CT-216

LDPC DECODING FOR 5G NR

Group: 24

Professor : Yash Vasavada
Mentor TA : Darpan Lunagariya

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.

Team Members:

Rishik Yalamanchili (202301258)

Chirag Katkoriya (202301259)

Mahek Jikkar (202301260)

Nakul Patel (202301261)

Priyanka garg (202301262)

Yug Savalia (202301263)

Krish Patel (202301264)

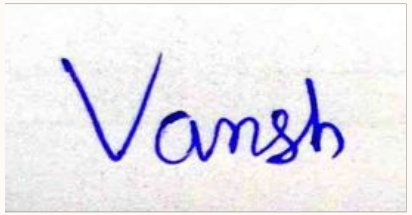
Jalu Rishabh (202301265)

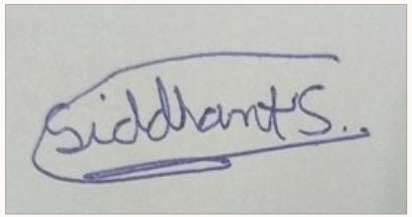
Vansh Vora (202301266)

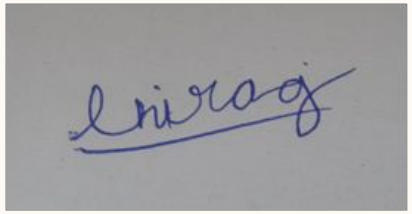
Arav Vaitha (202301267)

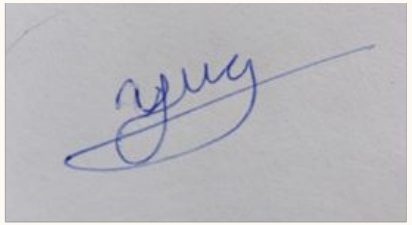
Siddhant Shekhar (202301268)

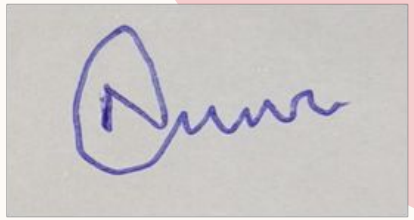
Signatures

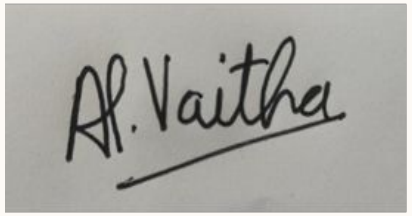


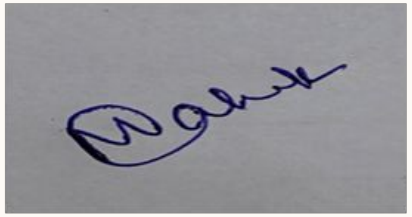


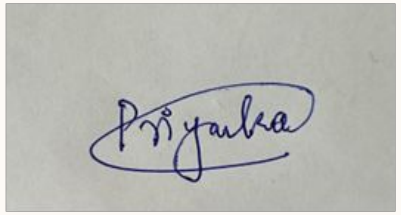


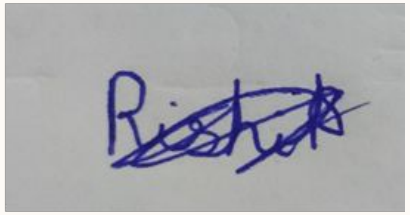












INTRODUCTION

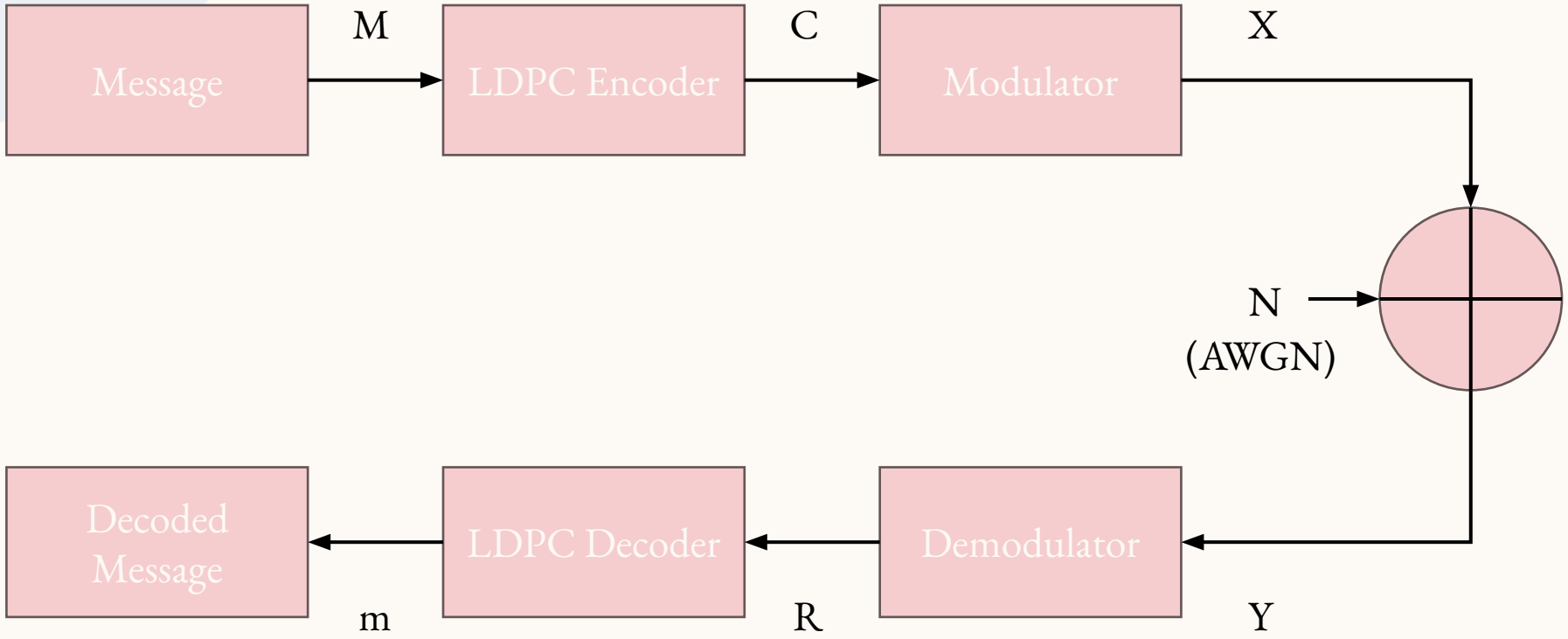
ABOUT LDPC CODES

- LDPC codes can be used in error correction of transmission errors which may occur through various stages of communication, mainly reducing the errors brought about by noise by the channel.
- The LDPC codes can provide a channel capacity close to the theoretical Shannon's Channel Capacity Limit.
- These codes were proposed by R.G. Gallager in 1962, but were not applied since the decoding techniques were too complex for practical implementation.
- Since the rediscovery of the codes by David MacKay, they were introduced in a wide range of communication devices, both wired and wireless.
- Currently, the most significant use of LDPC codes is in the new generation of wireless communications standards called 5G New Radio (5G-NR).

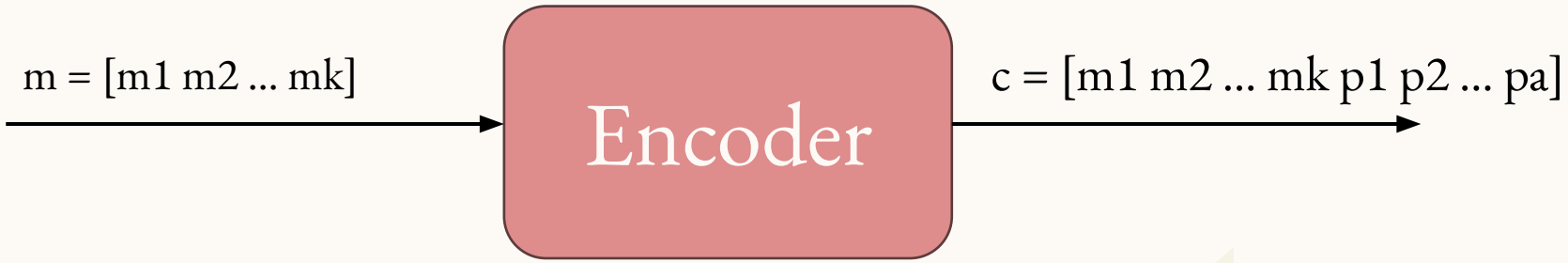
INTRODUCTION

- Low Density Parity Check (LDPC) codes are a class of linear block codes with a sparse parity check matrix H i.e. it contains small number of ones per row or column.
- LDPC code has a block length n with a mxn parity check matrix where each column contains a small fixed number of ones (wc = parity constraints) and each row contains (parity bits) associated with each Wc
- Wc >= 3 and each row contains Wr >= Wc number of ones.
- Low Density implies that Wc << m and Wr << n.
- Number of ones in the parity check matrix H= Wc*n = Wr*m.

FLOW DIAGRAM



ENCODING


m = [m1 m2 ... mk]
Encoder
c = [m1 m2 ... mk p1 p2 ... pa]

Encoding

Protograph : LDPC Code's parity-check matrix **H** is constructed from **BG** (base graph matrix) by using lifting size (expansion factor (Z)).

➔ The following steps are performed to construct the code.

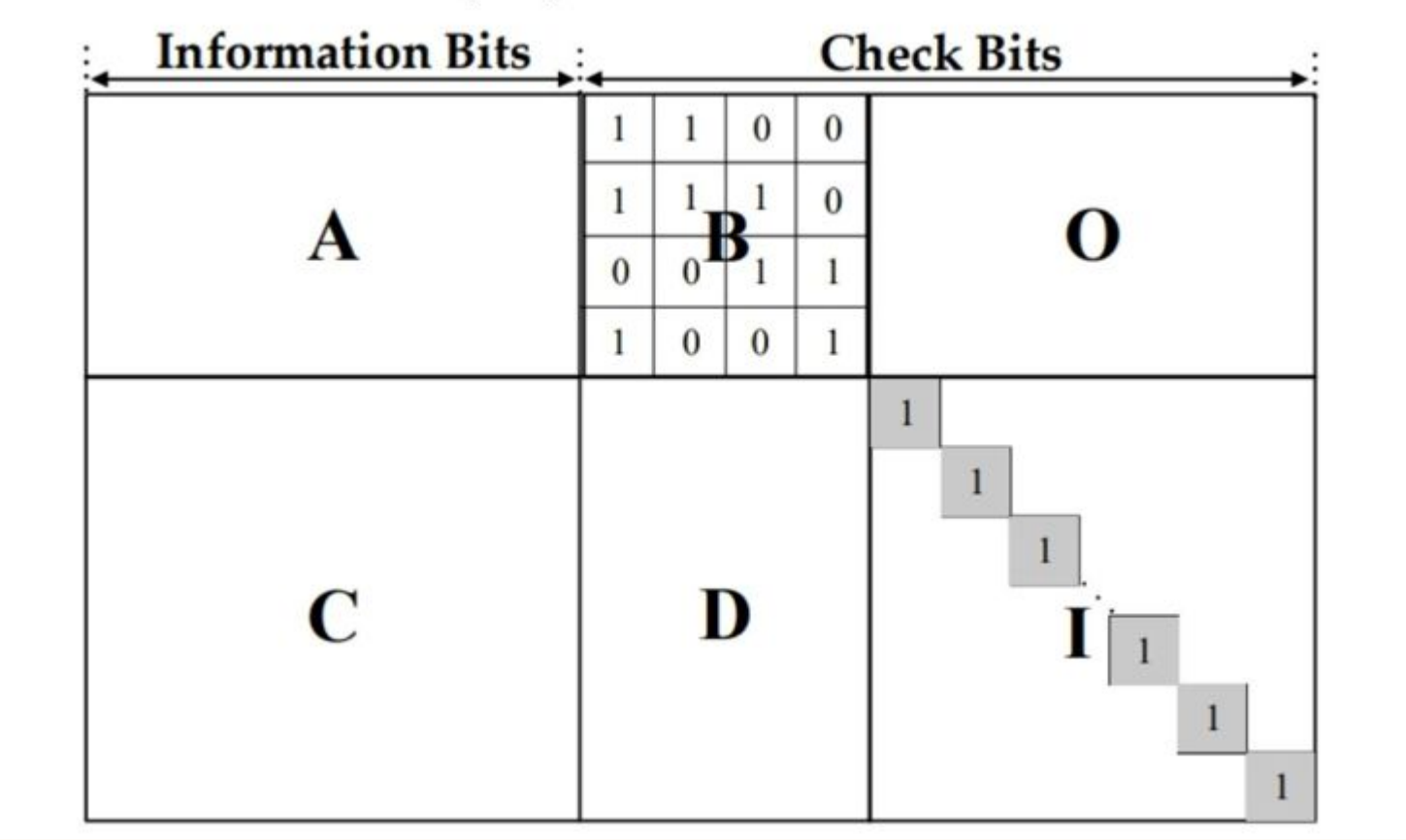
- Base Graph Matrix (BG)
- Expansion factor (Z)
- Rate Matching

Base Graph Matrix

LDPC has two base graph matrices

- BG1 : 46 x 68
- BG2 : 42 x 52

Base Matrix is a proper Block Like Structure



Characteristics of Block

- **A** : High dense part
- **B** : Double diagonal structure
- **O** : Zero matrix
- **C** : Low dense part
- **D** : Low dense part
- **I** : Identity matrix

For BG1,

A:4x22 , B:4x4 , O:4x42,

C:42x22 , D:42x4 ,I: 42x42

For BG2,

A:4x10 , B:4x4 , O:4x38,

C:38x10 , D:38x4 , I: 38 x 38

*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

Base Matrix to H Matrix

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

Expansion factor: 5

$H =$

0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	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	1	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	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	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	0	0	0	0	0	1	0	0	0	0
0	0	1	0	0	1	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	0	1	0	0	1	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	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
1	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	0	1	0
0	1	0	0	0	0	0	0	0	1	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	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	0	1	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	0	1	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	0	1	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	0	1	0	0	0	0	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	0	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	0	0	0
0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	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	0	1	0	0	0	0	0	1	0	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	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

Expansion

The non-binary BG is converted to binary H matrix through **expansion**.

- Expansion Factor (Zc) :
  - iLs (index) : 0,1,2,3,4,5,6,7
  - a: 2,3,5,7,9,11,13,15
  - Ja : 7,7,6,5,5,5,4,4
  - **Zc = a x 2^j** , j=0,1,2,...Ja (max value of Zc is 384)
- For each base matrix entry specified -1, 0, 1, ..., Zc -1.
  - -1 : Zc x Zc Null Matrix
  - I in [0, Zc -1] : Identity shifted right i times.
- BG1 supports 22z message bits(max.).
- BG2 supports 10z message bits(max.).

Encoding Algorithm

- First we expand the Base matrix

For all entries of B[i][j]:

if(B[i][j] == -1)

Replace B[i][j] by all zeros matrix of Zc x Zc

Else

Replace B[i][j] by shifted identity matrix of Zc x Zc by B[i][j]

- a = row(H) = n-k (message bits)
- c = [m1 m2 ... mk p1 p2 ... Pa]
- We want to find p1p2 ... Pa
- We multiply with H matrix $\mathbf{H} \times \begin{bmatrix} m^T \\ p^T \end{bmatrix} = \mathbf{0}$
- After that using double diagonal structure we find parity bits

Using this double diagonal structure it is easy to find parity.

I_k: identity matrix column shifted k times to right

Message m = [m1 m2 m3 m4]

where every m_i has 5 bits

Codeword c = [m1 m2 m3 m4 p1 p2 p3 p4]

where each pi and mi has 5 bits

p1 p2 p3 and p4 are unknown

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

H matrix with Z_c: 5

*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

$H\,[m1\ m2\ m3\ m4\ p1\ p2\ p3\ p4]^T = 0$

1: $I\_1m1 + I\_3m3 + I\_1m4 + I\_2p1 + I\_p2 = 0$

2: $I\_2m1 + I\,m2 + I\_3m4 + I\,p2 + I\,p3 = 0$

3: $I\_4m2 + I\_2m3 + I\,m4 + I\_1p1 + I\,p3 + I\,p4 = 0$

4: $I\_4m1 + I\_1m2 + I\,m3 + I\_2p1 + I\,p4 = 0$

Adding all 4,we get :

$I\_1p1 = I\_1\,m1 + I\_3m3 + I\_1\,m4 + I\_2m1 + I\,m2 + I\_3\,m3 +$

$I\_4m2 + I\_2m3 + I\,m4 + I\_4m1 + I\_1m2 + I\,m3$

Find p1 from above

p2: use p1 in 1, p3: use p2 in 2, p4: use p3 in 3

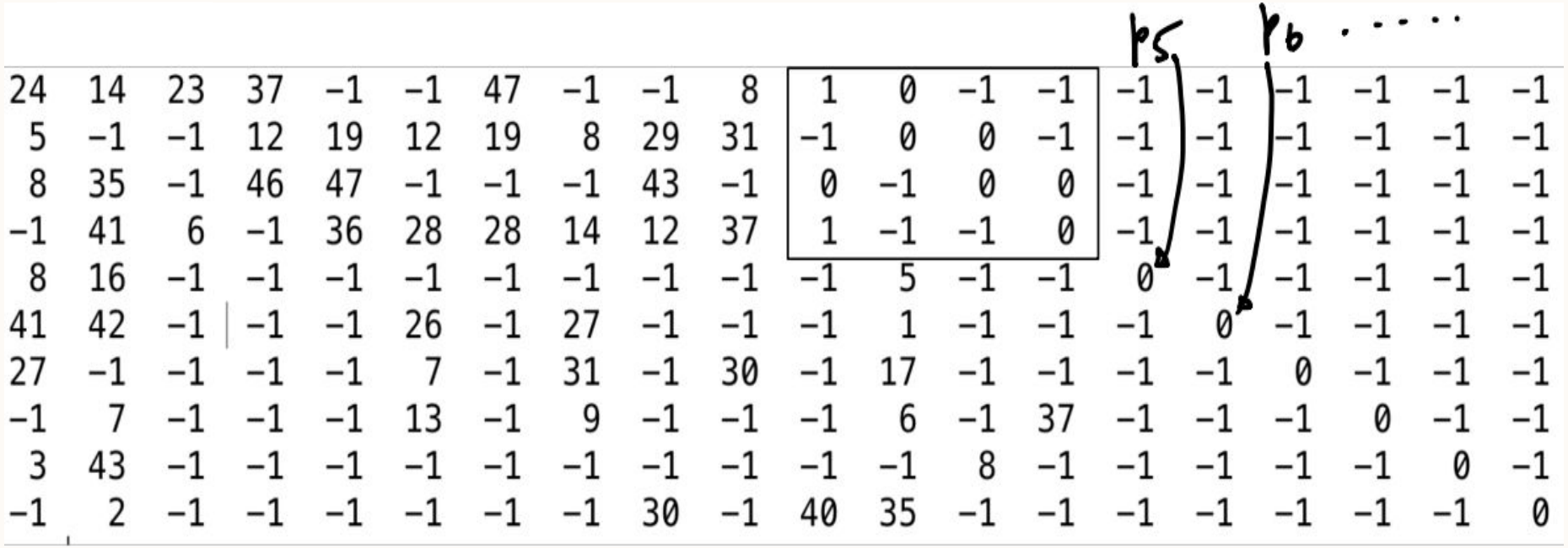
Expansion: 5

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

*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

Below given is a BG2 base matrix only 10 rows and 20 columns are shown

- Message: [m1 m2 ... m10], each 48 bits
- Parity: [p1 p2 p3 p4 p5 p6 ...]
- First four rows: use double-diagonal encoding to find p1, p2, p3, p4 solved in above slide
- Row 5: p5, Row 6: p6, and so on are found using the diagonals mentioned in the picture below

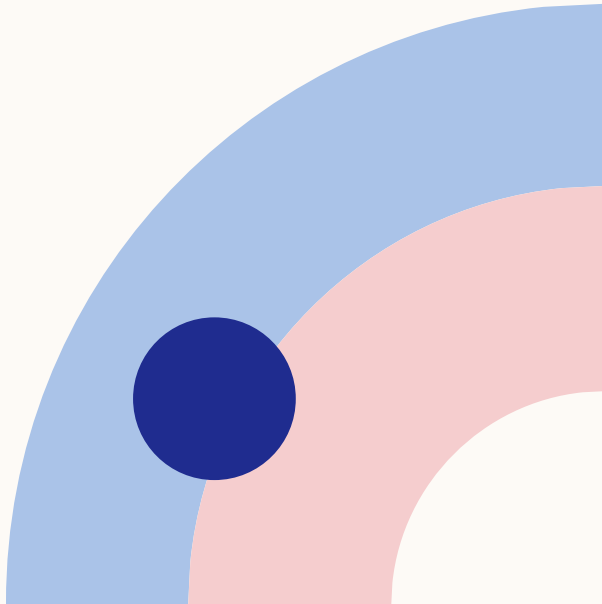


24	14	23	37	-1	-1	47	-1	-1	8	1	0	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	12	19	12	19	8	29	31	-1	0	0	-1	-1	-1	-1	-1	-1	-1
8	35	-1	46	47	-1	-1	-1	43	-1	0	-1	0	0	-1	-1	-1	-1	-1	-1
-1	41	6	-1	36	28	28	14	12	37	1	-1	-1	0	-1	-1	-1	-1	-1	-1
8	16	-1	-1	-1	-1	-1	-1	-1	-1	-1	5	-1	-1	0	-1	-1	-1	-1	-1
41	42	-1	-1	-1	26	-1	27	-1	-1	-1	1	-1	-1	-1	0	-1	-1	-1	-1
27	-1	-1	-1	-1	7	-1	31	-1	30	-1	17	-1	-1	-1	-1	0	-1	-1	-1
-1	7	-1	-1	-1	13	-1	9	-1	-1	-1	6	-1	37	-1	-1	-1	0	-1	-1
3	43	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	8	-1	-1	-1	-1	-1	0	-1
-1	2	-1	-1	-1	-1	-1	-1	30	-1	40	35	-1	-1	-1	-1	-1	-1	-1	0

Rate Matching

What is Rate Matching and Why is it needed?

- Rate matching is the process of selecting a specific number of bits from the parity matrix in order to meet the needs of transmission over a channel with fixed resources.
- Generally we match the code using Puncturing of bits.



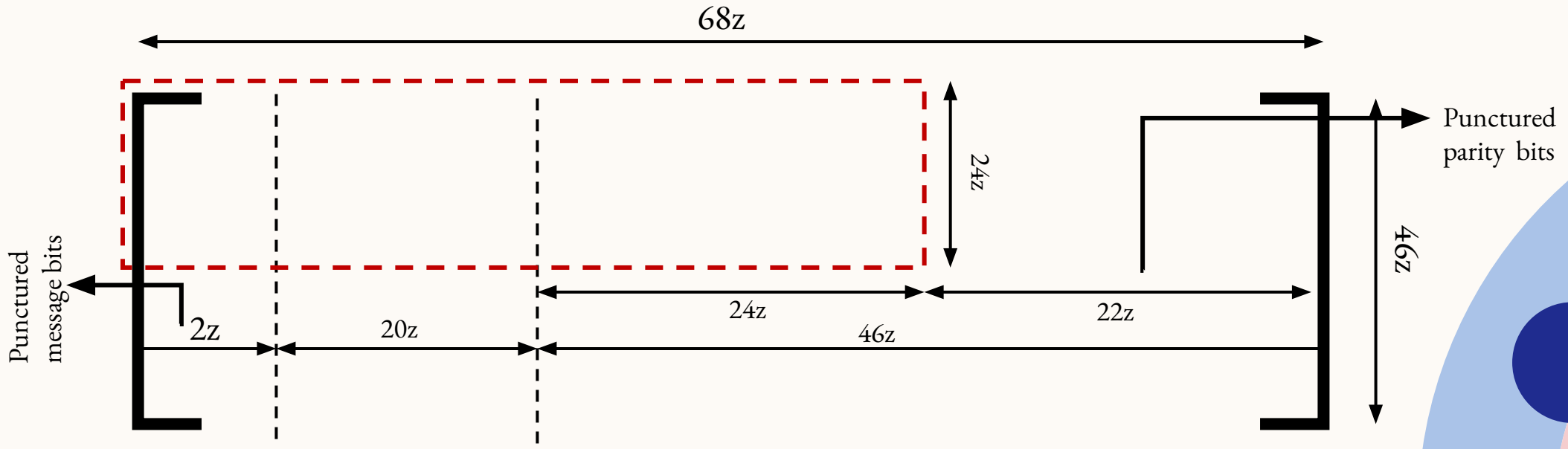
Rate Matching : An illustration for puncturing bits in BG1

- **To achieve a code rate of r=1/2**

We have 22z information bits, and 44z (excluding the first 2z bits) need to be transmitted.

Hence we need to transmit : 20z message bits + 24z Parity Bits .

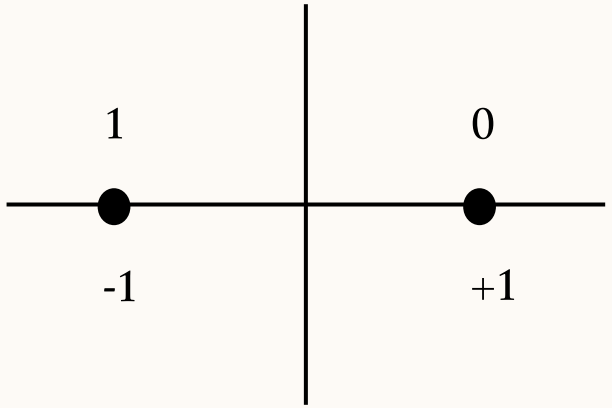
Therefore, the final dimension of the H matrix will be 24x46 (dotted box) ,punctured bits are set as 0 while receiving.



MODULATION

BPSK MODULATOR

- In **BPSK (Binary Phase Shift Keying)** , encoded bits are transformed into **BPSK** symbols before transmission. This can be achieved by representing the bits (ones and zeros) by shifting the phase of the carrier wave (codeword bits).
- The encoded codeword bit is mapped to a **BPSK** symbol using the mapping, **s =1-2c**, where s represents the **BPSK** symbol and c represents the codeword bit.
- Specifically, a bit is represented by shifting the phase of the carrier wave by 180° (pi radians) for a binary 1, and keeping it unchanged for a binary 0.

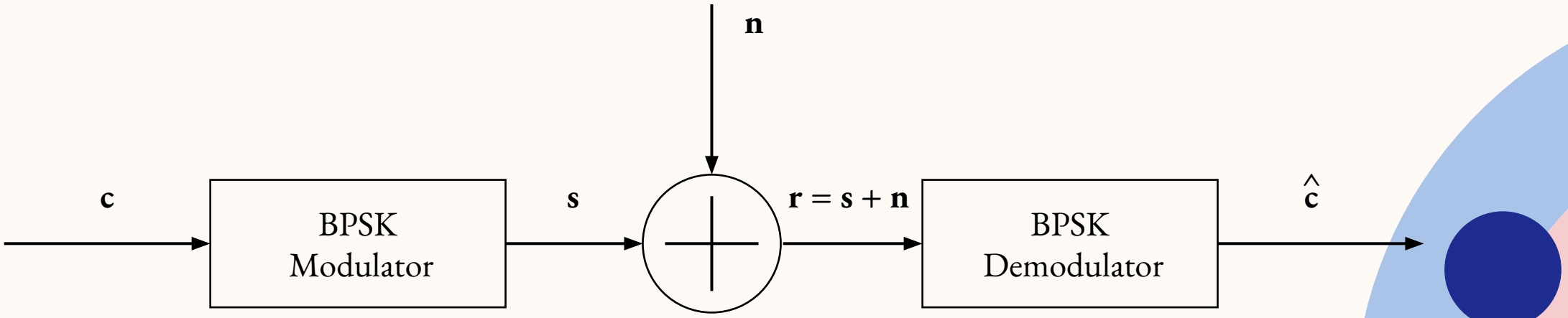


AWGN (ADDITIVE WHITE GAUSSIAN NOISE) CHANNEL

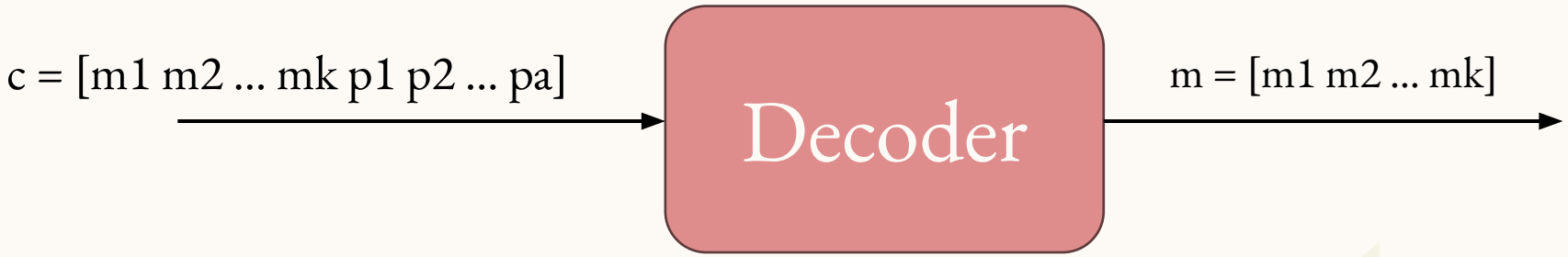
- The **BPSK** symbols are transmitted through an **AWGN** channel, which adds noise to the signal. The noise power is determined by the **SNR** (Signal-to-Noise Ratio) parameter, represented by y = Es/N0.
- Since **BPSK** symbols have energy, Es=1 Joule, the noise power is 1/y . The noise, n is generated as a Gaussian random variable with zero mean and variance is calculated per **SNR**.
- Thus, the received bits are represented by, **r** =**s** + **n**.

BPSK Demodulator

- To recover the original encoded bits, the **BPSK** demodulator makes decision by mapping received values to bits (ones and zeros) by drawing threshold at zero.
- This works as: If the received value is positive, then it will be mapped to 0, and negative received value will be mapped to 1.



DECODING

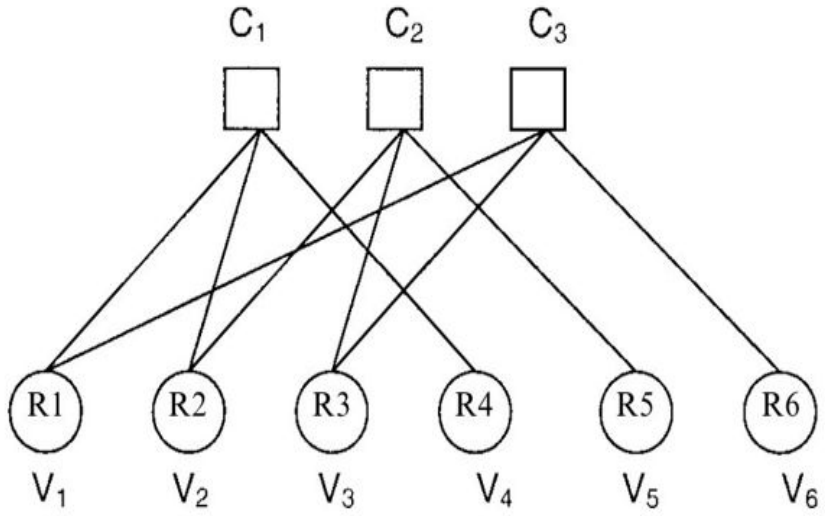

c = [m1 m2 ... mk p1 p2 ... pa]
Decoder
m = [m1 m2 ... mk]

HARD DECODING

Decoding Using Tanner Graph

This parity check matrix H can be represented by a Tanner Graph, which is a Bipartite Graph:

$$H = \begin{array}{c} \begin{matrix} v1 & v2 & v3 & v4 & v5 & v6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} c1 \\ c2 \\ c3 \end{matrix}$$



*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

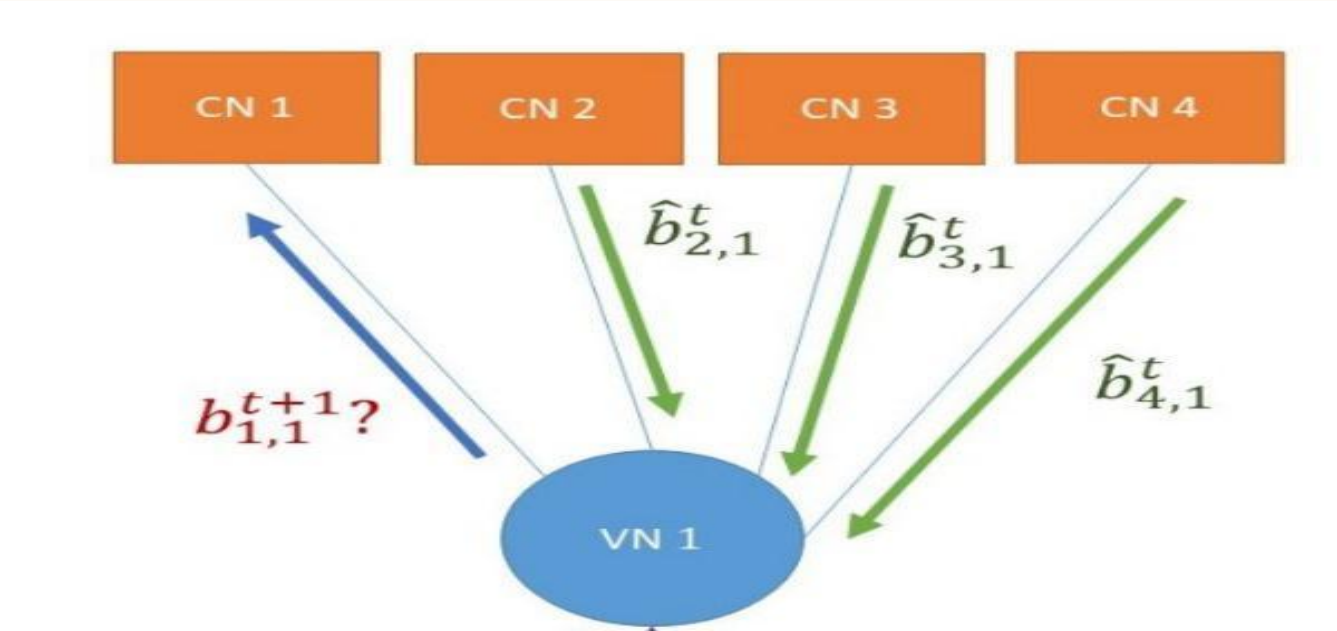
Hard Decision Decoding

Message Passing Iterative Decoding Algorithm

- The received bits are used to compute initial messages from check nodes (CNs) to variable nodes (VNs).
- Iteratively, messages are exchanged between CNs and VNs
- Each CN computes its outgoing messages based on incoming messages from connected VNs
- Similarly, each VN computes its outgoing messages based on incoming messages from connected CN.

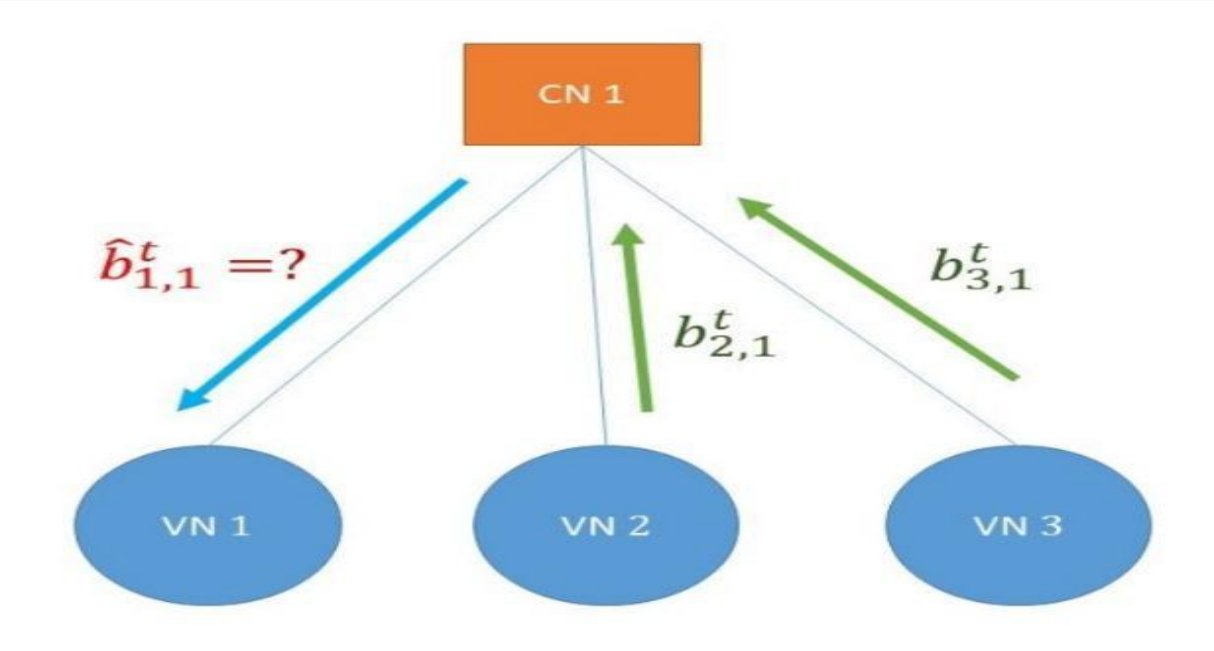
VN To CN

- First we load all the received values into the VN.
- Then, for first iteration:
  - We directly send the received value to all connected CNs of respective VNs.
- For all other iterations:
  - To send message to CN i, we take all the messages received to VN j and the bit it had originally, except value sent by CN i. Then we perform majority voting of these messages and send it to CN i.



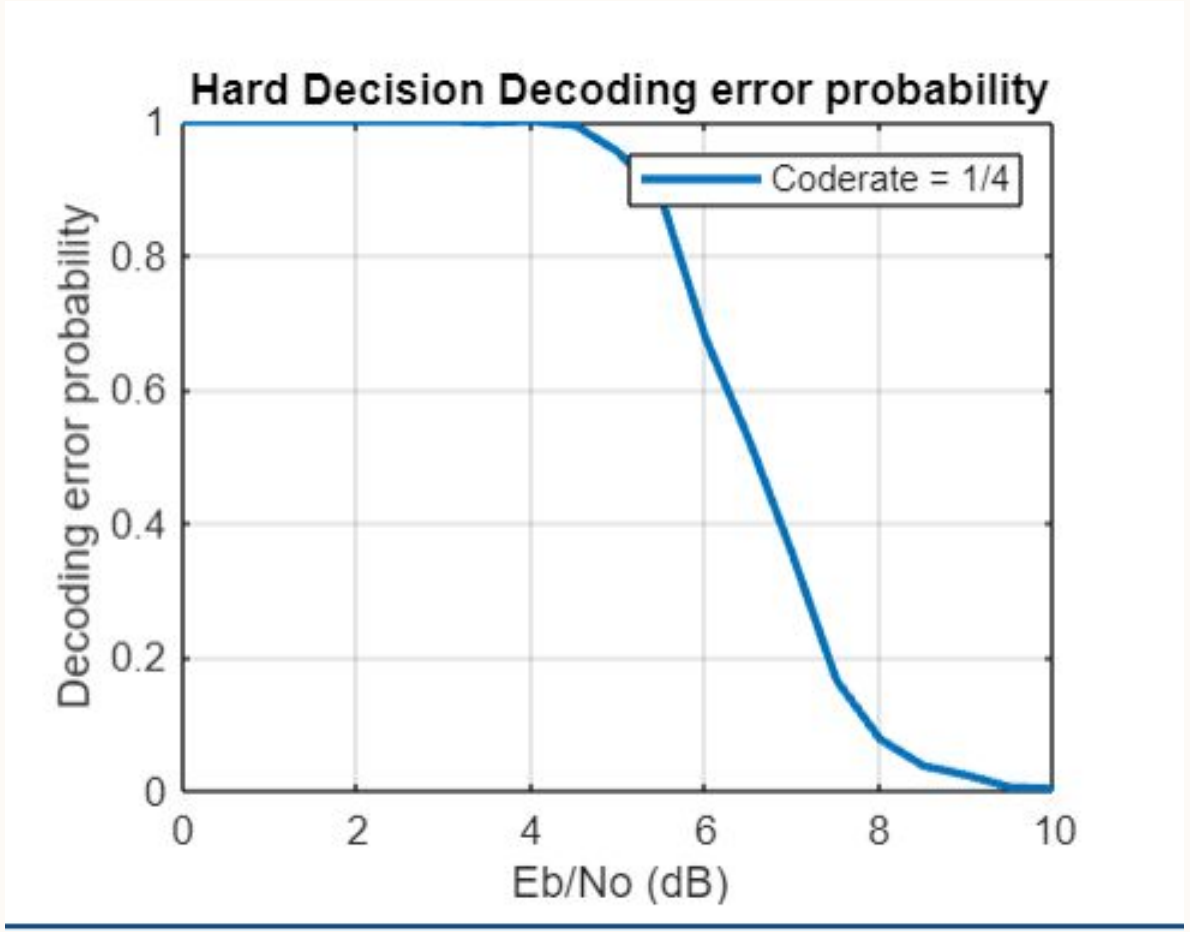
CN To VN

- Since each CN is a SPC code, the algorithm to send the message from CN to VN is simple performing an XOR operation .
- We take all the values received from the connected VNs except that sent by VN j. Then we take the xor of these values and send the message to VN j.
- To make the code efficient, we initially took XOR of all the messages received from each VN. The, to send the message to VN j, we again took the xor of the total with the message sent by VN j, and sent back that value.

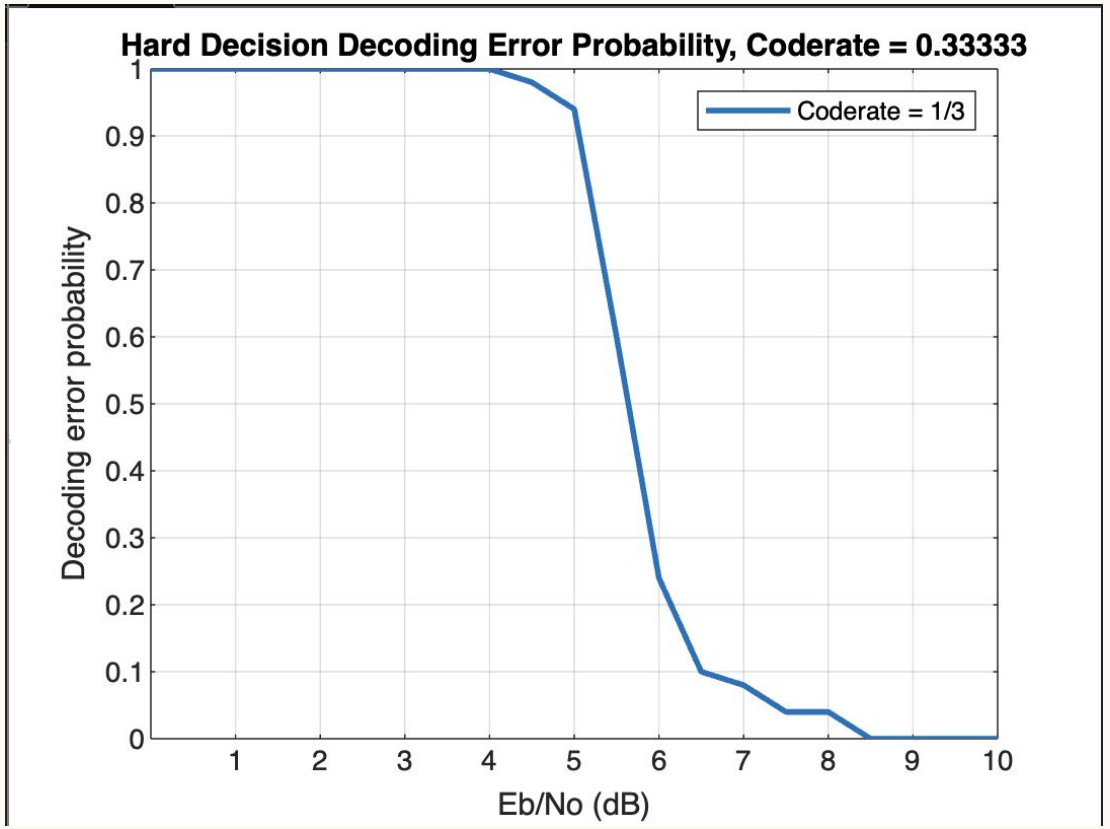


Results For Base Graph NR_2_6_52

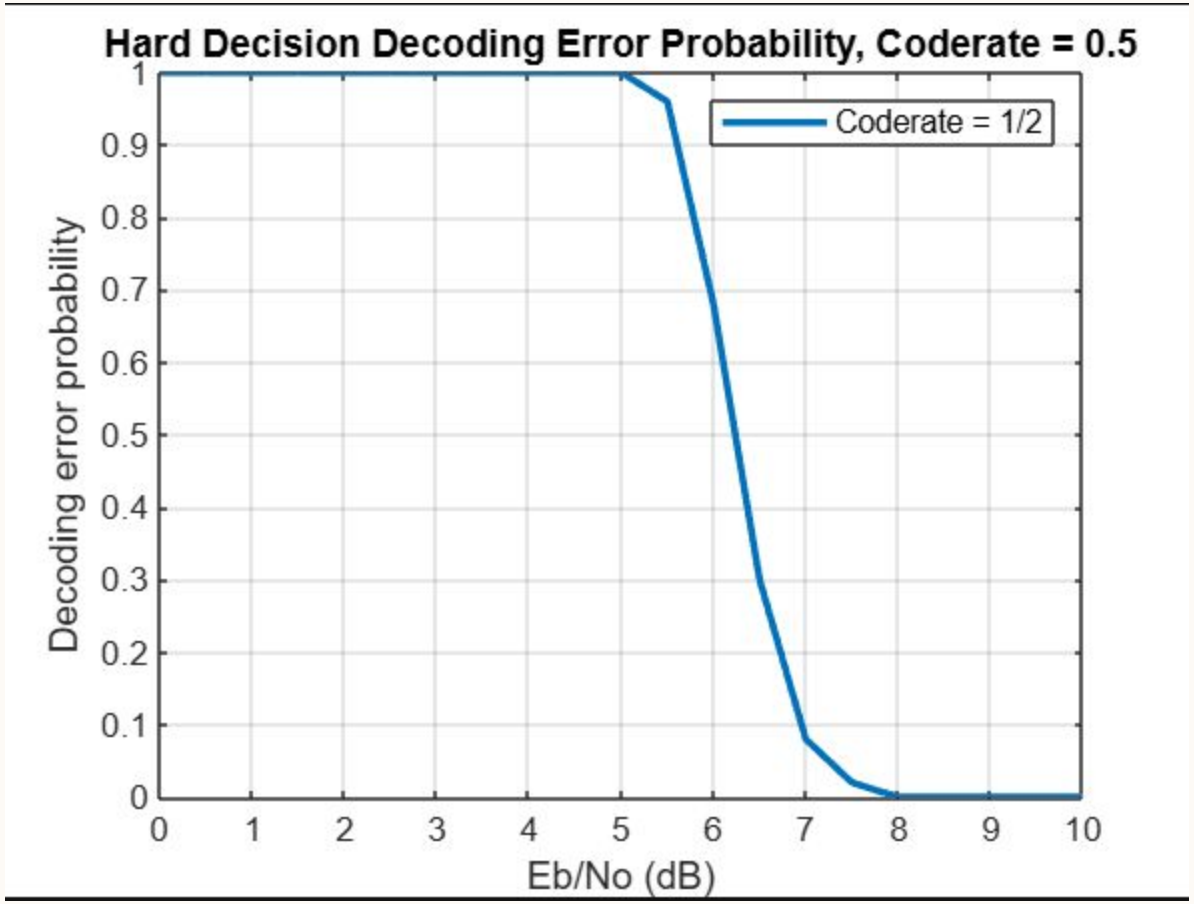
Coderate 1/4 (Hard Decoding)



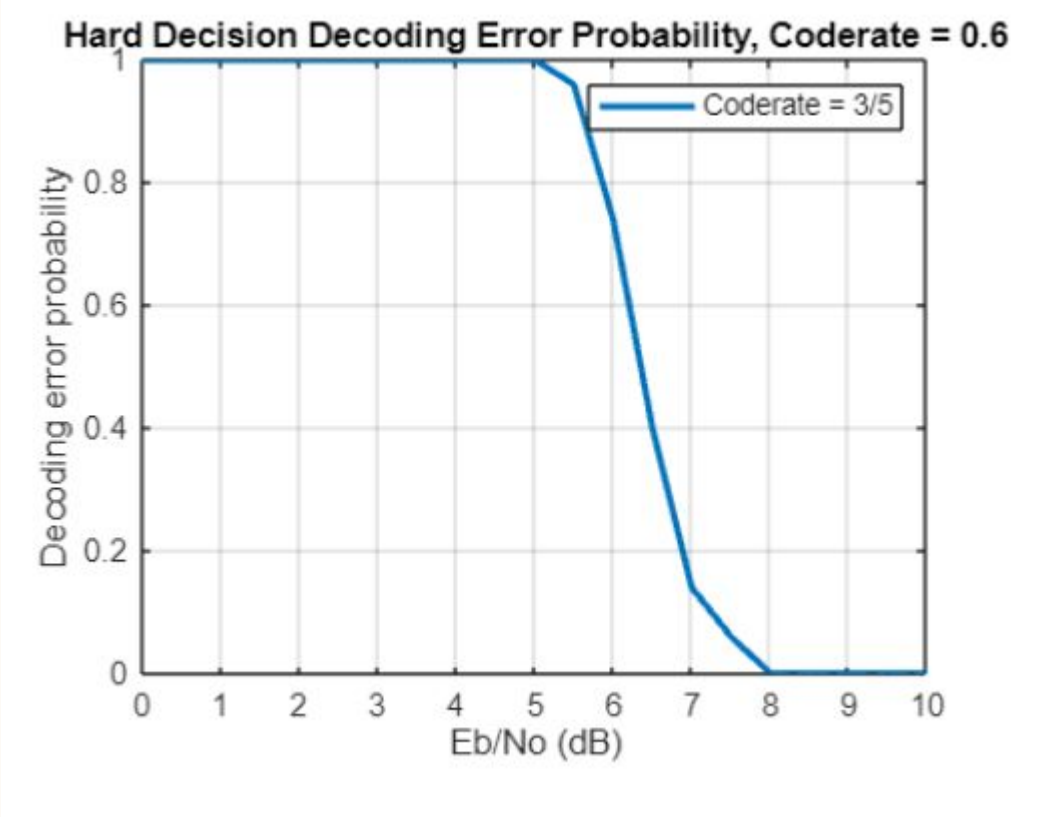
Coderate 1/3 (Hard Decoding)



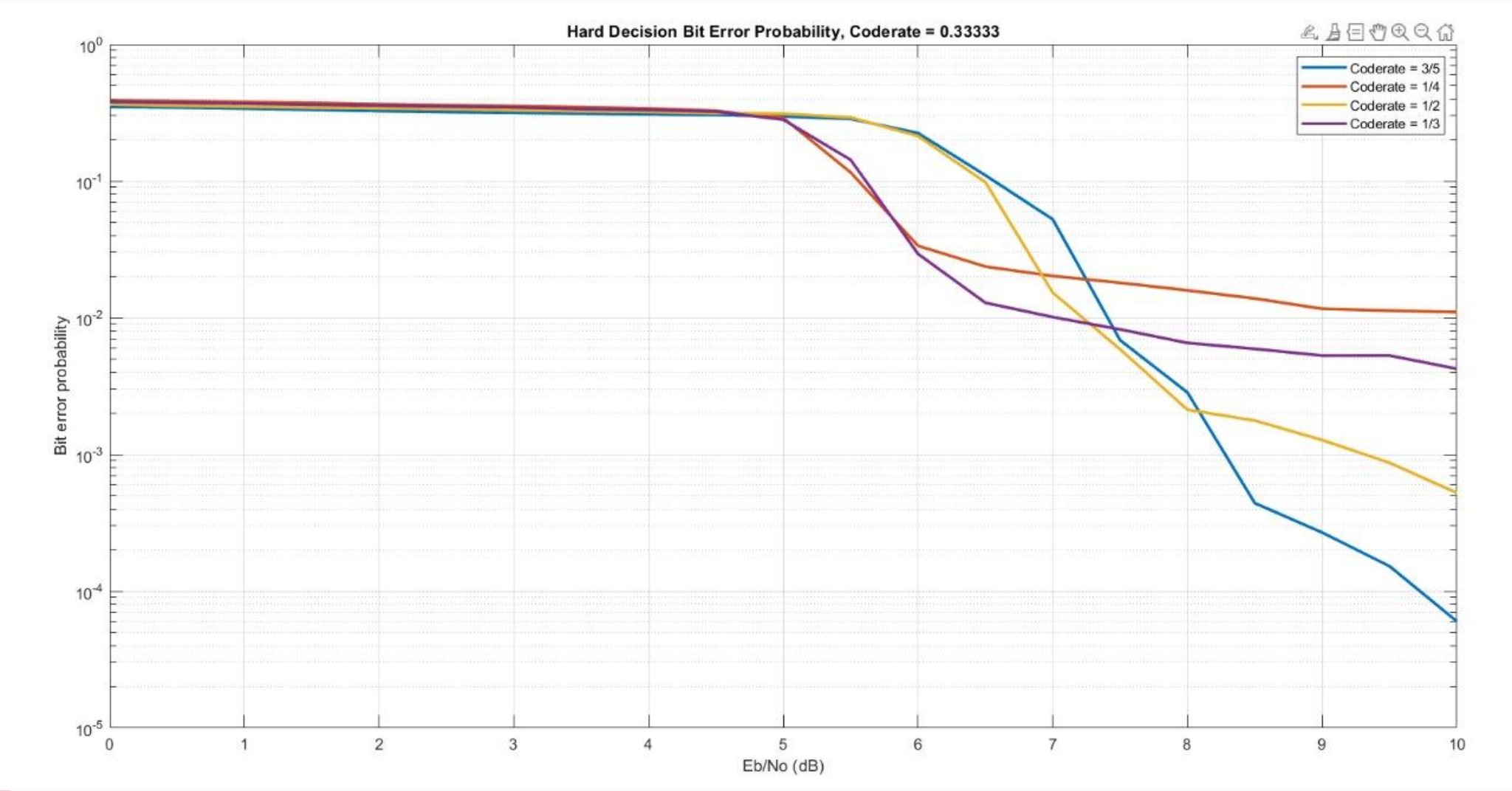
Coderate 1/2 (Hard Decoding)



Coderate 3/5 (Hard Decoding)



BER VS Eb/No


Hard Decision Bit Error Probability, Coderate = 0.33333
Coderate = 3/5
Coderate = 1/4
Coderate = 1/2
Coderate = 1/3
Bit error probability
Eb/No (dB)
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
0
1
2
3
4
5
6
7
8
9
10

The Problem with Hard Decision Decoding

- Let's explain the difference with the help of an example:-
  Suppose we want to send a message m:-[1 0 1 0 1].
- According to the modulation we have implemented during our encoding :-0 maps to 1 ($1-2m_i, m_i=0$ then $1-2(0)=1$)
  1 maps to -1 ($1-2m_i, m_i=1$ then $1-2(1)=-1$)
- Let us suppose that the the bits received are
  r=s+n=[0.2 0.5 -1.3 -0.1 -0.3]
- After the implementation of our Hard Decoding Code we would decode our received bits as [0 0 1 1 1].Now as visible there are errors at 2 places.
- To minimise the chances of bit errors we shall also implement Soft Decision Decoding

SOFT DECODING

Why Soft Decision Decoding?

- Hard Decision decoding relies on the absolute threshold of received bits.In contrast to that Soft Decision Decoding relies on the actual received values to make the the decoding more accurate and effective i.e. the log likelihood ratios(LLRs) which reflect belief we have that our bit is 0 or 1.
- For example, a received value of -1.3 is a stronger indication of 1 than -0.1. Soft Decision relies on the Euclidean distance rather than the minimum Hamming Distance which is used in Hard Decision Decoding.

Soft Decision Decoding on Tanner Graph

Message Passing Iterative Decoding Algorithm

- The received bits are used to compute initial messages from check nodes (CNs) to variable nodes (VNs).
- Iteratively, messages are exchanged between CNs and VNs
- Each CN computes its outgoing messages based on incoming messages from connected VNs
- Similarly, each VN computes its outgoing messages based on incoming messages from connected CN.

L(Storage Matrix)

L is the sparse matrix which has the same dimension as that of the Parity Check Matrix(H).It stores the LLR based messages exchanged between VNs and CNs. Each updated LLR in the L matrix is stored at the positions corresponding to that in H.

Initialisation

- Each VN is initialised with channel Log-Likelihood Ratio(LLRs) from the channel.
- LLR=log(P(bit=0|r)/P(bit=1|r))
- For BPSK modulation with noise variance σ^2 the channel LLR will be $2r/\sigma^2$
- These LLRs now will be the initial messages sent from VNs to CNs

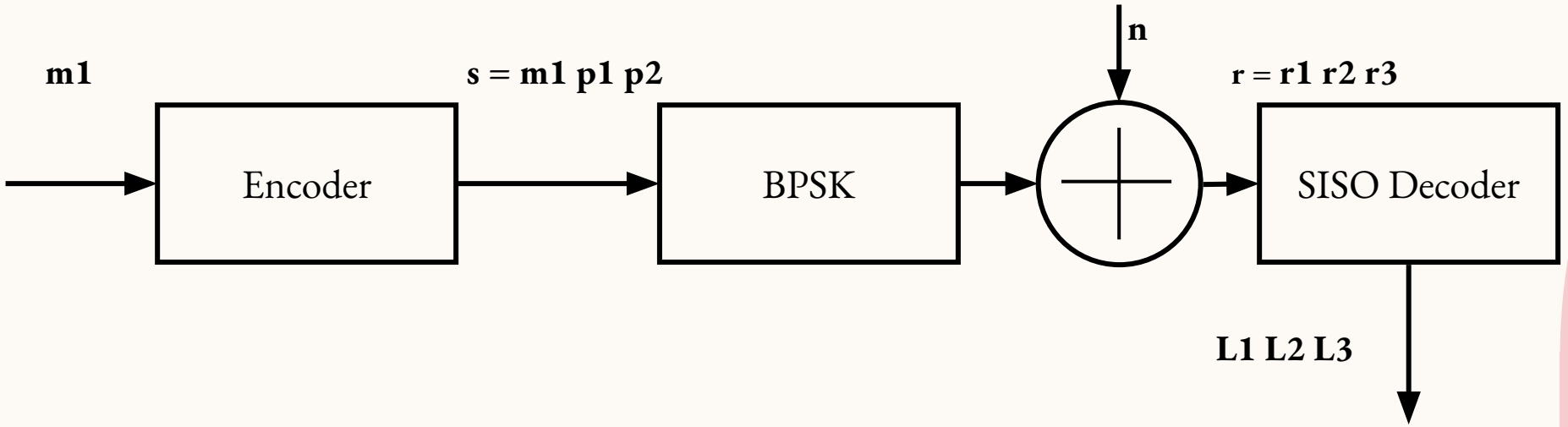
VN to CN

- For the first iteration we send all the LLR values of VN to all its connected CNs
- For all other iterations we take the sum of channel LLR and the sum of messages received by all CNs except the one received by the CN we are passing to.
- Once the min sum approximation is done by CNs and belief each entry of L is updated we calculate the sum of all the entries of a column of L and also add the value of receive vector element to it.
- The updated belief is that sum-(L_i).
- $L_{VN\_to\_CN} = L_{channel} + \Sigma L_{CN\_to\_VN}$ (for all CNs except the CN to which the message is being passed).

CN to VN

- The CNs do the SPC SISO decoding of all l_is received from the VNs
- The message passed from CN_i to VN_j is computed using tanh rule(which is then approximated to minsum rule).(described in slides ahead)
- For each row we calculate product of signs of entries in the row i.e.$\prod \text{sgn}(L_i)$.
- Each entry is then updated to $\prod \text{sgn}(L_i).\min(|L_{in}|)$ ($\text{sgn}(L_i)$ and is then sent from CNs to VNs
- The belief is updated after every iteration.After several iterations $L_{final}=L_{channel}+\sum L_{CN\_to\_VN}$(for all VNs).If $L_{final}>0$ then bit=0 else bit=1

SISO Decoder For (3,1)Repetition Code



Encoder:
Takes bits m1 and creates a codeword like s = m1 p1 p2.
BPSK Modulation:
Transmits bits using BPSK: 0 → +1, 1 → -1
Additive Noise:
Gaussian noise n is added, resulting in received values r = r1 r2 r3.
SISO Decoder:
Soft-In Soft-Out decoder estimates the likelihood of each bit.

Li = belief that bit si = 0 (log-likelihood ratio or LLR)
Where s1 = s2 = s3 = c1, then:
L1 = r1 + r2 + r3
where:
r1: intrinsic (direct observation)
r2 + r3: extrinsic (from repetition info)

SISO Decoder For (3,1)Repetition Code

Calculations for L1:

Split L1 into intrinsic and extrinsic parts:

$L_1 = l_1 + l_{ext,1}$

$l_i = (2 / \sigma^2) * r_i$

$l_{ext,1} = \log [P(c_1 = 0 \mid r_2 r_3) / P(c_1 = 1 \mid r_2 r_3)]$

$= \log (p_{e1} / (1 - p_{e1}))$

where:

$p_{e1} = \beta * p_2 * p_3$

$(1 - p_{e1}) = \beta * (1 - p_2) * (1 - p_3)$

$l_{ext,1} = \log (p_2 / (1 - p_2)) + \log (p_3 / (1 - p_3))$

$l_{ext,1} = l_2 + l_3$

*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

Example:

Received values:

$r = [0.9, -0.2, 1.4]$

Assume: $\sigma^2 = 1$

Using the above formula:

$L_1 = l_1 + l_{ext,1}$

Intrinsic part:

$l_i = (2 / \sigma^2) * r_i = 2 \cdot r_i$

$\Rightarrow l_1 = 2*0.9 = 1.8$

Extrinsic part (from r_2 and r_3):

$l_{ext,1} = l_2 + l_3$

$l_2 = 2*(-0.2) = -0.4,\ l_3 = 2*1.4 = 2.8$

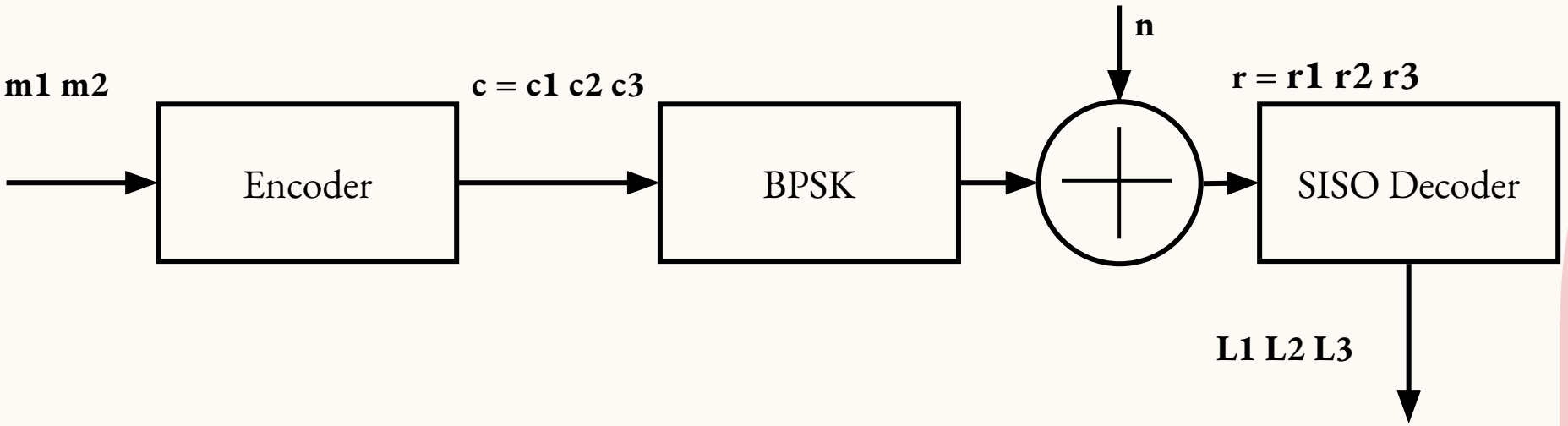
$l_{ext,1} = -0.4 + 2.8 = 2.4$

$L_1 = l_1 + l_{ext,1} = 1.8 + 2.4 = 4.2$

Decision:

- Since $L_1 > 0 \rightarrow$ bit is decoded as 0
- Large LLR means **strong confidence**

SISO Decoder For (3,2)SPC Code



- $L1 = l_1 + l_{ext,1}$
- channel LLR $l_1 = (2/\sigma^2)*r1$
- $l_{ext,1}$ = sign(r2)*sign(r3)* min(r2,r3) (in min-sum approximation)

SISO Decoder For SPC Code

Calculations for L1:

$$L_1 = l_1 + l_{ext,1}$$

$$l_i = \frac{2}{\sigma^2} r_i$$

$$l_{ext,1} = log\left(\frac{P(c_1 = 0 \,|r_2 r_3)}{P(c_1 = 1 \,|r_2 r_3)}\right) = log\left(\frac{p_{e1}}{1 - p_{e1}}\right)$$

As we know c1 = c2 $\oplus$ c3

$$p_{e1} = p_2 p_3 + (1 - p_2)(1 - p_3)$$

where $p_i = log\left(\frac{c_i = 0 \,|r_i}{c_i = 1 \,|r_i}\right)$

From given equation we can derive:-

$$\frac{(p_{e1} - (1 - p_{e1}))}{(p_{e1} + (1 - p_{e1}))} = \frac{(p_2 - (1 - p_2))}{(p_2 + (1 - p_2))} \frac{(p_3 - (1 - p_3))}{(p_3 + (1 - p_3))}$$

$$\frac{\left(1 - \frac{(1-p_{e1})}{p_{e1}}\right)}{\left(1 + \frac{(1-p_{e1})}{p_{e1}}\right)} = \frac{\left(1 - \frac{(1-p_2)}{p_2}\right)}{\left(1 + \frac{(1-p_2)}{p_2}\right)} \frac{\left(1 - \frac{(1-p_3)}{p_3}\right)}{\left(1 + \frac{(1-p_3)}{p_3}\right)}$$

$$\frac{(1 - e^{-l_{ext,1}})}{(1 + e^{-l_{ext,1}})} = \frac{(1 - e^{-l_2})}{(1 + e^{-l_2})} \frac{(1 - e^{-l_3})}{(1 + e^{-l_3})}$$

$$tanh\left(\frac{l_{ext,1}}{2}\right) = tanh\left(\frac{l_2}{2}\right) tanh\left(\frac{l_3}{2}\right)$$

$$log\left(tanh\left(\frac{l_{ext,1}}{2}\right)\right) = log\left(tanh\left(\frac{l_2}{2}\right)\right) + log\left(tanh\left(\frac{l_3}{2}\right)\right)$$

*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

SISO Decoder For SPC Code

Calculations for L1:

we can separate the magnitude and sign of lext,1

$$\log\left(\tanh\left(\frac{|l_{ext,1}|}{2}\right)\right) = \log\left(\tanh\left(\frac{|l_2|}{2}\right)\right) + \log\left(\tanh\left(\frac{|l_3|}{2}\right)\right)$$

$$sgn\left(l_{ext,1}\right) = sgn\left(l_2\right)sgn\left(l_3\right)$$ because tanh is odd function

let $f = \left|\log(\tanh\left(\frac{|x|}{2}\right))\right|$ we can prove that f is self inverse function.

Minsum Approximation

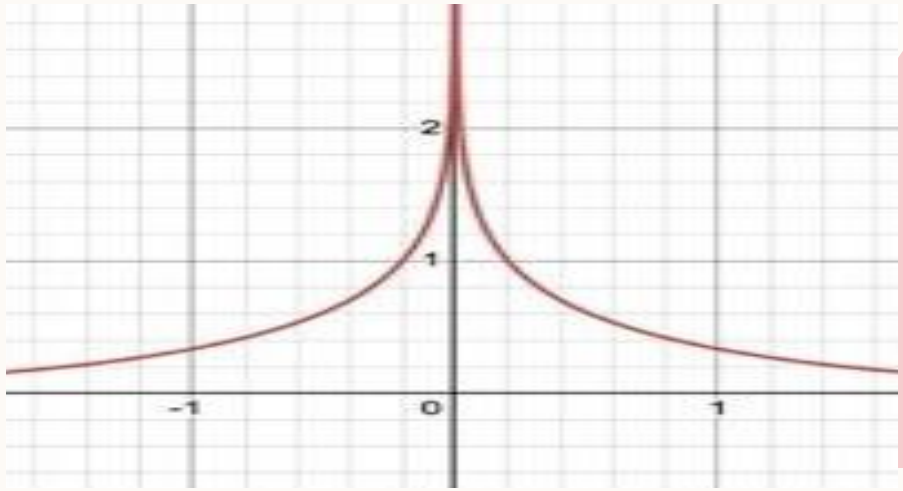
$$f\left(l_2\right) + f\left(l_3\right) \approx f\left(min\left(|l_2|, |l_3|\right)\right)$$

$$|l_{ext,1}| = f\left(f\left(l_2\right) + f\left(l_3\right)\right)$$

$$|l_{ext,1}| \approx f\left(f\left(min\left(|l_2|, |l_3|\right)\right) \approx min\left(|l_2|, |l_3|\right)\right.$$

How min sum approximation works

Graph of f(x) = |log(tanh(x/2))|



- The function is very large for small x.
- So when computing the sum, small LLR values dominate the result.
- Since the **smallest** term controls the outcome, the sum can be approximated by the **minimum** value.
- As you can see in the graph, for small x, f(x) shoots up — this is why Min-Sum uses the minimum instead of the full sum.

SISO Decoder For (n-1,n)SPC Code

- Absolute value of extrinsic LLR

$$|L_i^{ext}| = f\left(\sum_{j\neq i} f(|l_j|)\right)$$

$$f(x) = |\log(\tanh(x/2))|$$

$$f(x) = f^{-1}(x)$$

- Absolute value of extrinsic LLR after min-sum approximation

$$|L_i^{(ext)}| = \min_{j\neq i} |l_j|$$

- sign of extrinsic LLR

$$\text{sign}(L_i^{(ext)}) = \prod_{j\neq i} \text{sign}(l_j)$$

- Final Value

$$L_i^{(ext)} = \left(\prod_{j\neq i} \text{sign}(l_j)\right) \cdot \min_{j\neq i} |l_j|$$

Soft Decision Decoding (in algorithm)

MinSum decoder :

- Storage matrix L : sparse matrix of same dimensions as parity check matrix.
- L(i,j) = 0 if H(i,j) = 0
- L(i,j) can be non-zero only if H(i,j) = 1

$$\boldsymbol{r} = [\, r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5 \quad r_6 \quad r_7 \,]$$

$$L = \begin{bmatrix} r_1 & r_2 & r_3 & 0 & r_5 & 0 & 0 \\ 0 & r_2 & r_3 & r_4 & 0 & r_6 & 0 \\ r_1 & r_2 & 0 & r_4 & 0 & 0 & r_7 \\ r_1 & 0 & r_3 & 0 & r_5 & r_6 & r_7 \end{bmatrix}$$

*This example is taken from, Andrew Thangaraj. Ldpc and polar codes in the 5g standard, 2019.

For each Row (Check Nodes (SPC))

- **Magnitude**
  - min1 = minimum absolute value of all non-zero entries in row
  - min2 = next higher absolute value
  - set magnitude of all values

Set magnitude of all values except minimum = min1

Set magnitude of minimum value = min2

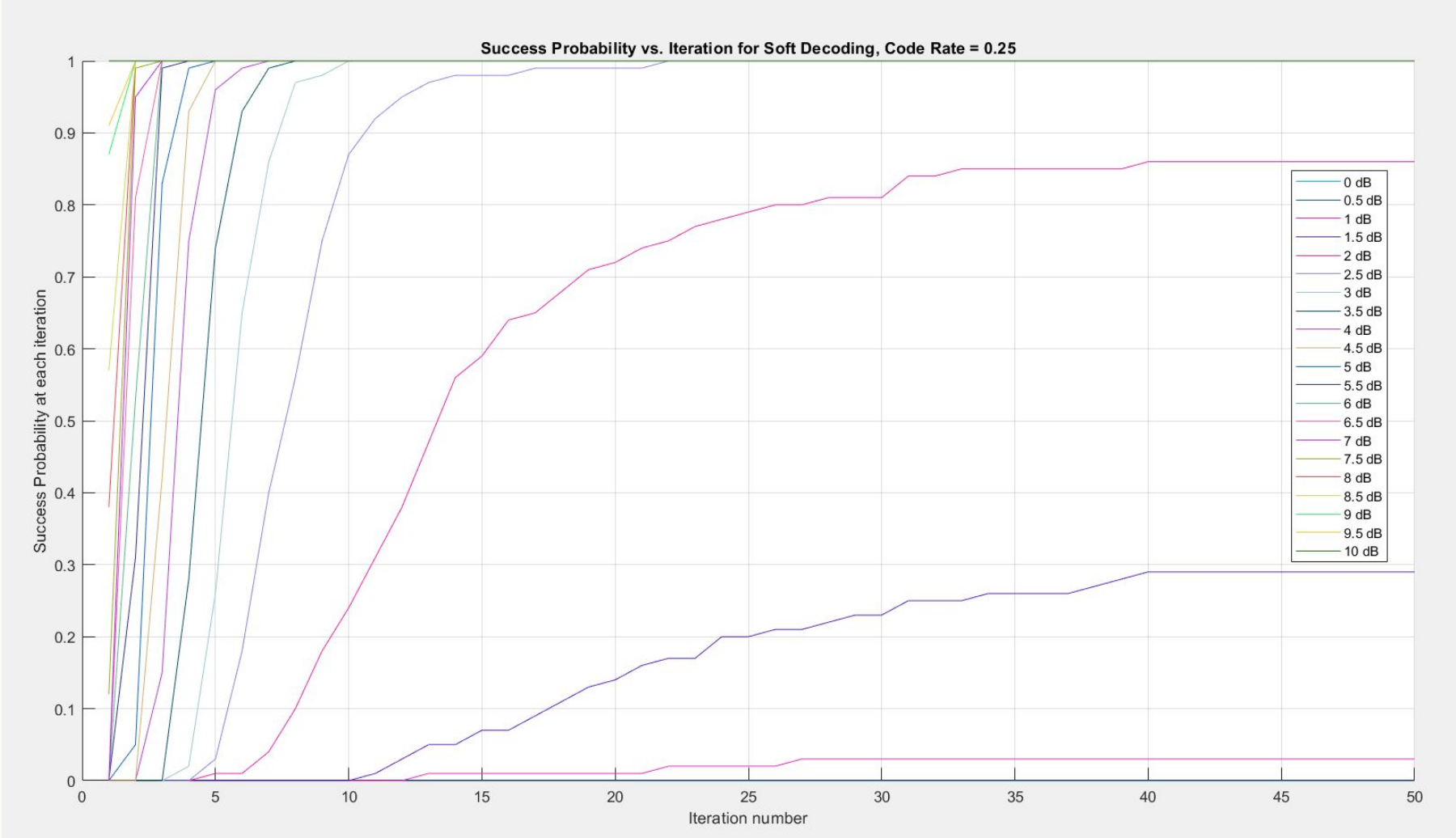
- **Sign**
  - Parity = Product of signs of entries in row
  - New sign of entry = (old sign) (parity)

For each Column (Variable Nodes (repetition))

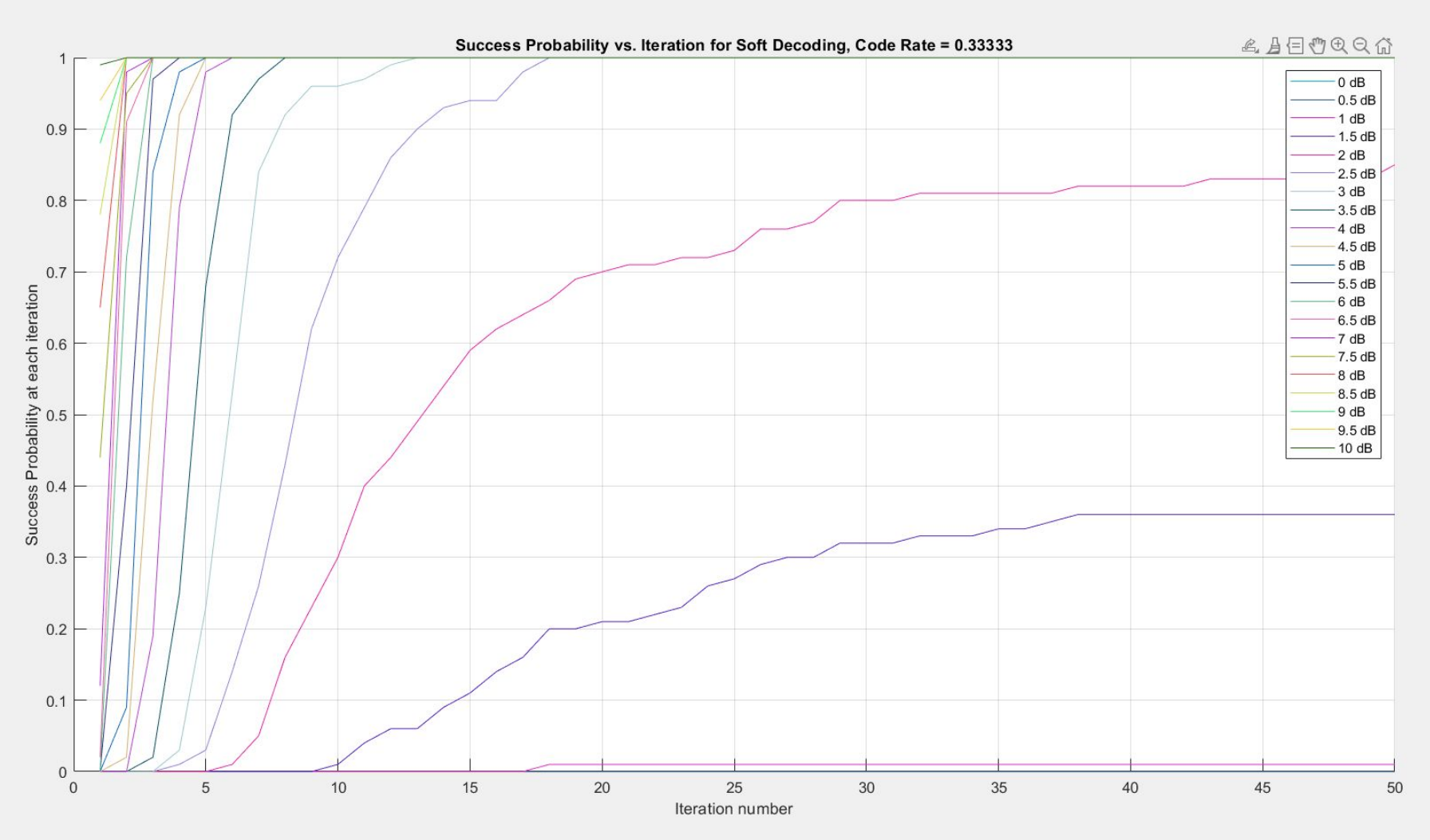
- **New Values**
  - Sum_j = r_j + sum of all entries in column j.
  - New Entry = Sum - (Old Entry)
  - Sum is the new belief generated about values after the ith iteration
- **Decoding Values**
  - If sum_j > 0, decision on bit j = 0
  - If sum_j <0, decision on bit j = 1

Results For Base Graph NR_2_6_52

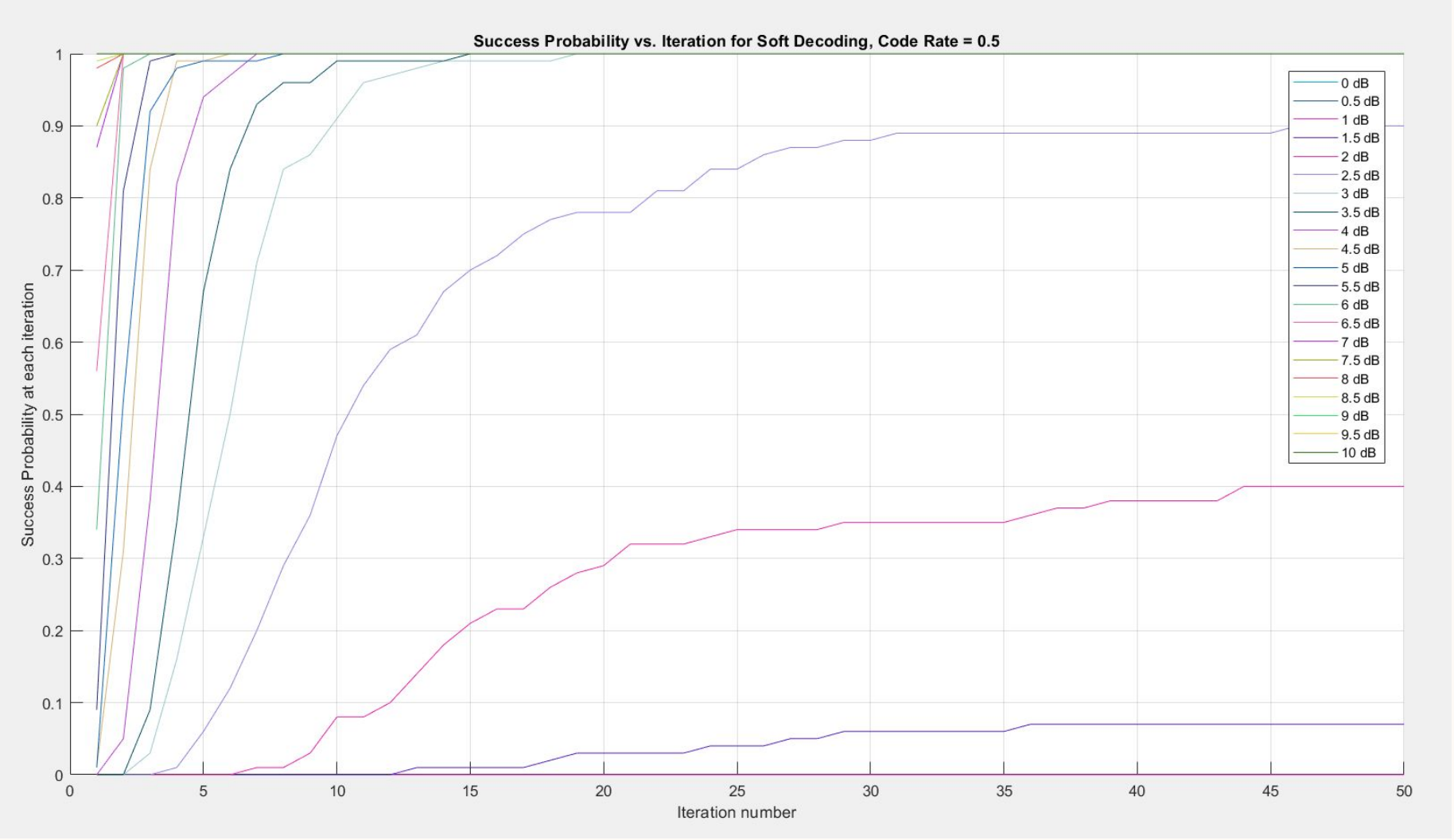
Coderate 1/4 (Soft Decoding)



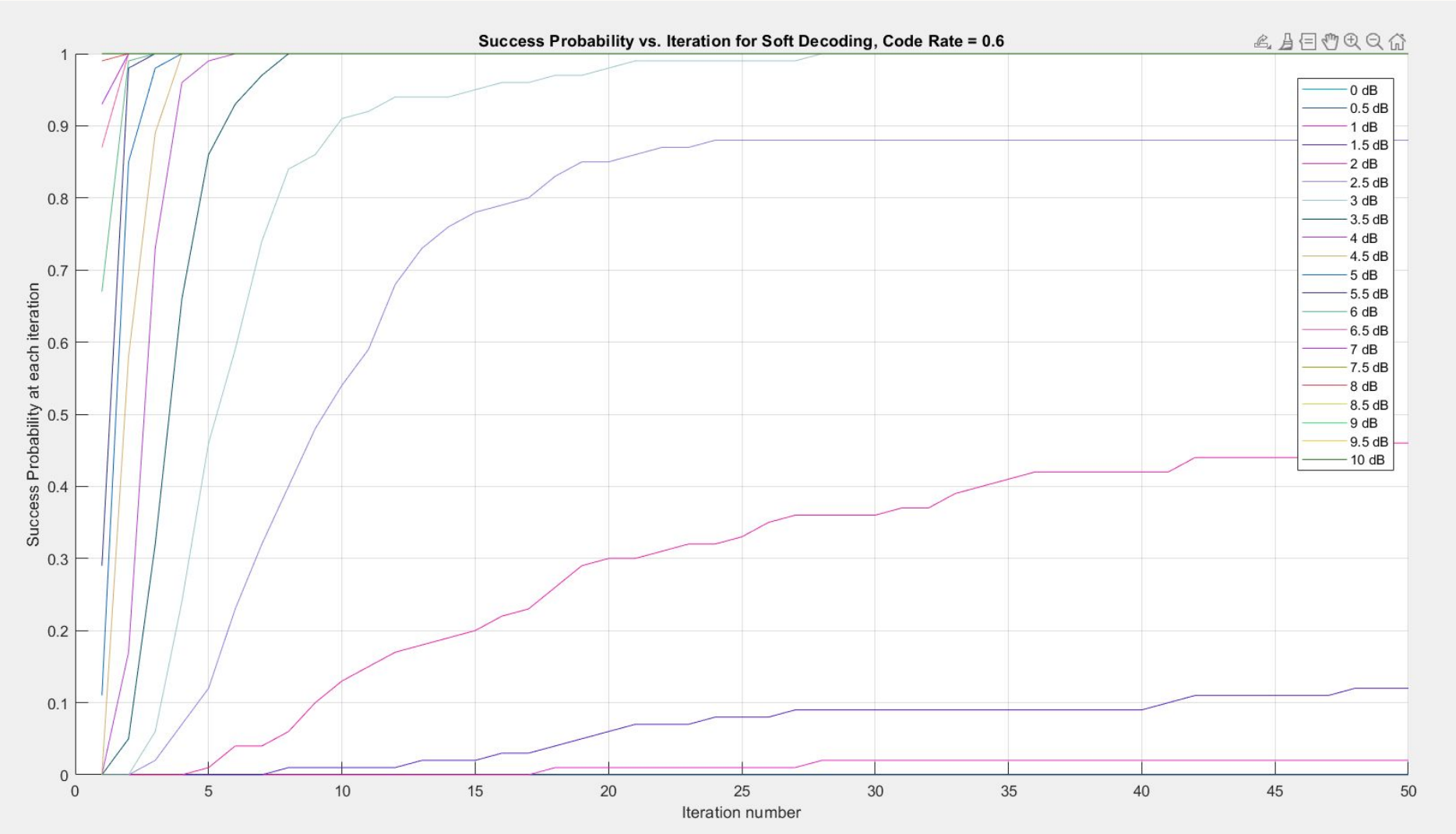
Coderate 1/3 (Soft Decoding)



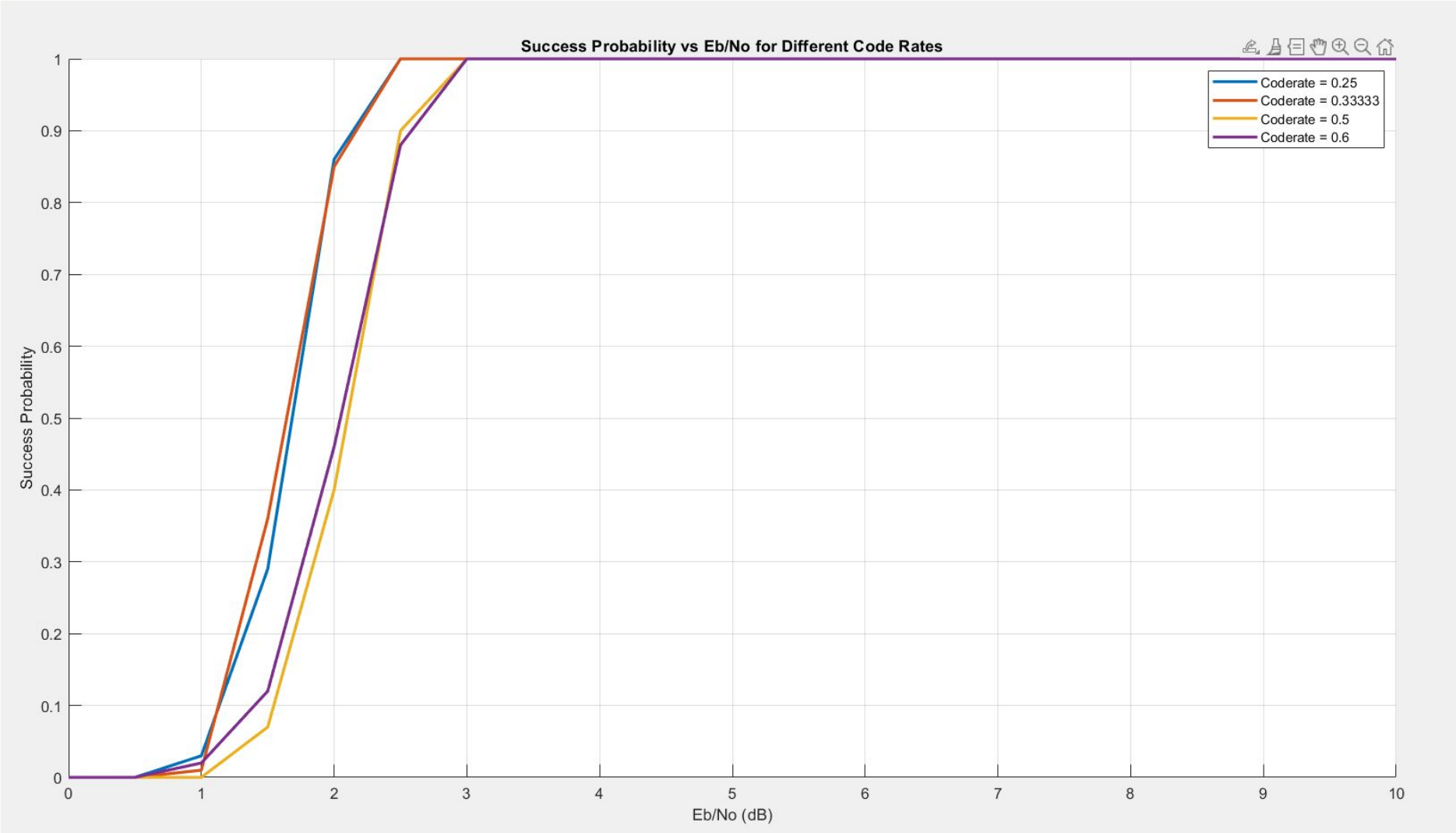
Coderate 1/2 (Soft Decoding)



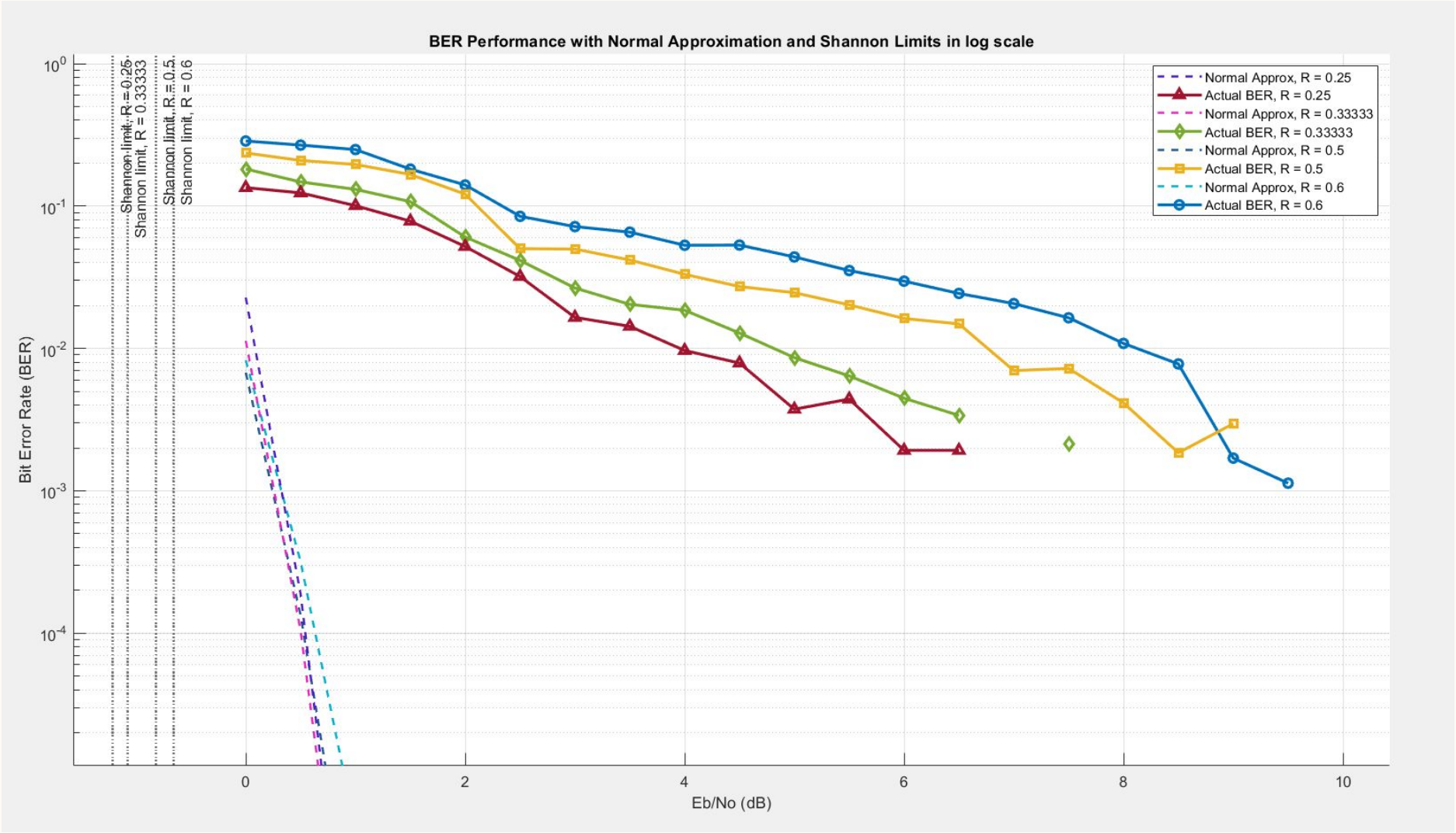
Coderate 3/5 (Soft Decoding)



Success Probability vs Eb/No for Different Code Rates



BER Performance with Normal Approximation and Shannon Limits in log scale



Analysis

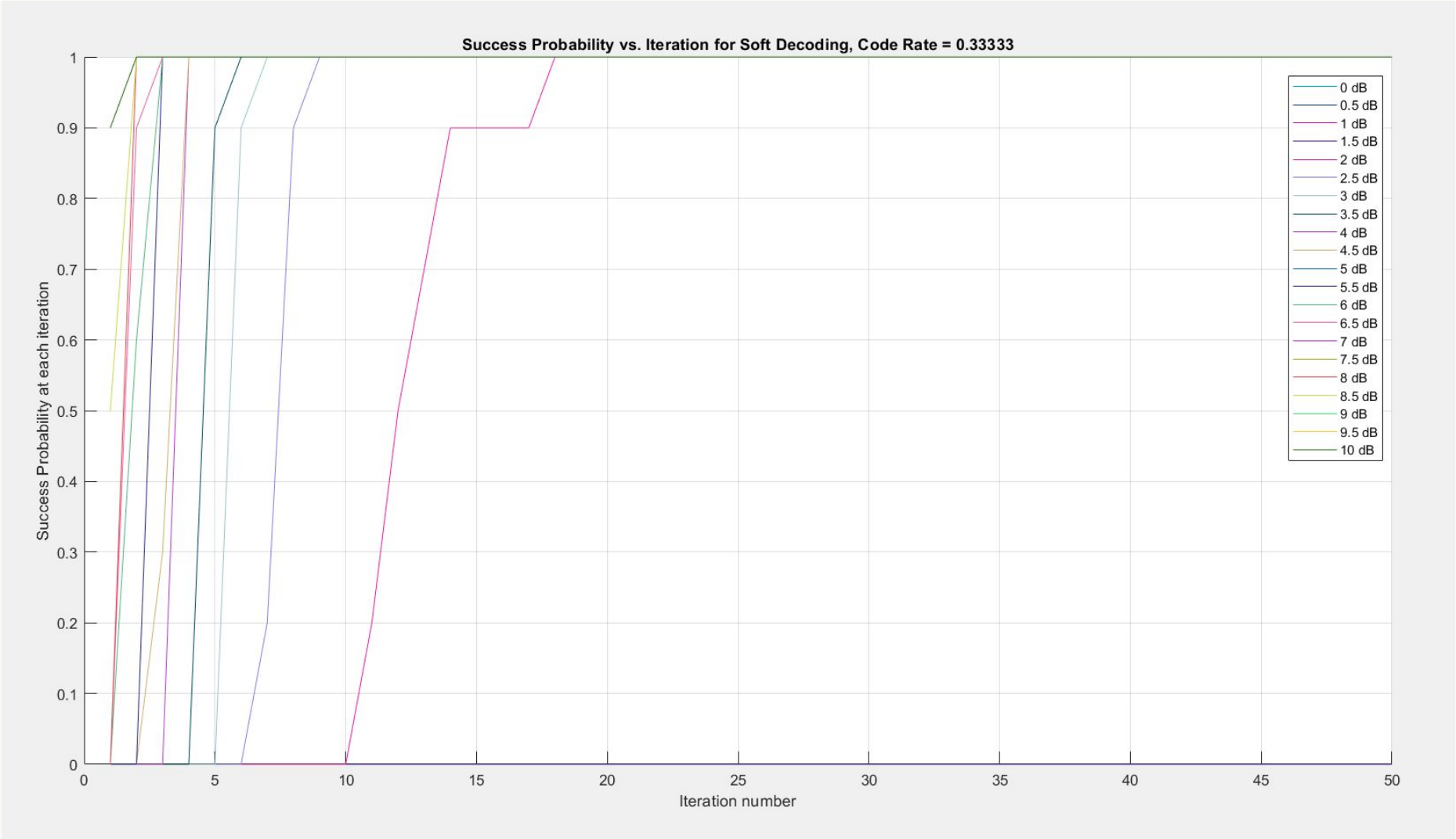
- We have plotted the Success Probability vs the No. of Iterations graphs for different coderates(¼,⅓,½,⅗).We observe that the Success Probability reaches(tends to) 1 in very few no.of iterations for lower coderates such as ¼ and ⅓.(in 3 to 10 iterations), whereas for higher coderates like ½ and ⅗ it usually takes more number of iterations to achieve a high success probability.
- The Success Probability Graph vs Eb/No for Different Code rates show a waterfall region in the graph where with the increase in Eb/No ratio the success probability rapidly increases from 0 to 1.

Analysis

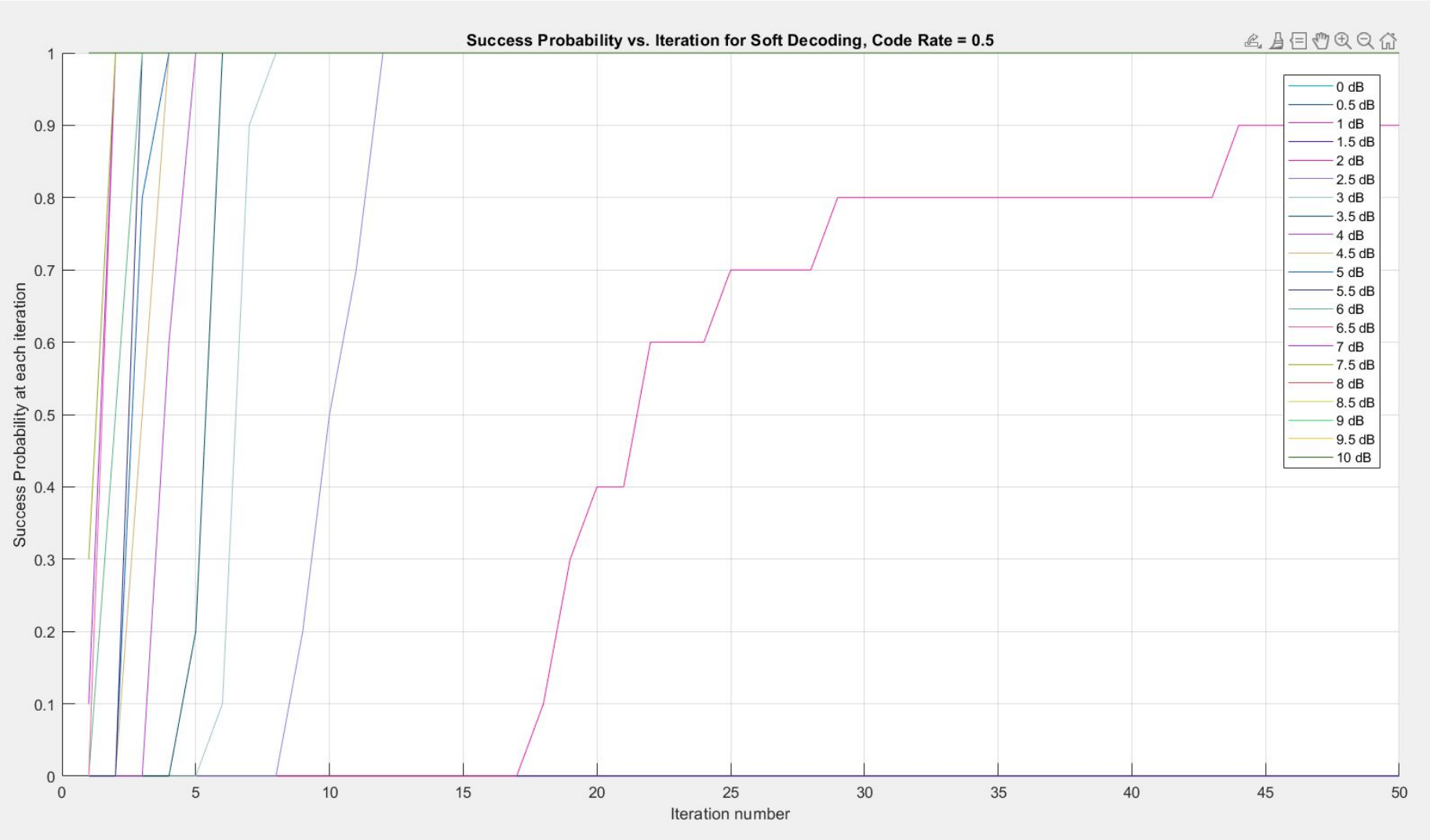
- Lower R values (0.25) show better BER performance than higher R values (0.5) at the same Eb/N0, demonstrating the rate-reliability tradeoff.
- A significant gap exists between normal approximations and actual BER performance, highlighting theoretical versus practical limitations.
- All curves approach an error floor at high Eb/N0 values (>4/5dB), where increasing power brings diminishing returns i.e it stagnets.

Results For Base Graph NR_1_5_352

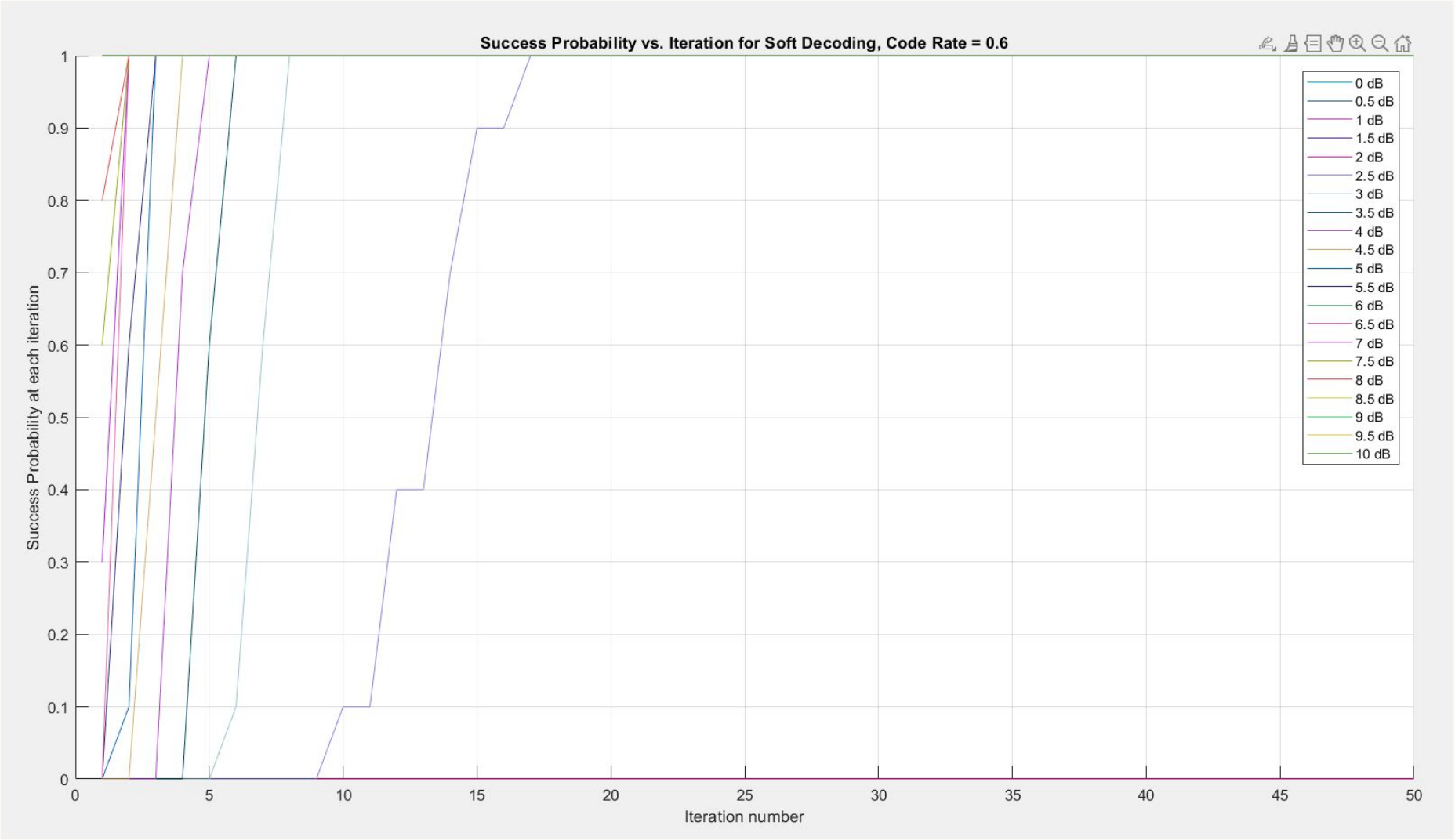
Coderate 1/3 (Soft Decoding)



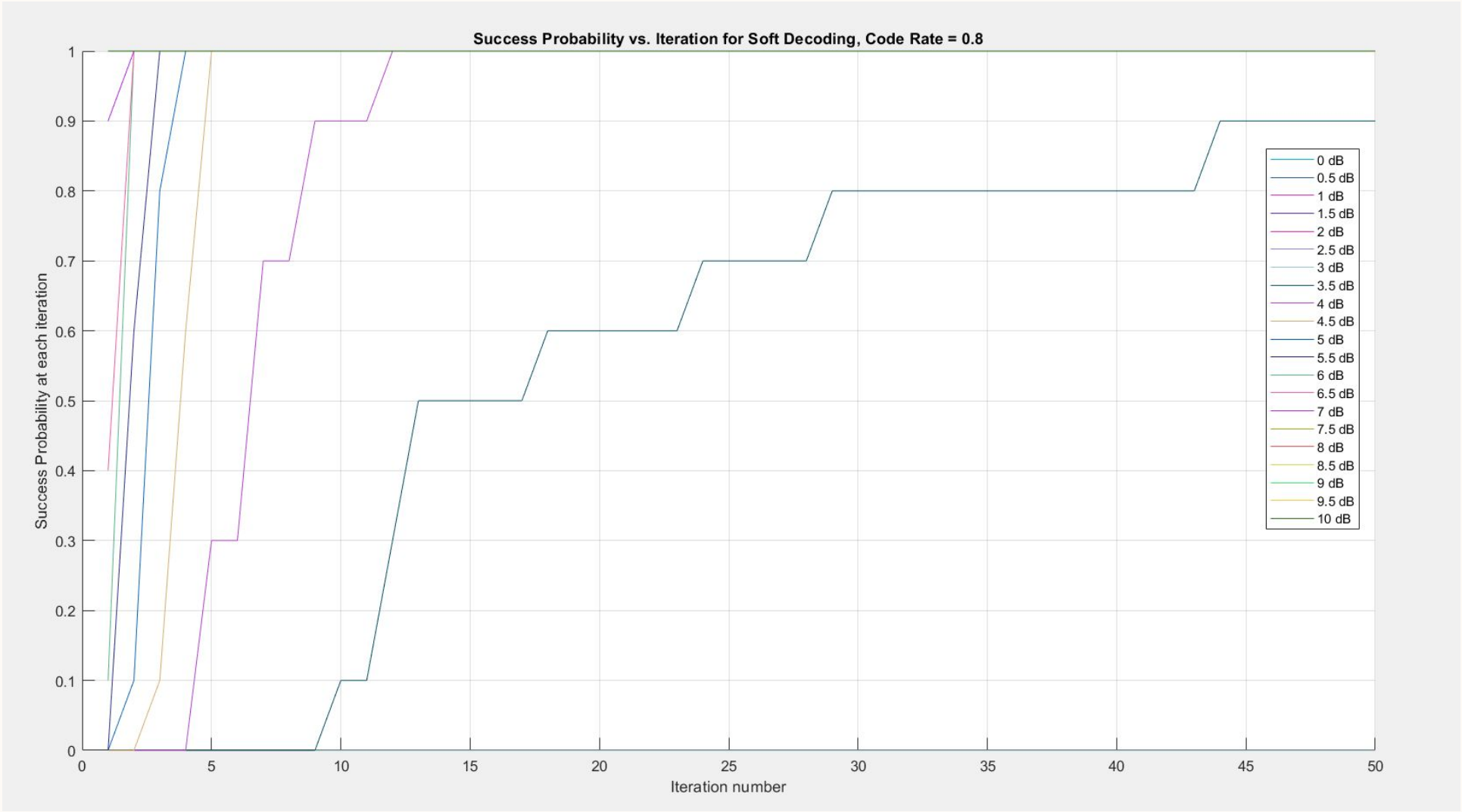
Coderate 1/2 (Soft Decoding)



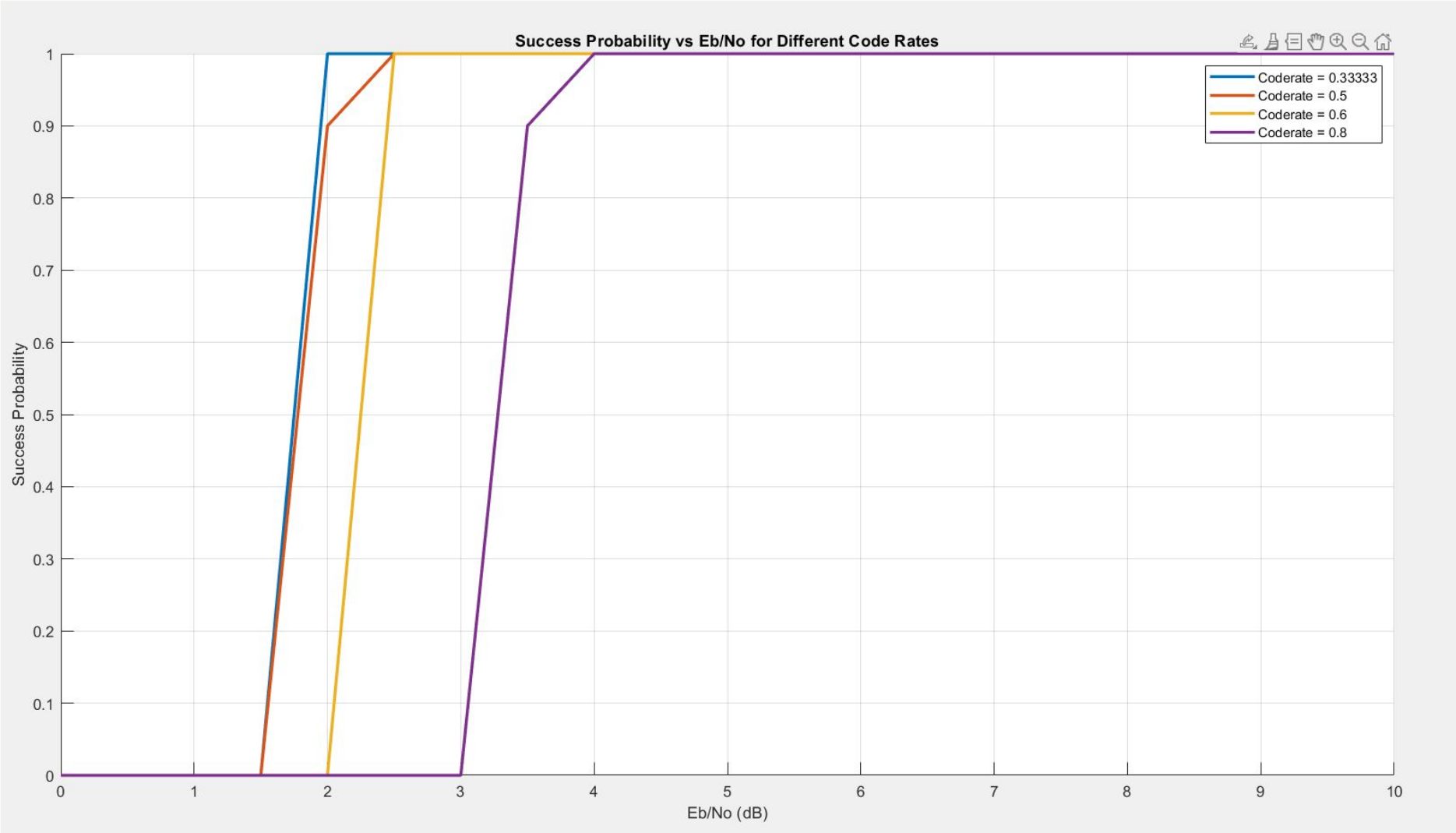
Coderate 3/5 (Soft Decoding)



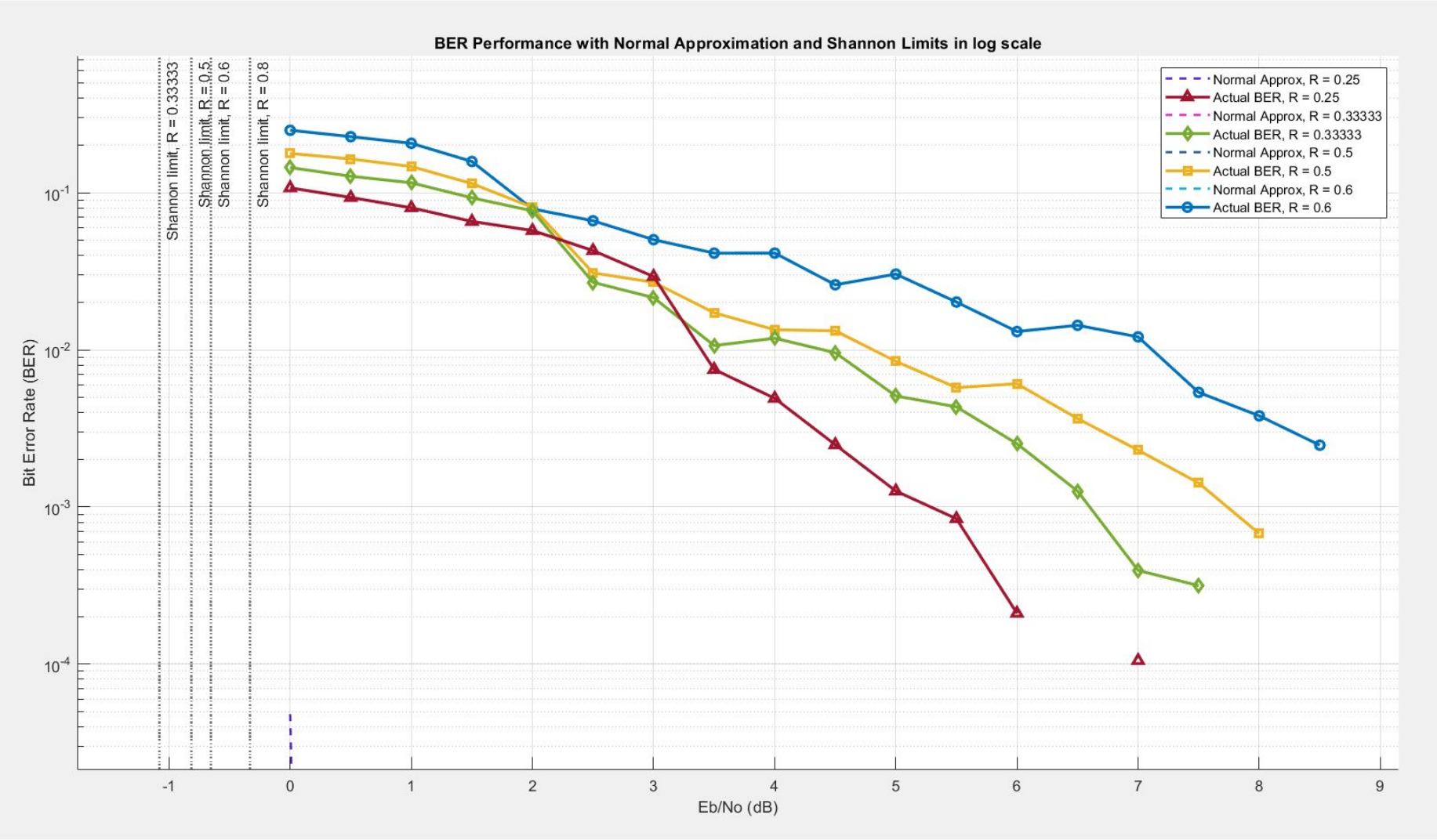
Coderate 4/5 (Soft Decoding)



Success Probability vs Eb/No for Different Code Rates



BER Performance with Normal Approximation and Shannon Limits in log scale



Analysis

- We have plotted the Success Probability vs the No. of Iterations graphs for different coderates$(\frac{1}{3}, \frac{1}{2}, \frac{3}{5}, \frac{4}{5})$.We observe that the Success Probability reaches(tends to) 0.9 in very few no.of iterations for lower coderates such as $\frac{1}{3}$ and $\frac{1}{2}$.(in 10 to 15 iterations), whereas for higher coderates like $\frac{3}{5}$ and $\frac{4}{5}$ it usually takes more number of iterations(above 20) to achieve a high success probability.
- For high SNR values we achieve faster convergence and and higher success probability regardless of code rate.
- For lower SNR values there is no guaranteed convergence and the results vary from graph to graph

Analysis

- Lower R values (0.33) show better BER performance than higher R values (0.5) at the same Eb/N0, demonstrating the rate-reliability tradeoff.
- A significant gap exists between normal approximations and actual BER performance, highlighting theoretical versus practical limitations.
- All curves approach an error floor at high Eb/N0 values (>5dB), where increasing power brings diminishing returns i.e it stagnets.

THANK YOU