CT-216 Introduction To Communication Systems

Present By Group-21

END SEMESTER PROJECT

LDPC decoding for 5G NR

Prof. Yash Vasavada
Mentor: Aditya Pithadiya

HONOR CODE

We, the members of Group 21 , hereby solemnly declare that:

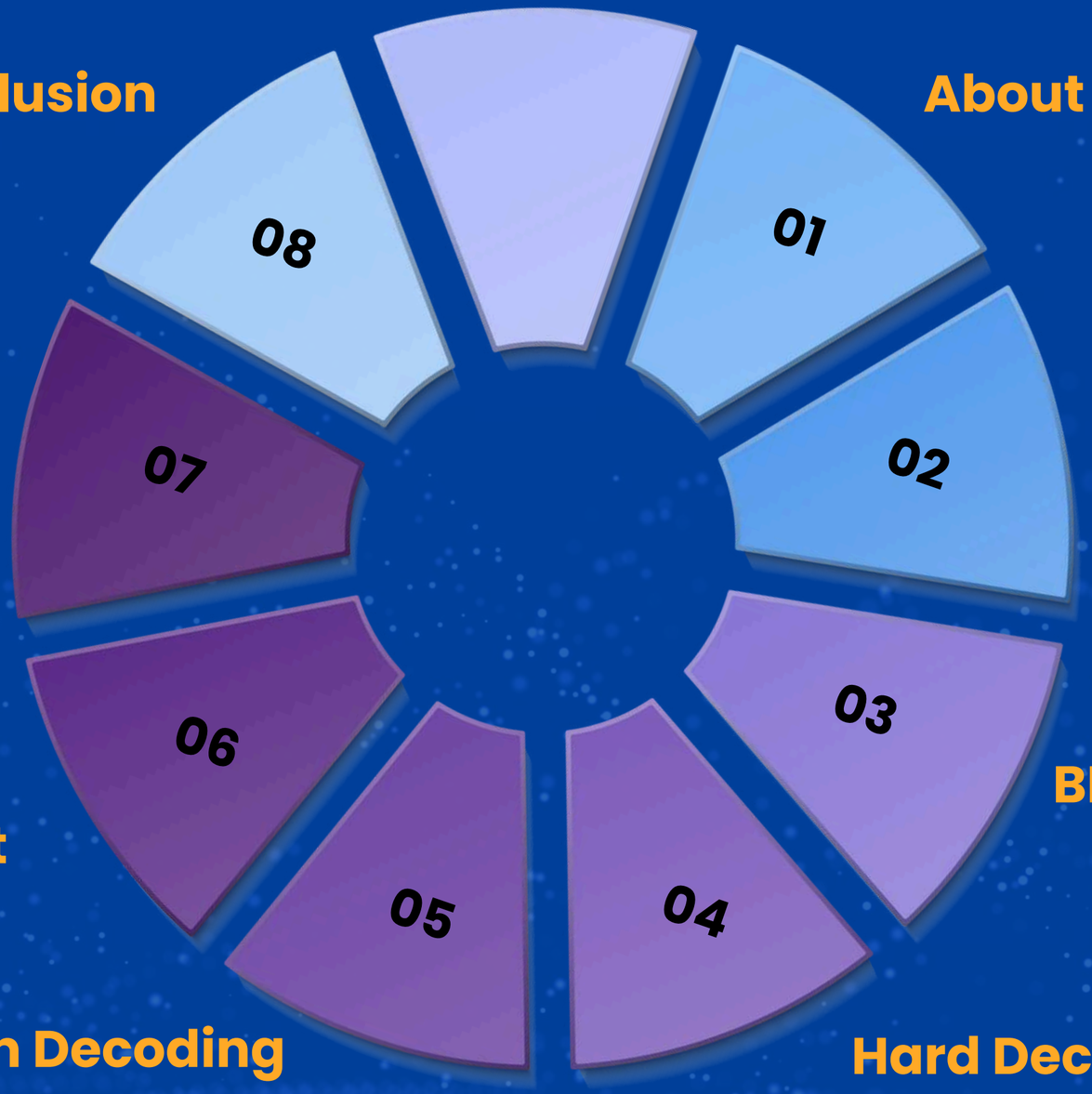
- The work presented in this project is entirely our own.
- We have not copied or reproduced any MATLAB code, simulation results or analysis from any other individual or group.
- All concepts, interpretations and insights have been developed independently through our own effort, study and understanding.
- Where external sources or references have been used, they have been properly acknowledged and cited.
- We make this declaration truthfully and in good faith, fully aware that any breach of academic integrity may result in serious consequences.

SIGNATURE OF GROUP MEMBERS

GROUP MEMBERS

STUDENT ID	NAME
202301223	RAIYANI RUDRA CHETANBHAI
202301224	PRAJAPATI ANUSHKA
202301225	DWARKESH VAGHASIYA
202301226	GOTHI PRIYANSHI MUKESHBHAI
202301227	PATEL VEDANT DINESHKUMAR
202301228	PATEL NAITIKKUMAR DILIPBHAI
202301230	PATEL APURV ASHOKBHAI
202301231	VORA KRESHA MANOJBHAI
202301232	GAADHE JAYANSH MANUBHAI
202301233	AYUSH NAKUM

CONTENTS



INTRODUCTION TO LDPC

- LDPC stands for low-density parity-check codes. They are used to correct transmission errors that can occur at various stages of communication, effectively minimizing the impact of channel noise.
- Originally proposed by R.G. Gallager in 1962, LDPC codes were not widely adopted at the time due to the complexity of decoding methods.
- Their practical use began after being rediscovered by David MacKay, leading to their adoption in a wide array of communication technologies, both wired and wireless.
- Today, LDPC codes play a crucial role in modern wireless communication systems, particularly in 5G New Radio (5G-NR) standards.

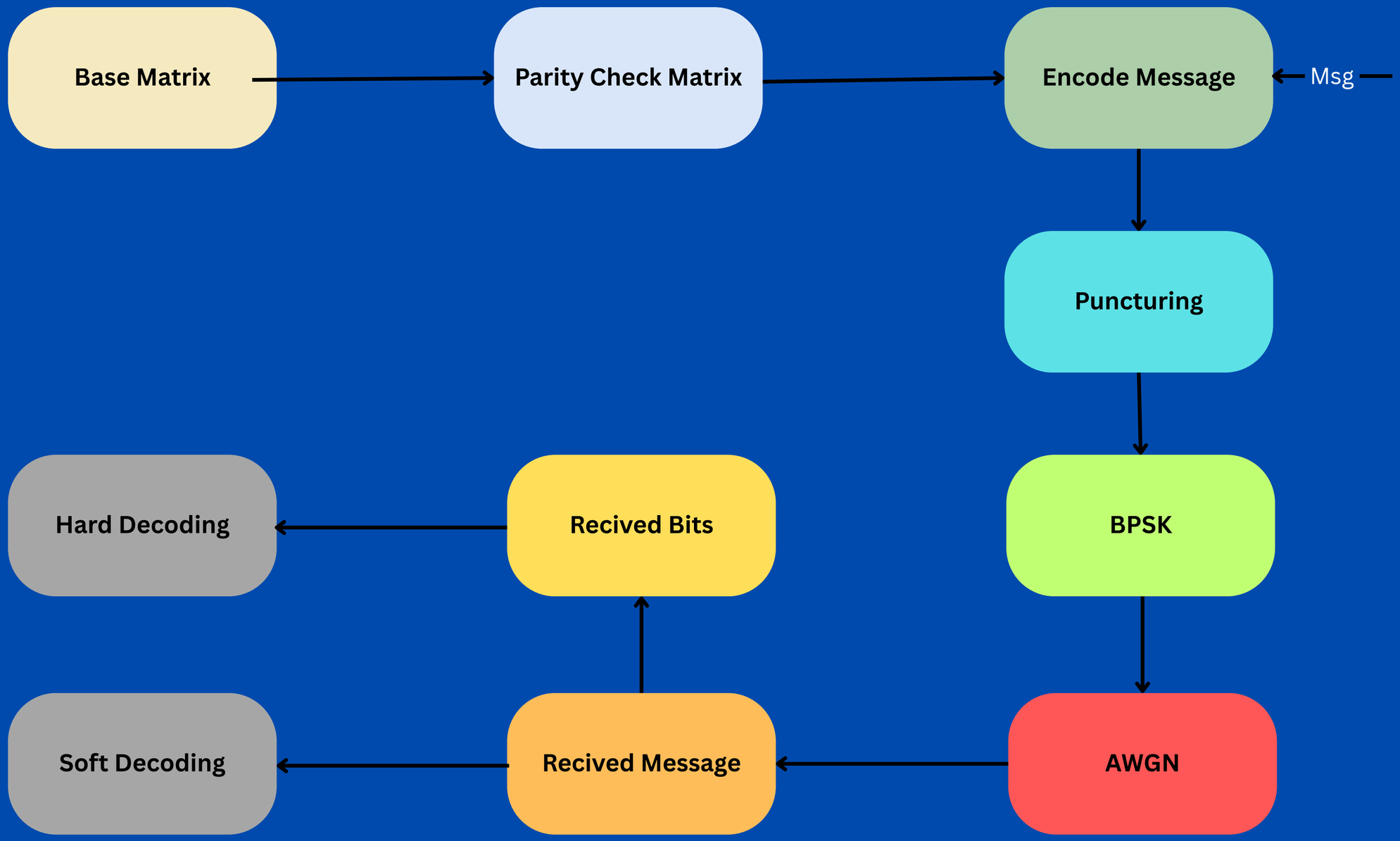
CHARACTERISTICS

- The **parity-check matrix** of LDPC codes is a sparse matrix, containing mostly zeroes with relatively few 1s. This reduces computational complexity during decoding.
  (Where number of 1s<< n * (n-k))

- LDPC codes can achieve error correction performance very close to **Shannon's theoretical limit**, making them extremely efficient.

- They use iterative decoding algorithms like belief propagation, which pass messages between variable and check nodes in a graphical model (Tanner graph).

PROGRAM FLOW



BASE GRAPH MATRIX

5G NR utilizes two base graphs : 1) Base Graph 1 (BG1)

2) Base Graph 2 (BG2)

selected based on the required code rate and input data size.

BG1 Dimensions: **46 × 68**

BG2 Dimensions: **42 × 52**

The number of message (information) bits, denoted as k, is calculated as:

k = n - m, where n is the number of columns and m is the number of rows in the base graph.

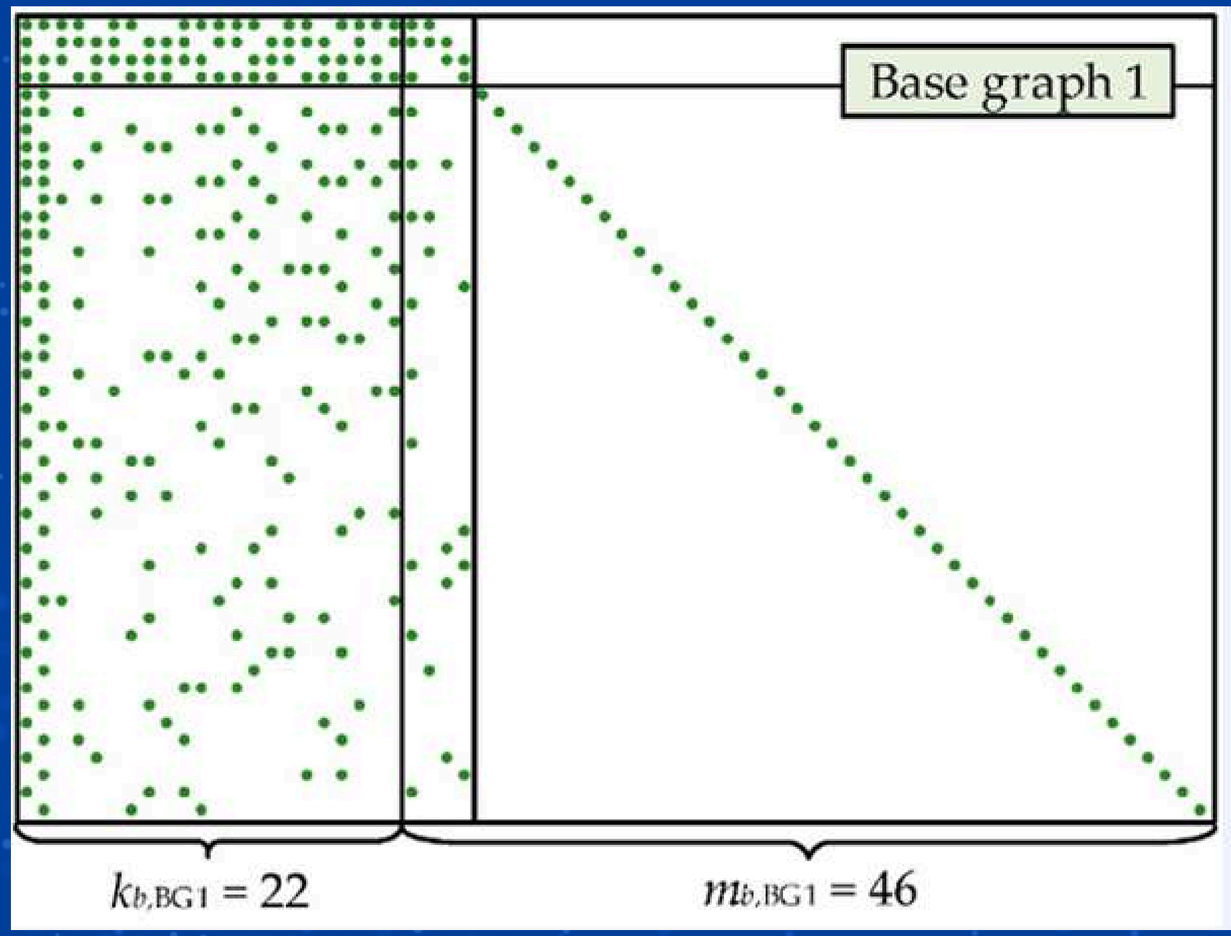
Accordingly:

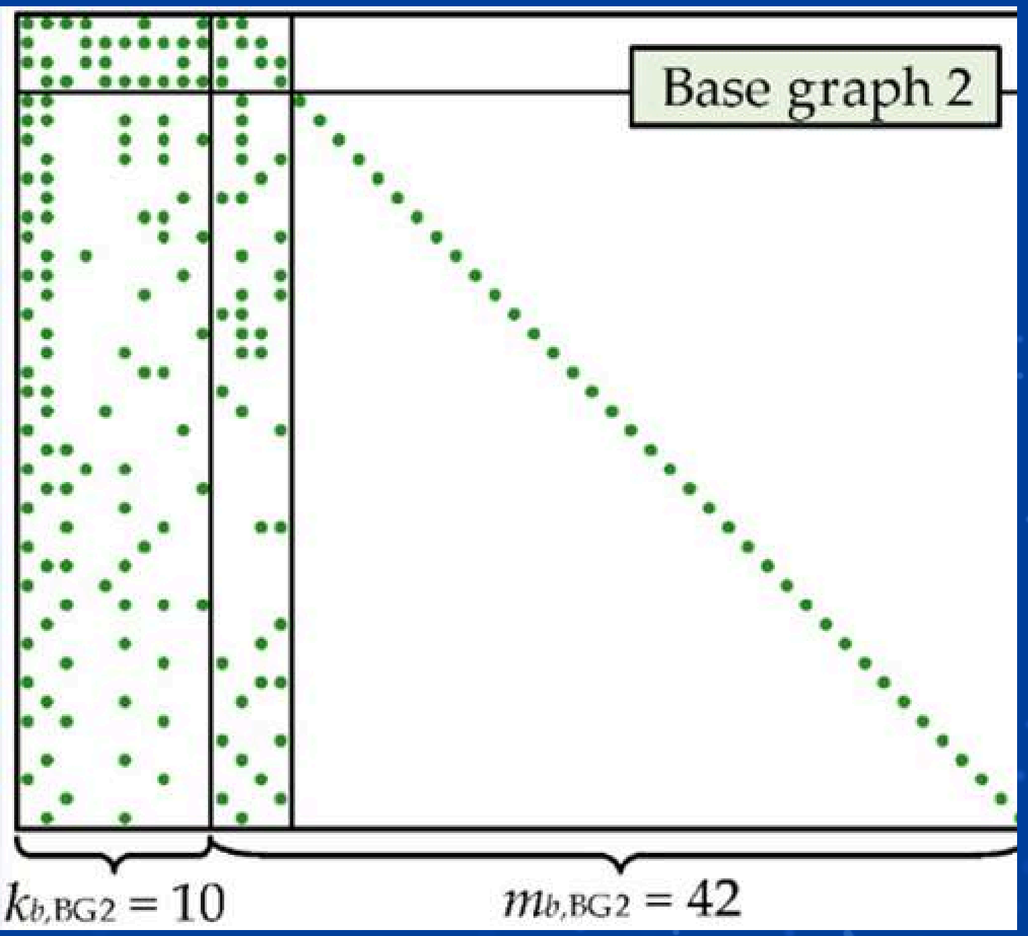
BG1 supports up to **22** information bits

BG2 supports up to **10** information bits

The differing dimensions of BG1 and BG2 correspond to the different numbers of information bits that can be used during encoding.

TYPES OF BASE GRAPH MATRIX

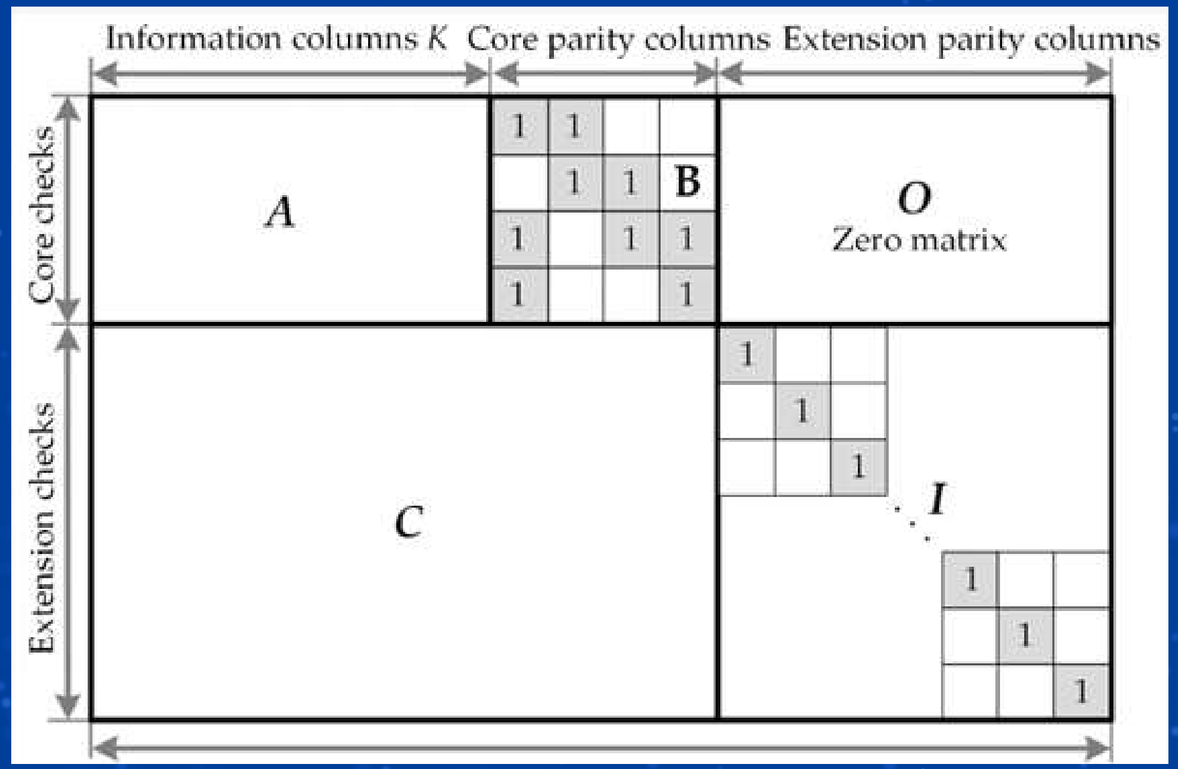




BG1 : matrix size of 46x68

BG2 : matrix size of 42x52

BASE GRAPH MATRIX

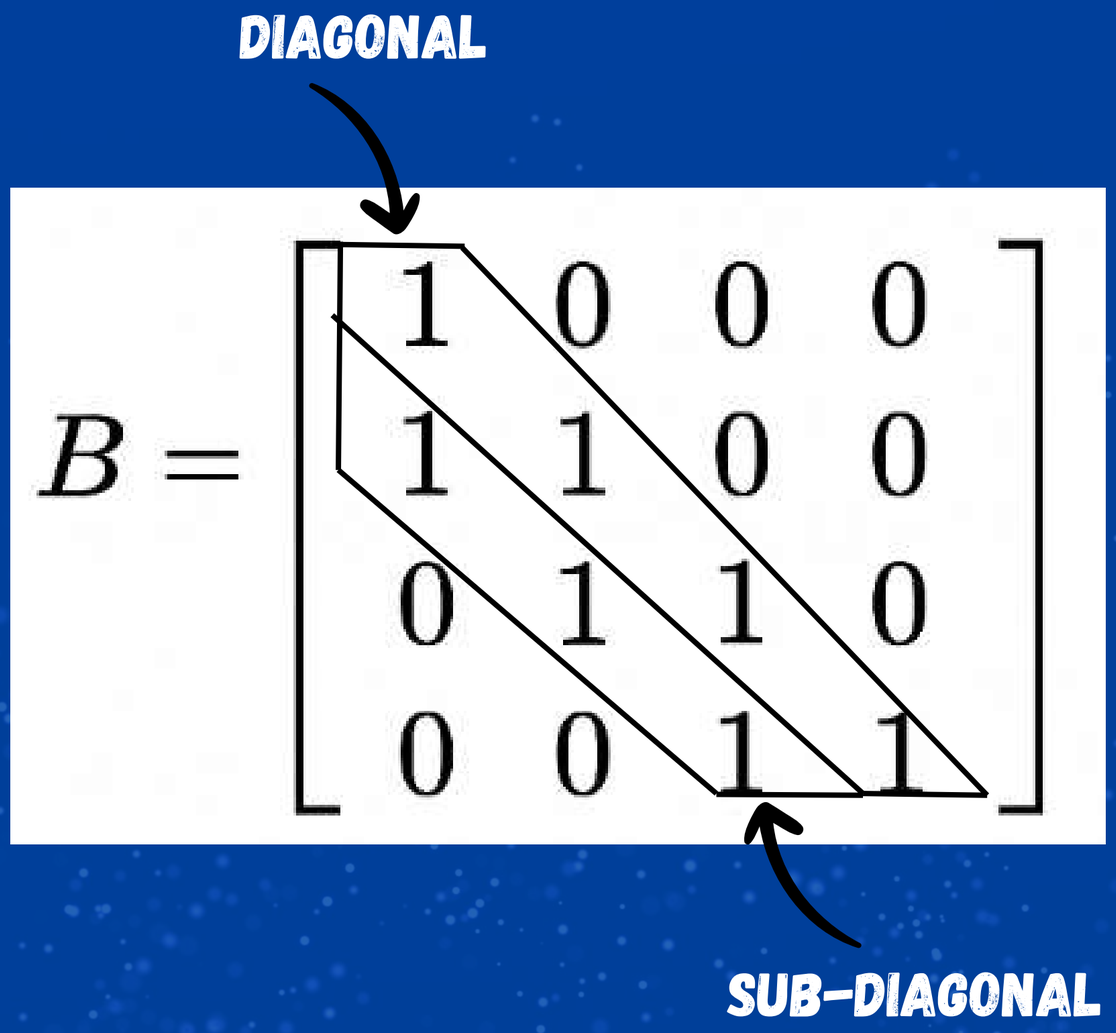


- A – Information Matrix
- B – Double Diagonal Matrix
- C – Less Dense Matrix
- I – Matrix with only diagonal entries = 1
- O – Matrix with all entries equal to -1

DOUBLE DIAGONAL MATRIX

A Double Diagonal Matrix is a 4×4 square matrix where:

- The main diagonal (top-left to bottom-right) contains 1s
- The lower sub-diagonal also contains 1s
- All other entries are 0s



PARITY CHECK MATRIX CONSTRUCTION

- For each element in the base graph, a z × z square matrix is generated, where z is the expansion factor.
- For every non-negative entry in the base matrix, the corresponding matrix is a circular right-shifted identity matrix, with the value indicating the number of shifts.
- For any -1 entry, the corresponding matrix is a zero matrix

$$B = \begin{bmatrix} 2 & -1 & 1 & 3 & 0 & -1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 3 & 2 & 1 & -1 & 0 \end{bmatrix}$$

Figure 3.3: Base graph **B**

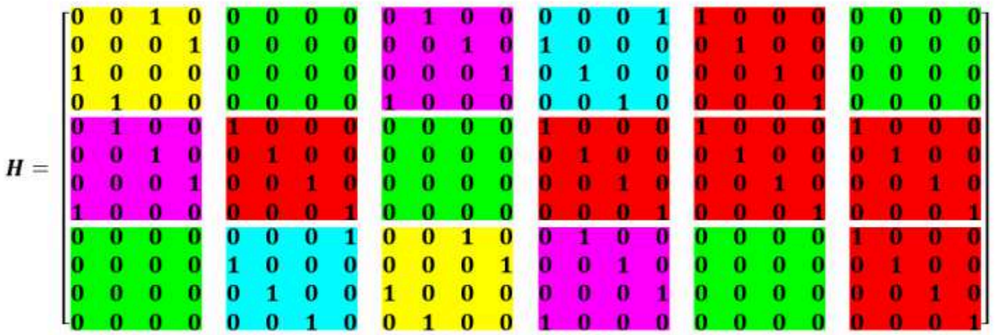


Figure 3.4: Parity-check matrix **H**

ENCODING USING BASE MATRIX

Codeword Vector Representation:
[v1 v2 v3 v4 v5 v6 v7 v8 v9]=[m1 m2 m3 m4 m5 p1 p2 p3 p4]

Equations Obtained from $H \cdot c' = 0$:
Equation 1:
$I_1v_1+I_3v_3+I_1v_4+I_2v_5+Iv_6=0$
Equation 2:
$I_2v_1+Iv_2+I_3v_4+Iv_6+Iv_7=0$
Equation 3:
$I_4v_2+I_2v_3+Iv_4+I_1v_5+Iv_7+Iv_8=0$
Equation 4:
$I_4v_1+I_1v_2+Iv_3+I_2v_5+Iv_8=0$
Combining the Equations:
Summing all equations gives:
$I_1v_5=I_1v_1+I_2v_1+I_4v_1+Iv_2+I_4v_2+I_1v_2+I_3v_3+I_2v_3+Iv_3+I_1v_4+I_3v_4+Iv_4$
Solving for Parity Bits:
From the combined equation, solve for p1 using known message bits m1,m2,m3,m4,m5
Substitute p1p into Equation 1 to solve for p2, and repeat to find all parity bits p1,p2,p3,p4.
The encoded message must satisfy: $H \cdot c' = 0$

PUNCTURED MATRIX

A punctured matrix refers to a modification of the LDPC code structure where some of the encoded bits are intentionally removed (not transmitted) after encoding.
This technique is used to increase the code rate as it reduces the encoded bits.

HOW PUNCTURING WORKS

1. Start with a lower-rate LDPC code.
2. After encoding, selectively remove some parity bits before transmission.
3. At the receiver:

- The decoder treats the punctured (missing) bits as unknowns or erasures.

H MATRIX AFTER PUNCTURING

$$B = \begin{bmatrix} 2 & -1 & 1 & 3 & 0 & -1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 3 & 2 & 1 & -1 & 0 \end{bmatrix}$$

Figure 3.3: Base graph **B**

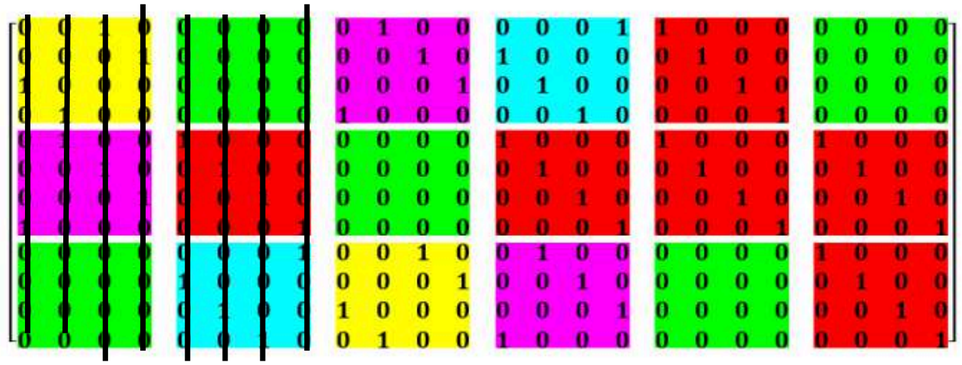
$H =$ 

Figure 3.4: Parity-check matrix **H**

To get x/y code rate :
Suppose we need to puncture a*z bits.

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

By this formula, we can find the lower limit of code rate for base graphs.

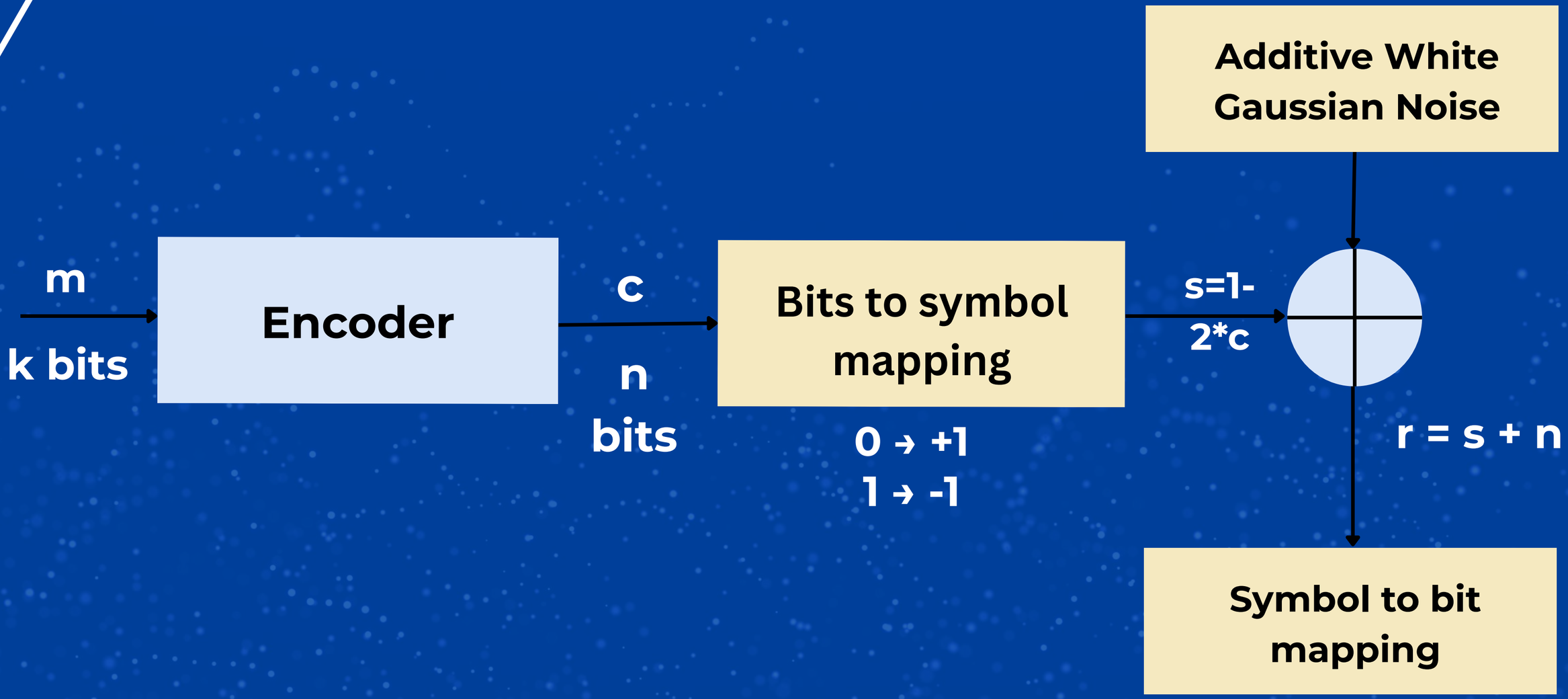
- For BG2, the lower limit is 1/5.
- For BG1, the lower limit is 1/3.

BPSK MODULATION AND NOISE

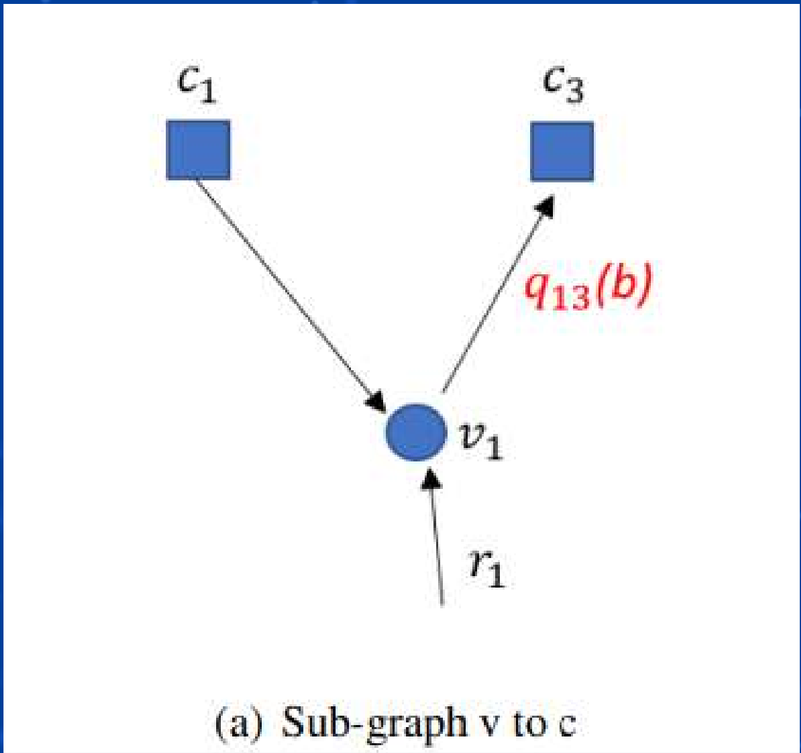
- Encode the message using LDPC → get an n-bit codeword.
- Modulate each bit using BPSK:
- Bit 0 → +1
- Bit 1 → −1
- Transmit through AWGN channel → adds random Gaussian noise.
- Receiver gets:
- Received Signal=BPSK Symbol+Noise
- Decoder uses this noisy signal to recover original bits.

AWGN NOISE CHARACTERISTICS

- The noise is random, following a Gaussian distribution:
- Mean (μ) = 0
- Variance (σ^2) depends on:
- Eb = Energy per bit
- No = Noise spectral density
- r = Code rate
- Standard deviation (σ) = $1 \div \sqrt[2]{2 \times SNR \times r}$


m
k bits
Encoder
c
n
bits
Bits to symbol mapping
0 → +1
1 → -1
s=1-
2*c
Additive White Gaussian Noise
r = s + n
Symbol to bit mapping

HARD-DECISION DECODING

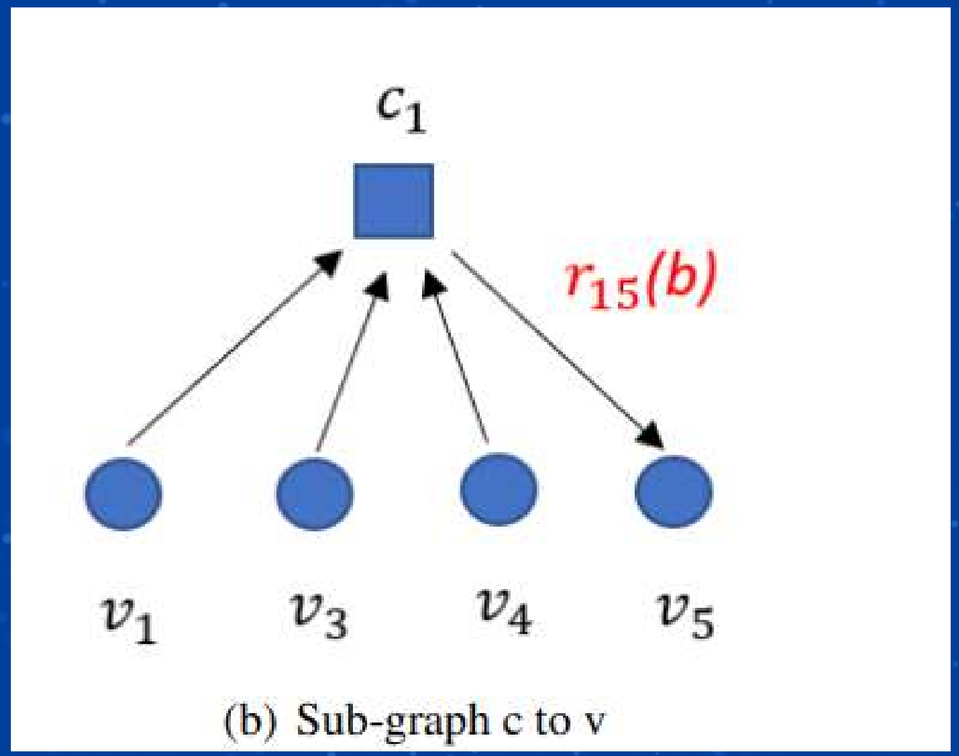


(a) Sub-graph v to c

VARIABLE NODE (VN) TO CHECK NODE (CN) MESSAGE PASSING:

- Initially, all entries of the received vector are loaded into the **Variable Nodes (VNs)**.
- The VNs follow a **repetition code encoding scheme**..
- In first iteration, each VN sends to a CN the modulo-2 sum of all incoming messages from other connected VNs, excluding the VN it is currently sending to.
- In subsequent iterations, each VN sends the connected CN majority voting of other connected CNs.

HARD-DECISION DECODING



(b) Sub-graph c to v

CHECK NODE (CN) TO VARIABLE NODE (VN) MESSAGE PASSING:

- CNs follow a single parity check coding scheme.
- For each connected VN, a CN computes the modulo-2 sum (XOR) of all incoming messages from the other connected VNs, excluding the one to which the message is being sent.

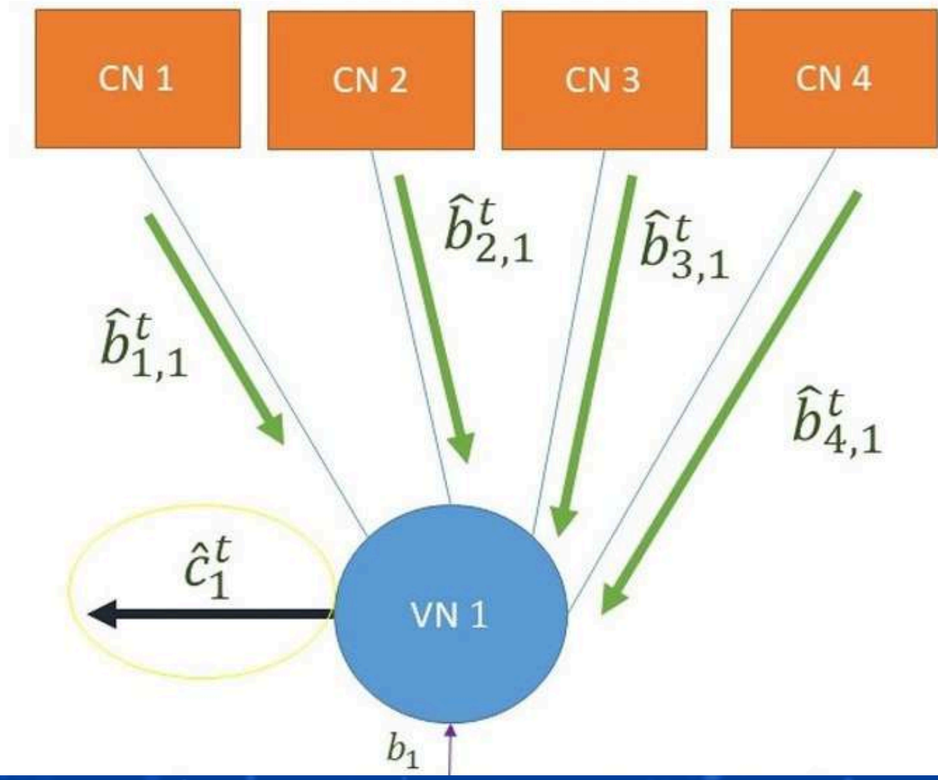
VARIABLE NODE UPDATE (MAJORITY VOTING):

EACH VN UPDATES ITS VALUE BASED ON MAJORITY VOTING:

- It considers all messages received from connected CNs, along with its own current value.
- The updated value is the majority among these

.

HARD-DECISION DECODING



DECODING TERMINATION CONDITION:

- The updated values of all VNs form the decoded codeword vector c.
- We compute the syndrome by multiplying the parity-check matrix (H) with the transpose of the decoded codeword vector.
- If the result is a zero vector, the codeword satisfies all parity checks and is considered valid.
- The iterative process continues until a valid codeword is obtained.

SOFT DECODING APPROACH:

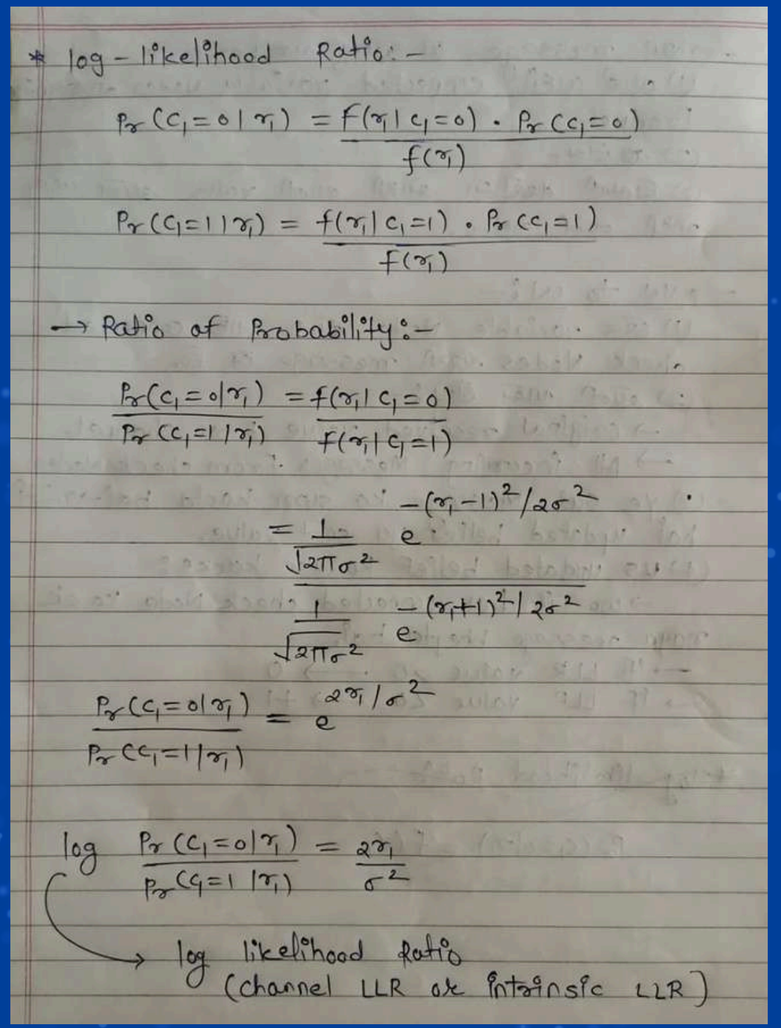
- In soft decision decoding, we work with probabilistic information. To handle this, a belief propagation (BP) algorithm is used. This involves an iterative process where messages are exchanged between variable nodes (bits) and check nodes (parity constraints) in the Tanner graph.
- The goal is to gradually refine the "beliefs" about the transmitted bits by incorporating both the received signal and the code structure.
- Two types of information guide this process:
- Intrinsic Information – Derived directly from the channel observations (what the channel output says about each bit).
- Extrinsic Information – Gathered from the code constraints (what neighboring nodes suggest about the bit).
- Belief propagation works using Log-Likelihood Ratios (LLRs), which express how likely a bit is to be 0 or 1 based on the observed data:

$$L\left(c_i\right) = log\left(P\left(c_{i=0}\left|y_i\right|\right) \div P\left(c_i = 1\left|y_i\right|\right)\right)$$

SOFT DECODING APPROACH:

MESSAGE PASSING DETAILS :

- At the start, LLRs based on channel outputs are assigned to each variable node.
- During each iteration, variable nodes send messages to connected check nodes, and check nodes respond based on these inputs. The message update rule follows a specific mathematical formula derived from probability theory.
- However, this exact formula can be computationally intensive, especially with repeated iterations for each check node.
- To simplify the implementation without significantly affecting performance, the Min-Sum Approximation can be used. It reduces complexity while still enabling effective decoding.



SOFT-DECISION DECODING

INITIALIZATION:

- We begin with a parity-check matrix (H) and a received vector obtained through BPSK modulation.

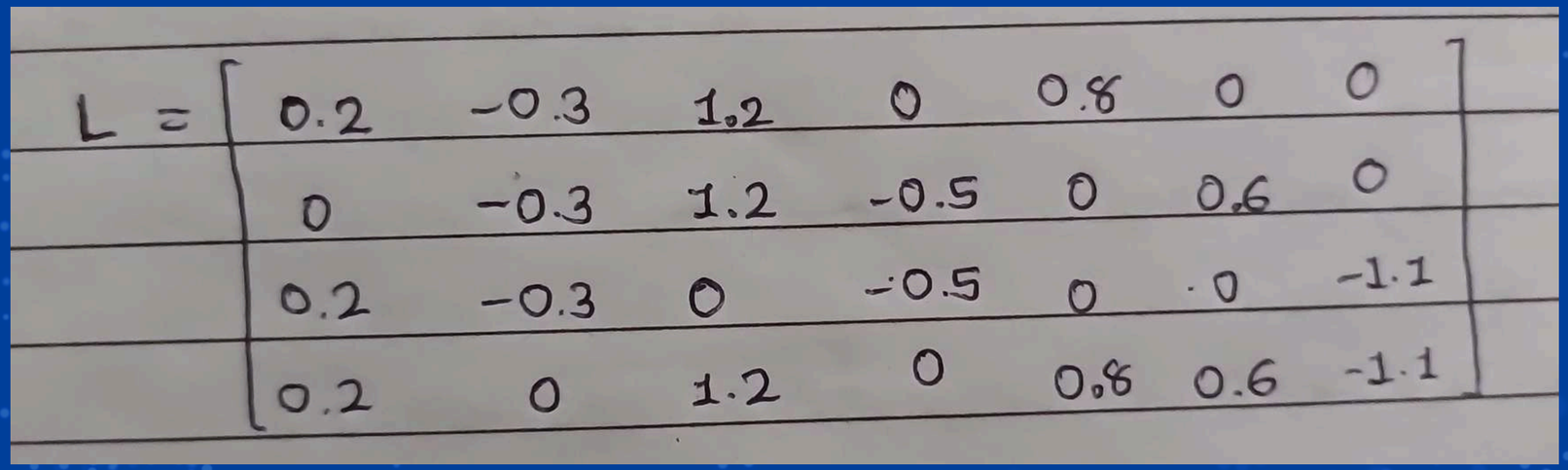
- We use the parity-check matrix H to construct the matrix L by replacing its 1s with the corresponding entries from the vector r.

Suppose H matrix :-

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 01 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

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

SOFT-DECISION DECODING



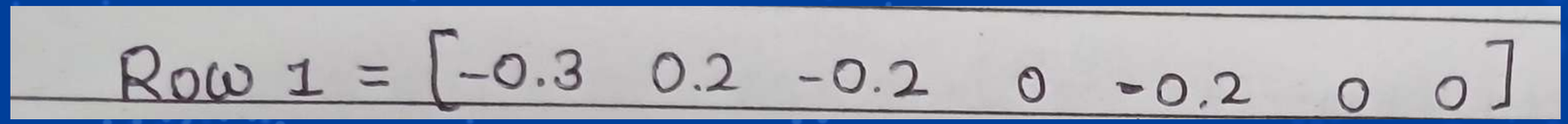
MINIMUM VALUE PROCESSING:

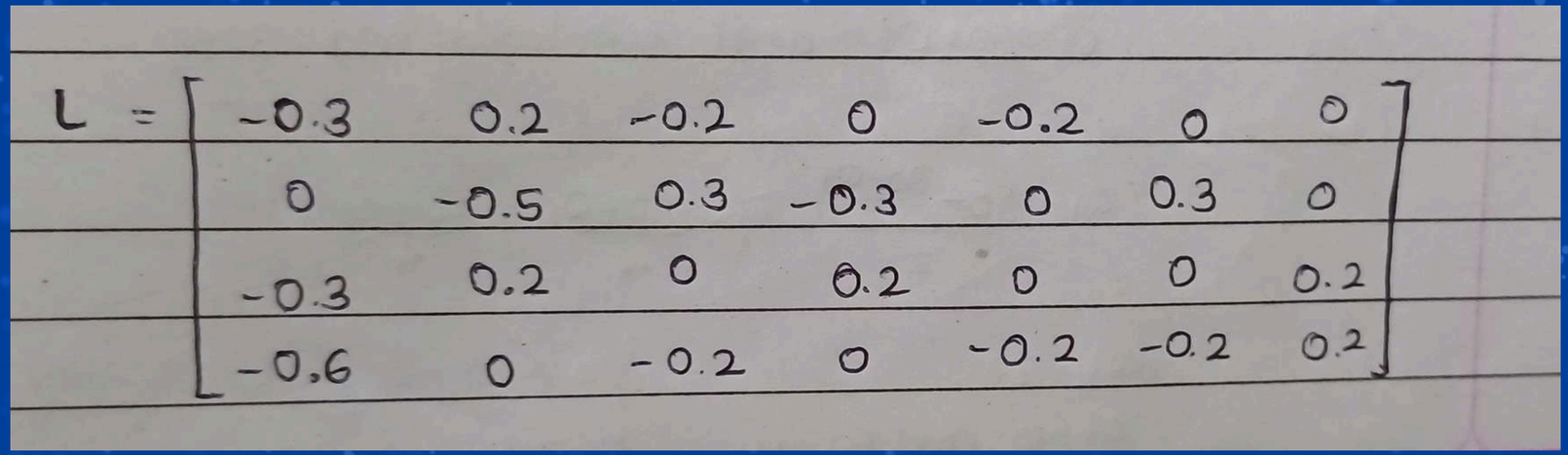
- For each row of the L matrix:
- Identify the smallest (minimum 1) and the second smallest (minimum 2) absolute values.
- Replace the minimum 1 with minimum 2.
- Replace all remaining non-zero values in the row with minimum 1
- Here, in first row, minimum1 = 0.2 and minimum2 = 0.3

SOFT-DECISION DECODING

SIGN ADJUSTMENT:

- Count the number of negative elements in each row of the L matrix.
- If the count is odd, invert the signs of all non-zero elements in that row.





SOFT-DECISION DECODING

VARIABLE NODE UPDATE:

- Compute the column-wise sum of the updated L matrix.
- Add this sum to the corresponding value in the received vector to obtain a new vector v.

UPDATED BELIEF AFTER 1ST ITERATION:

$$[-1 \quad -0.4 \quad 1.1 \quad -0.6 \quad 0.4 \quad 0.7 \quad -0.7]$$

$o_2 = [0.2]$

1st column of L matrix = $\begin{bmatrix} -0.3 \\ 0 \\ -0.3 \\ -0.6 \end{bmatrix}$ so that sum=-1

SOFT-DECISION DECODING

L MATRIX UPDATE:

- For each element in the L matrix:
  - Update its value as the difference between the corresponding v value (same column) and the current element value.

ITERATIVE REFINEMENT:

- This process is repeated across multiple iterations using Monte Carlo simulations.
- After convergence, we obtain the final updated received vector.

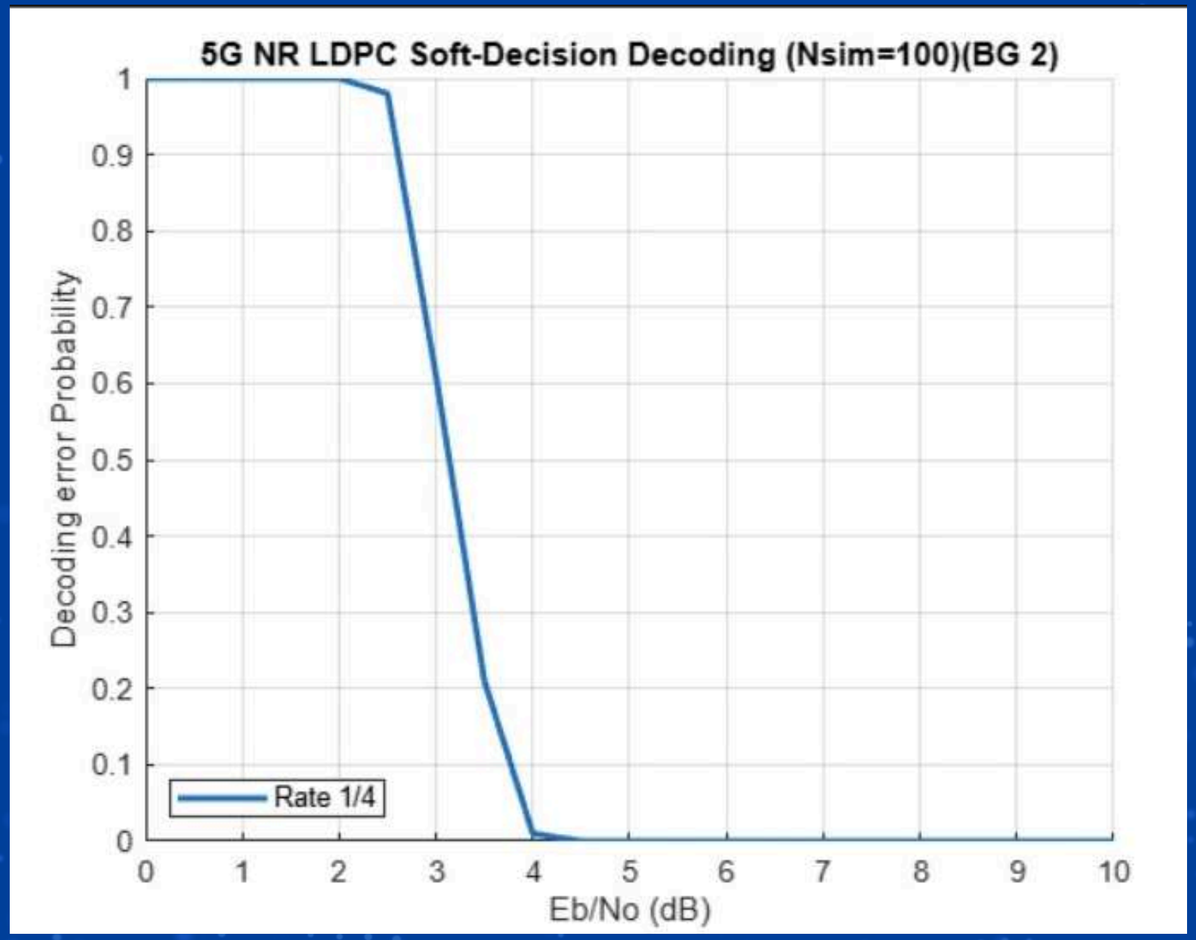
$$L = \begin{bmatrix} -0.7 & -0.6 & 1.3 & 0 & 0.6 & 0 & 0 \\ -0 & 0.1 & 0.8 & -0.3 & 0 & 0.4 & 0 \\ -0.7 & -0.6 & 0 & -0.8 & 0 & 0 & -0.9 \\ -0.4 & 0 & 1.3 & 0 & 0.6 & 0.9 & -0.9 \end{bmatrix}$$

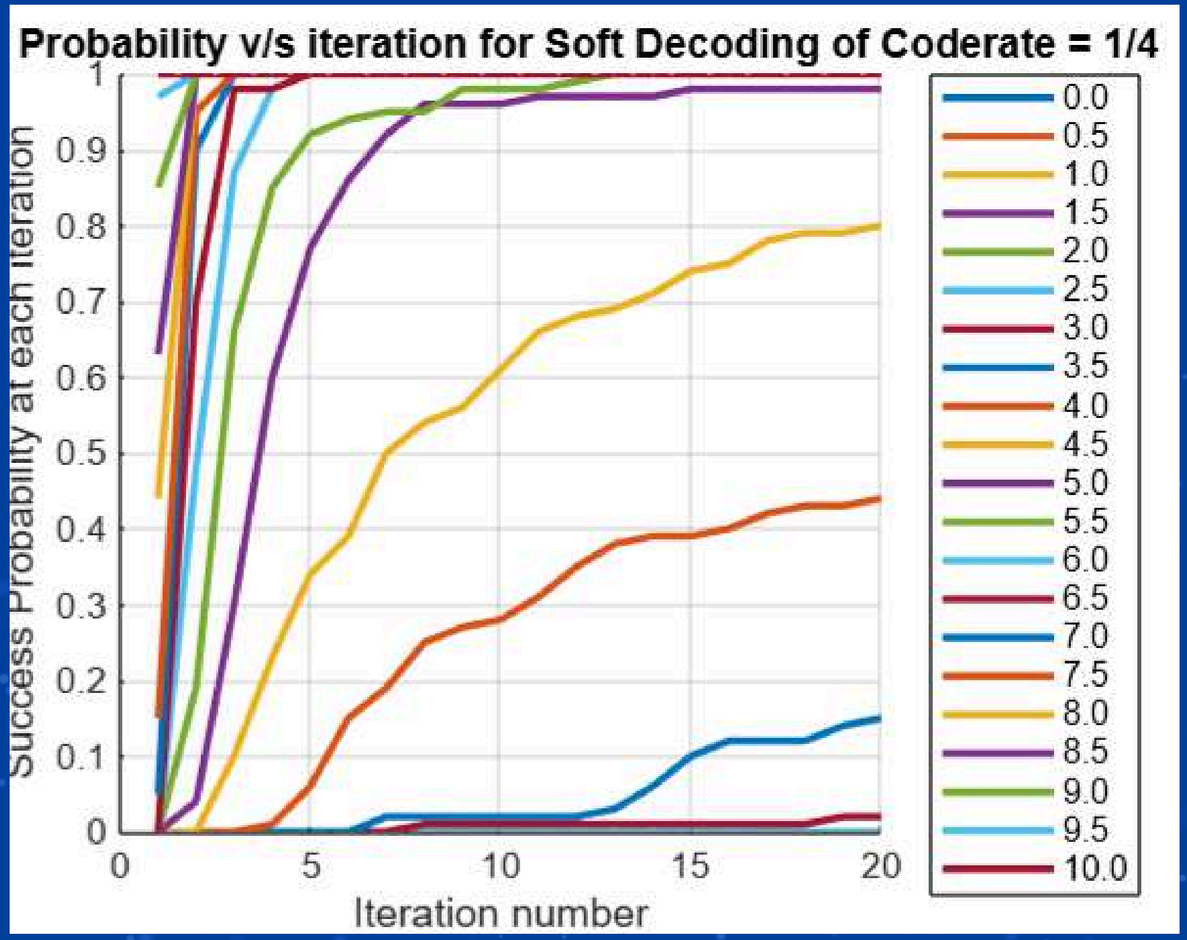
HARD DECISION:

- Apply a hard decision rule based on the BPSK scheme:
- Negative values in the final vector are decoded as 1.
- Positive values are decoded as 0.

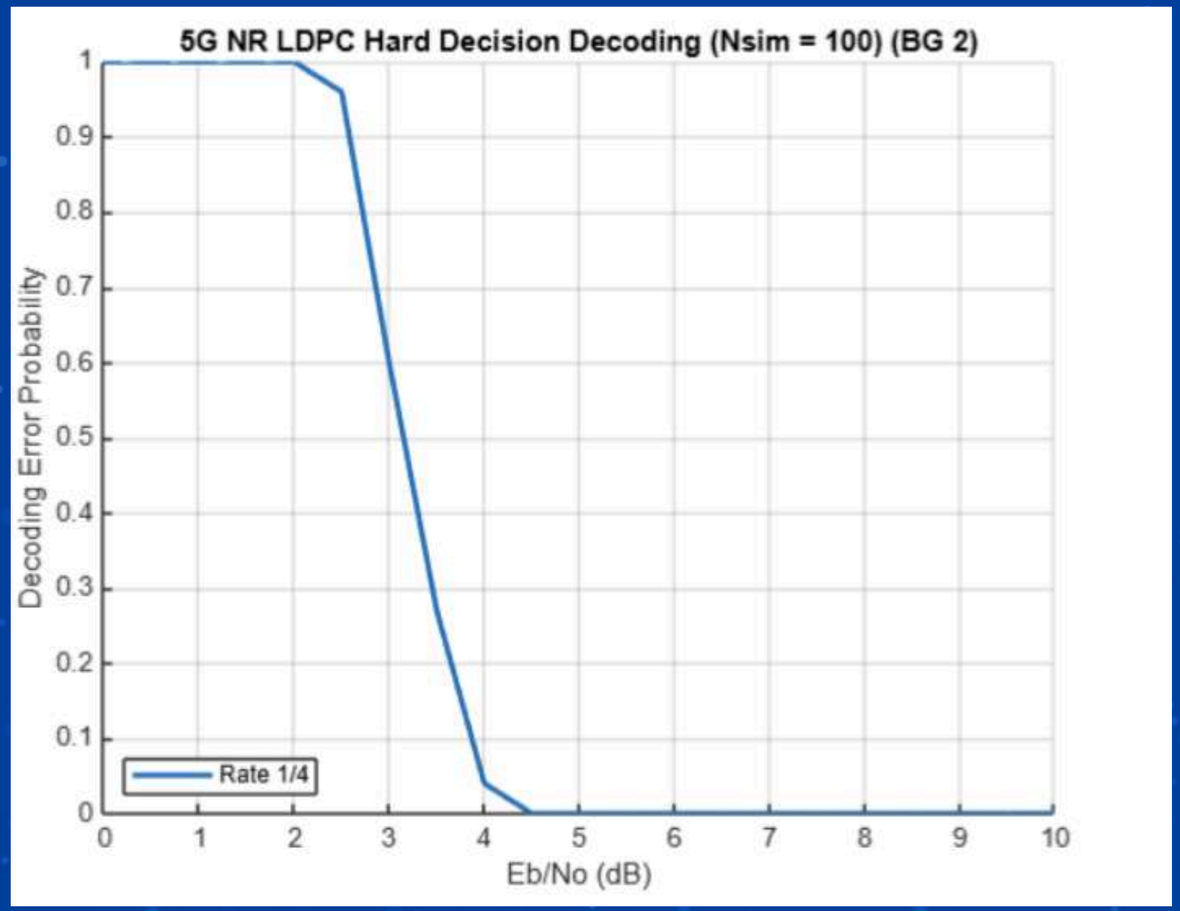
RESULTS OF MATRIX NR_2_6_52

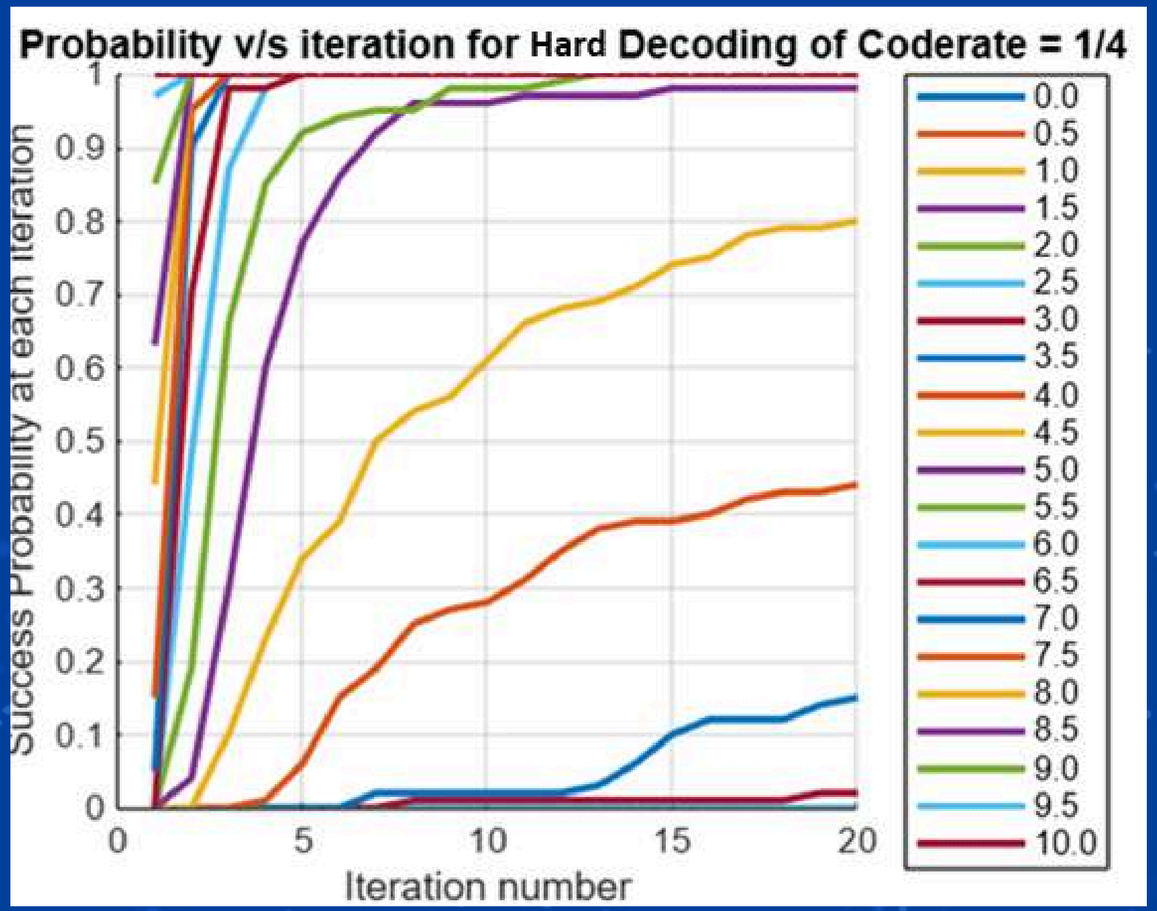
SOFT DECISION DECODING CODERATE = 1/4



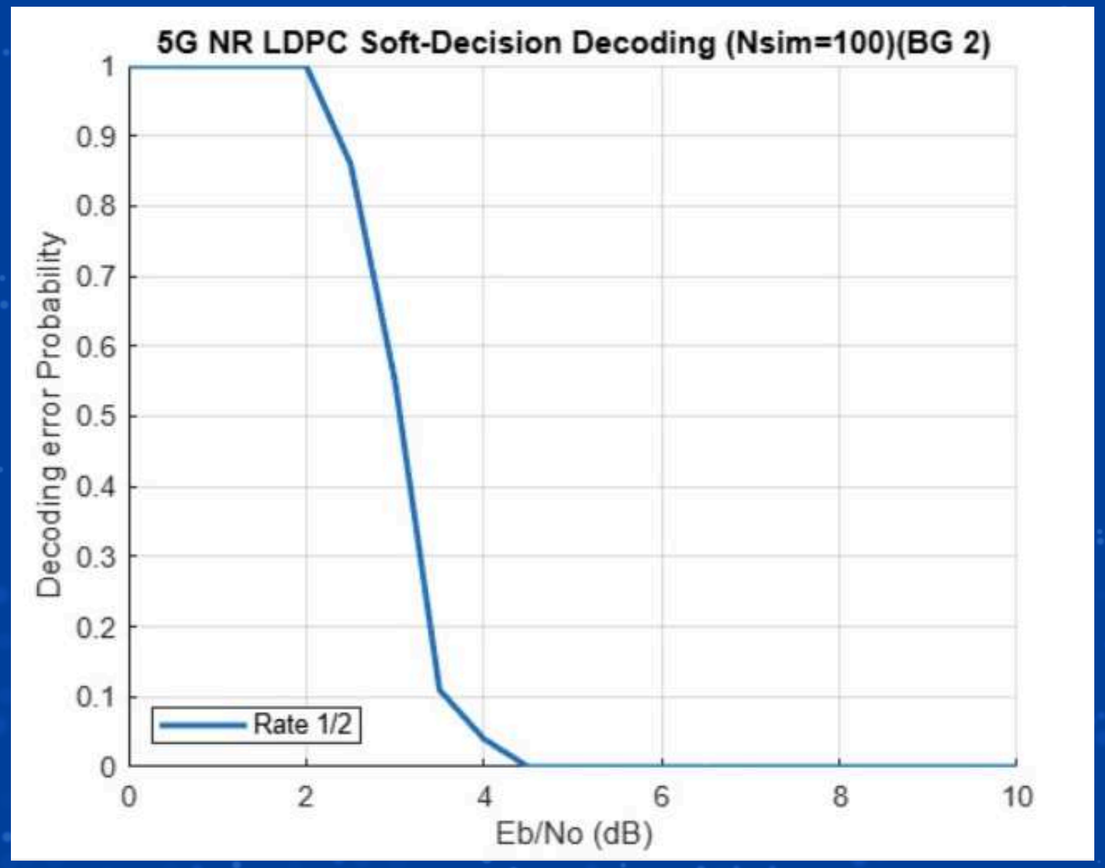


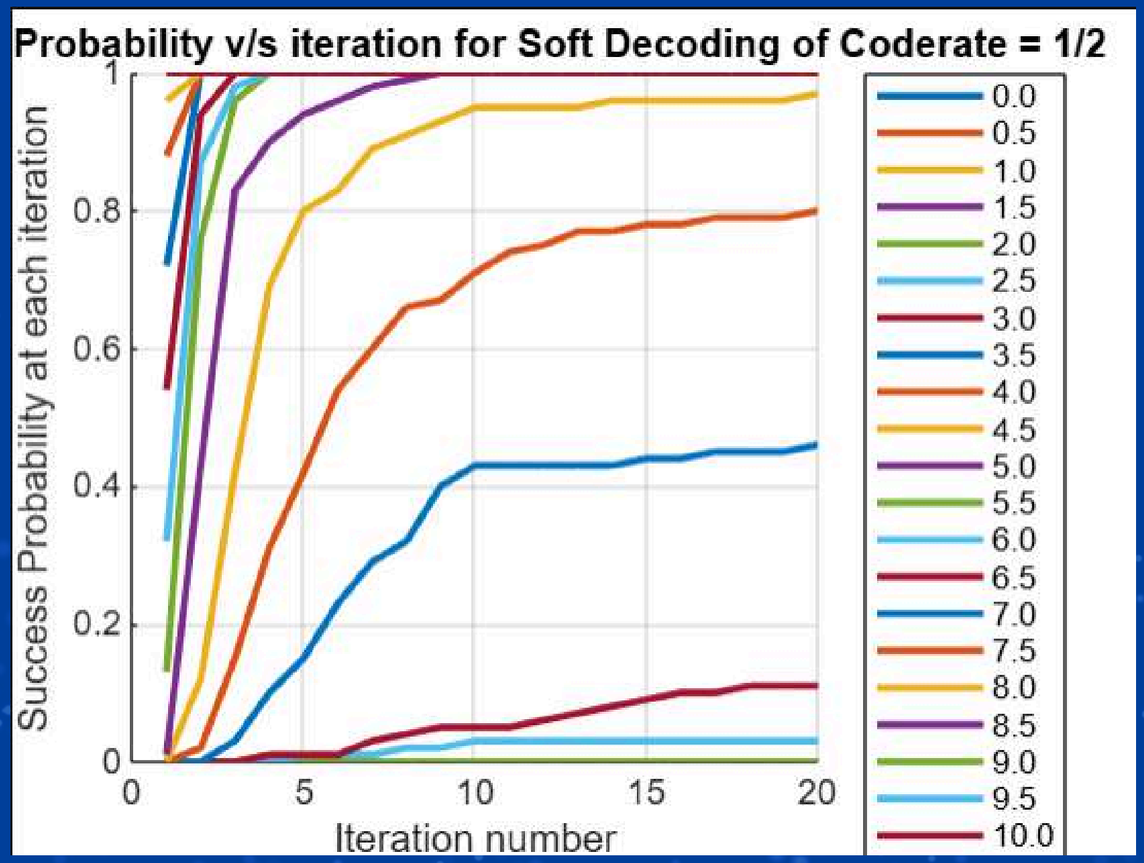
HARD DECISION DECODING CODERATE = 1/4



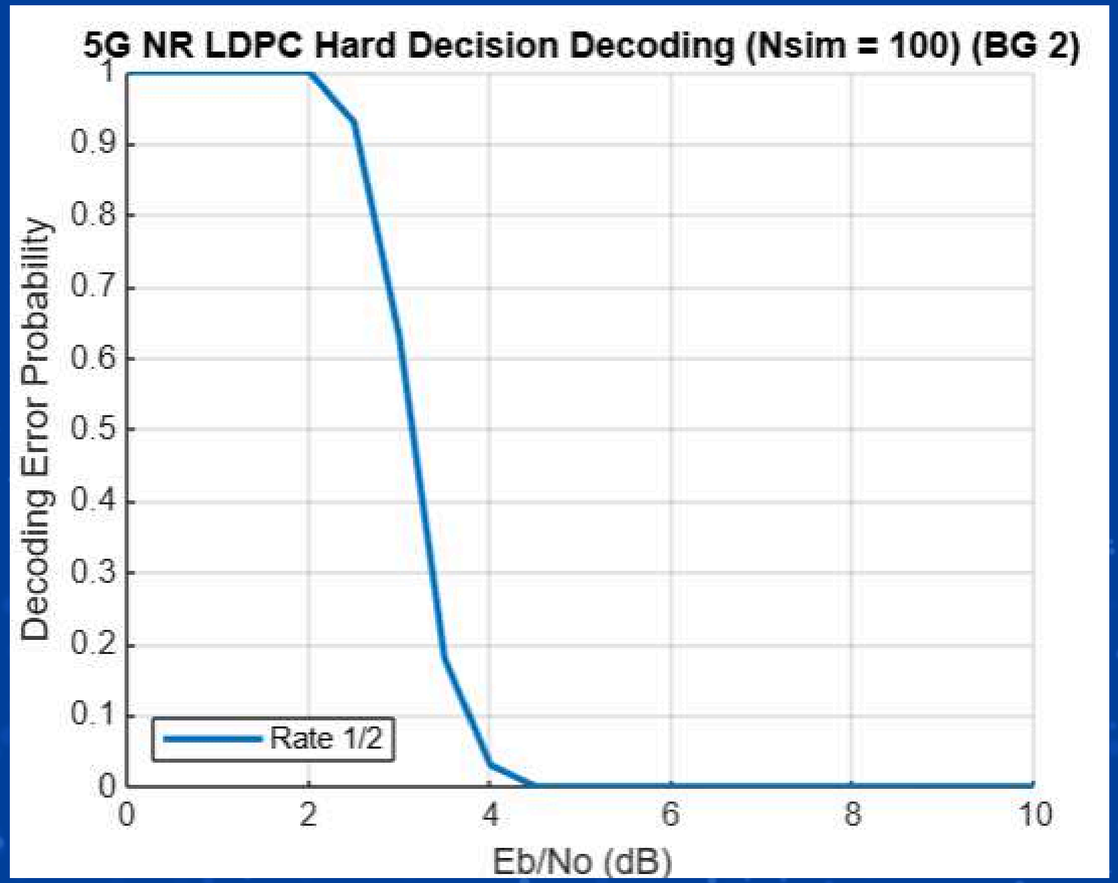


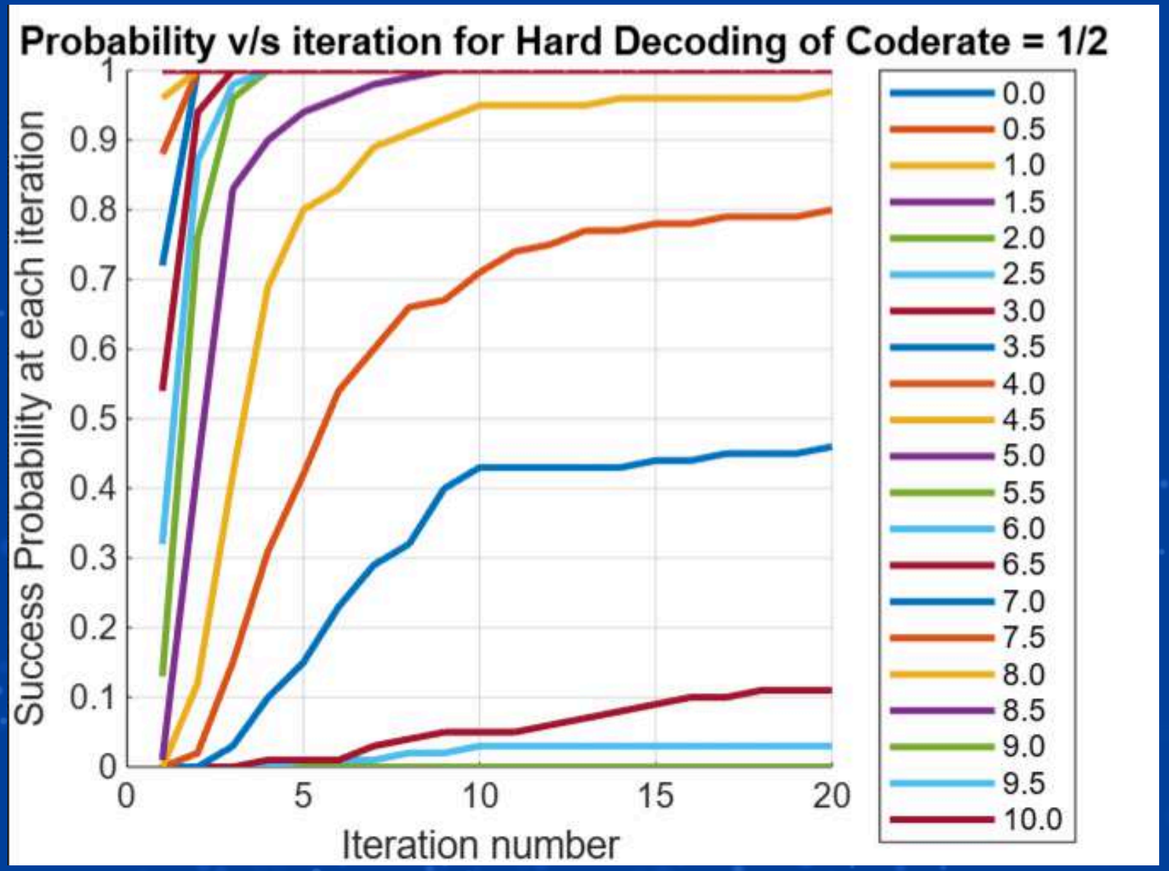
SOFT DECISION DECODING CODERATE = 1/2



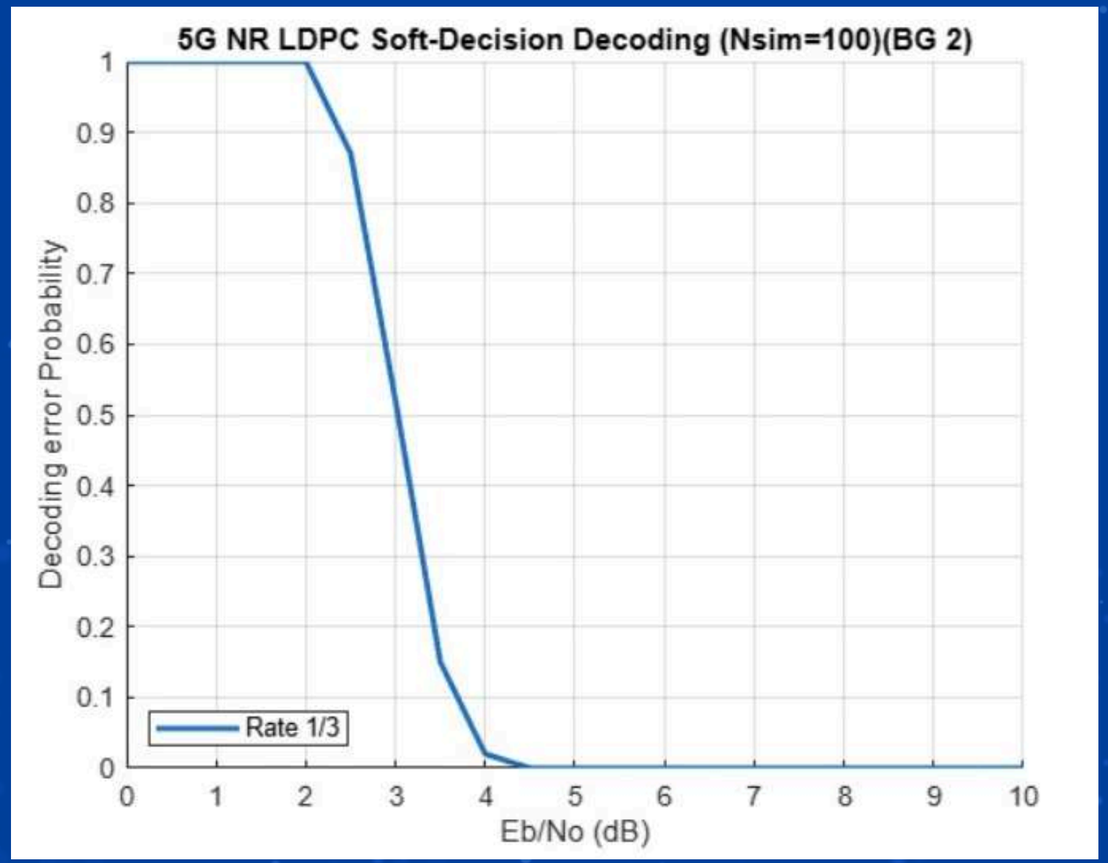


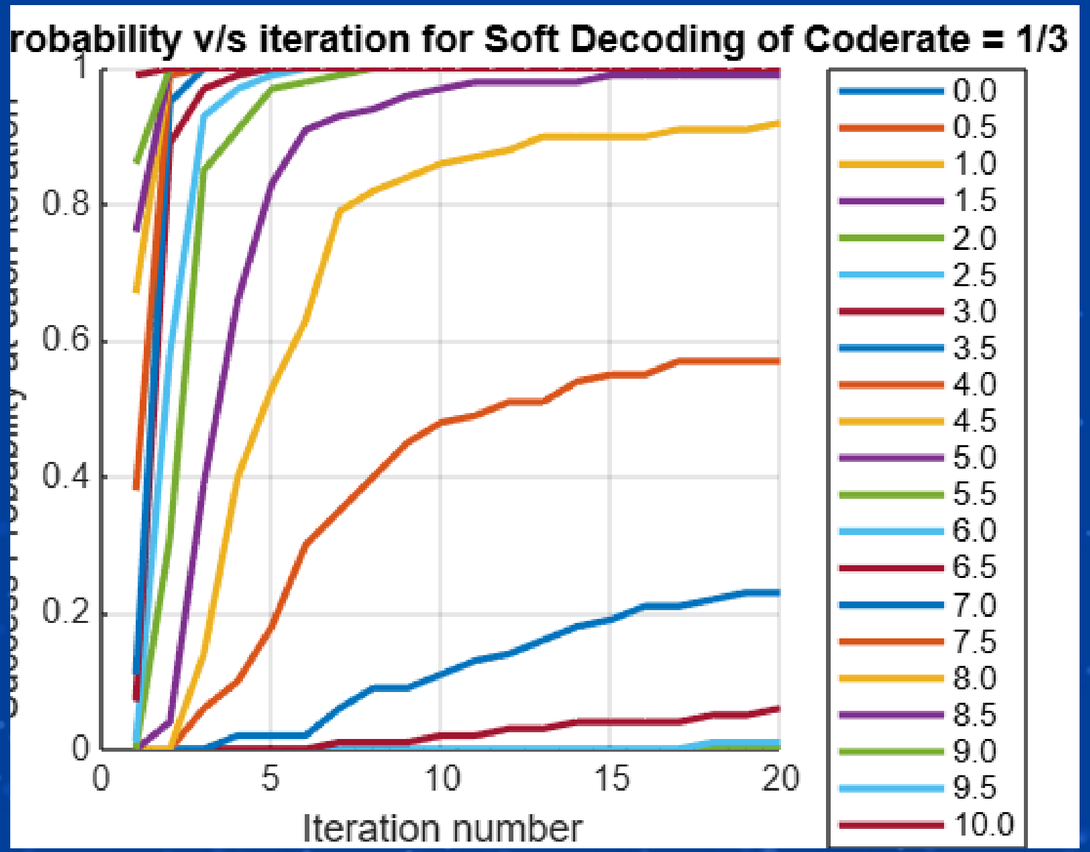
HARD DECISION DECODING CODERATE = 1/2



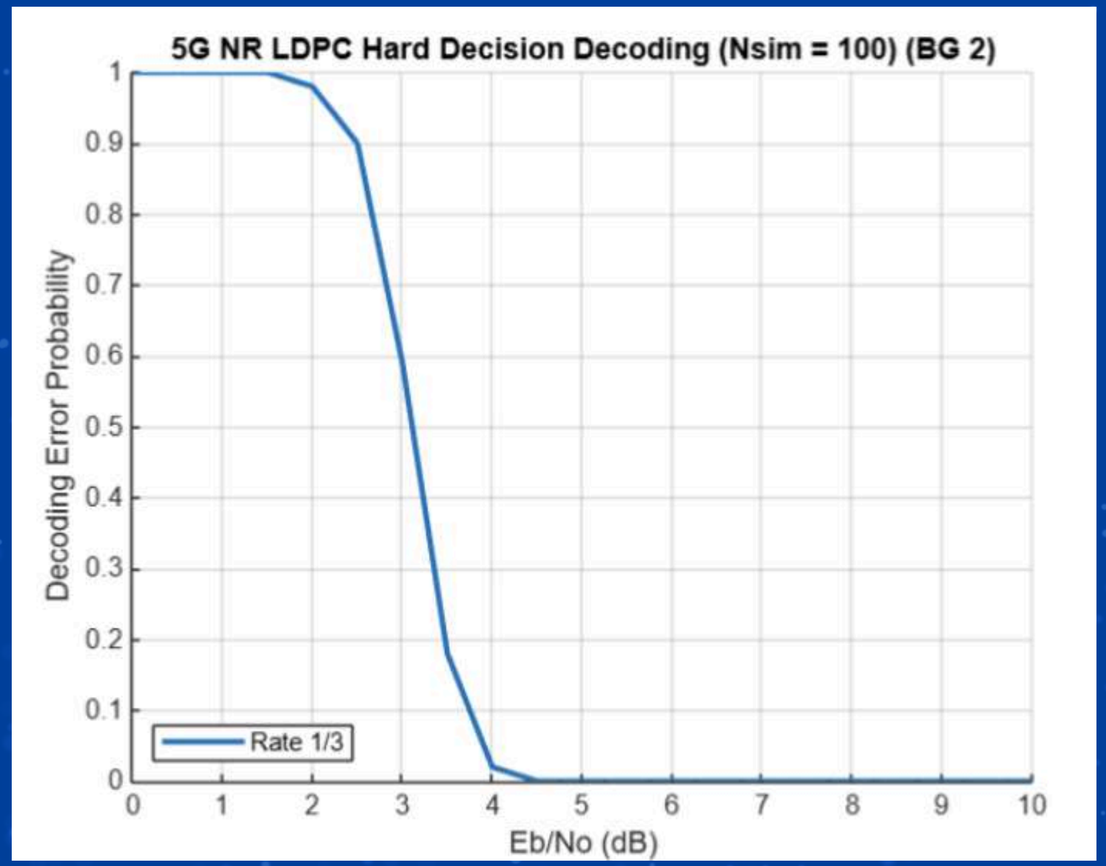


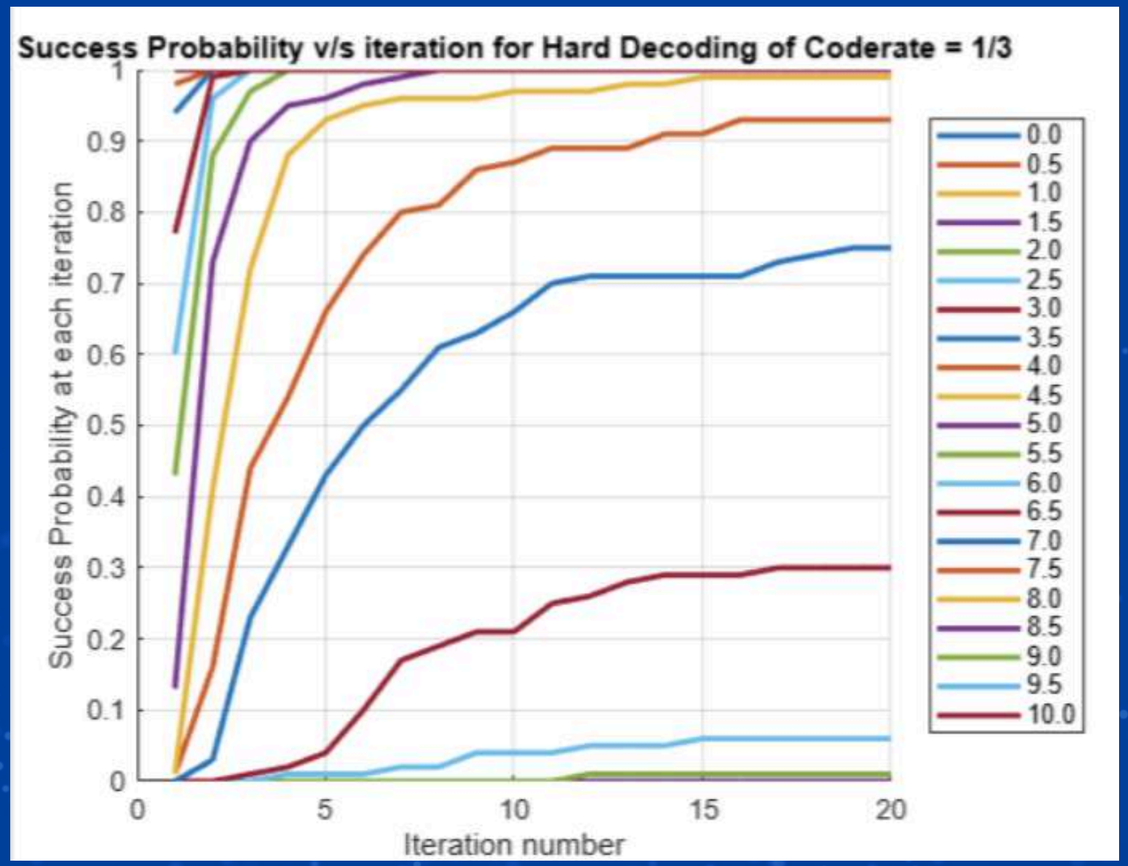
SOFT DECISION DECODING CODERATE = 1/3



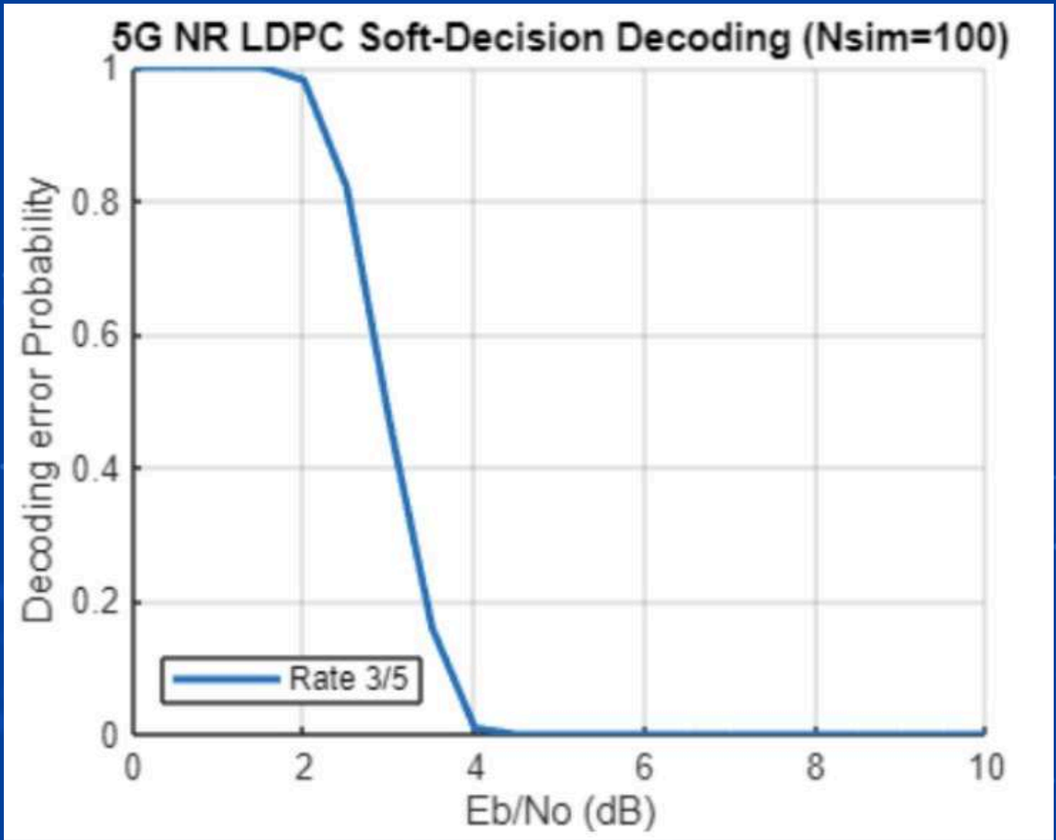


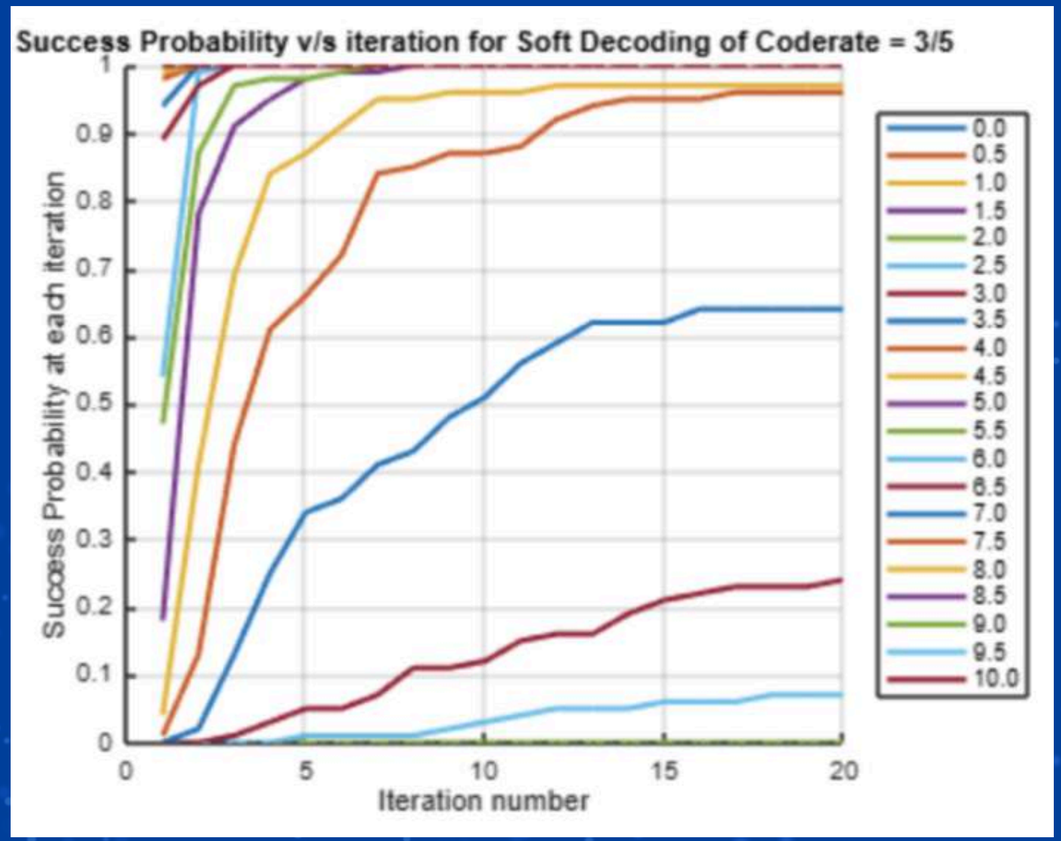
HARD DECISION DECODING CODERATE = 1/3



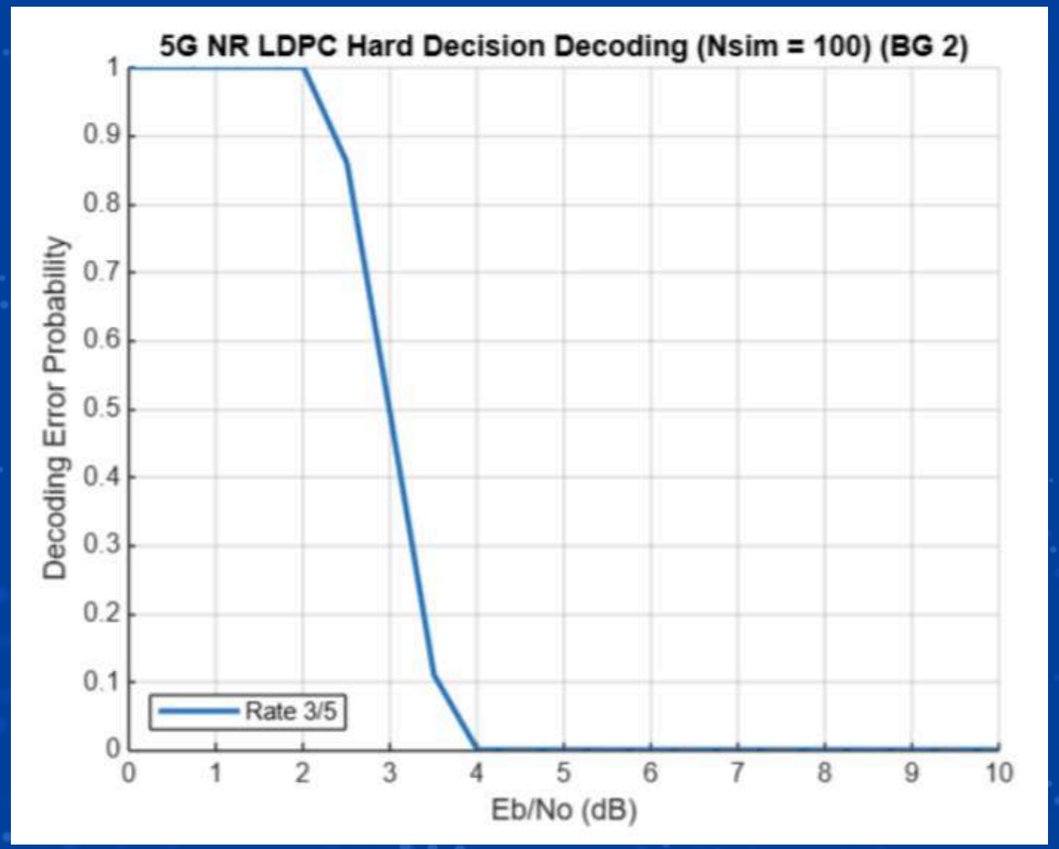


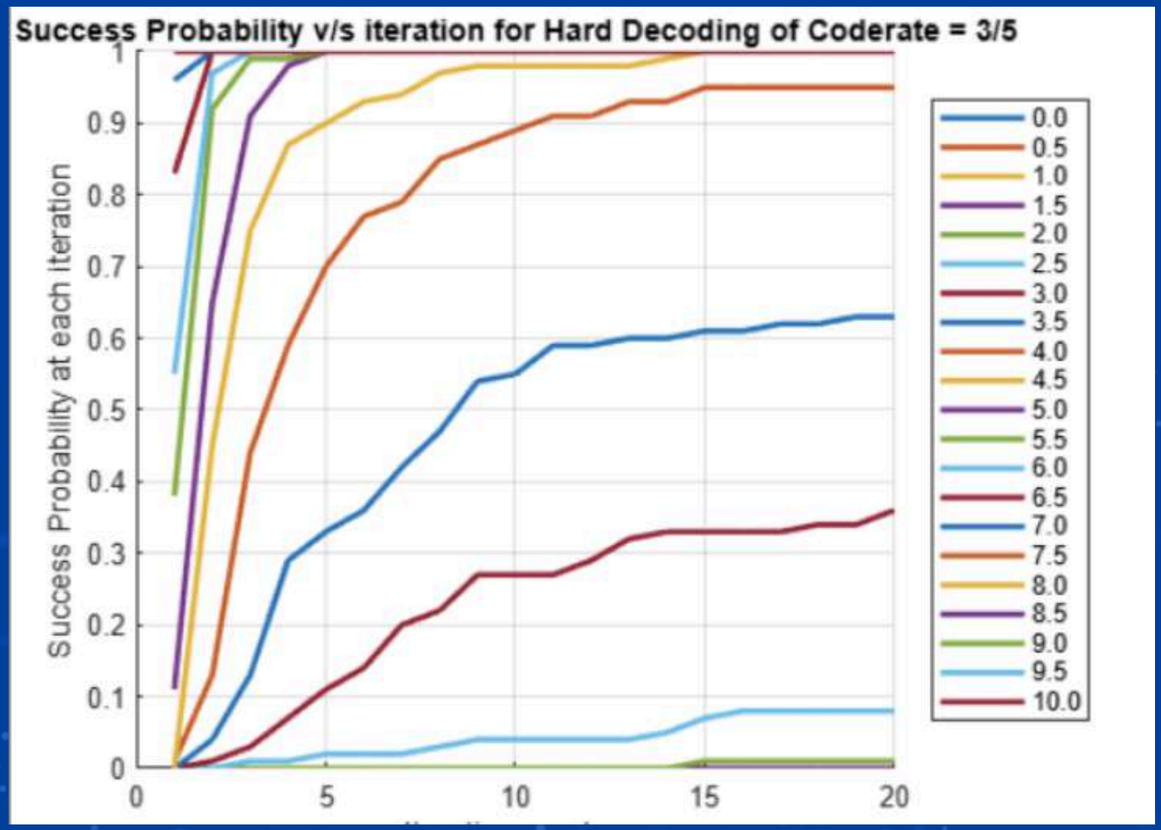
SOFT DECISION DECODING CODERATE = 3/5



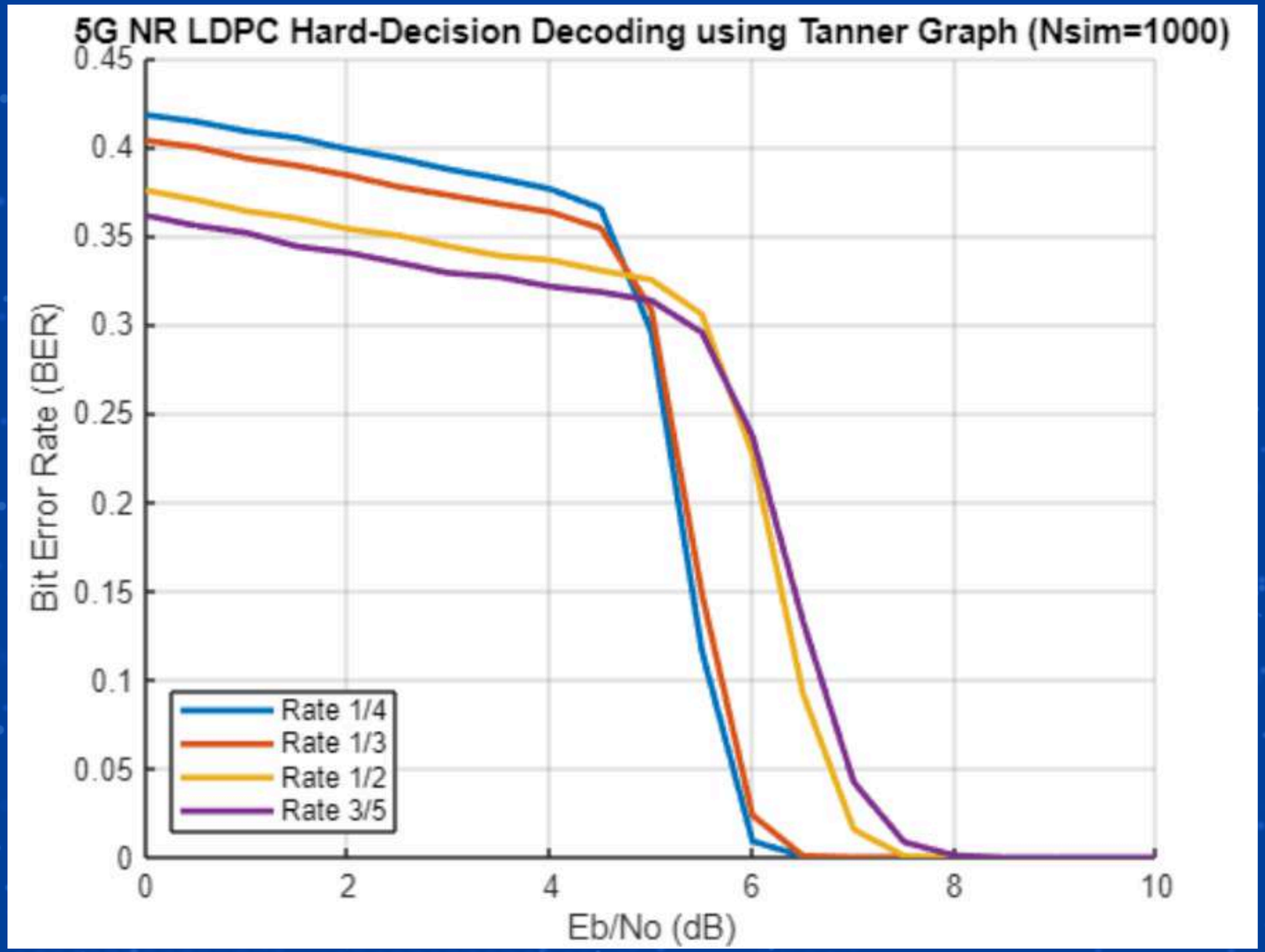


HARD DECISION DECODING CODERATE = 3/5

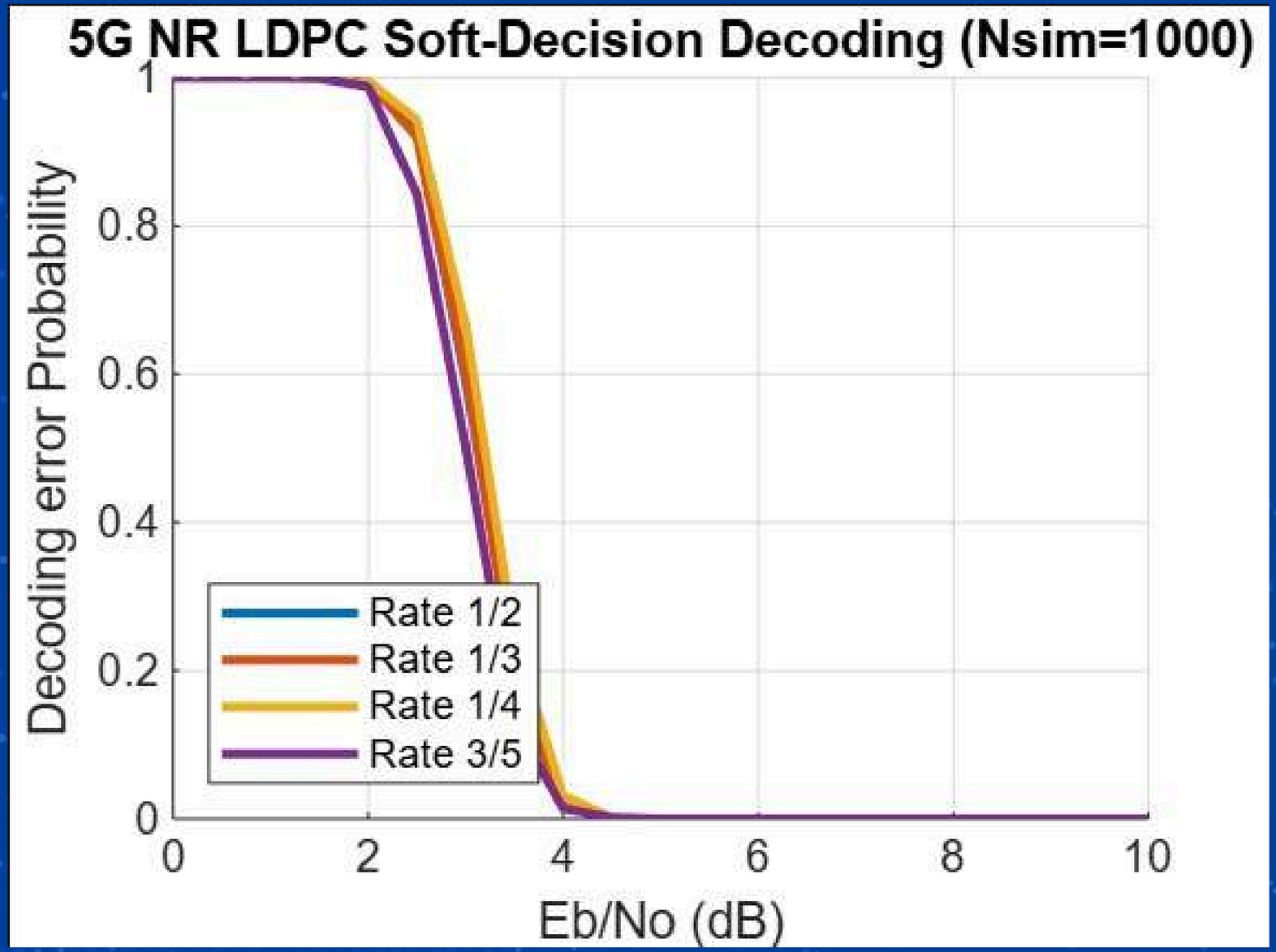




HARD DECISION DECODING COMPARE

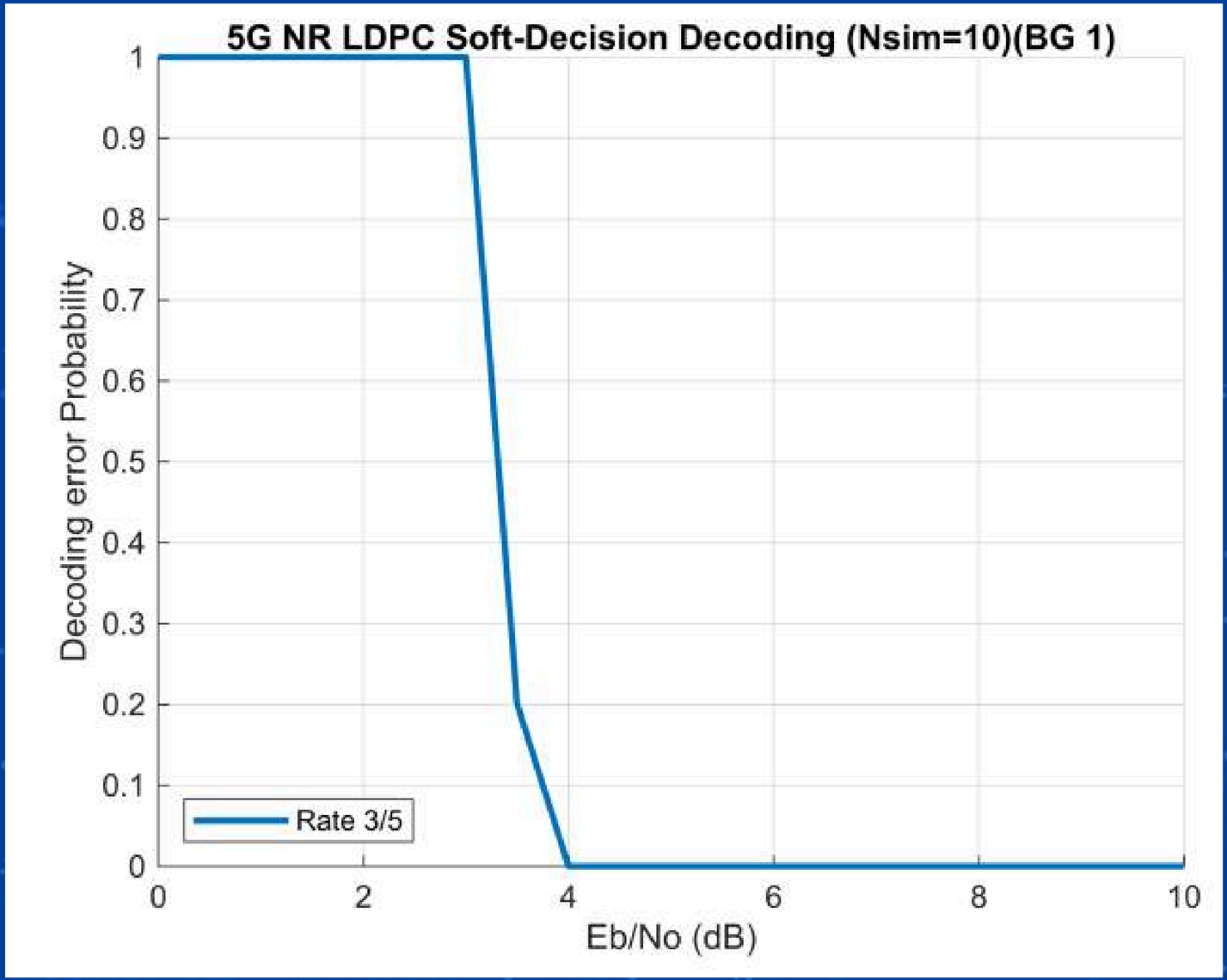


SOFT DECISION DECODING COMPARE

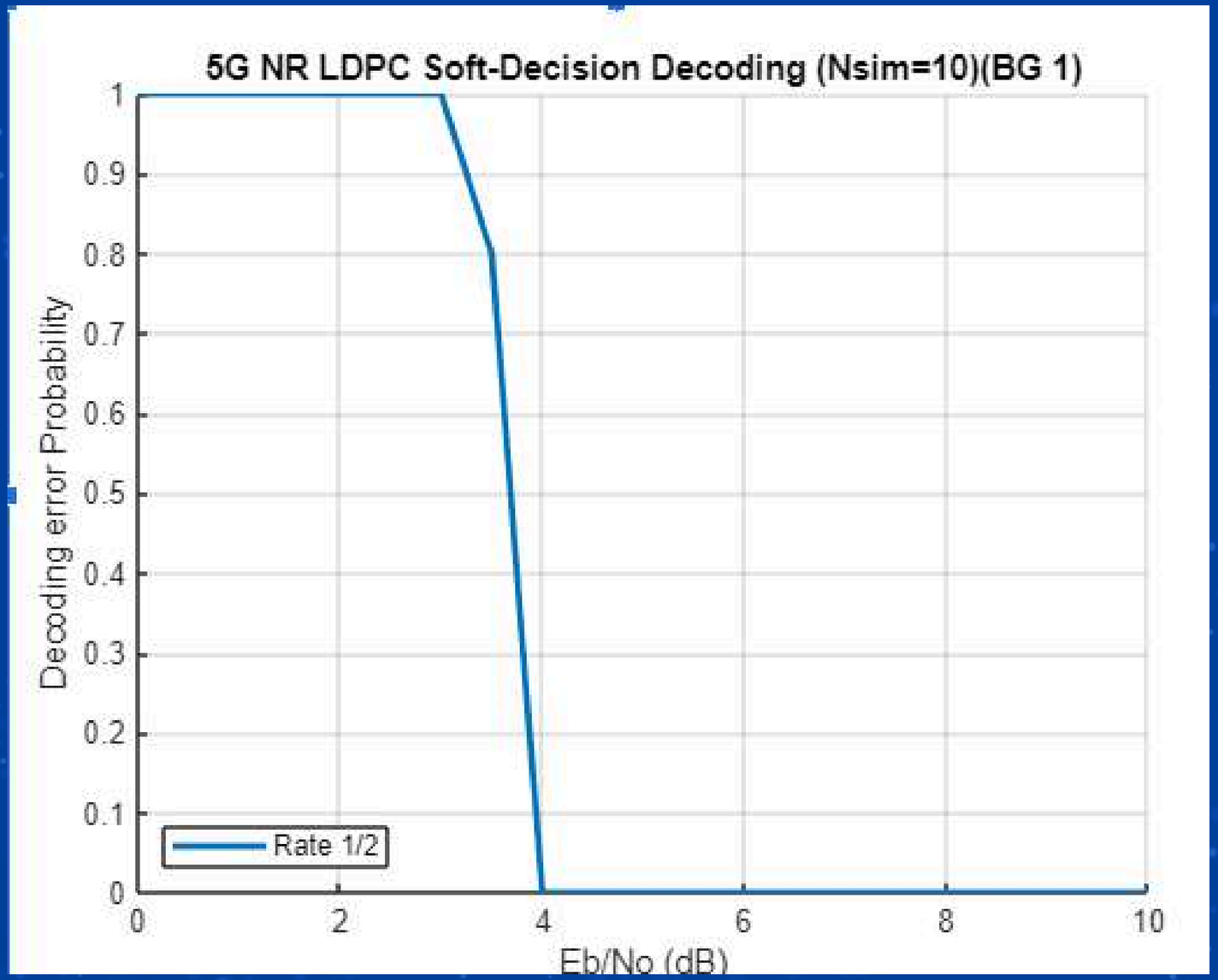


RESULTS OF MATRIX NR_1_6_52

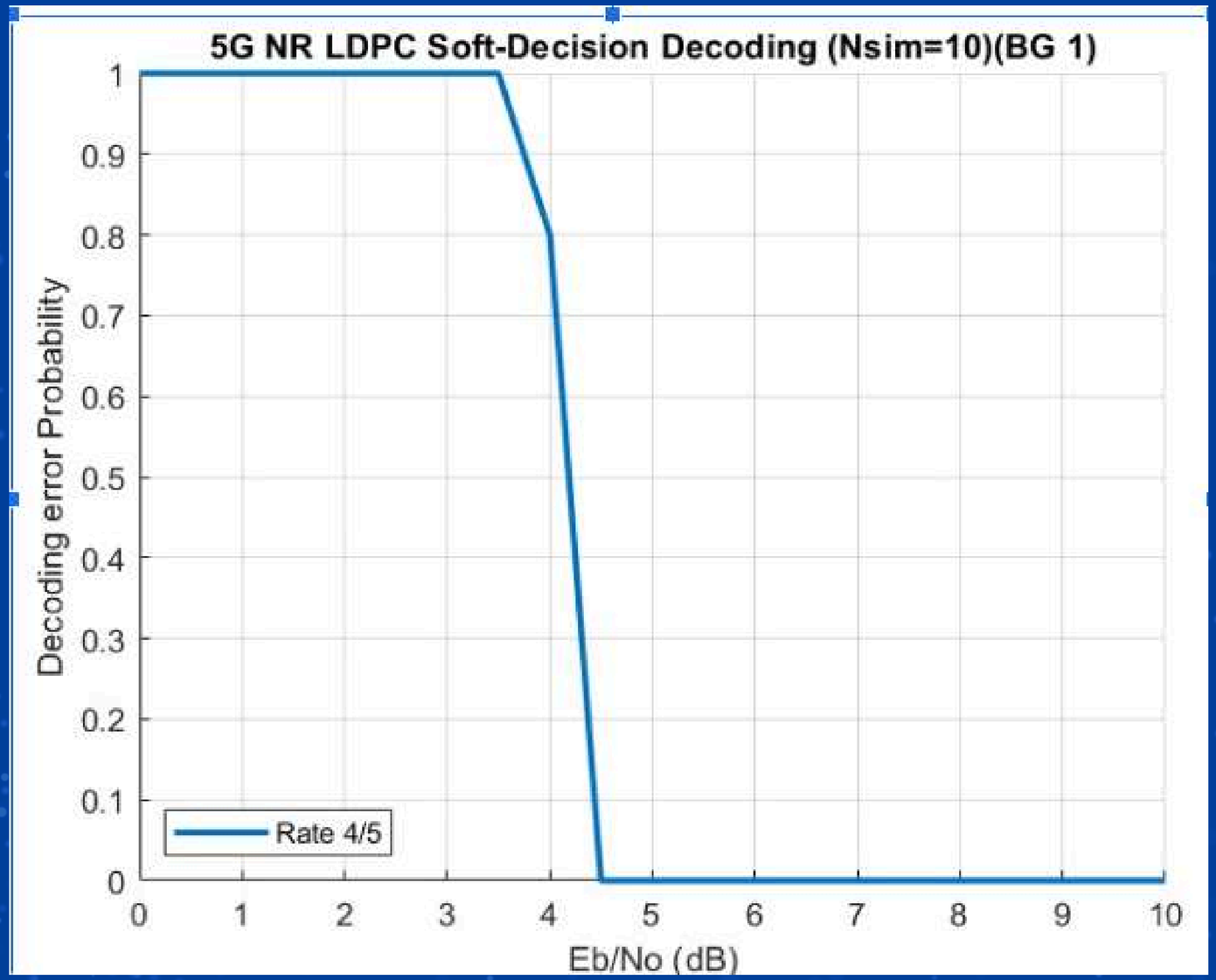
SOFT DECISION DECODING CODERATE = 3/5



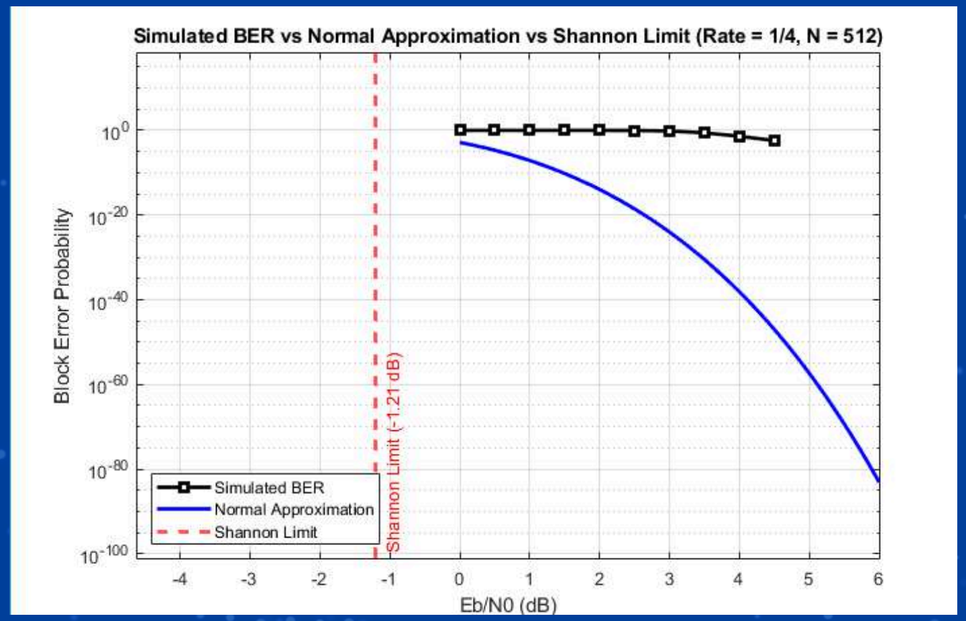
SOFT DECISION DECODING CODERATE = 1/2



SOFT DECISION DECODING CODERATE = 4/5



NORMAL APPROXIMATION AND SHANNON LIMIT



Validate LDPC decoder using benchmarks

Compare decoder performance with:

- Normal Approximation (NA)
- Shannon Limit
- Simulated BER(N=512,rate=1/4)

- Insufficient trials: With N = 512, the smallest measurable BLER is 1/512 ≈ 0.20%, so any error rate below this cannot be detected.
- Log-scale break: Beyond Eb/N_0 = 4.5 dB BER hits zero ($\log_{10}(0) = -\infty$), so the point isn't plotted and the curve gaps—raising Nsim ensures residual errors keep the line continuous.
- To estimate very low error probabilities like 10^(-100), we would need at least 10^(100) simulations.
- This is infeasible in practice due to time and resource constraints.
- Hence, we rely on theoretical tools like the normal approximation for such cases.

NORMAL APPROXIMATION

$$P_{N,}e = Q\left(\sqrt[2]{N \div V}\left(C - r + \left(log_2 N \div 2N\right)\right)\right)$$

Where, $C = log_2\left(1 + P\right)$

$$P = r.\, E_b \div N_o$$

$$V = \left(log_2 e\right)^2.\left(P\left(P + 2\right)\right) \div 2\left(P + 1\right)^2$$

SHANNON LIMIT

$$10log_{10}\left(\left(2^r - 1\right) \div r\right)\ \textbf{dB}$$

CONCLUSION

- LDPC codes are a key part of 5G technology, helping to improve data transmission by reducing errors.

- Their structure and decoding techniques make them well-suited for the demands of high-speed, modern communication systems.

REFERENCES

- Lecture Slides of Channel Coding by Professor Yash Vasavada.
- Video Lectures of NPTEL-NOM IITM by Prof. Andrew Thangaraj on LDPC codes.
- https://www.diva-portal.org/smash/get/diva2:1611415/FULLTEXT01.pdf
- Implementation of Low-Density Parity-Check Codes for 5G NR shared channels by LIFANG WANG

Thank you!

From Group-21