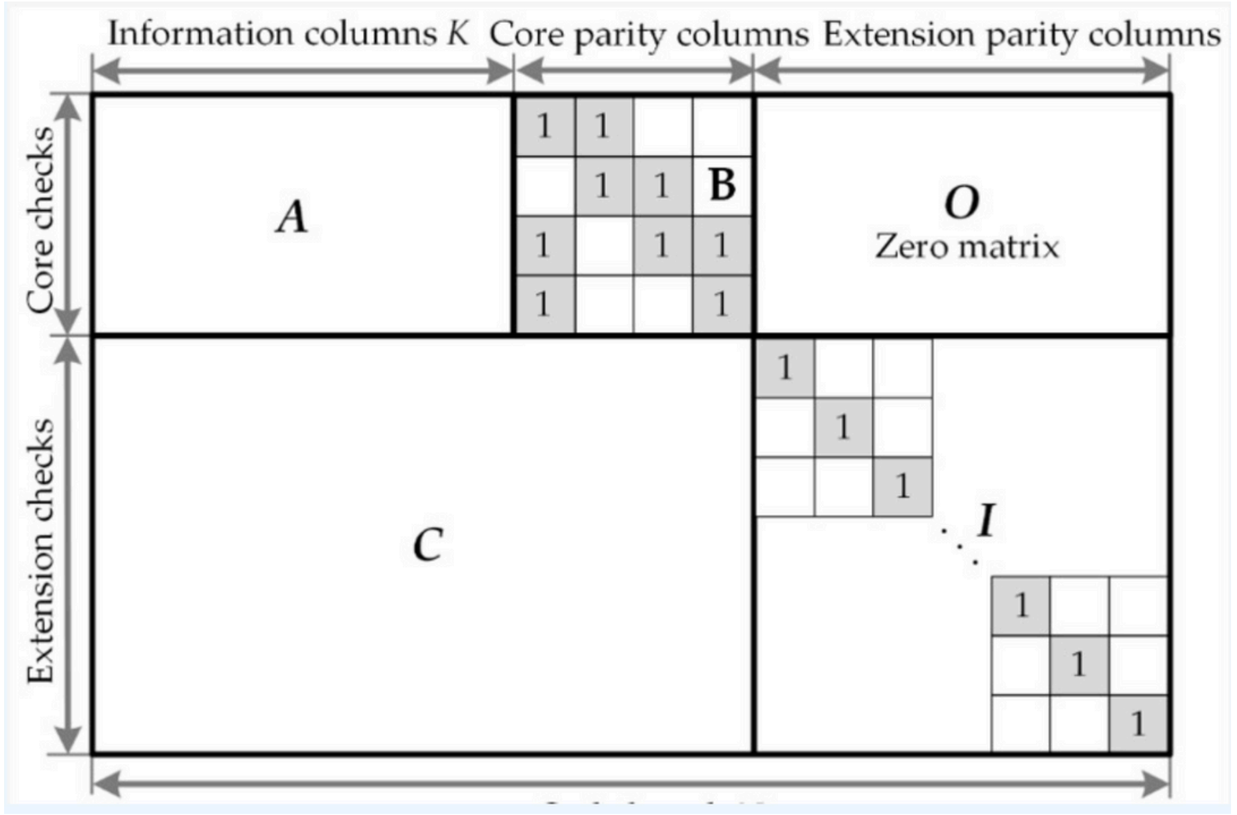
Low-Density Parity-Check (LDPC) codes are linear error-correcting codes introduced by Robert Gallager in the 1960s. They are defined by a sparse parity-check matrix (H matrix), which makes encoding and decoding computationally efficient.

LDPC codes are especially suited for modern communication systems like 5G NR, offering minimal latency, low decoding complexity, and very low error rates. Their adaptability to various block sizes and code rates stems from the use of rate-compatible base graphs in their design.

LDPC codes use parity-check matrices that can be regular or irregular. In regular LDPC codes, each row and column has the same number of ones, making them simpler but less flexible. Irregular LDPC codes allow varying numbers of ones in rows and columns, leading to better error correction performance, especially in modern systems. Though more complex to design, irregular codes are preferred in practice due to their efficiency near channel capacity.

Base Graph Matrix

A- Information Matrix
B- Double Diagonal Matrix
C- Less Dense Matrix
I- All Zero Diagonal Entries
O- All -1's Zero Matrices

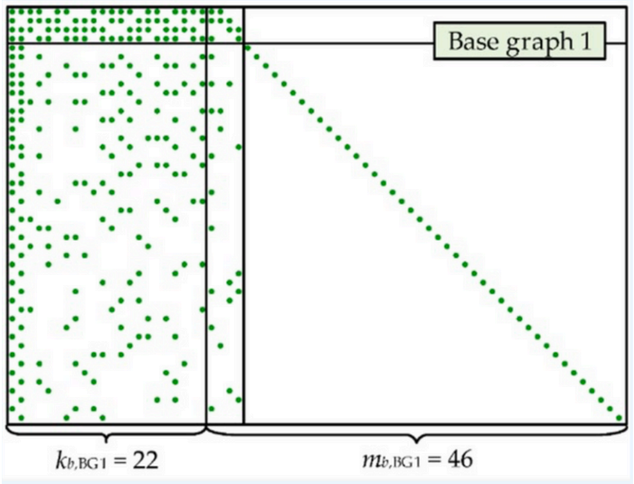


Base Graph Matrix

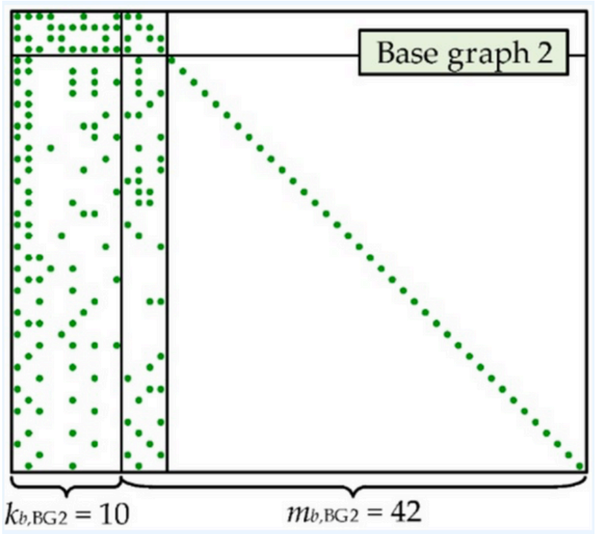
5G NR uses two base graphs for LDPC coding: BG1 (46×68) and BG2 (42×52), selected based on message size and code rate.

These graphs support up to 22z (BG1) and 10z (BG2) message bits, with a structured base matrix containing values like 0, -1, and numbers up to Z−1, where Z is the expansion factor.

The dual-graph design allows efficient adaptation to different data sizes and ensures reliable communication in diverse 5G scenarios.



(BG1): matrix size of 46x68



(BG2): matrix size of 42x52

H Matrix

- To construct the parity check matrix H from the base graph B, we use an expansion factor z (also called the lifting size). This factor varies based on the base graph.
- The construction rules are:
- If B_{ij} = -1, replace it with a z × z all-zero matrix.
- If B_{ij} = 0, replace it with a z × z identity matrix.
- If B_{ij} is between 1 and z−1, replace it with a z × z identity matrix circularly shifted to the right B_{ij} times.

H Matrix

Base Matrix, B = [

0 1 -1 3;

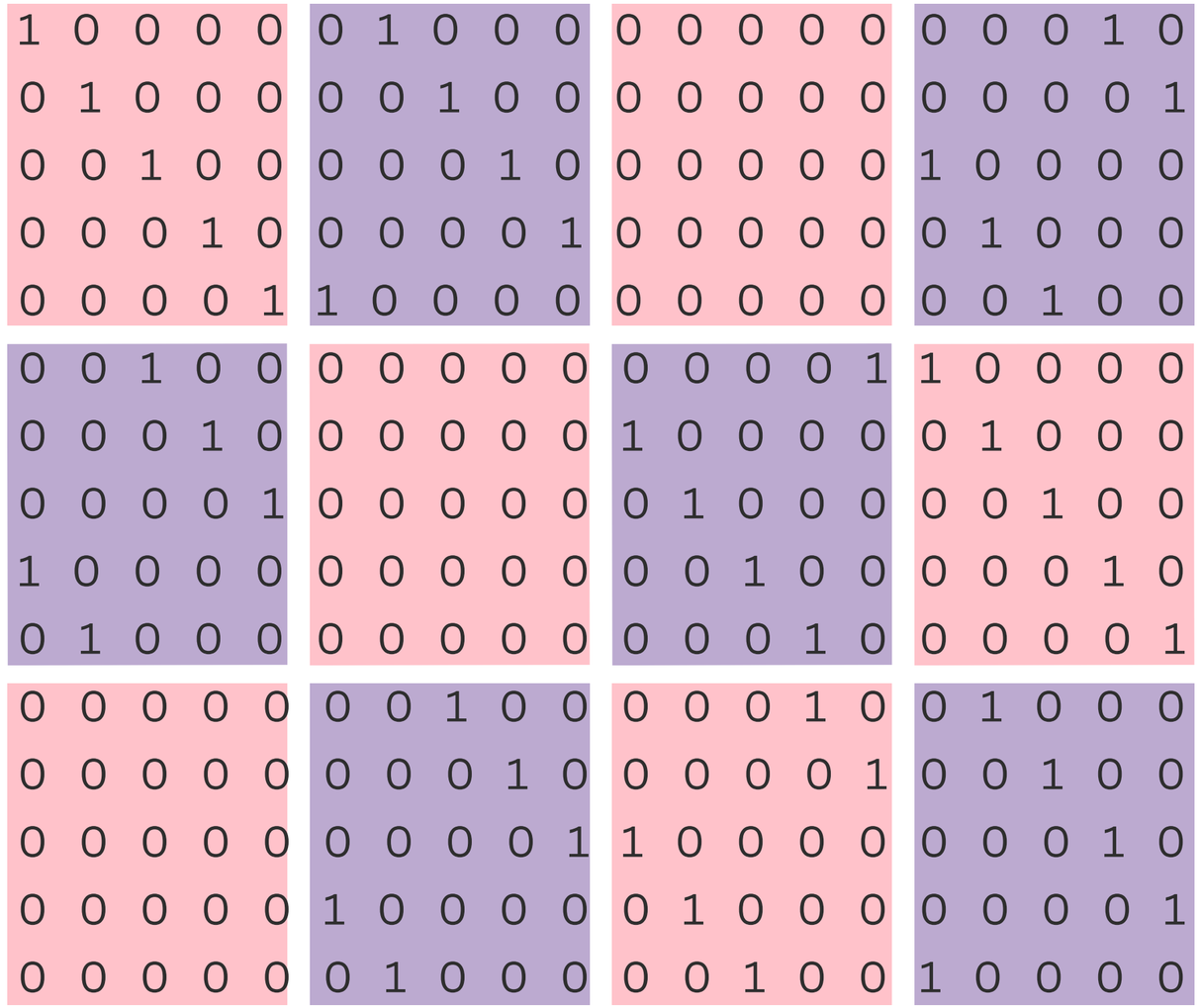
2 -1 4 0;

-1 2 3 1

];

Expansion Factor, Z = 5;

Parity Matrix, H =



Coderate matching (Puncturing)

$$\frac{(n-m)\cdot z}{(n-2)\cdot z - a\cdot z} = \frac{x}{y}$$

- Rate matching is used in LDPC codes to adjust the code rate to match the desired transmission rate. For a base matrix of size m × n and expansion factor z, the possible length of the codeword is n-z. Out of these n-z bits, the first 2z bits are punctured, leaving (n−2)·z bits available.

- To achieve a code rate of x/y, a·z bits are punctured from the available (n−2)·z bits. The number of message bits is calculated as (n−m)·z.

- The H matrix size is determined by the number of parity bits (calculated from the equation) and columns (calculated number + 2).

Encoding

The base matrix is given as follows:

$$H = \begin{bmatrix} I_1 & 0 & I_3 & I_1 & I_2 & I & 0 & 0 \\ I_2 & I & 0 & I_3 & 0 & I & I & 0 \\ 0 & I_4 & I_2 & I & I_1 & 0 & I & I \\ I_4 & I_1 & I & 0 & I_2 & 0 & 0 & I \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix}$$

Expansion: 5

Ik shows the 5 x 5 identity matrix
right shifted k times.
0 represents a 5 x 5 all zeros

Encoding

Equations and Encoding Process:

- Equations obtained:
  - ($l_1v_1 + l_3v_3 + l_4v_4 + l_5v_5 + l_6v_6 = 0$) (Equation 1)
  - ($l_2v_2 + l_4v_4 + l_6v_6 + l_7v_7 = 0$) (Equation 2)
  - ($l_4v_4 + l_5v_5 + l_6v_6 + l_7v_7 = 0$) (Equation 3)
  - ($l_1v_1 + l_2v_2 + l_7v_7 + l_8v_8 = 0$) (Equation 4)

- Combining the equations:
  - ($l_1v_5 = l_1v_1 + l_1v_4 + l_1v_2 + l_1v_3 + l_1v_5 + l_3v_3 + l_4v_4 + l_5v_5 + l_6v_6 + l_4v_4$)

- Message structure:
  - [($v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9$)] = [($m_1, m_2, m_3, m_4, m_5, p_1, p_2, p_3, p_4$)]

Encoding

- Rewriting the parity bit equation:
  - ($l_1p_1 = l_1m_1 + l_3m_1 + l_4m_1 + l_1m_2 + l_4m_2 + l_1m_3 + l_3m_3 + l_1m_4 + l_3m_4 + l_4m_4$)
- Solving for parity bits:
  - First, solve for (p_1) using the rewritten equation.
  - Substitute (p_1) into Equation 1 to find (p_2).
  - Continue solving to determine (p_3) and (p_4).
- Encoded message verification:
  - The encoded message must satisfy the equality (H*c = 0).

BPSK Modulation

The encoded output is passed to a BPSK modulator, which converts the matrix of encoded bits to the matrix of encoded symbols (1, -1). Specifically:

- 1 becomes -1
- 0 becomes 1

This process enables efficient transmission of the encoded message over a communication channel.
BPSK (Binary Phase Shift Keying) is a modulation technique used in LDPC encoding to represent binary data as symbols (1, -1). It is fundamental in ensuring that the encoded data, which consists of binary values, can be transmitted efficiently over noisy channels, where the reliability of error correction codes like LDPC becomes essential.
The encoded symbols are transmitted through an AWGN (Additive White Gaussian Noise) channel, which introduces noise into the system. The variance of the noise added is given by the formula:

BPSK Modulation

$$\sigma = \frac{1.0}{\sqrt{2 \cdot \left(\frac{E_\beta}{N_0}\right) \cdot r}}$$

E_β is the energy per bit.

N_0 is the noise power spectral density.

r is code rate

σ is the result of the calculation.

AWGN

- **Definition**: AWGN stands for Additive White Gaussian Noise:
  - Additive: Noise is added to the transmitted signal (or bit).
  - White: The noise follows a Gaussian distribution, meaning its values are spread in a predictable way, with a mean of zero.

- **Application**: In practical systems, AWGN is added to each transmitted bit. For example:
  - If a bit is 1 and a noise value of 0.4 is added, the result is 1.4.
  - Monte Carlo simulations are often used to simulate real-world conditions, running multiple iterations (e.g., 1,000) to ensure that the noise values are randomly applied to the matrices, so no two matrices have identical noise patterns.

Integration in Programs: AWGN is commonly integrated into simulation tools using functions like random.randn(), which generates random values with a normal (Gaussian) distribution.

Hard Decoding

- Hard decision decoding uses binary values (0 or 1) derived from the received signal and operates on the Tanner graph using iterative message passing. This approach is based purely on logic (XOR, majority voting), making it computationally efficient but less optimal than soft decision decoding

- The algorithm follows an iterative message passing process on the Tanner graph, where:
- Variable Nodes (VNs) represent bits of the codeword.
- Check Nodes (CNs) represent the parity constraints.
- Each iteration consists of the following 3 major steps:
    1. VN → CN Message Passing (Initialization & Updates)
    2. CN → VN Message Passing (Parity Check Updates)
    3. Bit Decision (Computing ĉ)

Hard Decoding

- **1.VN → CN Message Passing (Initialization & Updates)**
- First Iteration:

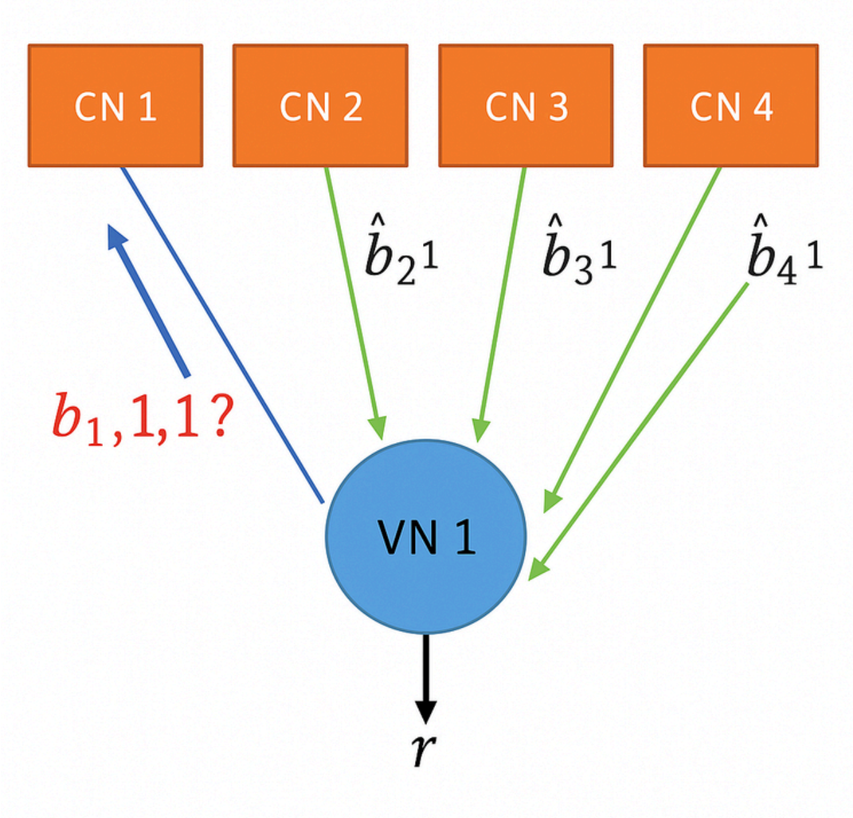
Each Variable Node (VN) sends the received hard value (from channel) to all its connected Check Nodes(CNs).

- Subsequent Iterations:

A VN computes the majority of all messages it has received from connected CNs (except the CN it's sending to).

This avoids message loops and provides extrinsic information.

The VN then sends the result of the majority voting to that CN.



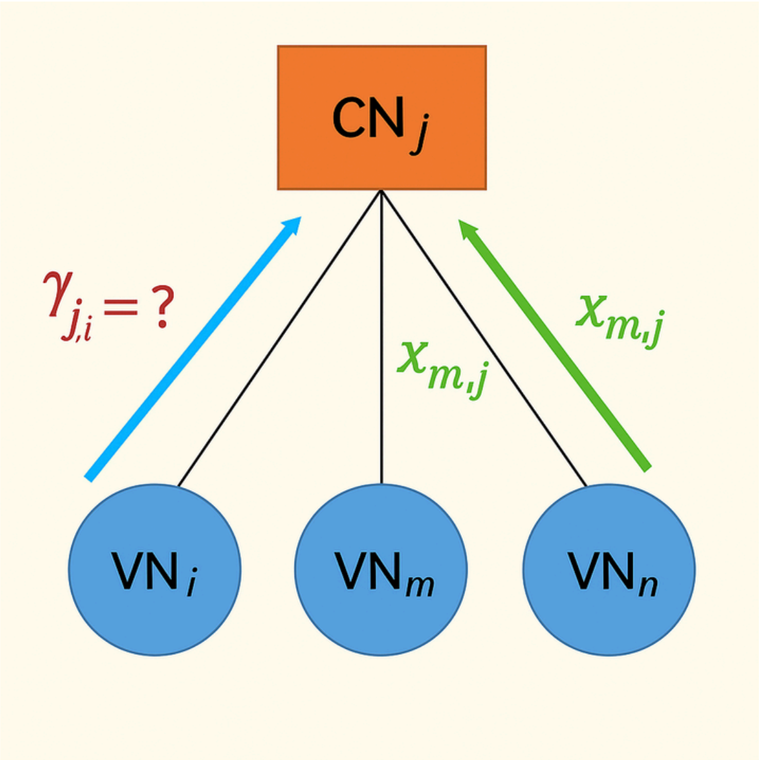
Hard Decoding

- **2.CN → VN Message Passing (Parity Check Updates)**

- Each Check Node implements a Single Parity Check (SPC).

- It receives values from all connected VNs.

- To send a message back to a VN, the CN:
  Computes the XOR of all incoming bits except the one from the target VN.
  Sends this XOR result back as its belief about the target VN's bit.



Hard Decoding

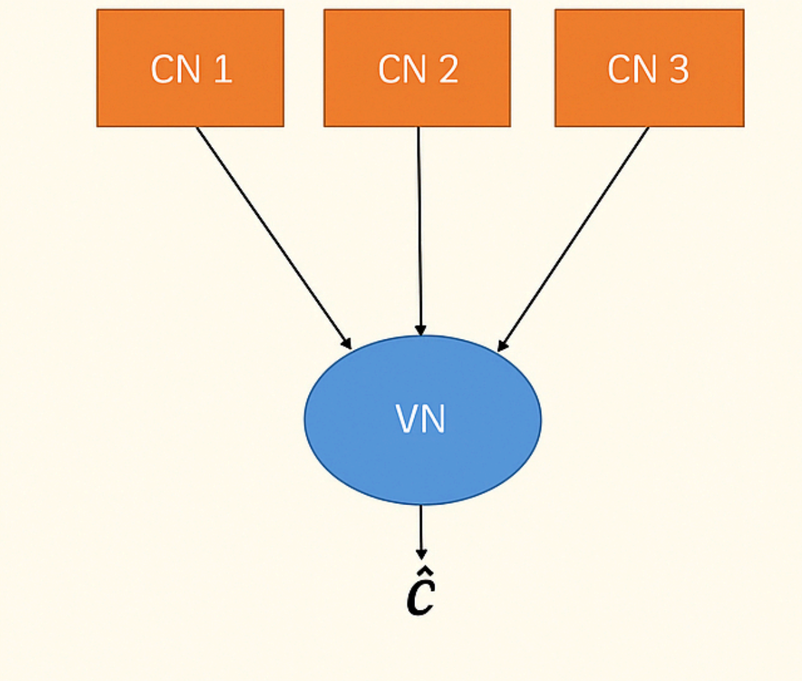
- **3.Bit Decision (Computing ĉ)**
- After CN-to-VN messages are received:

  Each VN again performs majority voting on:

  All messages from its connected CNs.
  Its originally received bit.

,

- The result is used to update the estimate of the decoded word ĉ



Soft Decision Decoding

Minimum Sum Approximation Algorithm (Min-Sum):

- **Soft decision decoding** uses real valued inputs i.e. **LLRs** instead of binary decisions to improve decoding accuracy.

- The **LLRs** are received in decoder and are calculated as:

$$L(c_i) = log\frac{P_r(c_i = 0|y_i)}{P_r(c_i = 1|y_i)}$$

- The **log-likelihood ratio (LLR)**, gives us a measure of how likely the next received bit is to be a 0 or a 1. To estimate this using the **Min-Sum Approximation**, we start with the **sum-product algorithm**. This algorithm involves applying the **inverse hyperbolic tangent ($\tanh^{-1}$)** function to the product of individual LLRs across bits, iterating over *n values*. The result is a sum that reflects the combined probabilities of these bits. Since $\tanh^{-1}$ can be expressed as a log function, it simplifies the computation. The min-sum approximation then comes in to further refine this output, using an approximation technique to simplify the results while maintaining accuracy.

Soft Decision Decoding

Minimum Sum Approximation Algorithm (Min-Sum):

- **VN → CN** Updates after first iteration.
- Here, after a CN sends a message to VN, each VN then calculates the next message which it will pass to every connected CN which is computed using the following expression:

$$L(q_{ij}) = L(c_i) + \sum_{j' \in C_i \setminus j} L(r_{j'i})$$

- This process continues iteratively, refining estimates.

Soft Decision Decoding

Minimum Sum Approximation Algorithm (Min-Sum):

- **VN → CN Transmission:**
  Each **VN** sends messages to connected **CNs** using the formula:

$$L(r_{ji}) = log\frac{r_{ji}(0)}{r_{ji}(1)} = 2tanh^{-1}(\prod_{i' \in V_{j}/i} tanh(\frac{1}{2}L(q_{i'j}))) = (\prod_{i' \in V_{j}/i} a_{ij})\,\varphi(\sum_{i' \in V_{j}/i} \varphi(\beta_{i'j}))$$

- Here the main problem would be that computing **tanh** would be much complex and this is where **Min-Sum Approximation** helps us.
- The sum of φ-values can be approximated by the smallest φ-value:

$$\sum \varphi(\beta_{ij}) \approx \varphi(\min(\beta_{ij}))$$

- Message from **CN** to **VN** then becomes:

$$L(r_{ji}) = \prod \alpha_{i'} \cdot \min(|\beta_{i'}|)$$

Soft Decision Decoding

Minimum Sum Approximation Algorithm (Min-Sum):

- **Final Bit Decisions**

- After each round of message passing between the **VNs** and **CNs**, we update our best guess for each bit. This is done by combining all the incoming messages to each variable node, giving us the most up-to-date belief about its value.
- To compute this, we define total LLR for bit *Ci* as:

$$L(Q_i) = L(c_i) + \sum_{r_j \in C_i} L(r_{ji})$$

- This total LLR value gives us an updated estimate of likelihood that bit *Ci* is *0* or *1*.
- Using this we make a hard decision for as follows for each bit:

$$\hat{c}_i = \begin{cases} 1, & \text{if } L(Q_i) < 0 \\ 0, & \text{otherwise} \end{cases}$$

Soft Decision Decoding

Minimum Sum Approximation Algorithm (Min-Sum):

- **Final Bit Decisions (continue)**
- This total LLR value gives us an updated estimate of likelihood that bit *Ci* is *0* or *1*.
- Using this we make a hard decision for as follows for each bit:

$$\hat{c}_i = \begin{cases} 1, & \text{if } L(Q_i) < 0 \\ 0, & \text{otherwise} \end{cases}$$

- Here,
  - ➔ If the **total LLR is -ve** → the bit is **more likely to be 1.**
  - ➔ If the **total LLR is 0(zero) or +ve** → the bit is **assumed to be 0.**
- The decoding stops when:
  - ➔ The maximum number of iterations has been reached
    - ➔ The estimated codeword $\hat{C}$ satisfies the parity-check condition i.e. $H \cdot \hat{C} = 0$

meaning it's a valid codeword according to the code's structure.

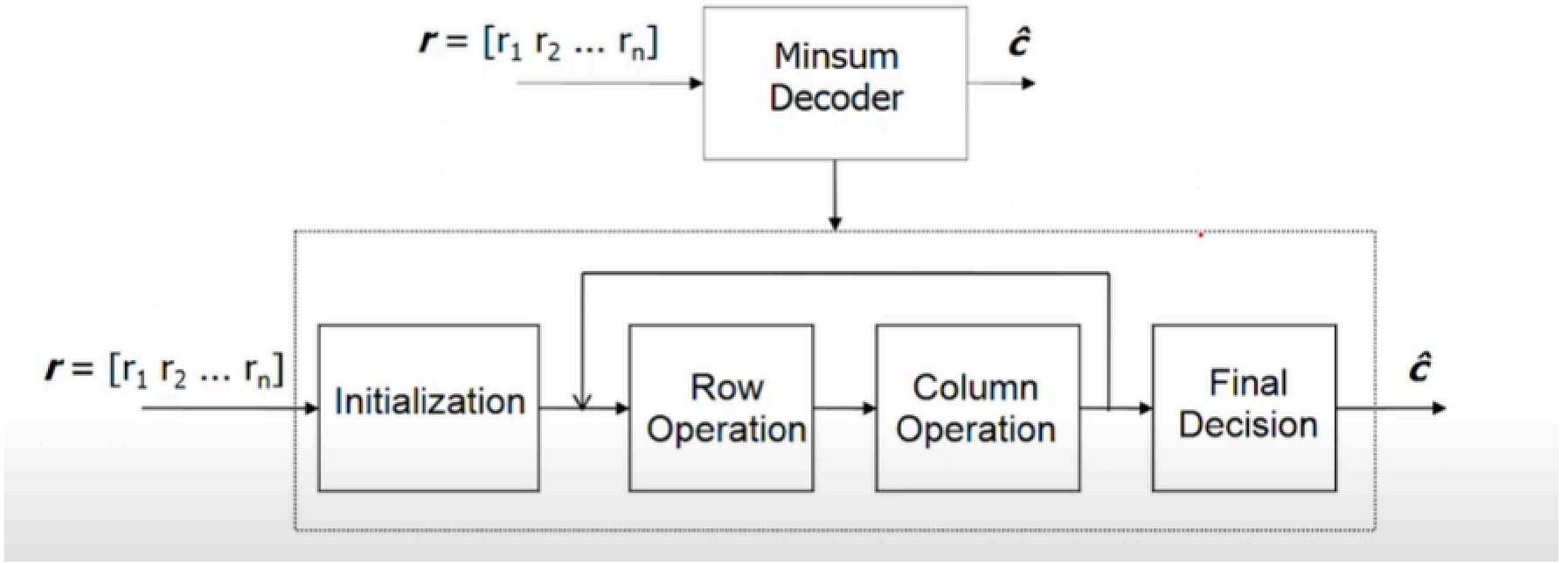
Soft Decision Decoding

Step-by-Step Implementation

- To implement the soft decision decoding algorithm, we start by using the **parity check matrix H**, where each row corresponds to a **CN** and each column to a **VN**. A value of **1** at position *(i, j)* in the matrix indicates a connection between CN *i* and VN *j*. Using this structure, we generate a new matrix called L, where every 1 in the H matrix is replaced with the received **LLR**, effectively initializing each variable node with the observed values from the channel.
- Once the **L matrix** is formed, we begin message passing between CNs and VNs. This is done by applying the **Min-Sum Approximation**: for each row in **L**, we replace each non-zero entry with the **minimum of the other non-zero values in the same row**, and adjust the sign based on the product of signs. After this CN to VN step, we update messages from VNs to CNs by summing values in each column of the L matrix.
- The updated values give us the new **LLRs**, which help in estimating the most likely decoded bits **Ĉ**. To check if decoding is complete, we compute the matrix product **H x Ĉ** if the result is zero or we've hit the iteration limit, the process stops, otherwise it continues.

Example for better Understanding

Min Sum decoder



Example for better Understanding

Suppose we have the H matrix as

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

received vector is :

$$\boldsymbol{r} = \begin{bmatrix} 0.2 & -0.3 & 1.2 & -0.5 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

We make the L matrix as :

$$L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

Example for better Understanding

For each row,

Magnitude

- Min1 = minimum absolute value of all nonzero entries in row
- Min2 = next higher absolute value
- Set magnitude of all values (except minimum) = Min1
- Set magnitude of minimum value = Min2

Sign

- Parity = product of signs of entries in row
- New sign of an entry = (Old Sign) × (Parity)

Example for better Understanding

Before row operation

$$L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

After row operation

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

Example for better Understanding

- For each column,
  - New values
    - $Sum_j = r_j$ + sum of all entries in Column j
    - New Entry = Sum – (Old entry)

Example for better Understanding

After Column Operation

$$r = [\ 0.2 \quad -0.3 \quad 1.2 \quad -0.5 \quad 0.8 \quad 0.6 \quad -1.1]$$

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

$$\text{Sum} = [\ -1 \quad -0.4 \quad 1.1 \quad -0.6 \quad 0.4 \quad 0.7 \quad -0.7\ ]$$

The updated beleif after 1st iteration:

$$L = \begin{bmatrix} -0.7 & -0.6 & 1.3 & 0 & 0.6 & 0 & 0 \\ 0 & 0.1 & 0.8 & -0.3 & 0 & 0.4 & 0 \\ -0.7 & -0.6 & 0 & -0.8 & 0 & 0 & -0.9 \\ -0.4 & 0 & 1.3 & 0 & 0.6 & 0.9 & -0.9 \end{bmatrix}$$

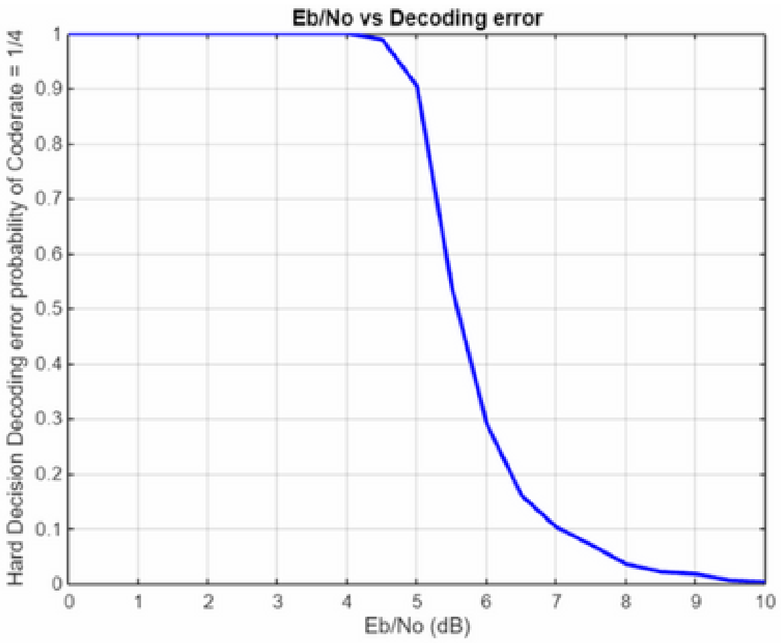
The updated L matrix is:

We then convert the new vector sum to 0s and 1s and compare it with original message to check if it is decoded successfully. This continues in a loop.

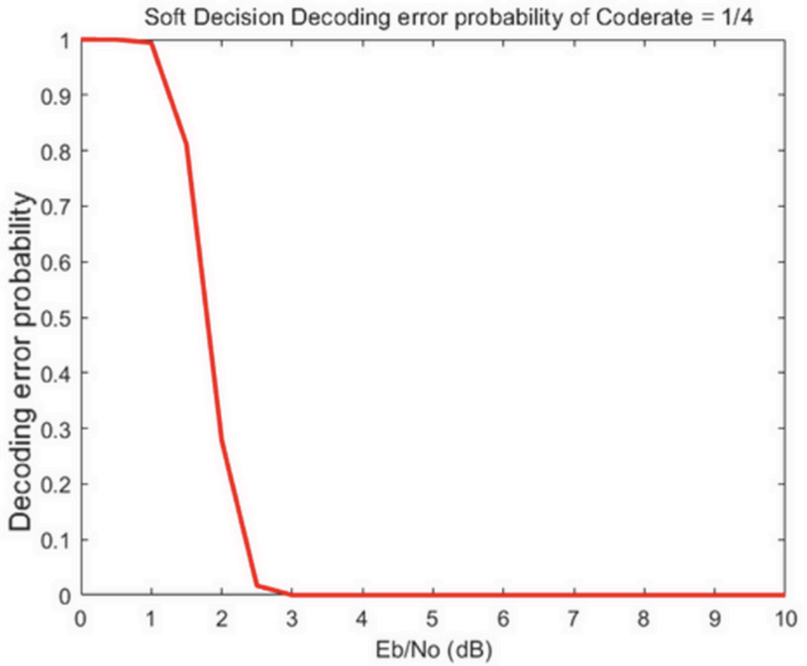
RESULTS OF MATRIX NR_2_6_52

CODE RATE = 1/4

HARD DECISION DECODING

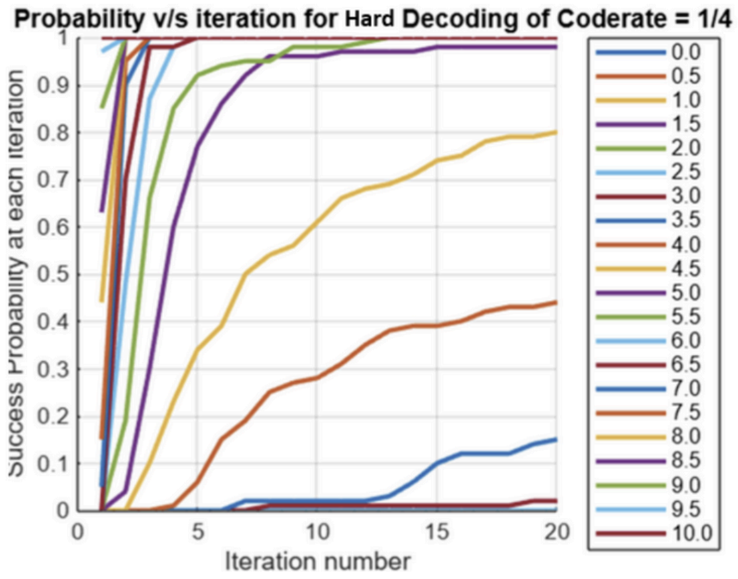


SOFT DECISION DECODING

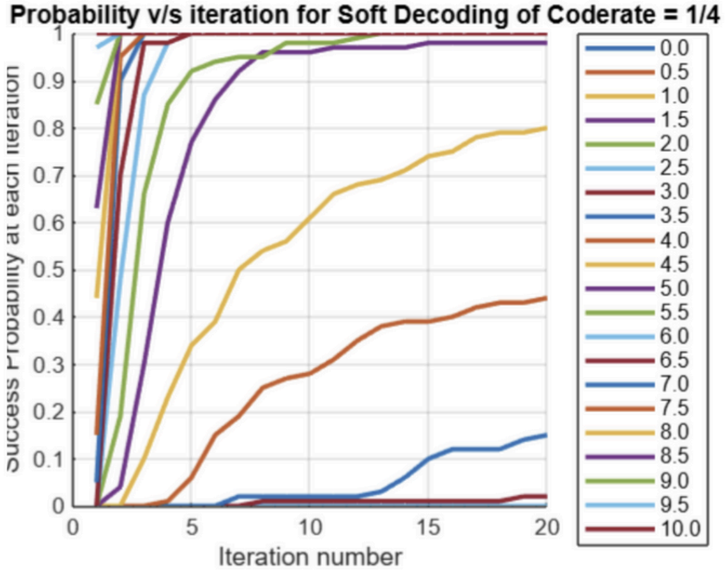


CODE RATE = 1/4

HARD DECISION DECODING

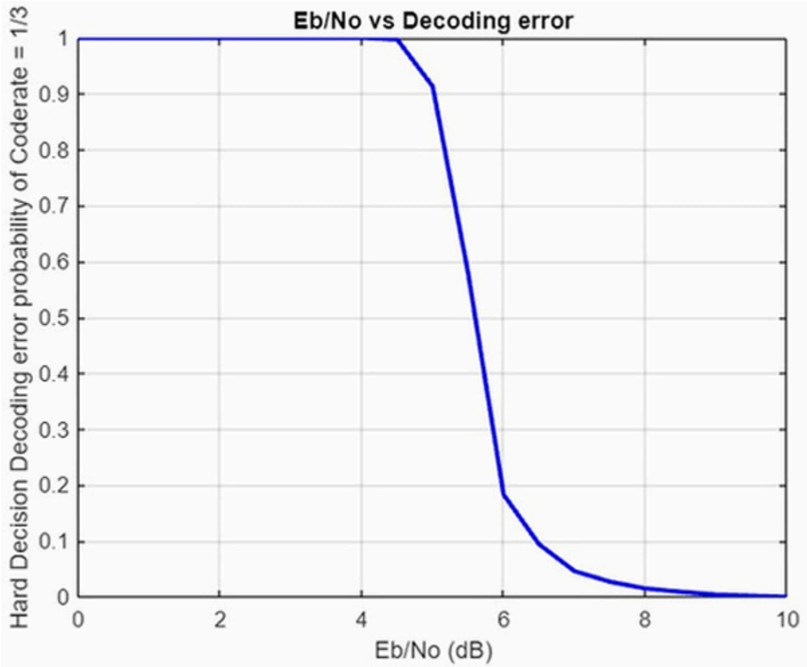


SOFT DECISION DECODING

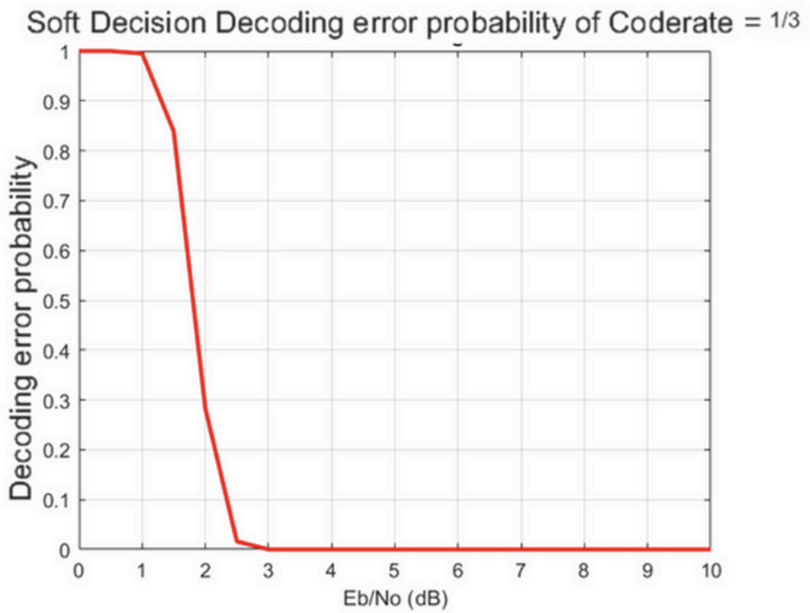


CODE RATE = 1/3

HARD DECISION DECODING

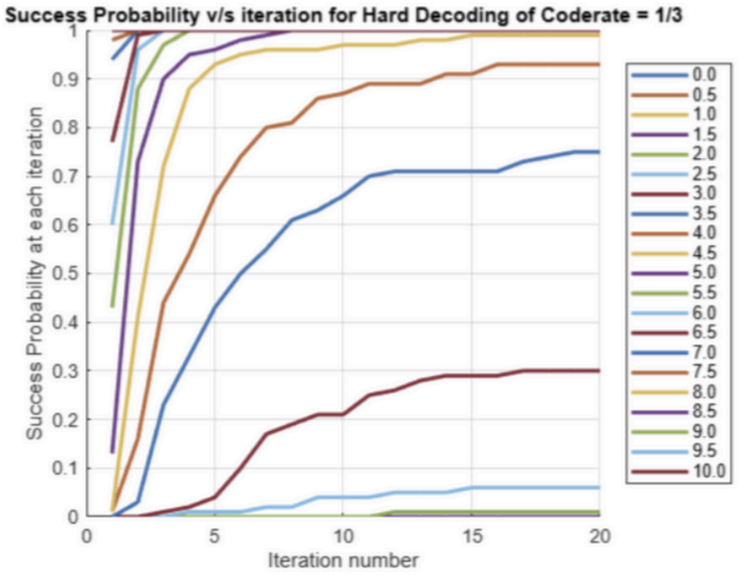


SOFT DECISION DECODING

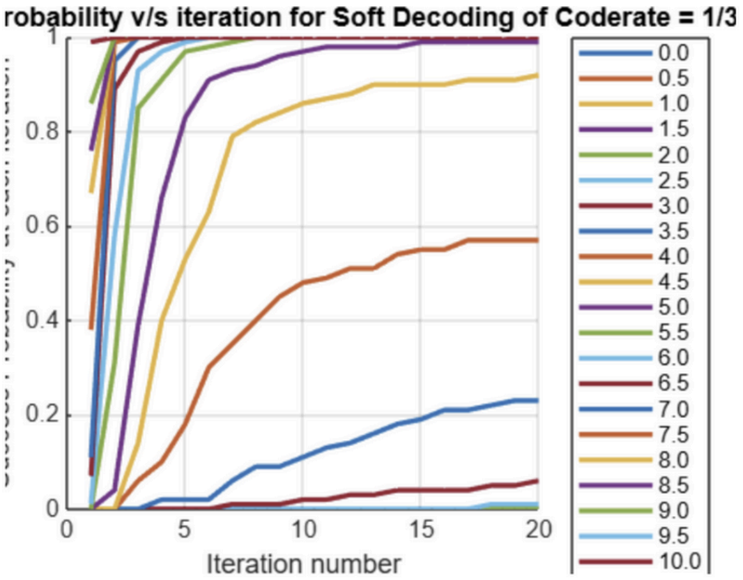


CODE RATE = 1/3

HARD DECISION DECODING

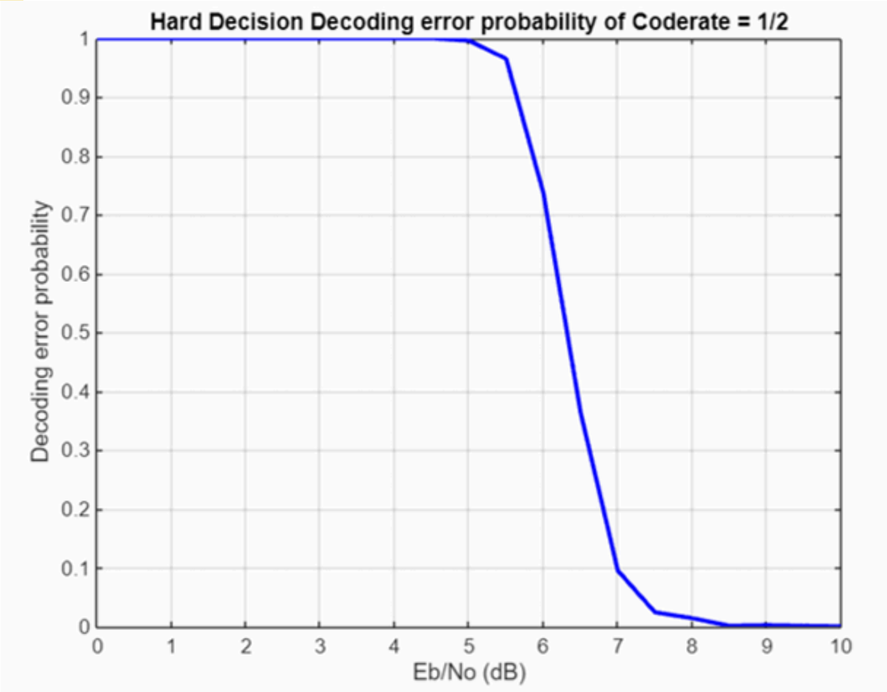


SOFT DECISION DECODING

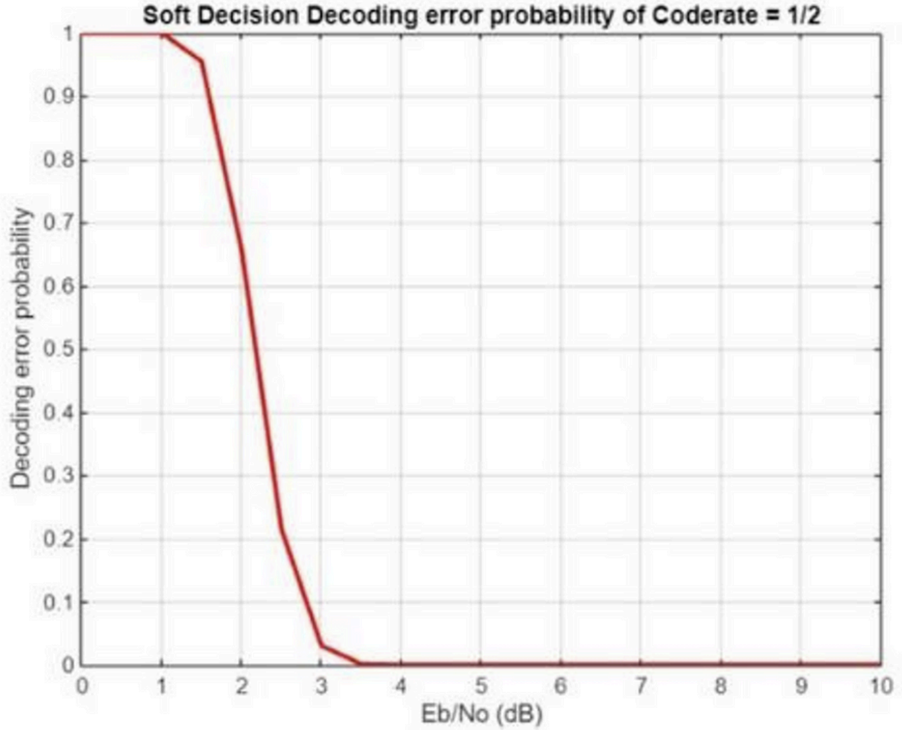


CODE RATE = 1/2

HARD DECISION DECODING

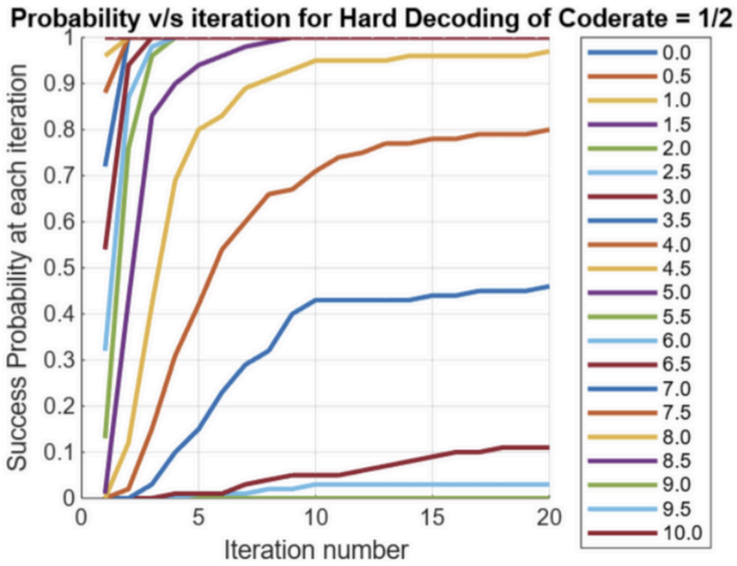


SOFT DECISION DECODING

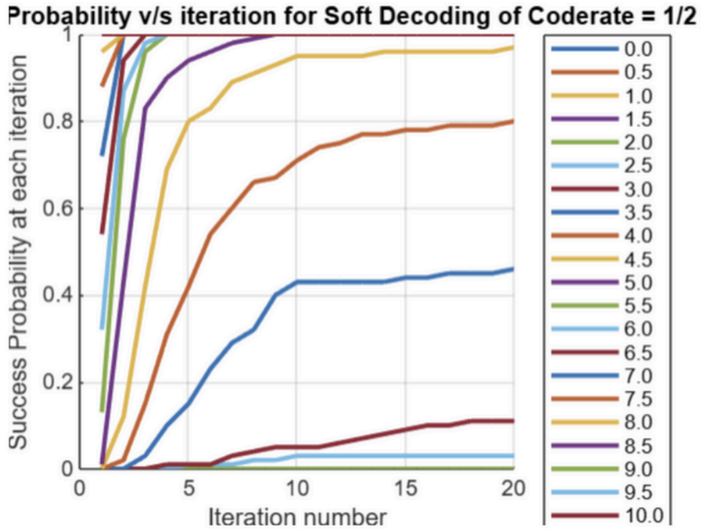


CODE RATE = 1/2

HARD DECISION DECODING

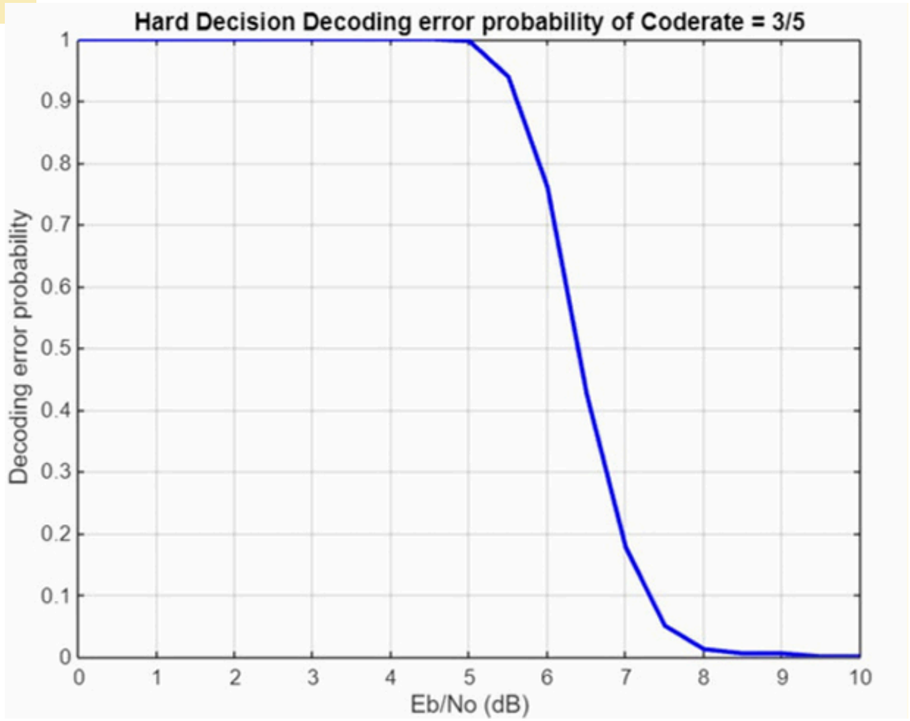


SOFT DECISION DECODING

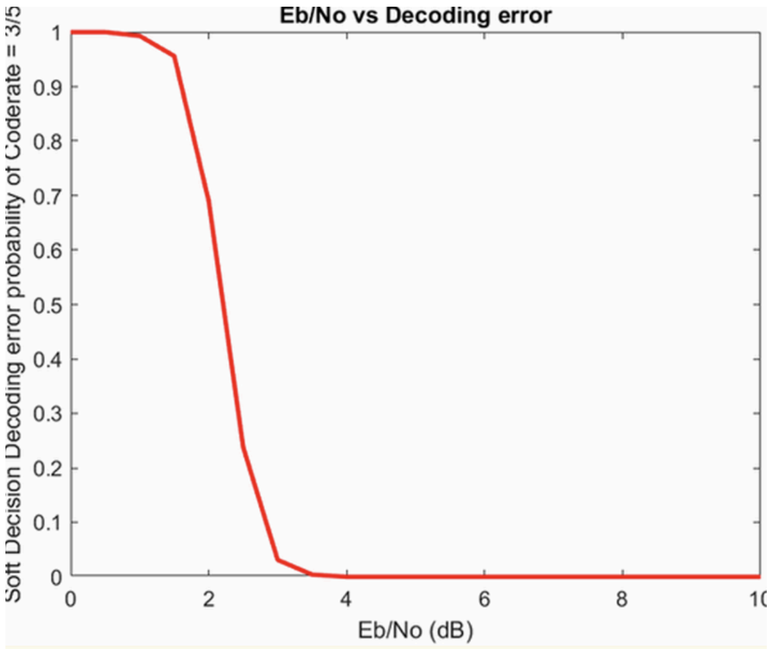


CODE RATE = 3/5

HARD DECISION DECODING

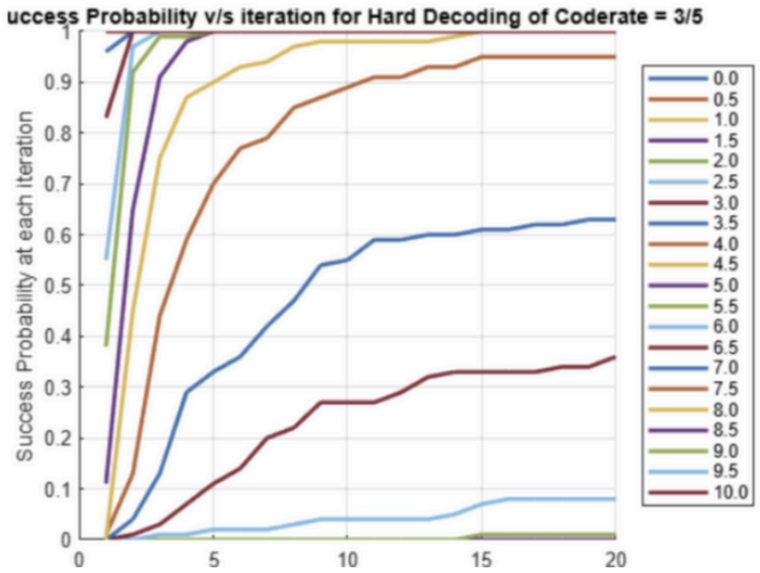


SOFT DECISION DECODING

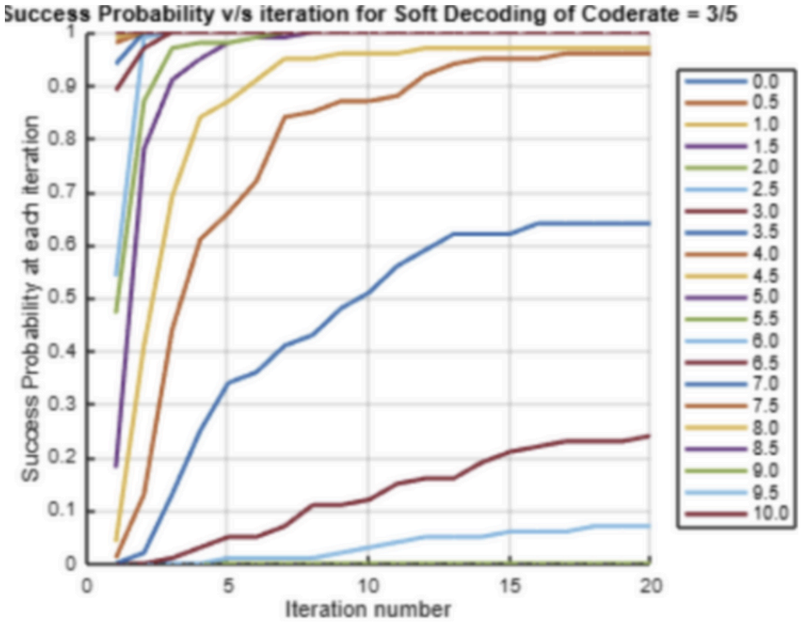


CODE RATE = 3/5

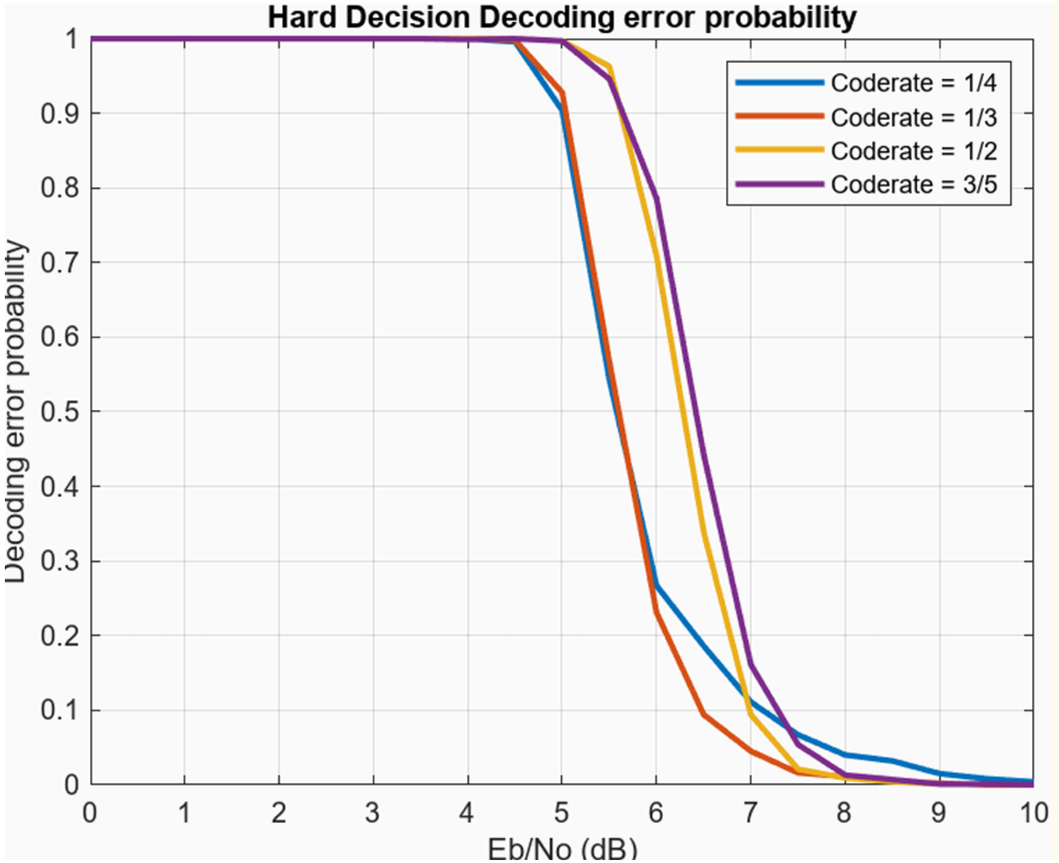
HARD DECISION DECODING



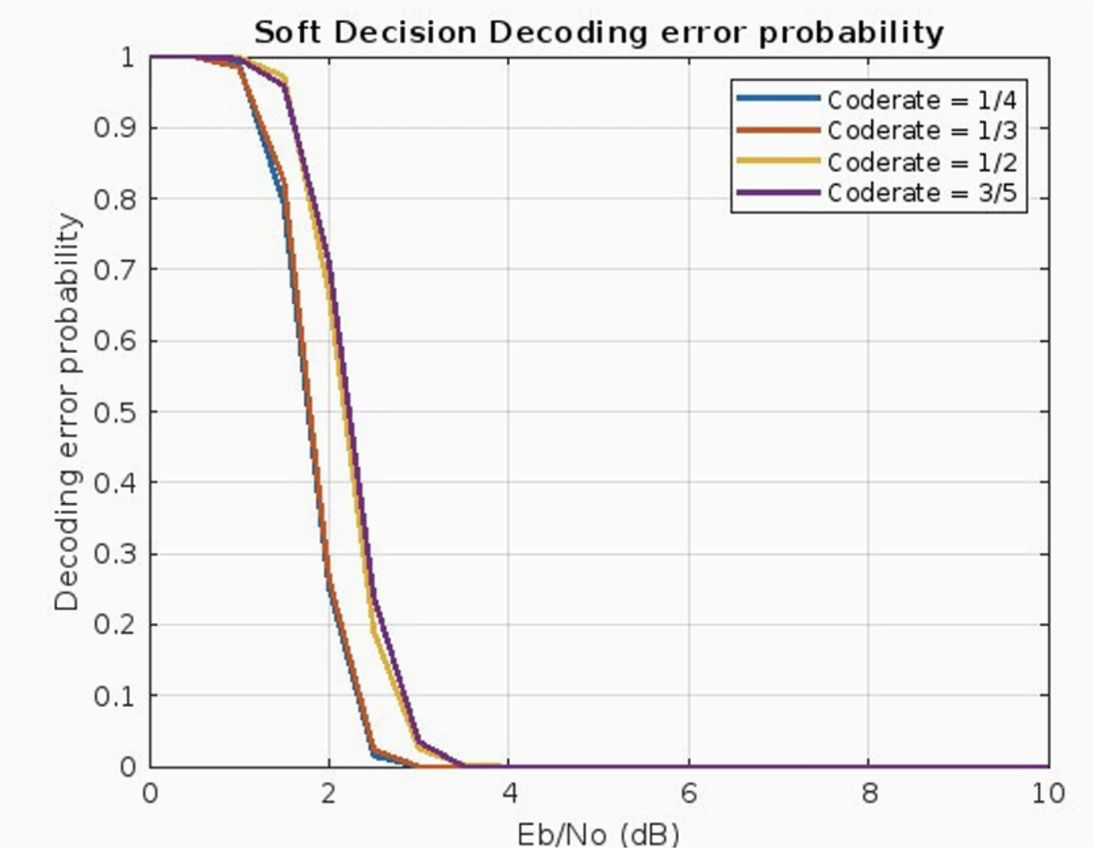
SOFT DECISION DECODING



HARD DECISION DECODING

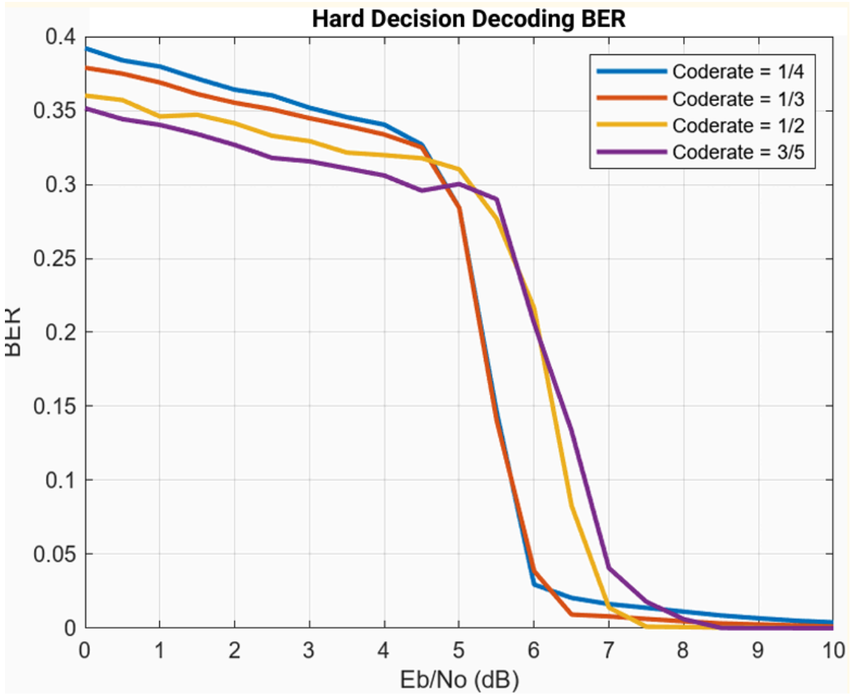


SOFT DECISION DECODING

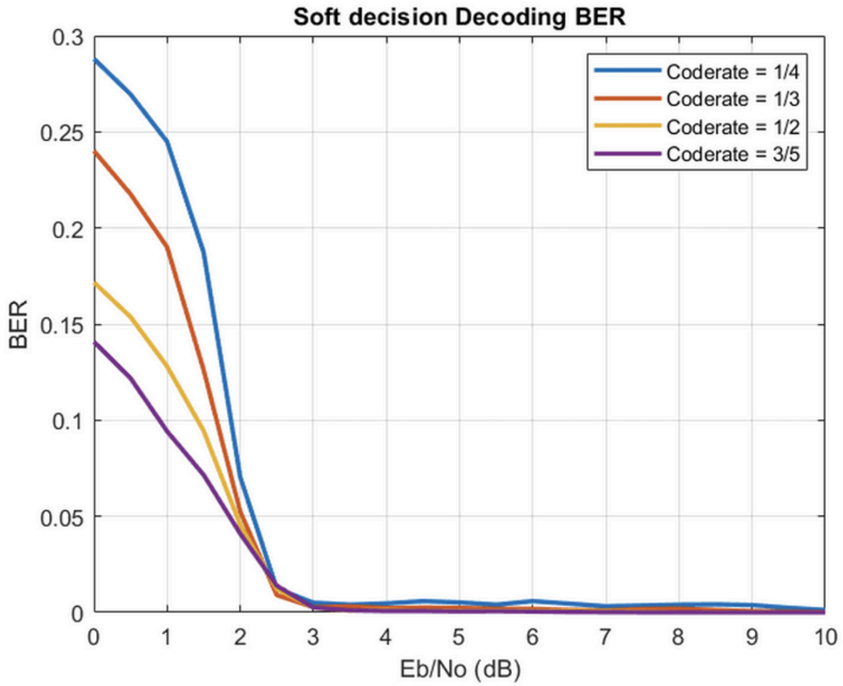


Bit Error Rate

HARD DECISION DECODING



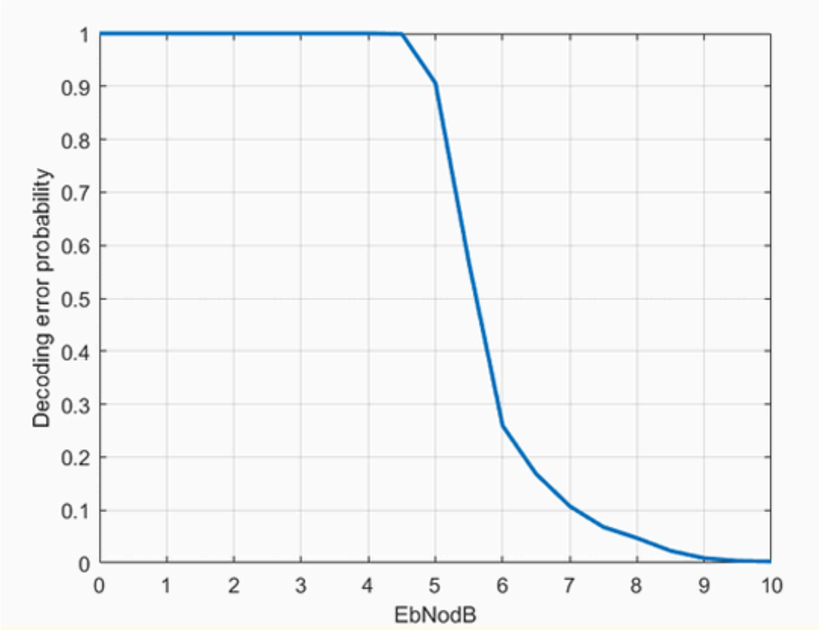
SOFT DECISION DECODING



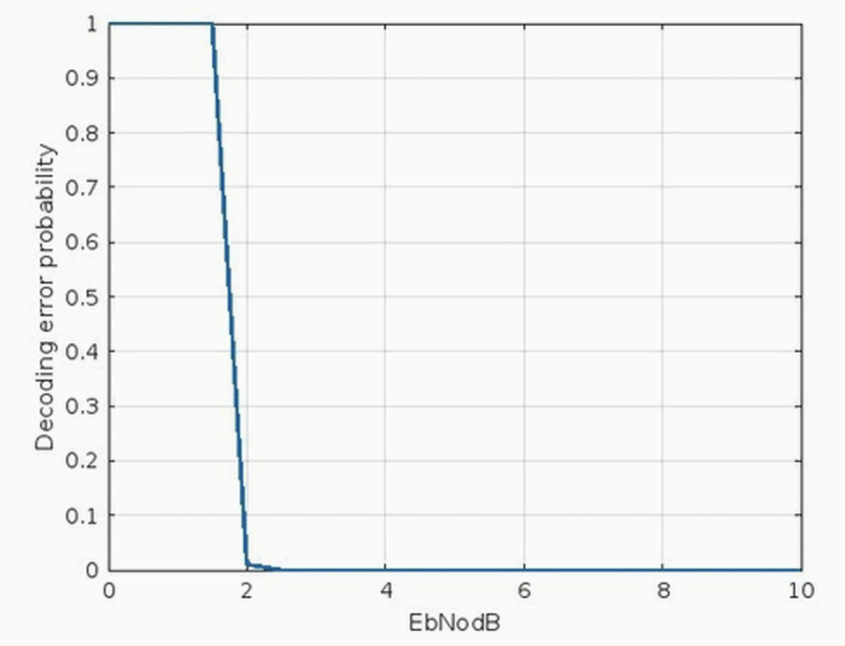
RESULTS OF MATRIX NR_1_5_352

CODE RATE = 1/3

HARD DECISION DECODING

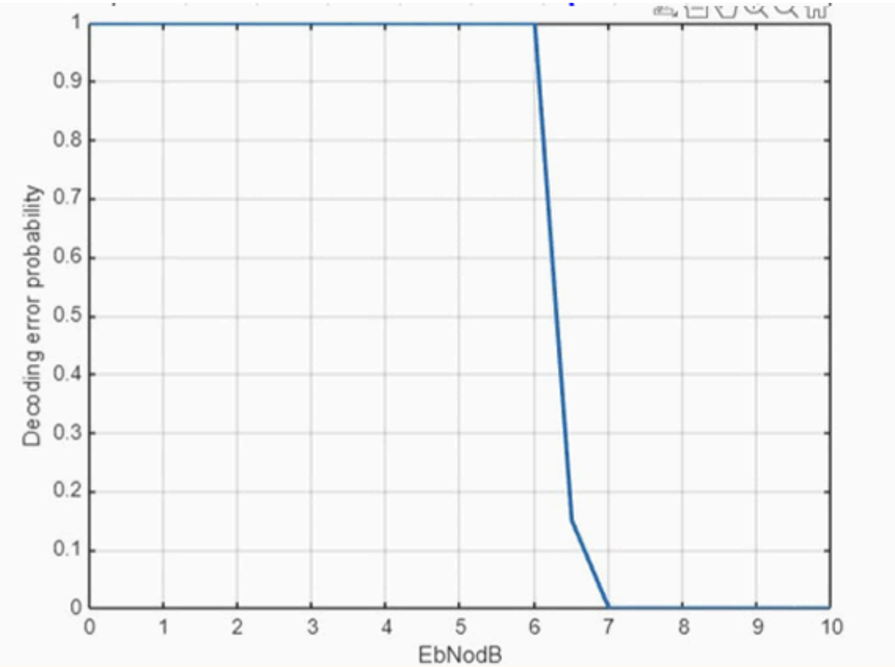


SOFT DECISION DECODING

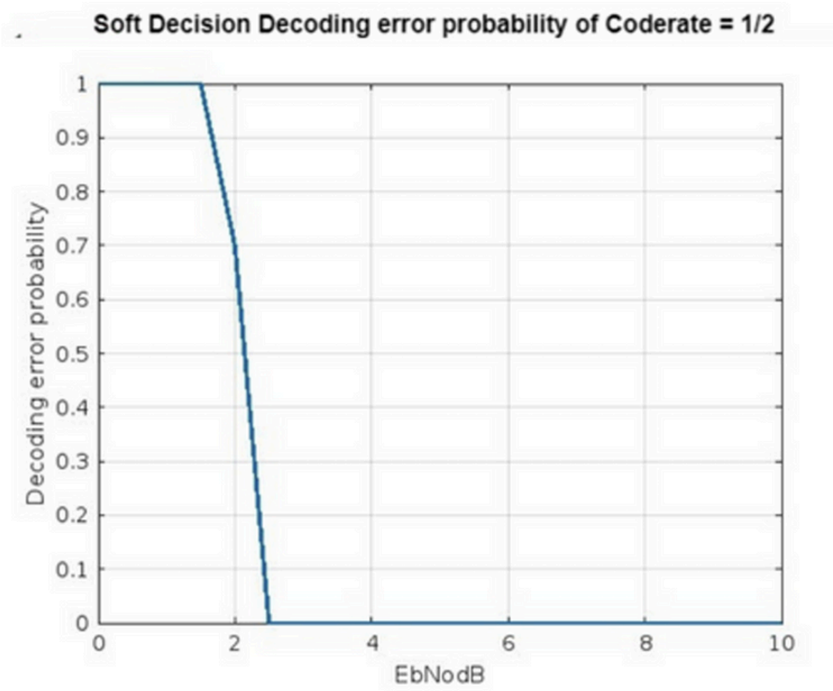


CODE RATE = 1/2

HARD DECISION DECODING

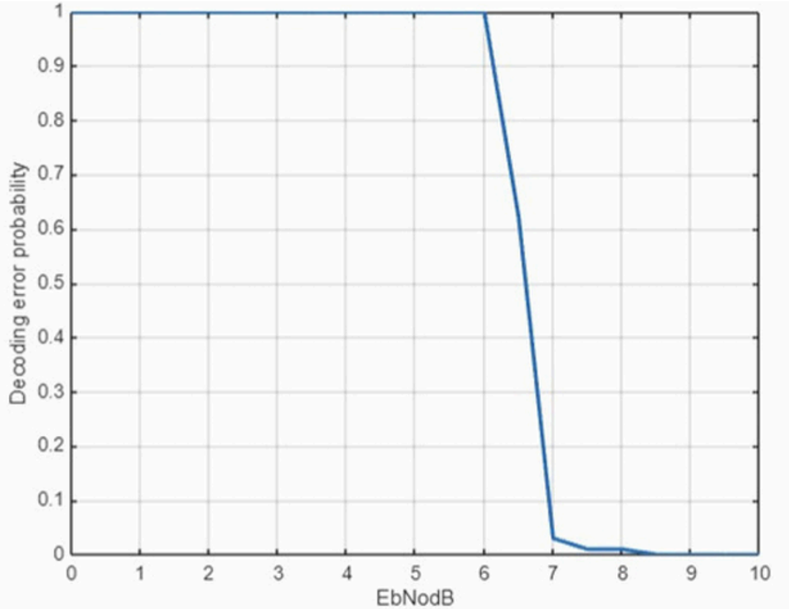


SOFT DECISION DECODING

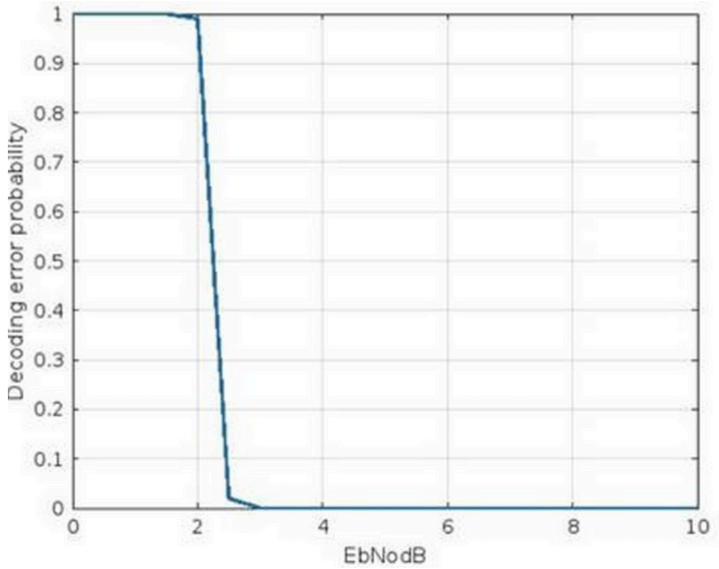


CODE RATE = 3/5

HARD DECISION DECODING

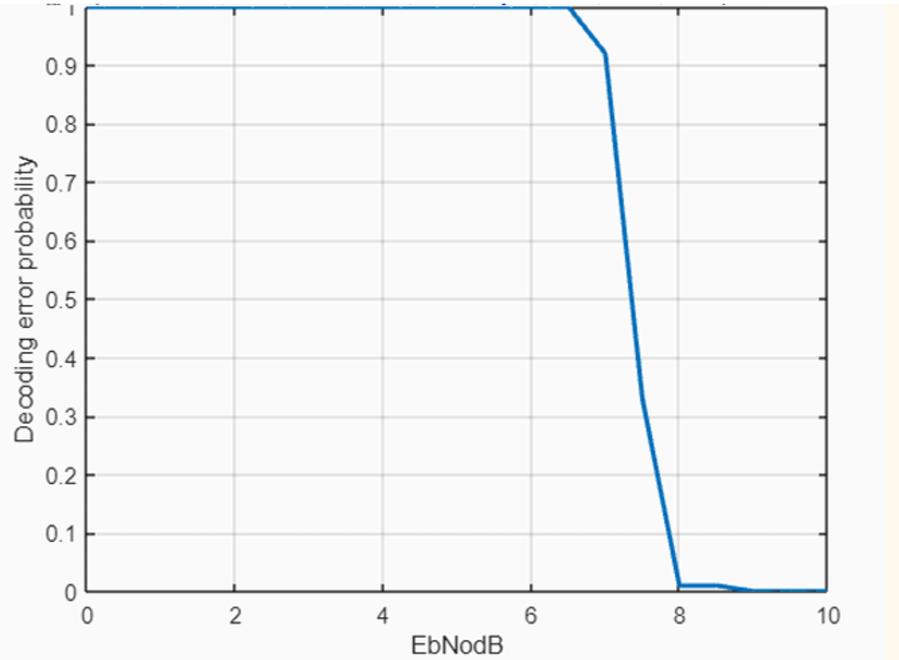


SOFT DECISION DECODING

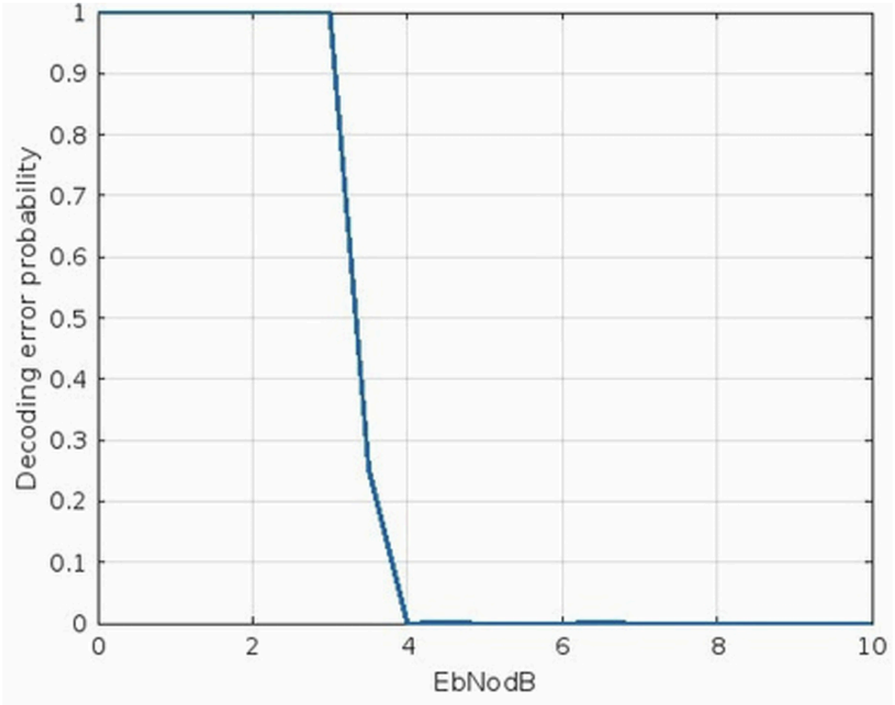


CODE RATE = 4/5

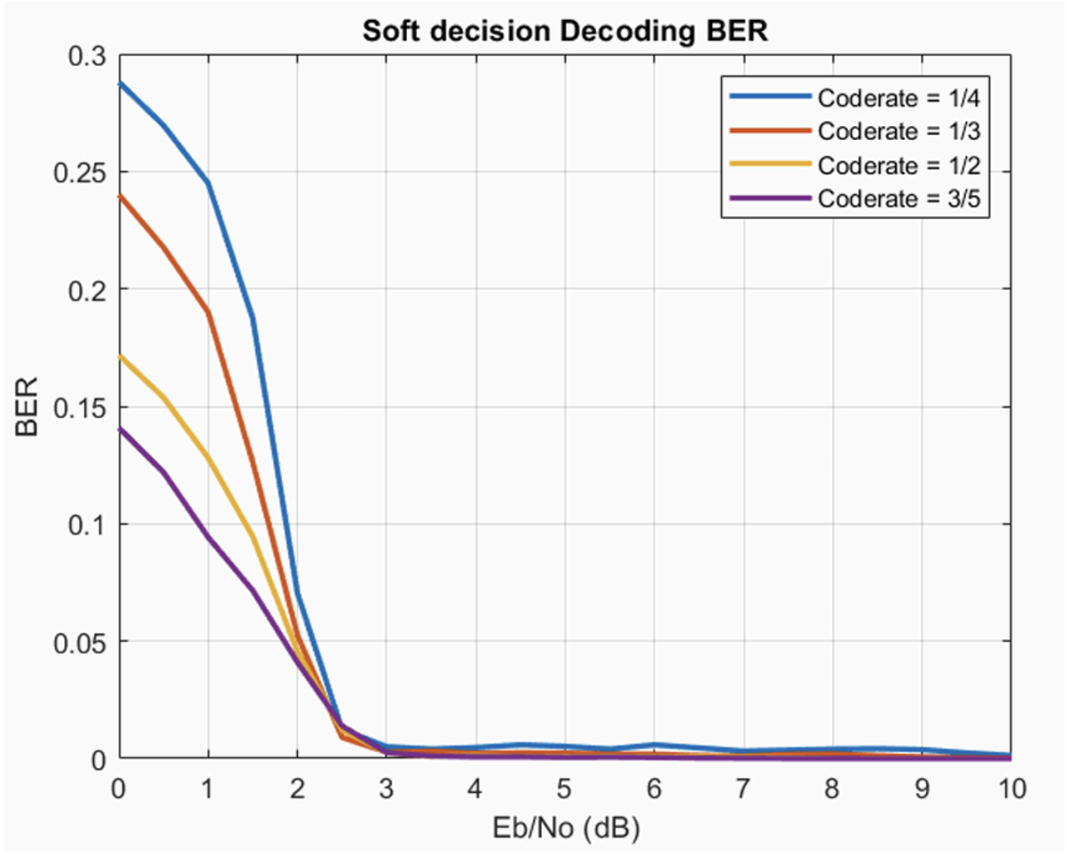
HARD DECISION DECODING



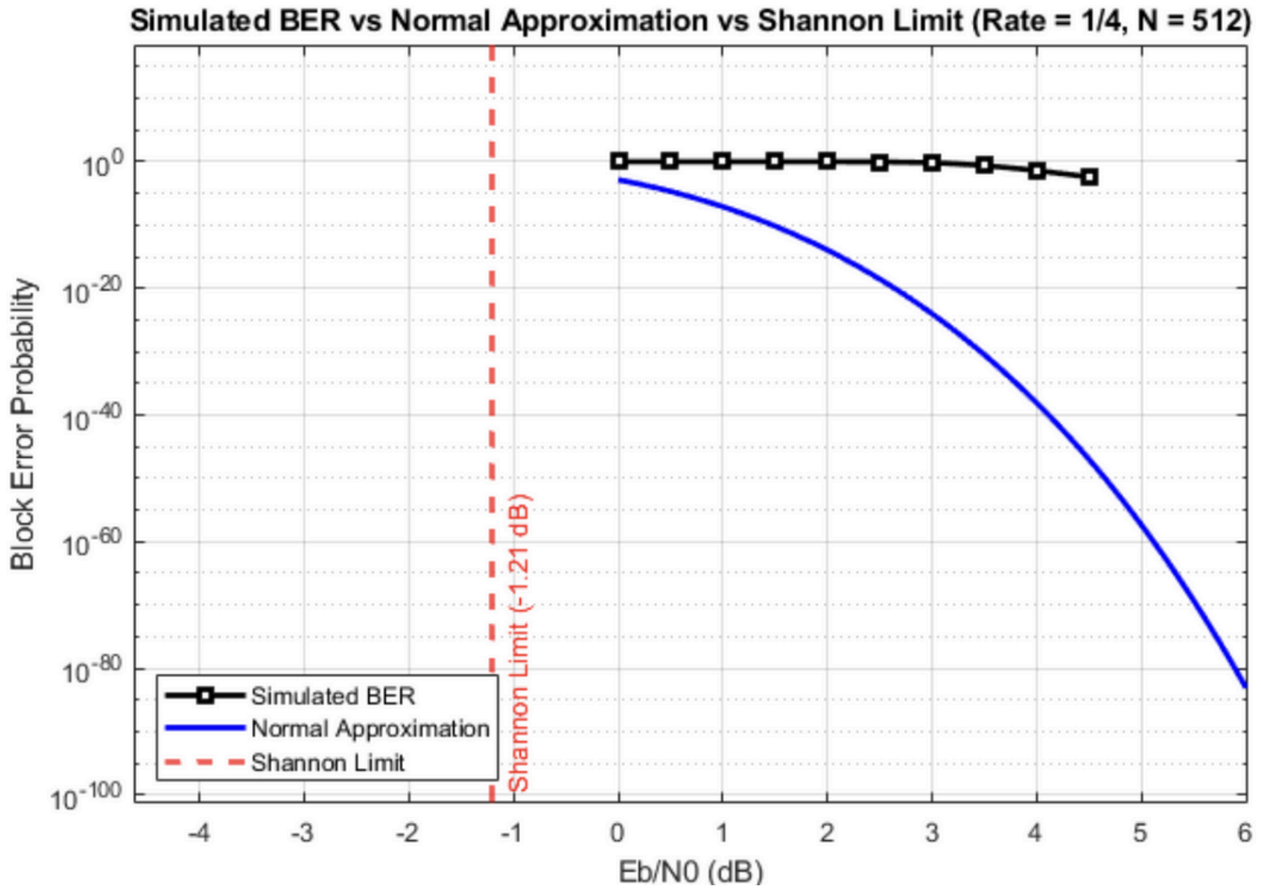
SOFT DECISION DECODING



SOFT DECISION DECODING



COMPARISON WITH SHANNON



Bibliography

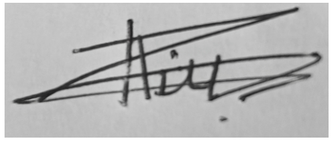
- Lecture Slides by Professor Yash Vasavada CT216 - Winter 2025.
- Video Lectures of NPTEL-NOM IITM by Prof. Andrew Thangaraj on LDPC codes.
- Implementation of Low-Density Parity-Check codes for 5G NR shared channels by LIFANG WAN

Team Members

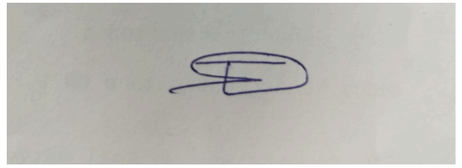
202301078	KRISH BHAVESHKUMAR PATEL
202301079	KUSHAL BHUPTANI
202301080	HARSHVIR SINGH
202301081	HET RANK
202301082	AAYMAN AMMAR SHAMS
202301083	DHAMELIYA UTSKER ARJANBHAI
202301084	AYUSH PATEL
202301085	KAVYA PARMAR
202301086	RITUL PATEL
202301087	AGRIM SHARMA
202301088	SANTOKI TAPAS ASHOKBHAI

Kavya Parmar

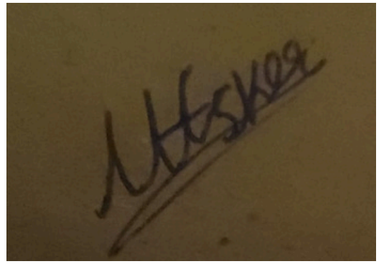
Agrim Sharma



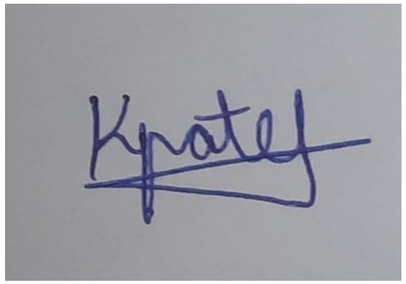
Ritul Patel



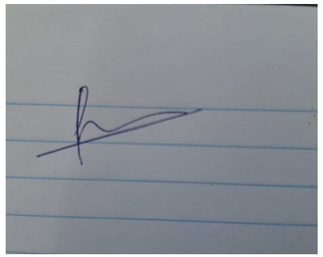
Tapas Santoki



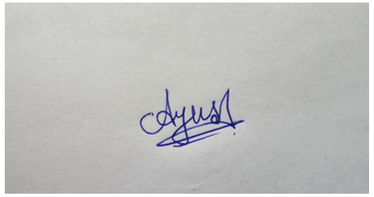
Dhameliya Utsker



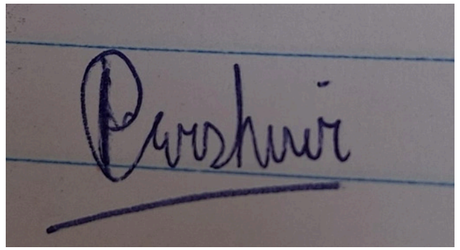
Krish Patel



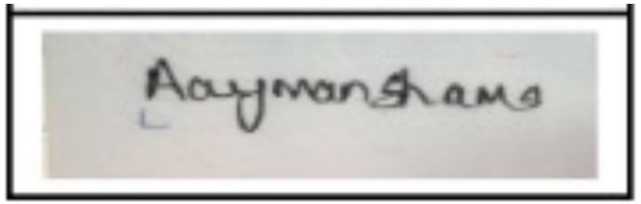
Het Rank



Ayush Patel



Harshvir Singh



Aayman Ammar Shams

Thank You