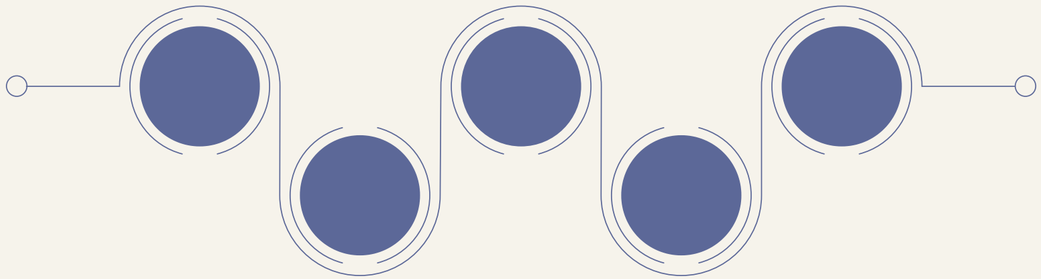
LDPC DECODING FOR 5G NR

CT216

Prof. Yash Vasavda

TA Mentor Akshat Jindal



LAB GROUP: 3

SUB GROUP - 16

Honor Code

We declare that:

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

Overview

- Introduction to LDPC codes
- Program flow
- How LDPC works
- Base Graphes
- Lifting
- Encoding
- Code Rate Management , puncturing
- BPSK Modulation , AWGN Channel
- Hard Decision Decoding
- Soft Decision Decoding
- Log Likelihood` Ratio(LLR)
- soft decoding algorithm

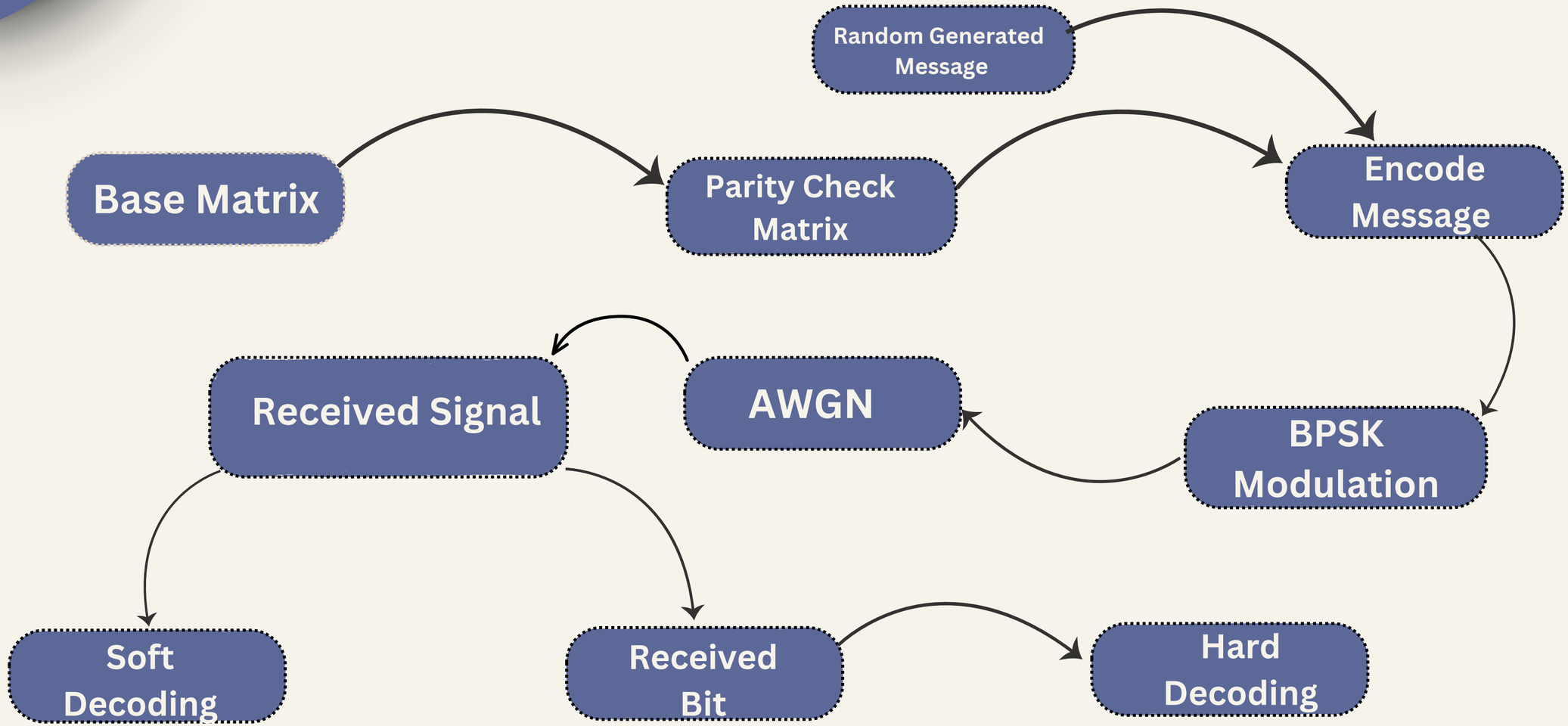
Overview

- Min-Sum Approximation(MSA)
- Practical Implementation
- Hard vs Soft Decoding
- Simulation Summary
- Conclusion
- performance graphs

Introduction

- Low-Density Parity-Check (LDPC) codes a type linear block code with **sparse parity-check** matrices.
- Introduced by Robert Gallager in the 1960s.
- LDPC codes work well with different block sizes and code rates due to the flexibility provided by base graphs.
- It provides minimal latency, very good rate compatibility, and very low error rate.

Program Flow



How LDPC Works

- The main goal in encoding is to calculate the parity bits.
- In LDPC, we find the parity bits using a parity-check matrix constructed from a base graph.
- The encoded codeword is then modulated.
- This modulated codeword is transmitted through an AWGN (Additive White Gaussian Noise) channel, which introduces noise.
- If soft-decision decoding is used, we work with Log-Likelihood Ratios (LLRs) for improved accuracy.
- If hard-decision decoding is used, we demodulate the signal and decode it using the Tanner graph method.

Base Graphs

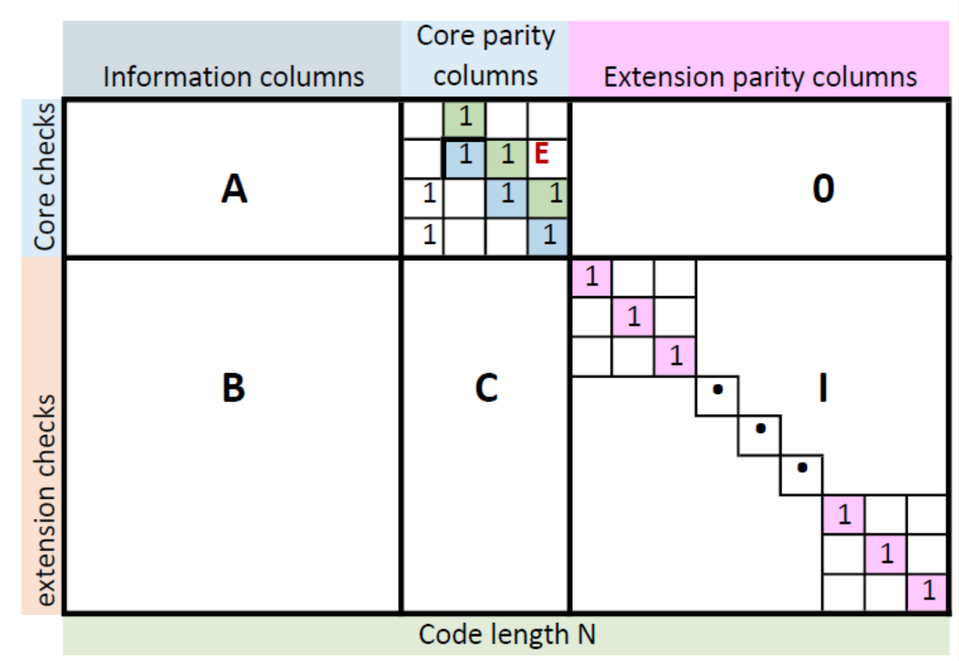
- 5G NR uses two base graphs defines below:
  > BG1: 46×68 matrix (used when block size is large or code rate is high).
  > BG2: 42×52 matrix (used for smaller blocks or lower code rates).

- Matrix structure: It has submatrices A, E ,O, B, C, and I.

- The selection of BG depends on transport block size and desired code rate.

- After Expansion of base graph we will get according parity check (H) matrix.
  By Having Two base graphs we can handle various data size and code rates.

Base graphs block structure;



BG1

- A is 4 x 22 matrix
- E is 4 x 4 matrix
- O is 4 x 42
- B is 42 x 22 matrix
- C is 42 x 4 matrix
- I is 42 x 42 matrix

BG2

- A is 4 x 10 matrix
- E is 4 x 4 matrix
- O is 4 x 38
- B is 38 x 10 matrix
- C is 38 x 4 matrix
- I is 38 x 38 matrix

O is a matrix filled with all -1s, and I is identity matrix according to H matrix means diagonal elements are 0s and rest are -1.

Lifting(expansion of base graphs)

- We use lifting to construct parity check matrix H from the base graph. We will use Z which is expansion factor or lifting size.
- Use the following rules to make parity check matrix.
- It replaces each element in the base graph with a Z x Z matrix.

  -1 → all-zero matrix (size Z x Z)

  0 → identity matrix (size Z x Z)

  P →Right circular shift of identity matrix by (P) positions (size Z x Z)

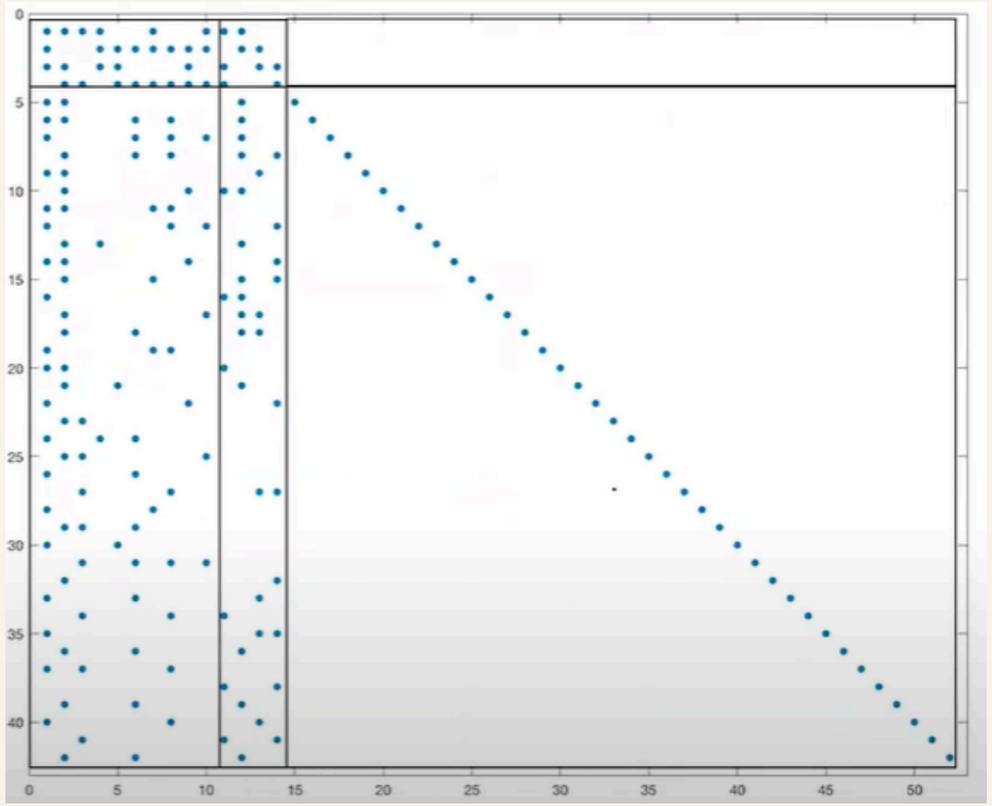
  Where P is positive integer < Z.
- Reduces memory requirements while preserving graph sparsity and structure.

Example :

$$B = \begin{bmatrix} 1 & -1 & 3 & 1 & 0 & -1 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 4 & 2 & 1 & -1 & 0 \end{bmatrix}$$

Expansion factor: 5

$$H = \left[\begin{array}{cccccccccccccccccccccccccccccc}
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0\\
\hline
0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0
\end{array}\right]$$



- Here -1 is represented as blank and others are represented as dots

Encoding using base graph

- To encode the message we will find the parities using the H matrix (or we can use base graph directly to encode the message.)
- In encoding the double diagonal structure of sub matrix E plays crucial role.
- Take this example

$$H = \begin{bmatrix} I_1 & 0 & I_3 & I_1 & I_2 & I & 0 & 0 \\ I_2 & I & 0 & I_3 & 0 & I & I & 0 \\ 0 & I_4 & I_2 & I & I_1 & 0 & I & I \\ I_4 & I_1 & I & 0 & I_2 & 0 & 0 & I \end{bmatrix}$$

Expansion: 5

- From this H matrix , codeword will be [m1 m2 m3 m4 p1 p2 p3 p4].
- $I_1 m_1 + I_3 m_3 + I_1 m_4 + I_2 p_1 + I p_2 = 0$. (eq.1)
- $I_2 m_1 + I m_2 + I_3 m_3 + I p_2 + I p_3 = 0$ (eq.2)
- $I_4 m_2 + I_2 m_3 + I m_4 + I_1 p_1 + I p_3 + I p_4 = 0$ (eq.3)
- $I_4 m_1 + I_1 m_2 + I m_3 + I_2 p_1 + I p_4 = 0$ (eq.4)

Encoding using base graph

- Solve this equation by adding all the eq. to get value of p1.Now, substitute p1 in eq. 1 to get p2 and same way we get p3 and p4 by substitution.
- Equation obtained by adding all the eq.

l1m1 + l3m3 + l1m4 + l2 m1 + lm2 + l3m3 + l4m2 + l2m3 + lm4 + l4m1 + l1m2 + lm3 =l1 p1.

Coderate Matching

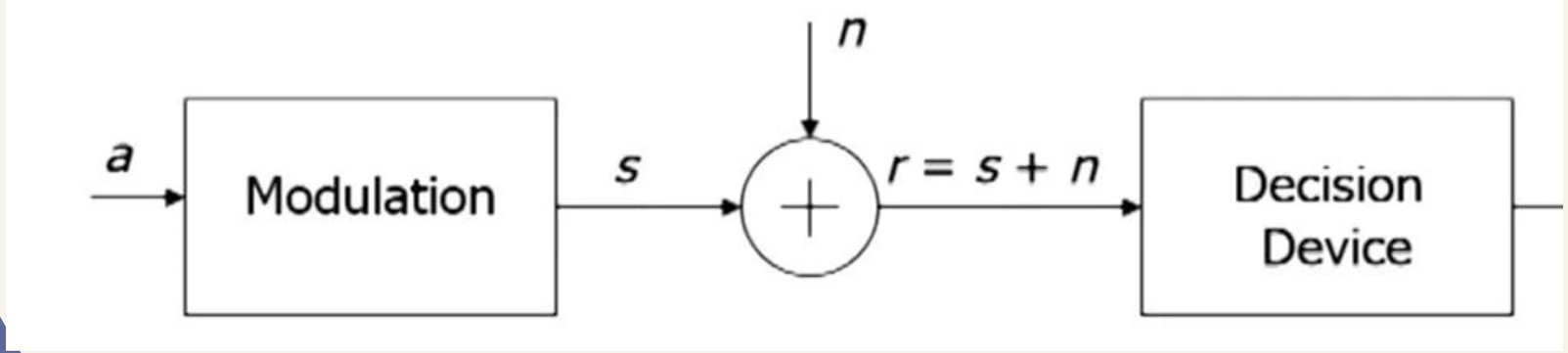
- If we want a specific code rate for transmission then we have to do code matching in LDPC.
- So we have to remove some parity bits to adjust the coderate, and this is called puncturing.
- Let base matrix with size m x n, and expansion factor Z.
- Total length of codeword is n*Z, out of this n*Z bits 2*Z are always punctured.
- Let, for desired coderate a*Z bits have to be removed so total number of bits after puncturing = (n-2) * Z - (a*Z).
- And out of this total bits the number of message bits = (n-m) * Z.

- So the formula for code rate will be,

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

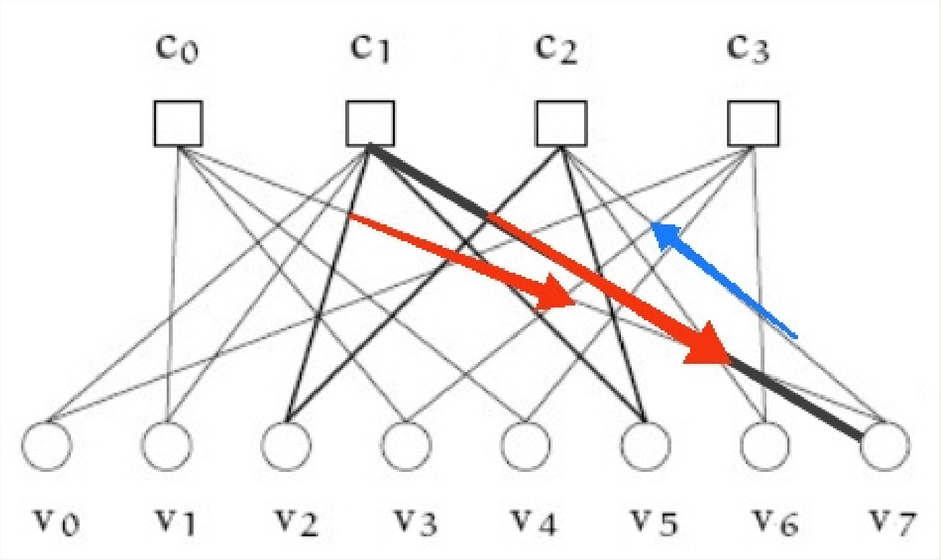
BPSK And AWGN noise:

- BPSK is used to map bits to symbols for transmission
- Bit 0 → mapped to 1
- Bit 1 → mapped to -1
- Then the modulated signal is transmitted through AWGN which introduces noise
- Variance of noise σ^2 = 1/(2*SNR*coderate)



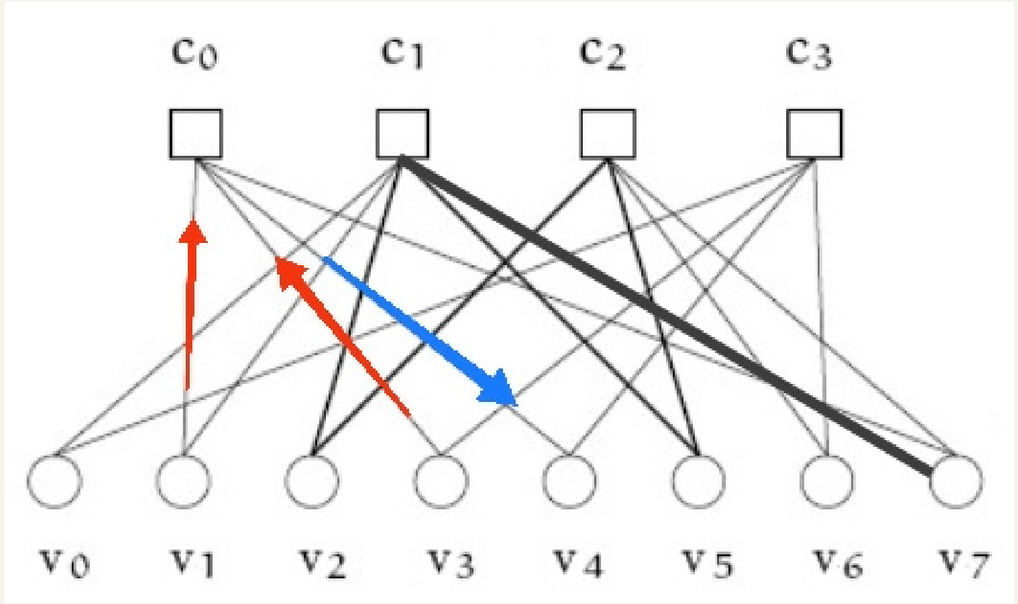
Hard Decision Decoding

- First all variable nodes has its value (bit)
  - In first iteration all variable nodes (VN) sends its value to their connected check nodes (CN).
  - in rest iterations, VNi sends CNj is the value that is majority of all values received by the VNi from all connected checknodes, except from CNj.



Hard Decision Decoding

- Now each check node is a SPC code, so if VNi is connected with CNj then the value at VNi should be XOR of values all other VNs connected to CNj.
- so each CNj sends VNi is the value that is XOR of all value received by the CNj from all connected variable nodes, except from VNi.



Hard Decision Decoding

- To make this communication we need to know that which variable node (VN) is connected by which check node (CN).
- We can simply do traverse of H matrix and make the communication between them, if H(i, j) is 1 then i and j are connected.
- But H matrix will be large, we can reduce this traversals by do mapping of which CNs are connected to VNi and which VNs are connected to CNi, plus H is sparse matrix so its more beneficial to do mapping first then just traverse according to map, than traverse whole H matrix each time.

Soft Decision Decoding

- In soft decision decoding the communication between CNs and VNs done using probabilities (log likelihood ratio) to make more accurate decoding decisions.
- There are two kind of beliefs
  - Intrinsic → the initial value of VN means given by channel
  - Extrinsic → the value we get in decoding by excluding node itself.
- In soft decision decoding we will be using tanner graph to make communication between CNs and VNs

Log-Likelihood Ratio (LLR)

- The Log-Likelihood Ratio quantifies the likelihood of a bit being 0 or 1, given the received noisy signal. It is defined as:

$$L(c_i) = \log\left(\frac{P(c_i = 0 \mid y_i)}{P(c_i = 1 \mid y_i)}\right)$$

where: c_i is the transmitted bit. y_i is the received noisy value.

- Interpretation:

  $L(c_i) > 0$ implies that bit is likely to 0

  $L(c_i) < 0$ implies that bit is likely to 1

  The larger the absolute value of LLR, the more confident the decision

- For BPSK modulation over an AWGN channel, the LLR can be approximated by:

  $L(c_i) \approx 2 * y_i / \sigma^2$, $L(c_i) \approx y_i$ (because we care only about sign and relative abs value)

  where y_i is the received value and σ^2 is the noise variance.

Soft Decoding Algo.

- It uses a message-passing framework over the Tanner graph. Messages passed are the LLRs.
- Two main steps per iteration:
    1. Variable Node to Check Node (VN → CN):

$$m_{v \to c}^{(t)} = L_{ch}(v) + \sum_{c' \in N(v) \setminus \{c\}} m_{c' \to v}^{(t)}$$

    2. Check Node to Variable Node (CN → VN):

$$m_{c \to v}^{(t)} = 2 \tanh^{-1} \left(\prod_{v' \in N(c) \setminus \{v\}} \tanh \left(\frac{m_{v' \to c}^{(t-1)}}{2} \right) \right)$$

        - Iteration stops when:
            - All parity-check equations are satisfied (i.e., $H * C^T = 0$), or
            - Maximum number of iterations is reached.

Min-Sum Approximation (MSA)

- Algo. described before is accurate but requires complex tanh and tanh-inverse functions. we can be simplifies this by replacing these with a minimum operation:

$$m_{c \to v}^{(t)} = \left(\prod_{v' \in N(c) \setminus \{v\}} \text{sign}(m_{v' \to c}^{(t-1)}) \right) \times \min_{v' \in N(c) \setminus \{v\}} \left| m_{v' \to c}^{(t-1)} \right|$$

  - Faster and easier to implement in hardware
  - Slightly less accurate due to approximation
  - Suitable for real-time embedded systems

Practical Implementation of LDPC Decoding

- To implement LDPC decoding in MATLAB or other environments:
  1. Construct H matrix from the base graph and lifting size.
  2. Initialize LLRs from the received noisy symbols (BPSK + AWGN).
  3. Create mapping structures:
     - CN_to_VN_map: All connected VNs to CNi
     - VN_to_CN_map: All connected CNs to VNi
  4. Create L matrix: same size as H, where each non-zero position holds the current LLR
  5. Iteratively update messages:
     - Use MSA for CN → VNi (skip value came from VNi)
     - Use summation for VN → CNj (skip value came from CNj)
  6. Estimate decoded message ĉ and check if H * ĉ^T = 0
  7. Stop after convergence or after reaching iteration limit

- **Example : let's take H matrix as follows -**

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

- **The received message be -**

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

- **Forming the L matrix by using the received message and the H matrix**

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

- **Let’s take the first row :**

$$\begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \end{bmatrix}$$

- **By substituting the non-zero values with minimum of all other non-zero values, we get the updated values of row1:**

$$\begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \end{bmatrix}$$

- **Doing the same for all other rows, we get the updated matrix as follows:**

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

- **We can find the updated belief values by adding every value present in a column for every column**

r = $\begin{bmatrix} 0.2 \end{bmatrix}$

$$\begin{bmatrix} -0.7 \\ 0 \\ -0.7 \\ -0.4 \end{bmatrix}$$

sum = -1

New entry in the column = Total sum + Old Entry.

- **The updated belief after 1st iteration is:**

$$\begin{bmatrix} -1 & -0.4 & 1.1 & -0.6 & 0.4 & 0.7 & -0.7 \end{bmatrix}$$

- **The updated L matrix is:**

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

- **We then convert the new vector c to 0s and 1s and compare it with original message to check if it is decoded successfully.**
- **This updated belief array or $\hat{C}$ can be checked using $H*\hat{C} = 0$ equation. If not, such iterations are repeated until we get convergence.**

*this whole example is taken from andrew thangaraj course

Hard vs Soft Decision Decoding

FEATURE	HARD DECISION DECODING	Soft Decision Decoding
INPUT TYPE	Binary (0 or 1)	Real Valued (LLR)
Uses Channel Reliability	Less than Soft	Yes
Accuracy	Moderate	High
Complexity	Low	Higher
5G NR Suitability	Not Used	Standard method

- Hard decoding simply applies thresholding (e.g., if received > 0 then 0 else 1), while soft decoding takes into account how reliable the received bit is.

Simulation Summary

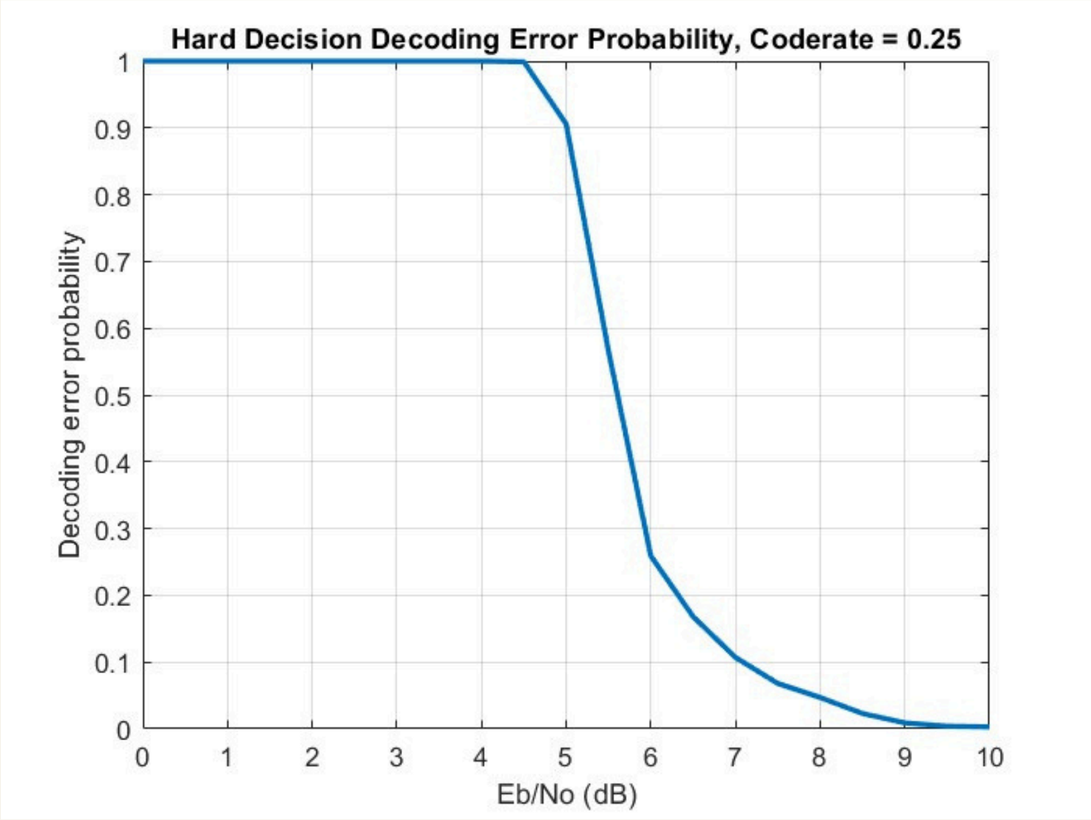
- Soft Decoding achieves lower BER at all SNR levels.
- At $Eb/N_0 \geqslant 2.5$ dB, BER drops to $\sim 10^{-5}$ for soft decoding.
- Hard decoding typically needs > 8 dB for similar performance.
- For code rates like 1/4, soft decoding is extremely robust.
- Lower code rates add more parity bits → better error correction.

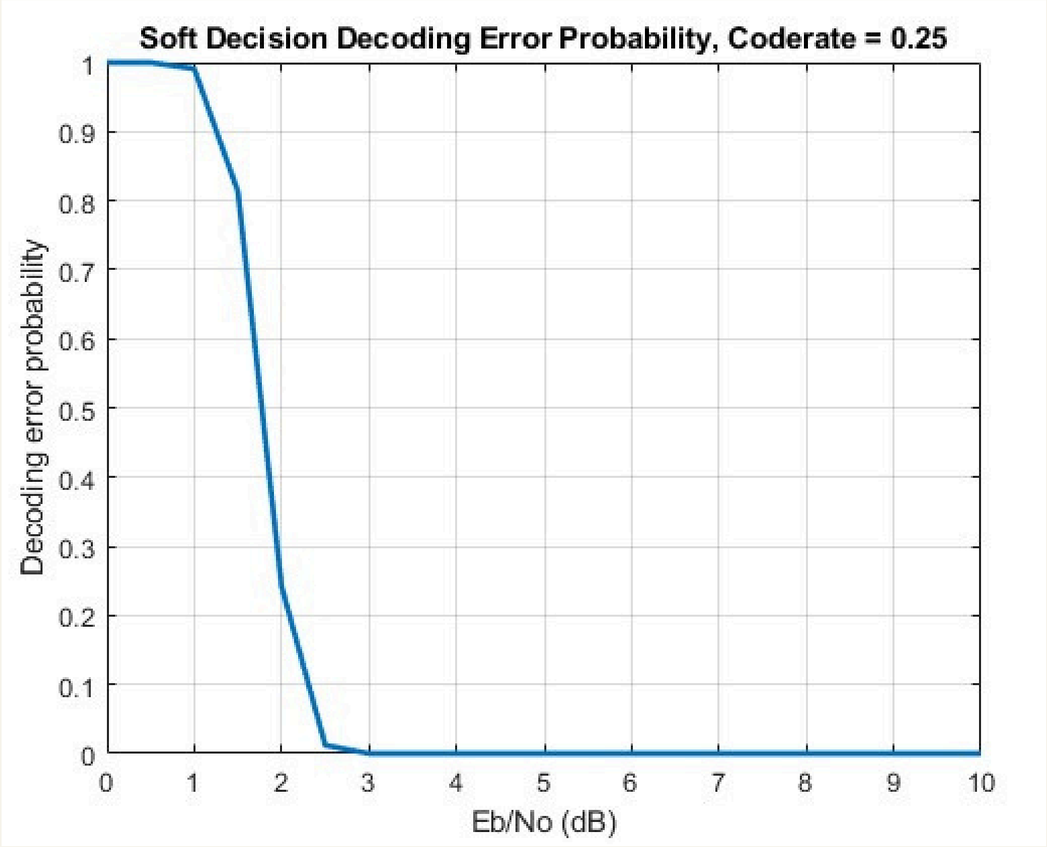
Conclusion

- Soft decision decoding with Min-Sum provides a good tradeoff between speed and performance.
- It is the standard for LDPC in 5G NR shared channels.
- Iterative decoding helps reach near-Shannon performance.
- LDPC codes combined with soft decision decoding offer low error rates, high speed, and adaptability.

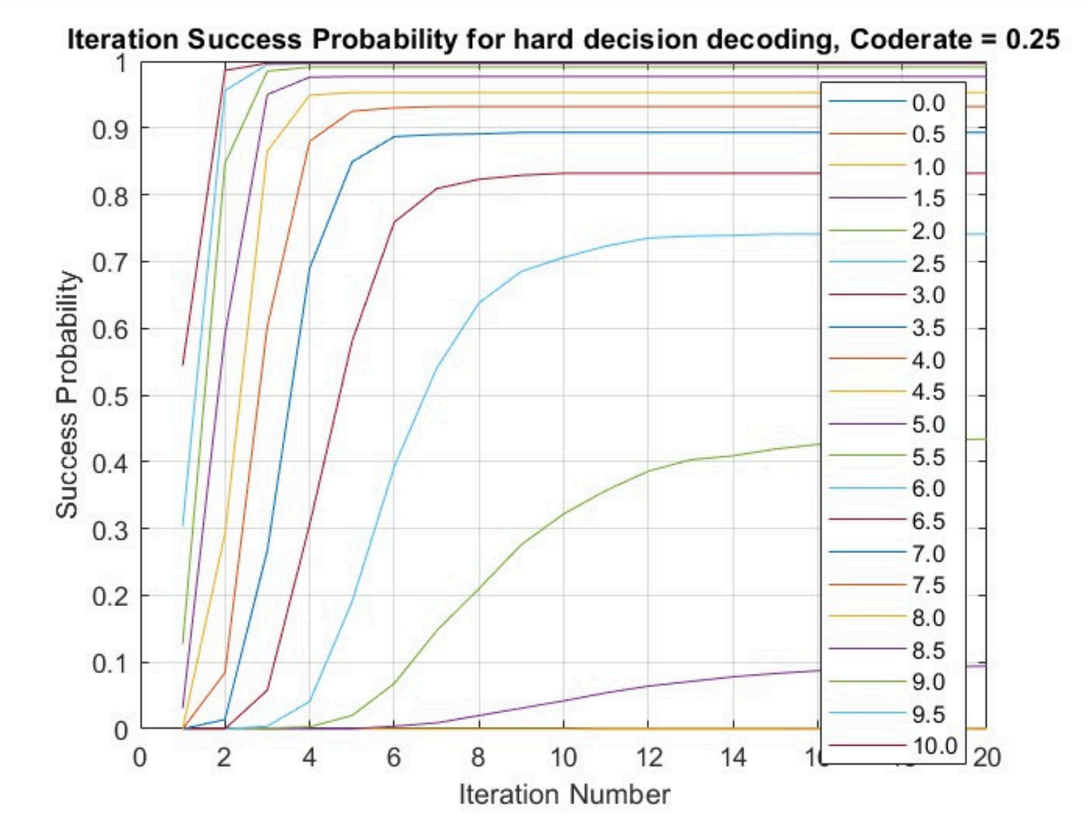
RESULTS OF NR_2_6_52

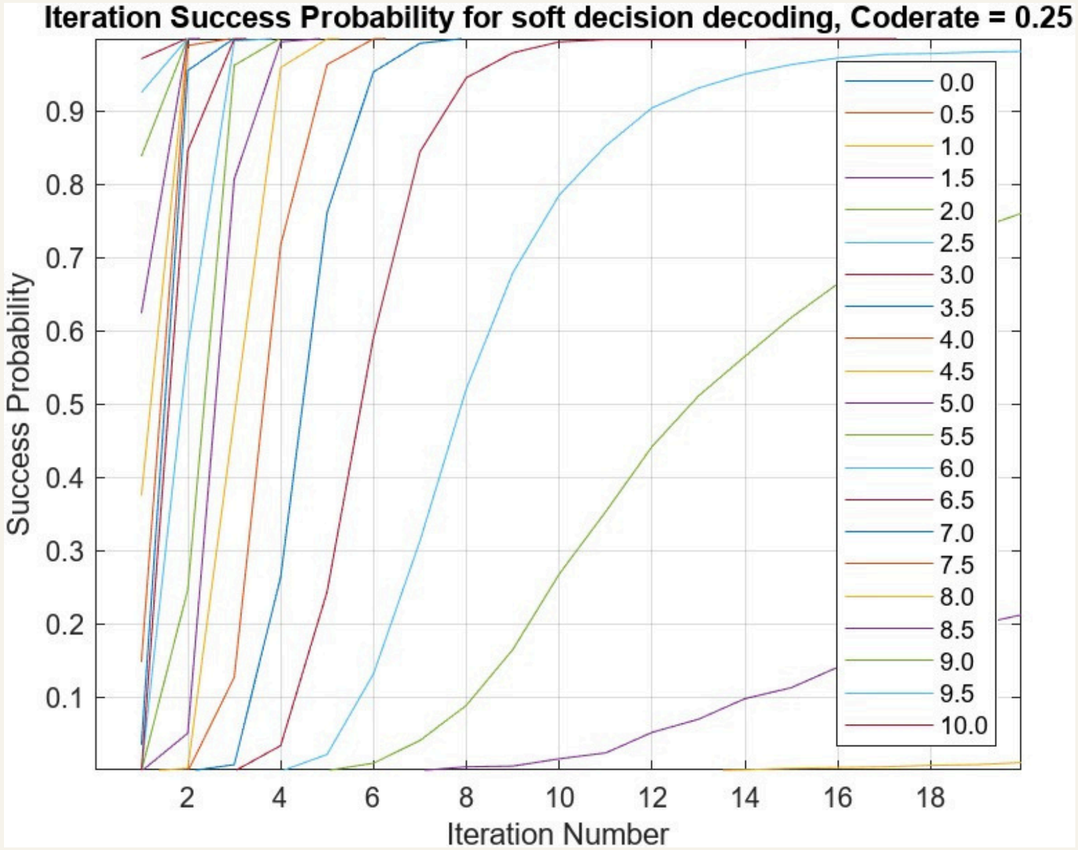
DECODING ERROR PROBABILITY vs Eb/No FOR CODE RATE = 1/4



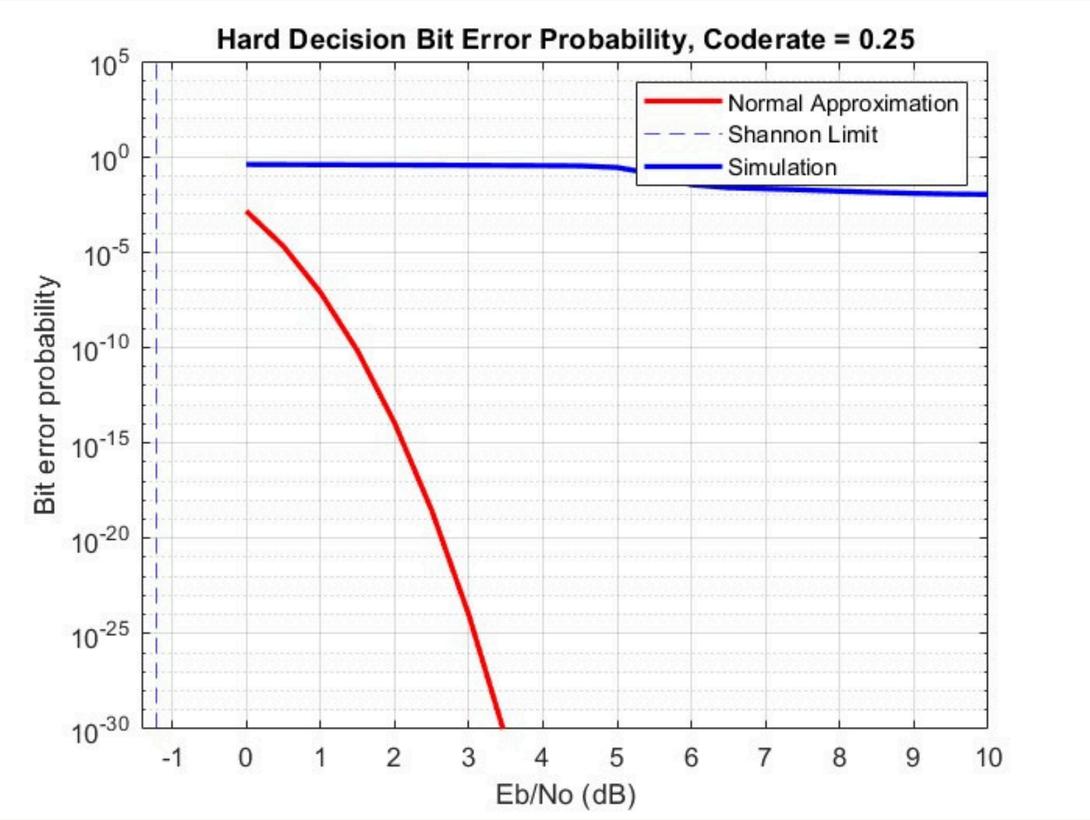


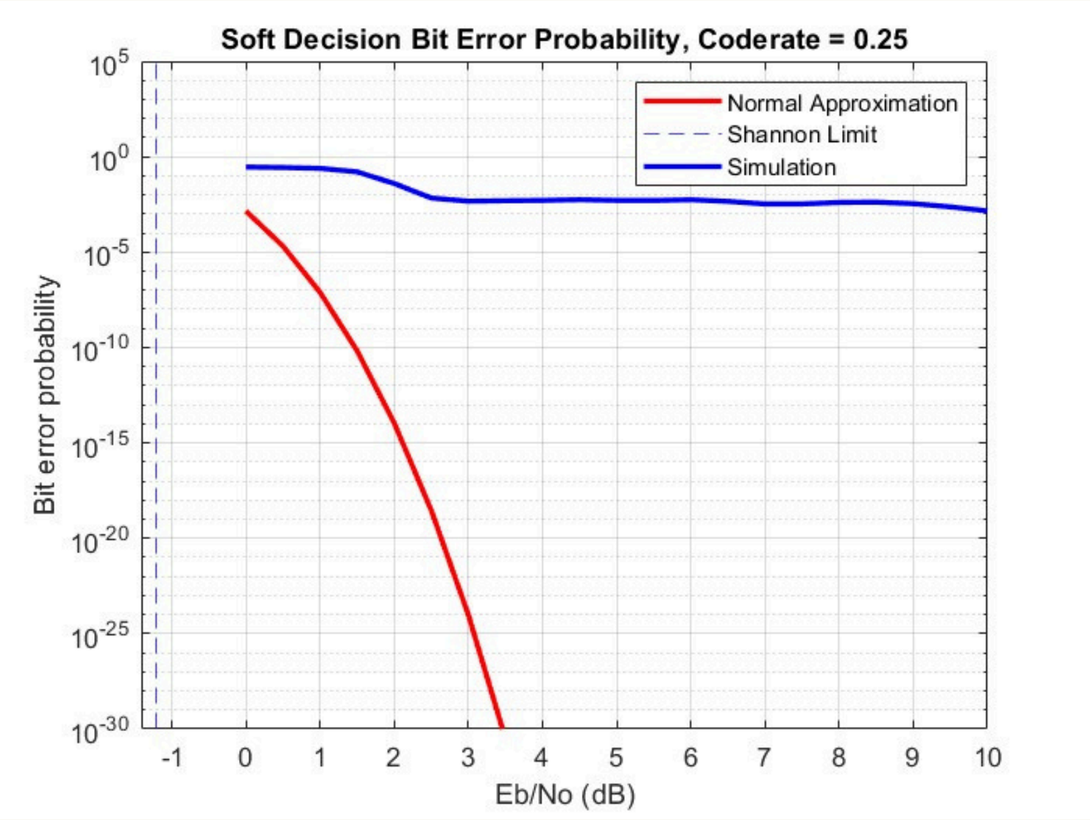
SUCCESS PROBABILITY vs ITERATION FOR CODE RATE = 1/4



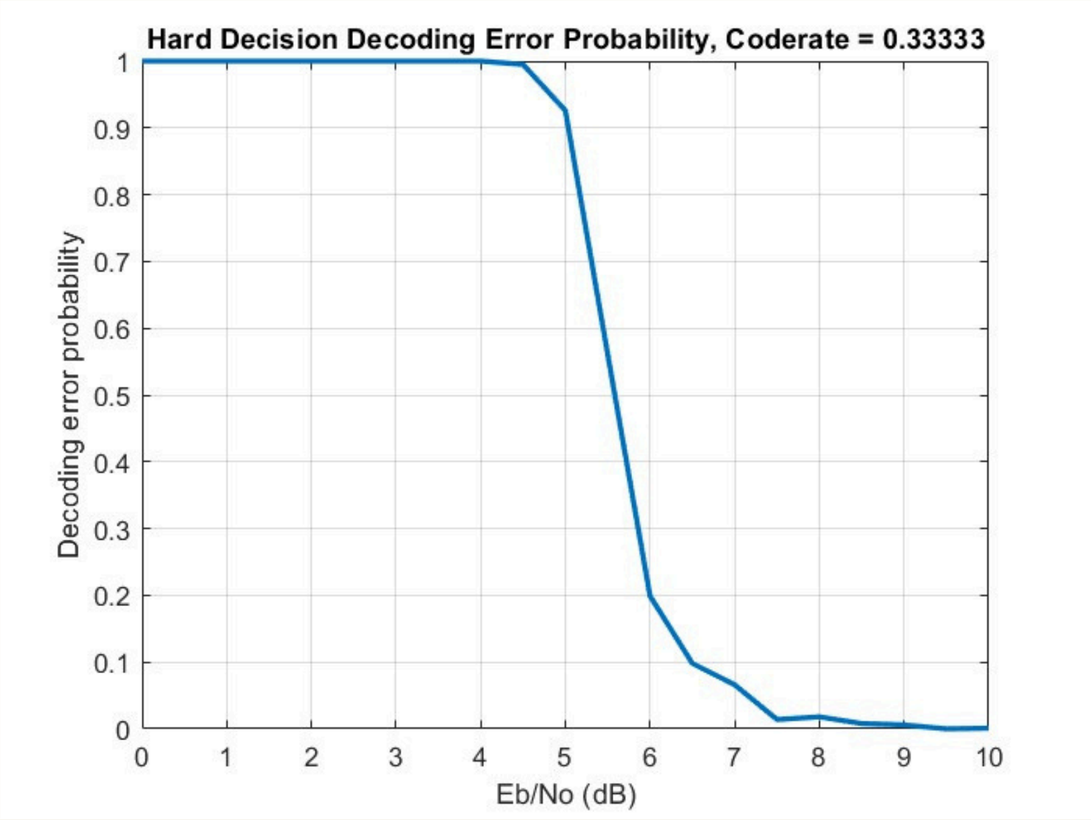


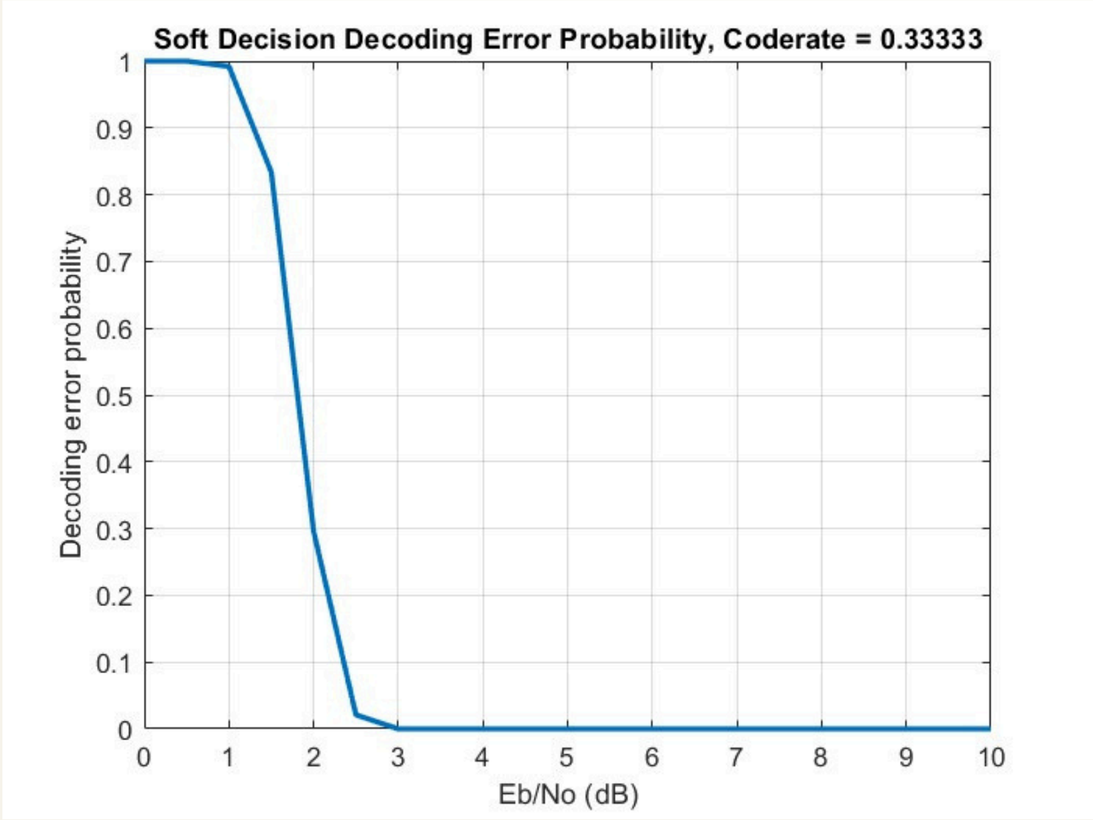
BIT ERROR PROBABILITY vs Eb/No FOR CODE RATE = 1/4



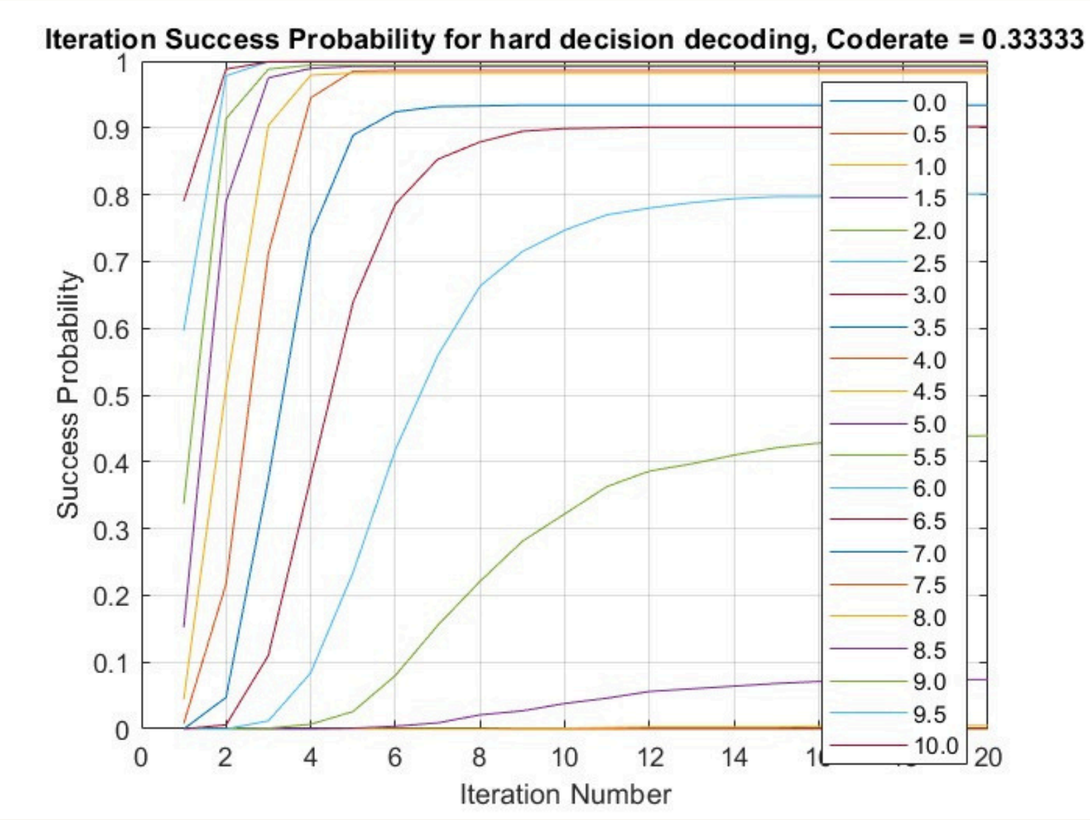


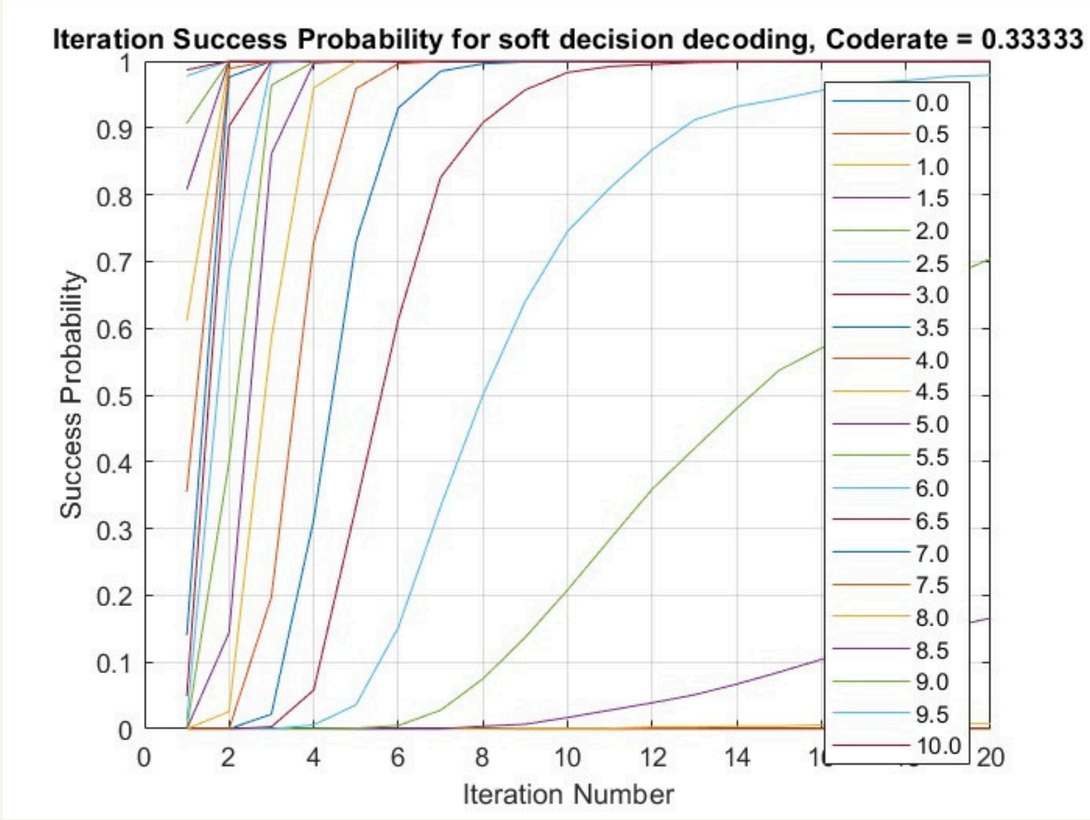
DECODING ERROR PROBABILITY vs Eb/No FOR CODE RATE = 1/3



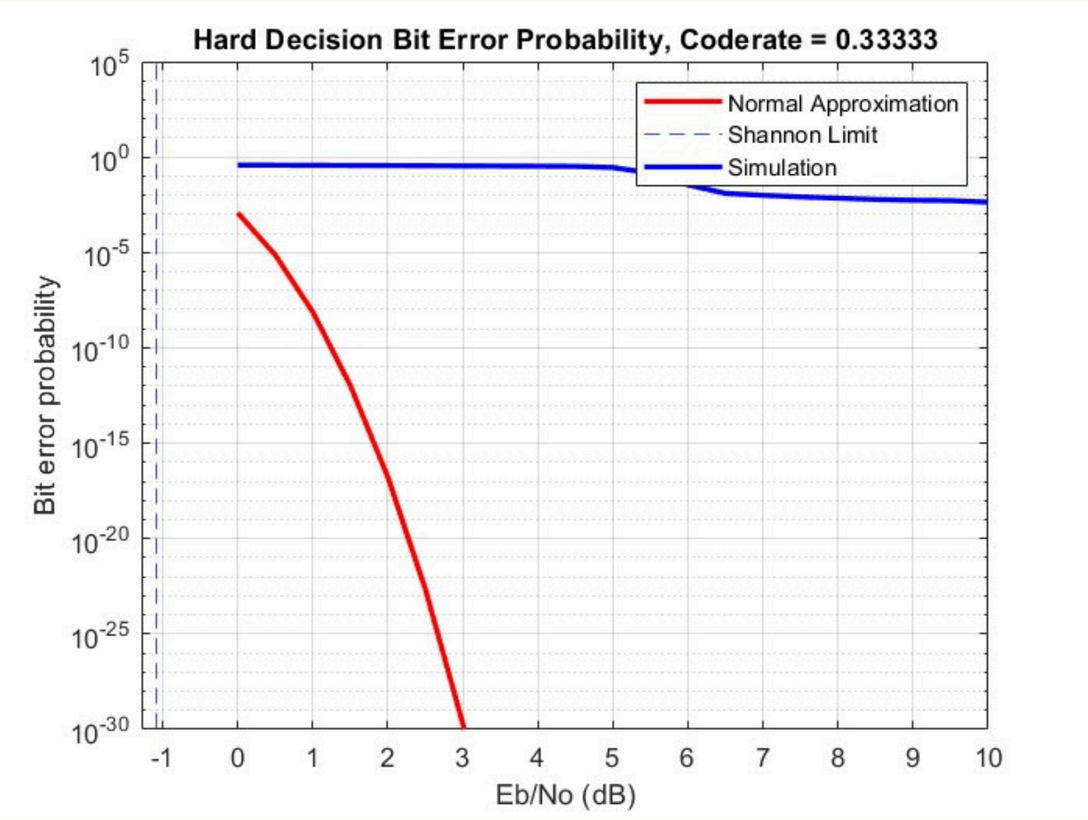


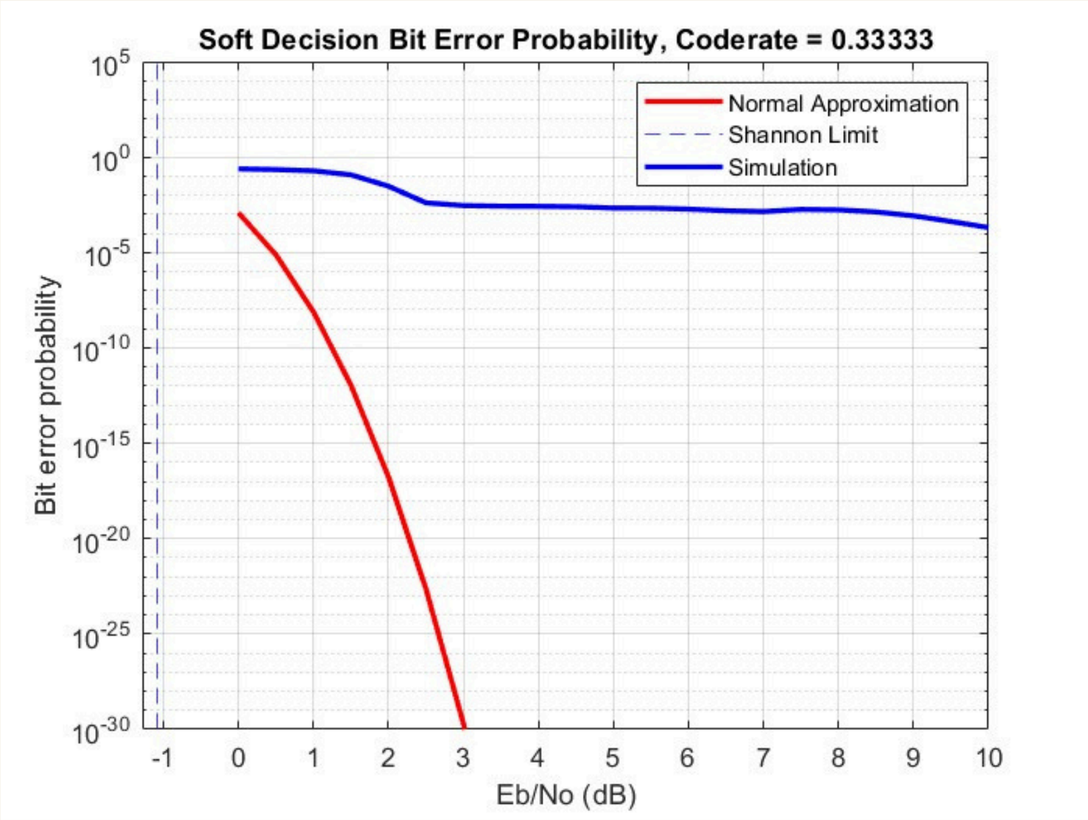
SUCCESS PROBABILITY vs ITERATION FOR CODE RATE = 1/3



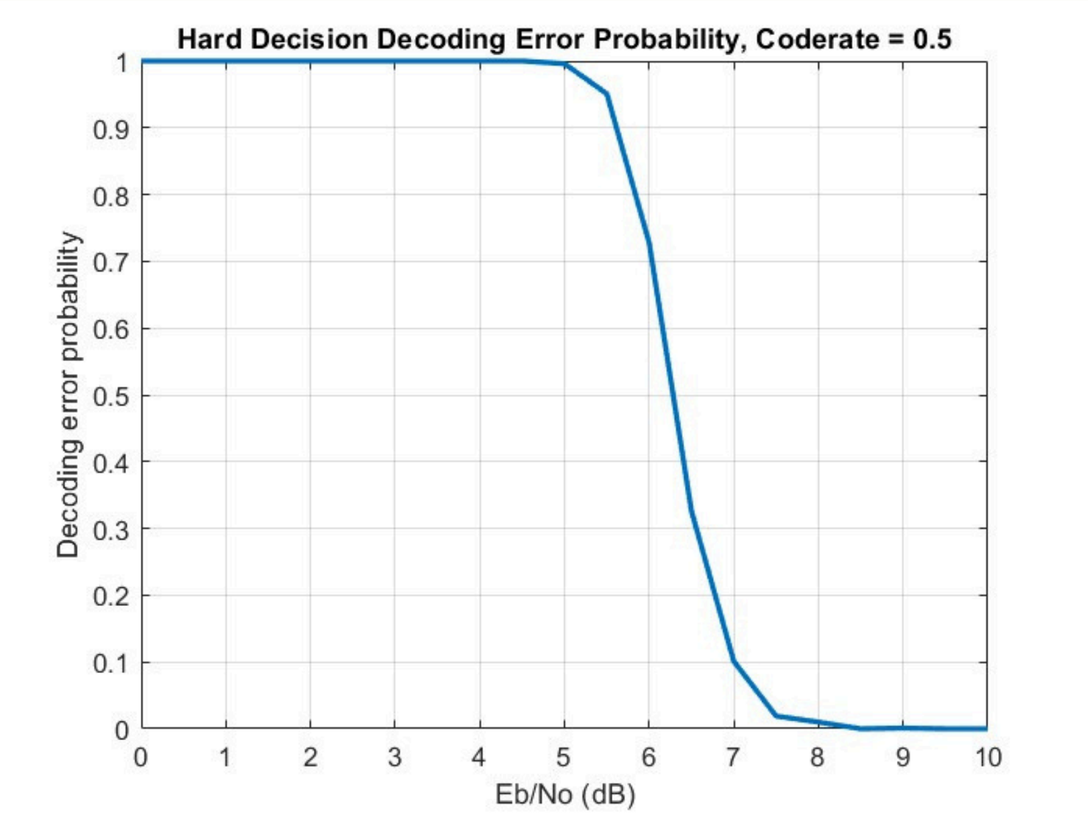


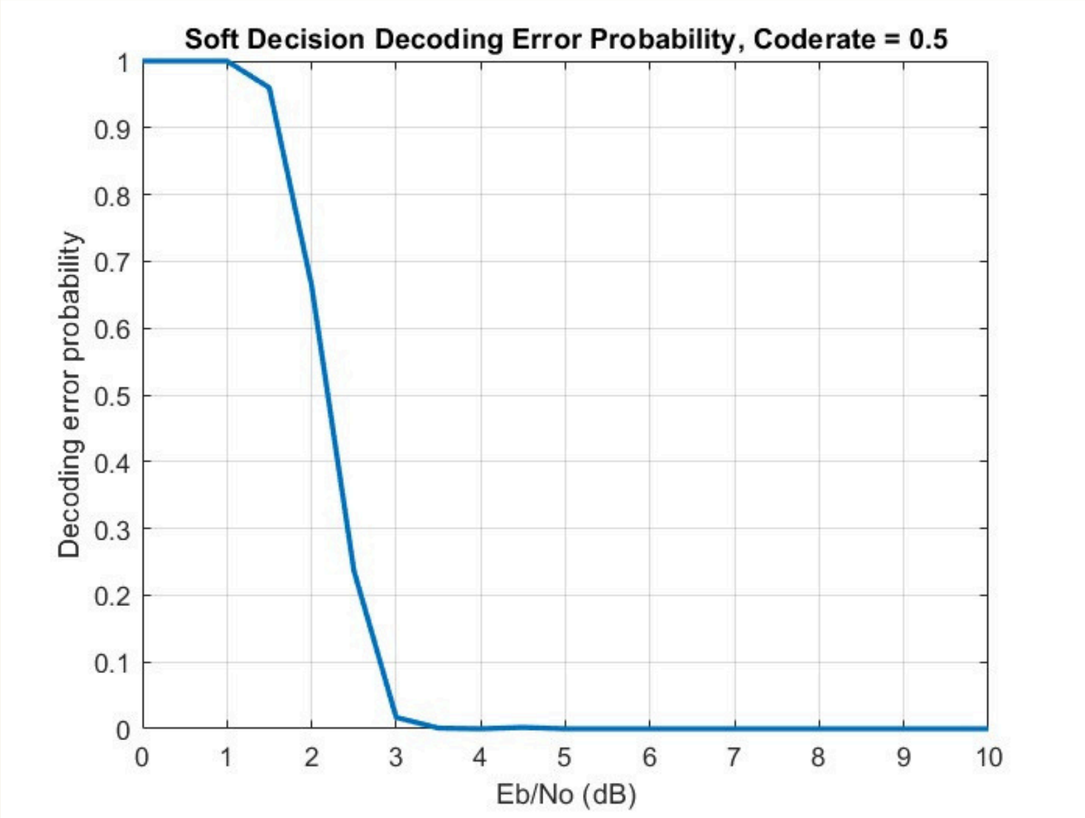
BIT ERROR PROBABILITY vs Eb/No FOR CODE RATE = 1/3



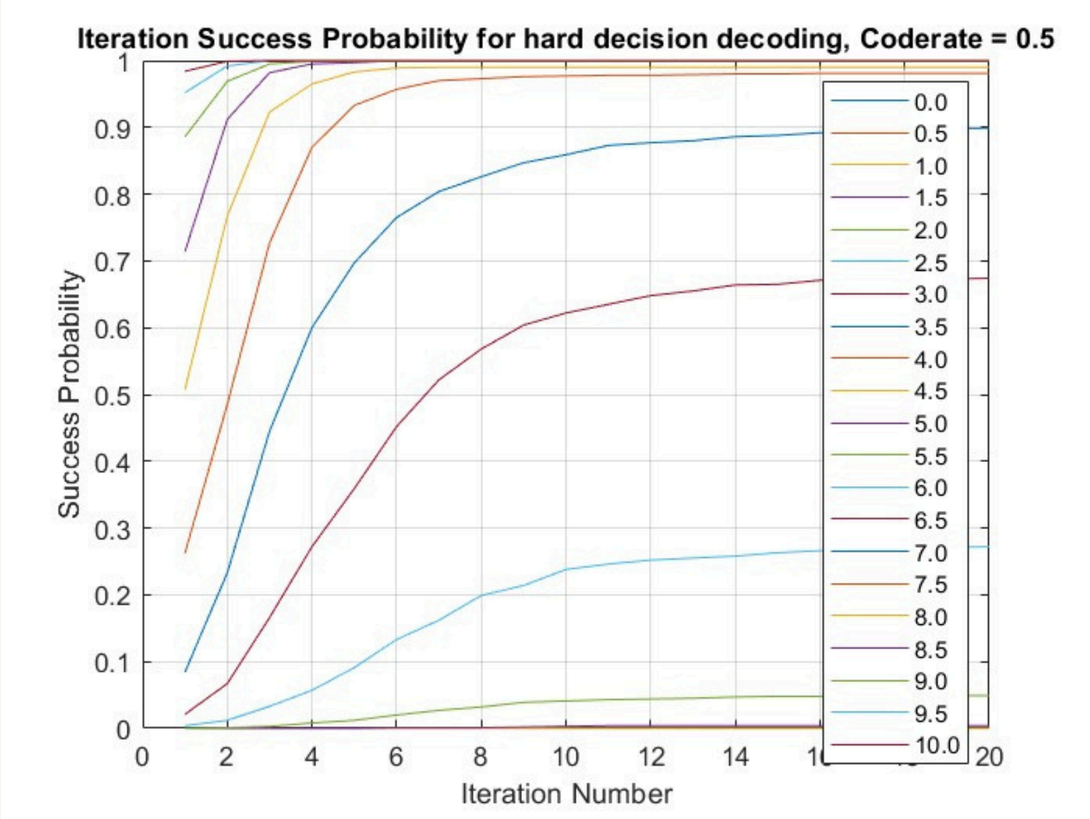


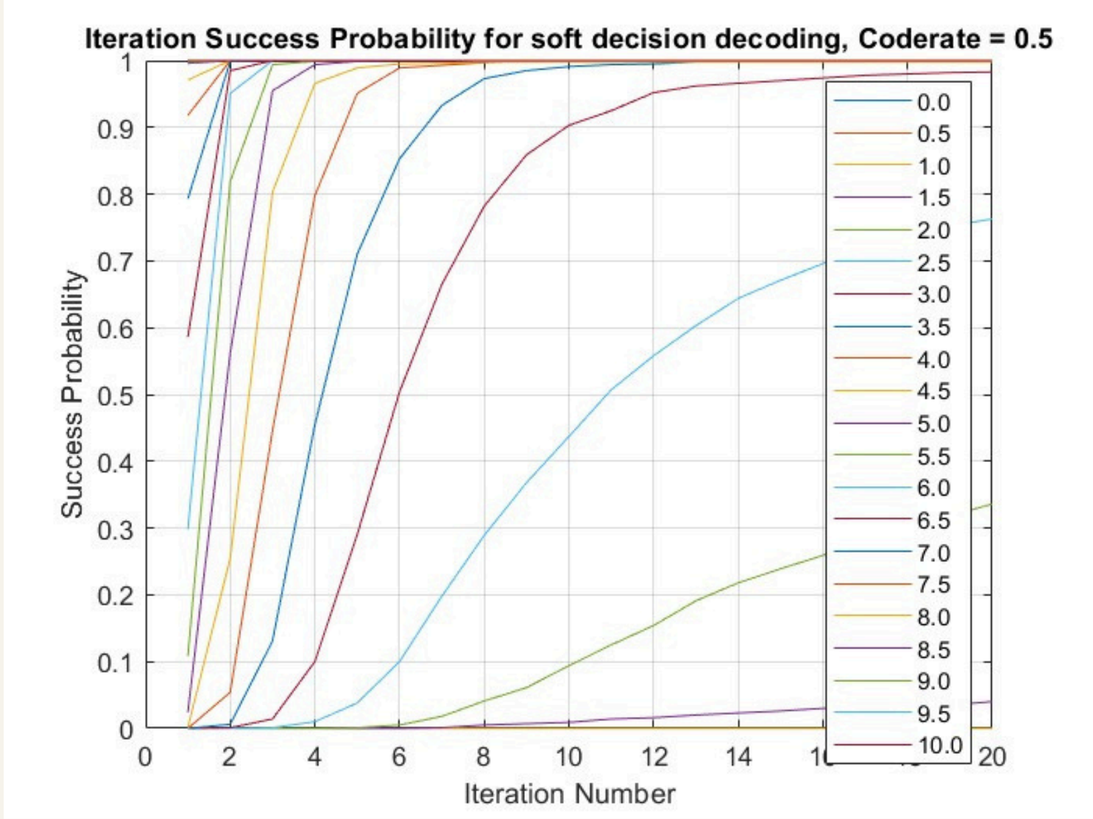
DECODING ERROR PROBABILITY vs Eb/No FOR CODE RATE = 1/2



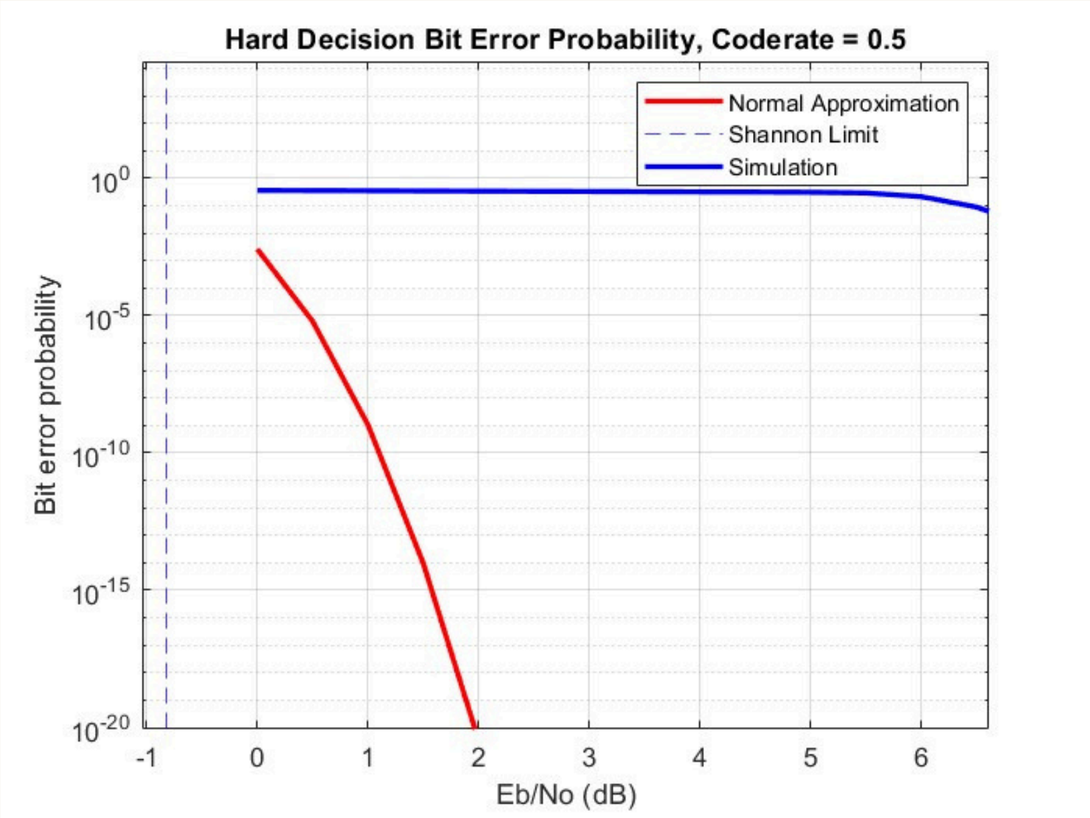


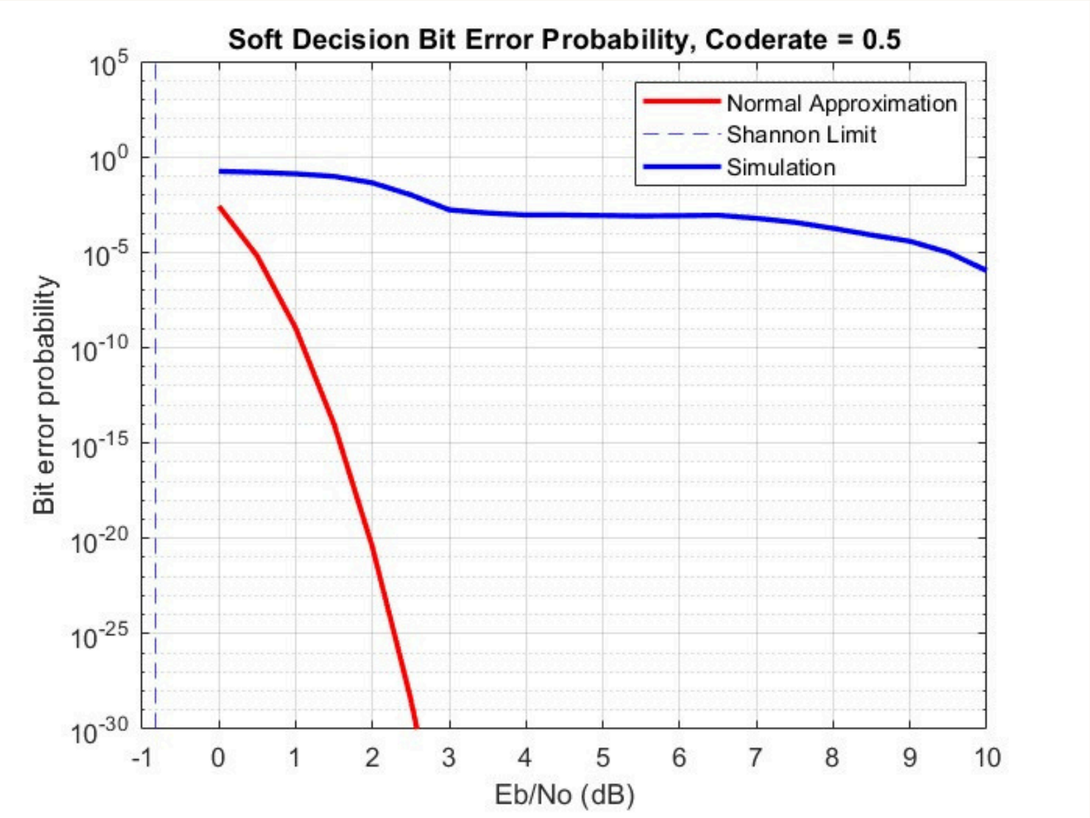
SUCCESS PROBABILITY vs ITERATIONFOR CODE RATE = 1/2



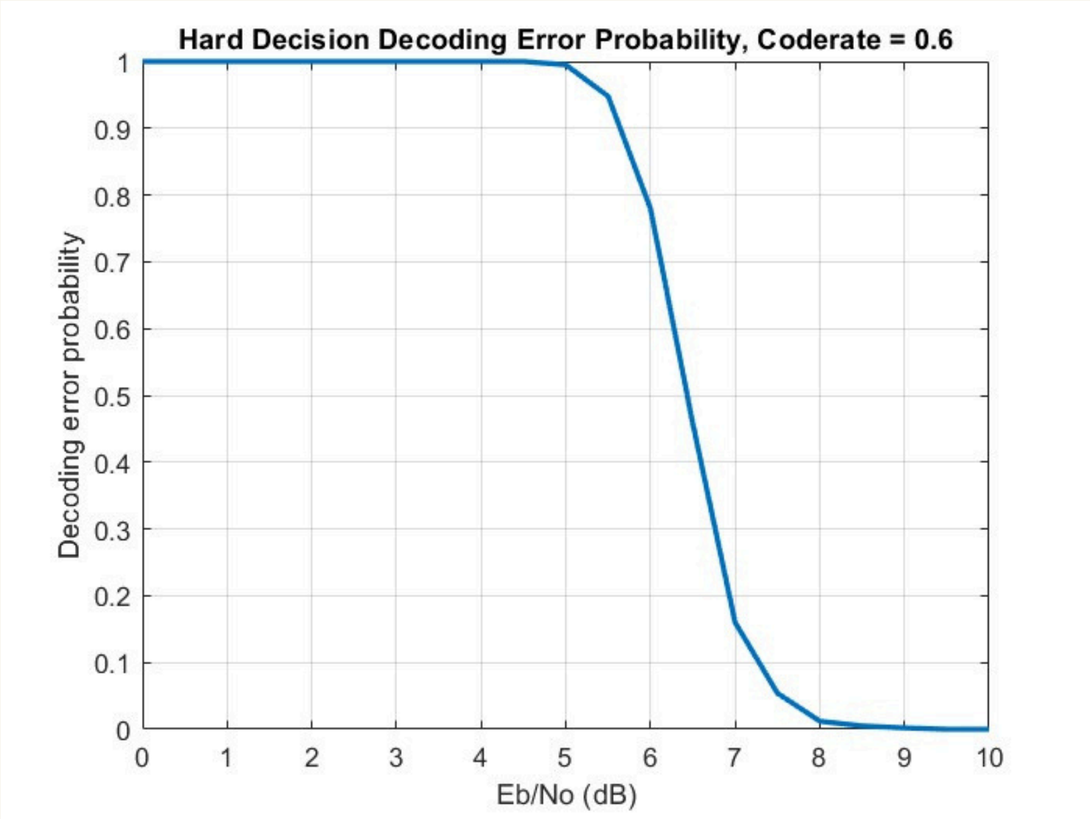


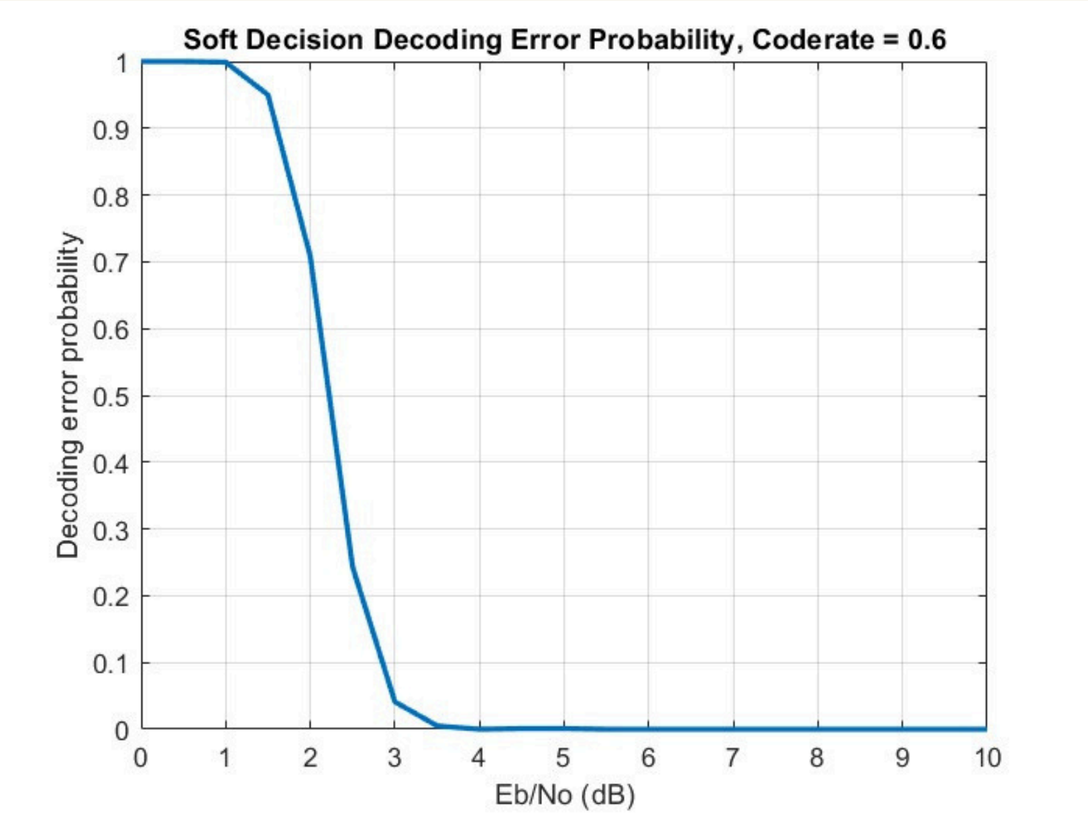
BIT ERROR PROBABILITY vs Eb/No FOR CODE RATE = 1/2



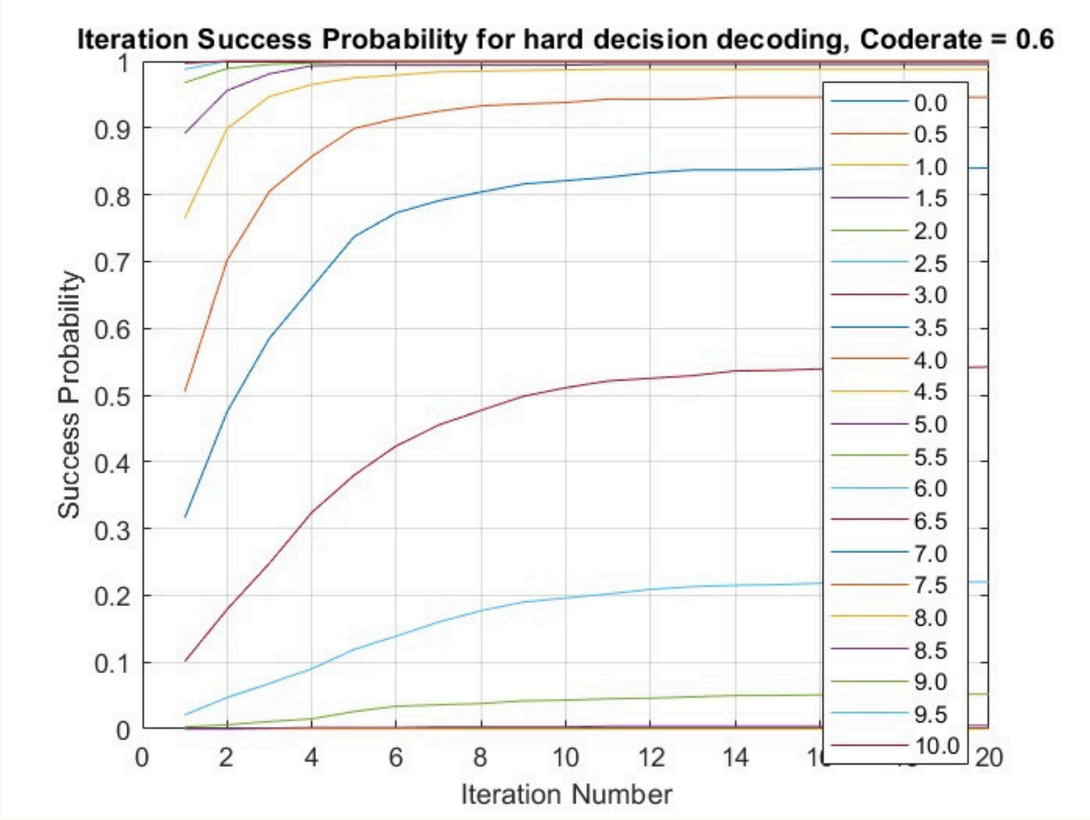


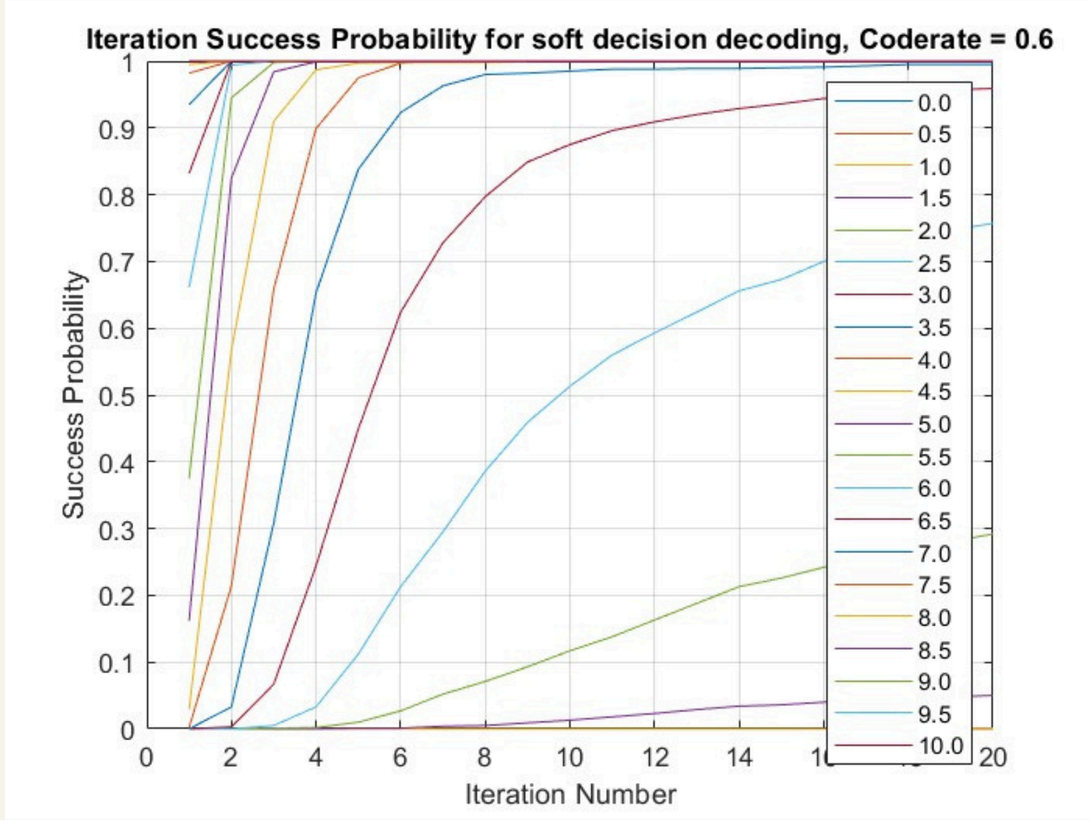
DECODING ERROR PROBABILITY vs Eb/No FOR CODE RATE = 3/5



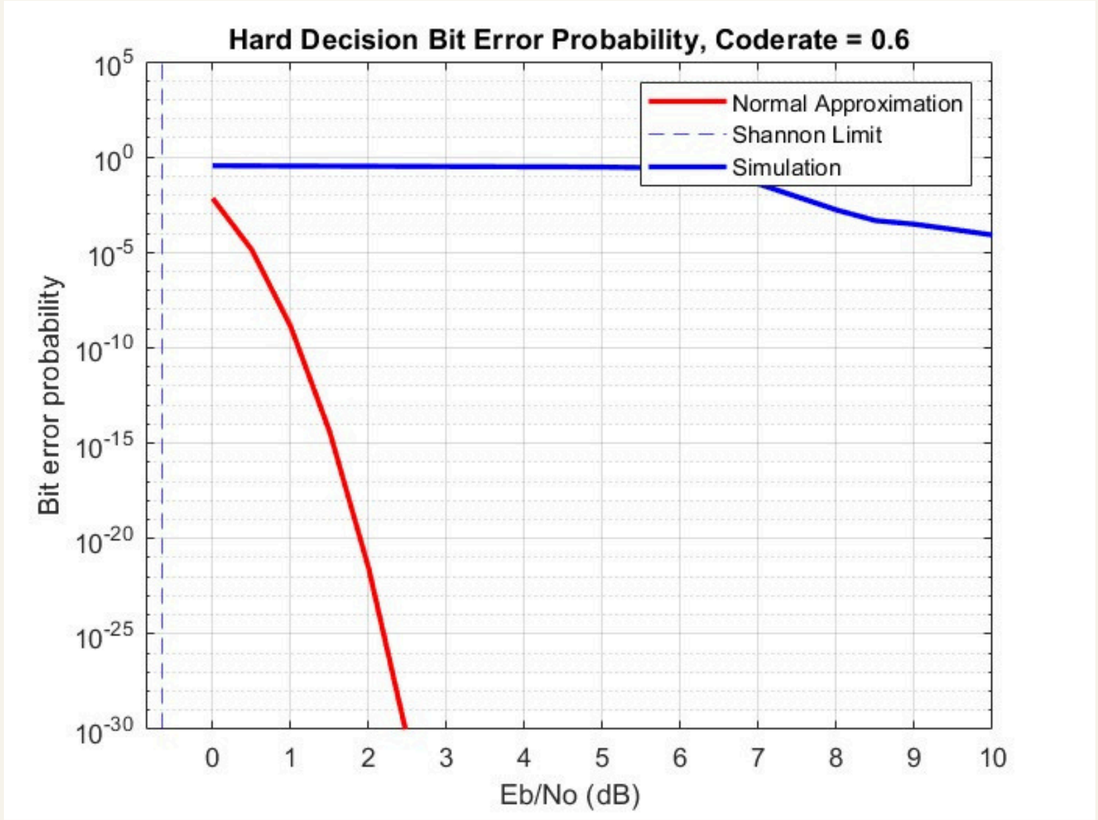


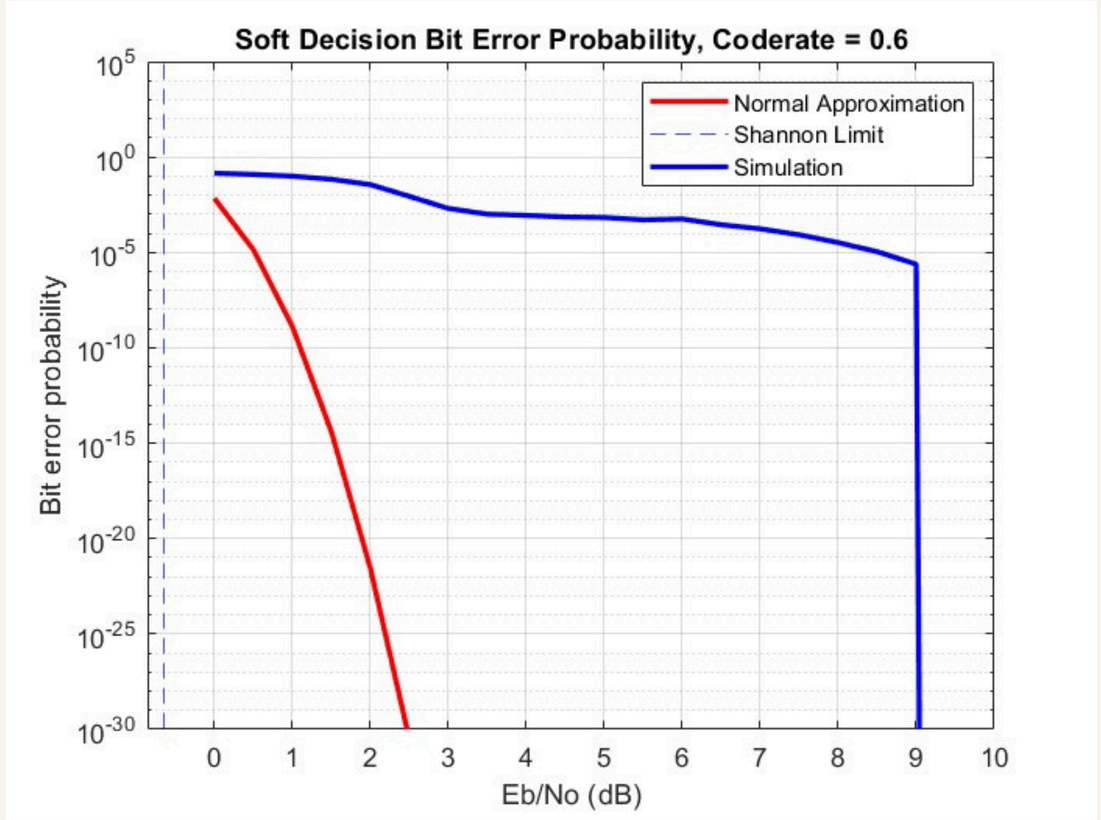
SUCCESS PROBABILITY vs ITERATION FOR CODE RATE = 3/5



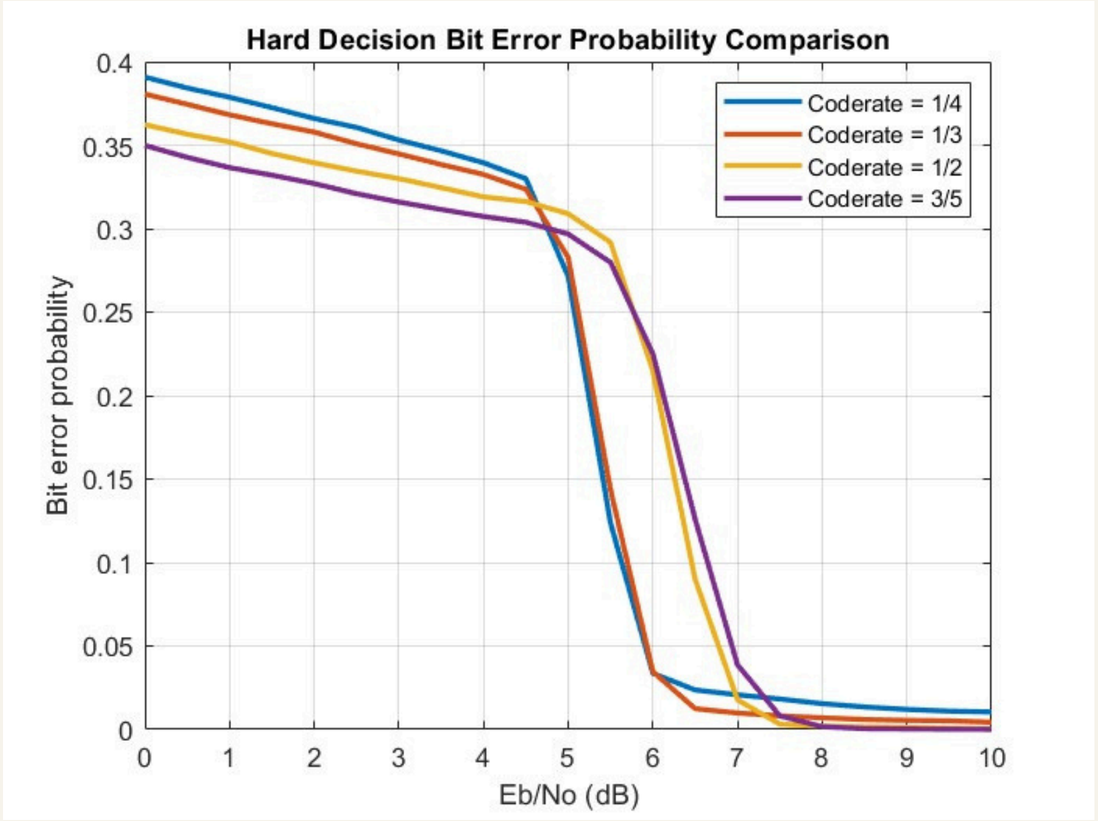


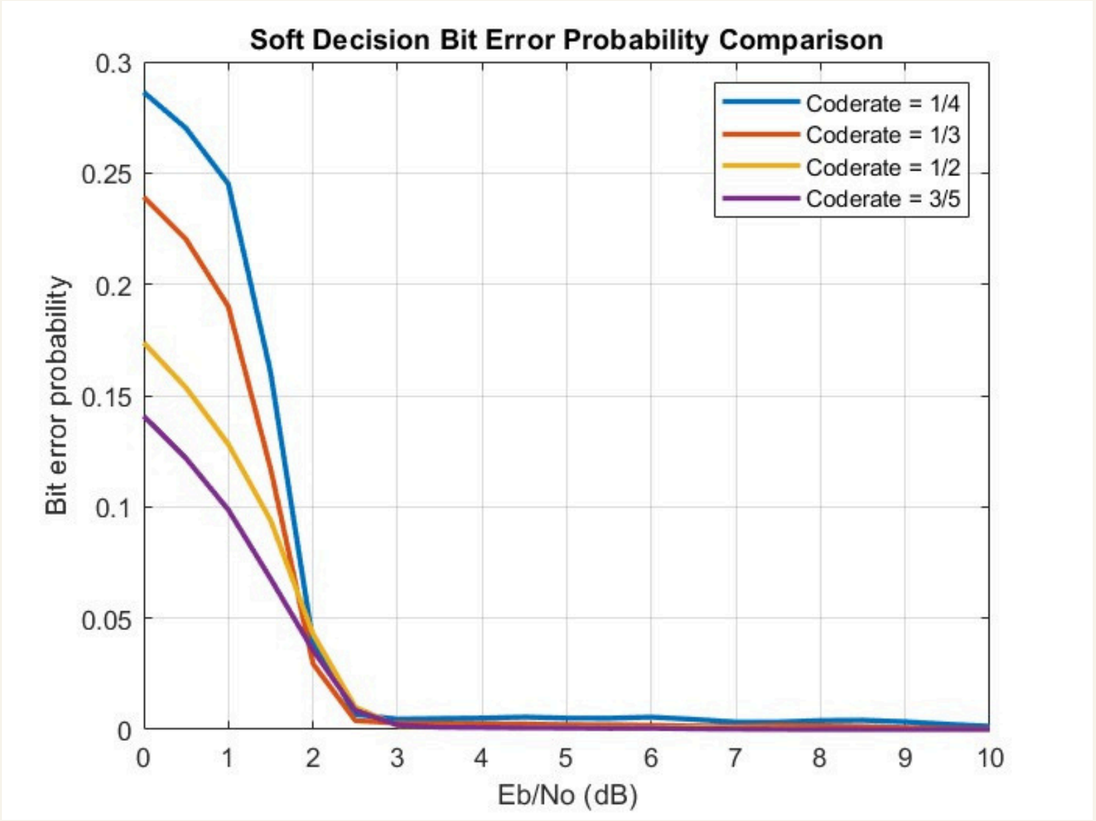
BIT ERROR PROBABILITY vs Eb/No FOR CODE RATE = 3/5



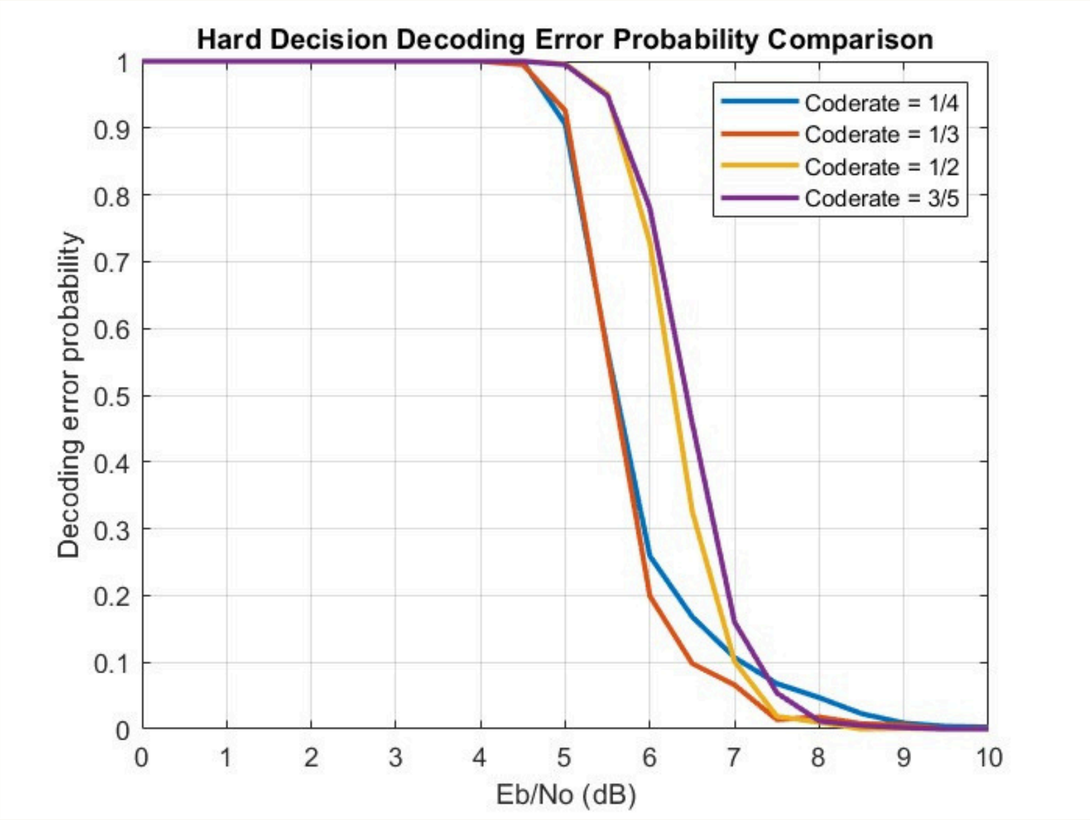


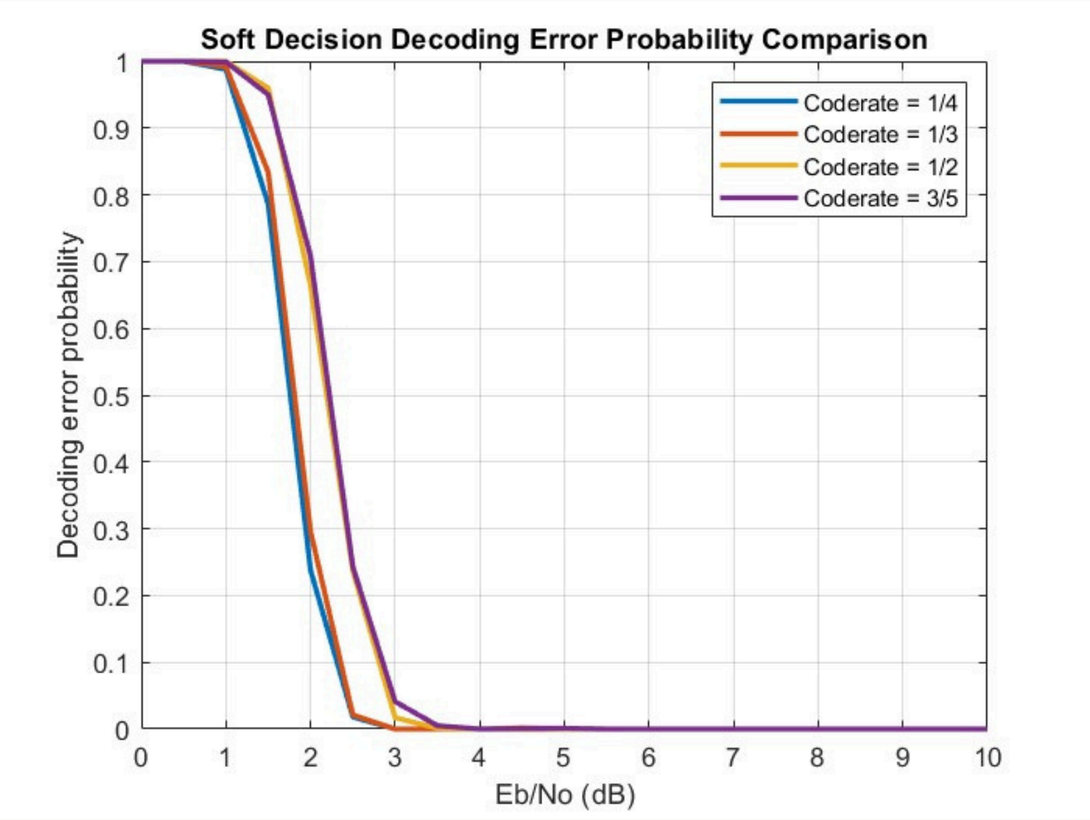
BIT ERROR PROBABILITY COMPARISON





DECODING ERROR PROBABILITY COMPARISON





Bibliography

- Video Lectures of NPTEL-NOM IITM by Prof. Andrew Thangaraj on LDPC codes
- Lecture Slides of Channel Coding by Professor Yash Vasavada
- Implementation of Low-Density Parity-Check codes for 5G NR shared channels by LIFANG WANG
- LDPC Codes – a brief Tutorial --Bernhard M.J. Leiner
- GFG, Quora.

Team Members

- DHRUVIKA RATHOD - 202301168
- BHAVYA THAKKAR - 202301169
- ANSHUMAN BHAGAT - 202301170
- ZEEL PARMAR - 202301171
- MACWAN JENISH - 202301172
- RUDRA CHAUHAN - 202301173
- RATHVA HARDIK KUMAR - 202301174
- JETHVA MANTHAN - 202301175
- BARASARA MEET - 202301176
- OM CHAVDA - 202301177
- PATEL PRINCE - 202301178

THANK YOU