CT-216

LDPC DECODING FOR 5G NR

Lab Group - 1, Sub Group - 3

Under Guidance of : Prof. Yash Vasavda
Mentor TA: Mr. Harshneel

HONOR CODE

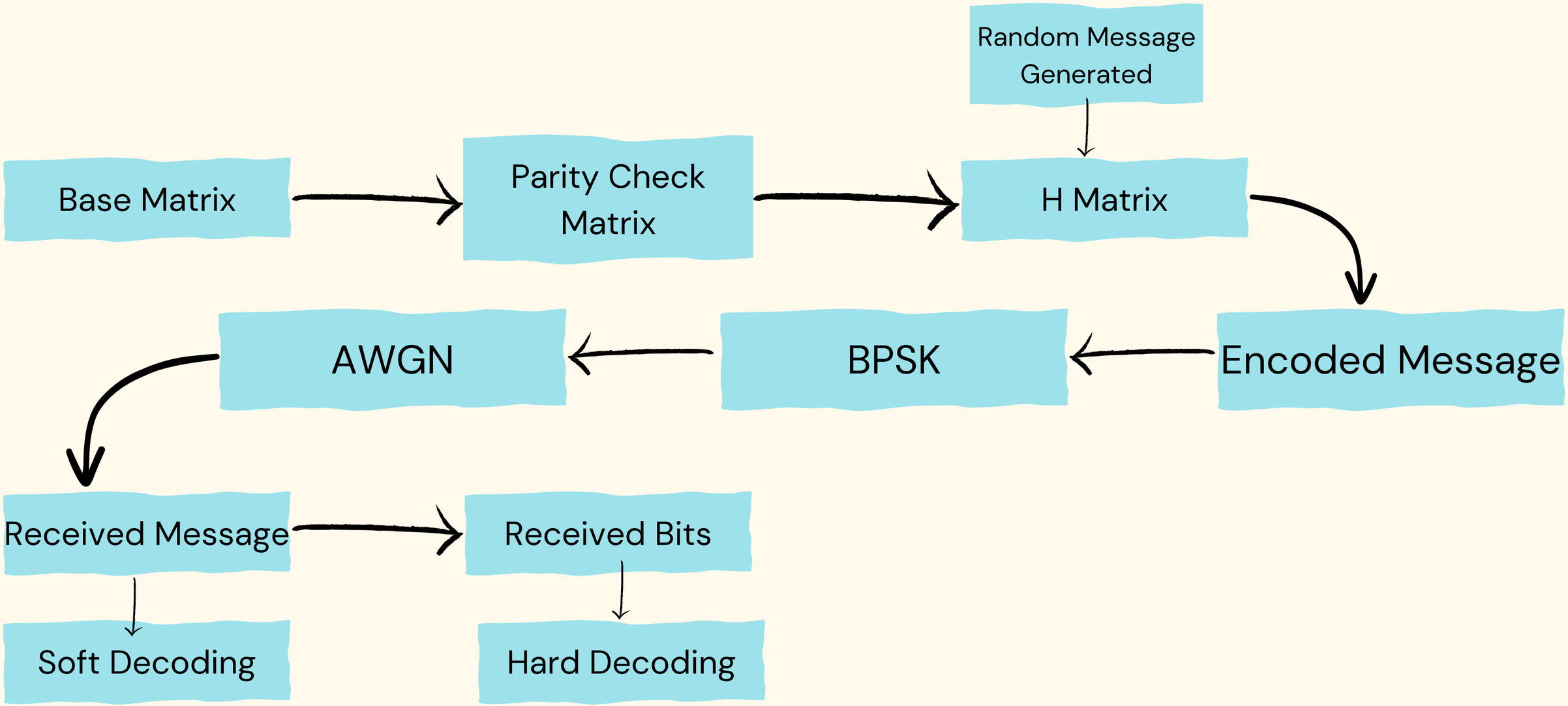
- We declare that:
  - The work that we are presenting is our own work.
  - We have not copied the work (the code, the results, etc.) that someone else has done.
  - Concepts, understanding and insights we will be describing are our own.
  - We make this pledge truthfully. We know that violation of this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

Contents

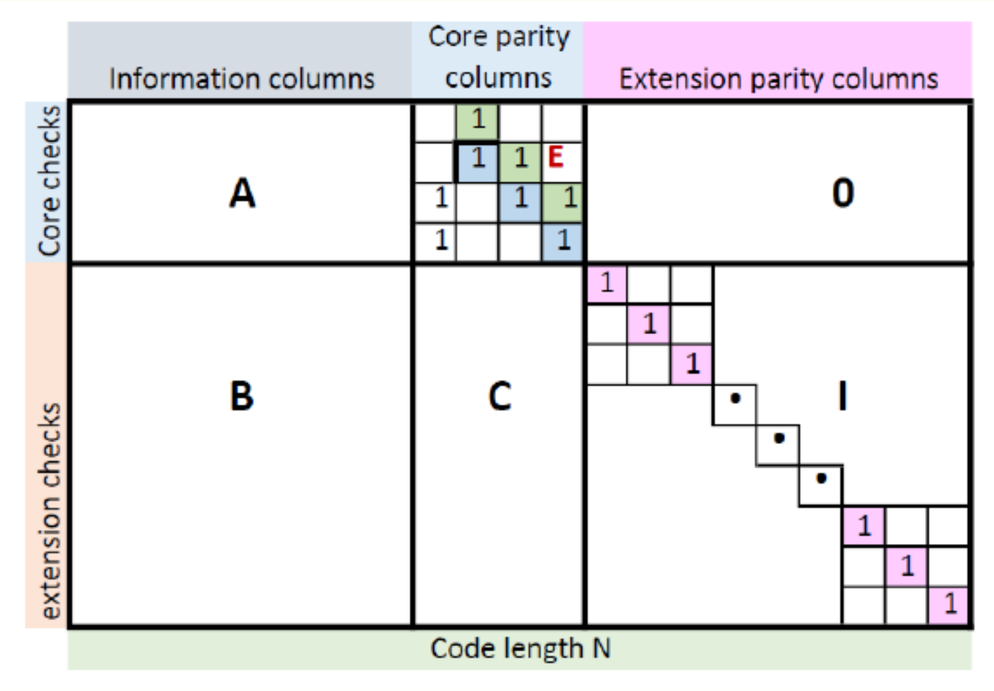
About LDPC Codes

- A type of linear block code introduced by Robert Gallager in 1960s.
- Have a special characteristic of sparse H matrix.
- Offer exceptional suitability for 5G NR shared channels like:
    - Minimal latency,
    - Low decoding complexity
    - Remarkable rate compatibility
    - Very low error rate
- LDPC codes are adaptable to various block sizes and code rates, owing to the innovative design of rate-compatible base graphs.

PROGRAM FLOW



Base Graph Analysis



- 5G NR makes use of two different sized base graphs, BG1 and BG2, whose usage is determined by the code rate and size of information bits.
  - BG1 has a size of 46 x 68.
  - BG2 has a size of 42 x 52.
- The number of message bits is determined by k=m-n where the base graphs are of size m x n. Thus,
  - BG1 has a maximum of 22 bits.
  - BG2 has a maximum of 10 bits.
- Both the base graphs have a distinctive block structure as shown in the figure which can be exploited during encoding.
- By having two base graphs, 5G NR LDPC codes can adapt to various data sizes efficiently, ensuring reliable communication across different scenarios.

Parity Check Matrix Generation

- To construct the parity check matrix H from the base graph B we use z, which is the lifting size or the expansion factor, which can be determined for every base graph, and make use of the following rules :
  - Every element in B having value -1 is replaced by a z x z sized all zero matrix.
  - Every element in B having value 0 is replaced by a z x z sized identity matrix
  - Every element in B having values from 1 to z - 1 is replaced by the z x z identity matrix circularly shifted right $B_{i,j}$ times where $B_{i,j}$ is the value of the element in B.

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

Expansion factor: 5

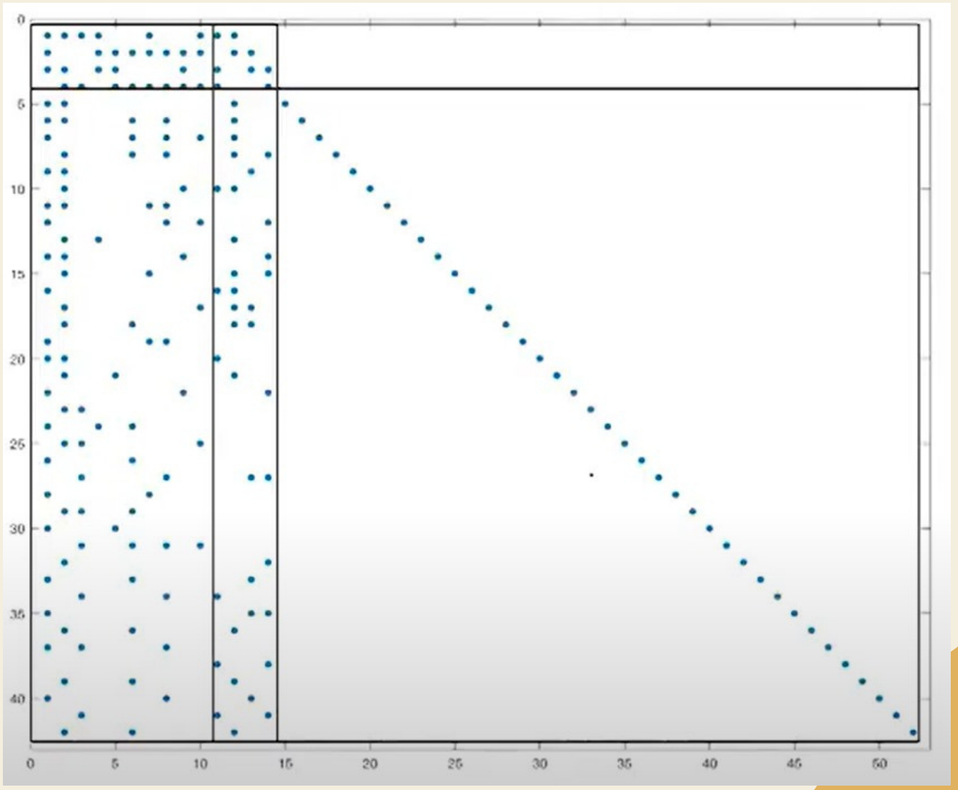
$H =$

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

Base Matrix

to

H Matrix



Base Matrix

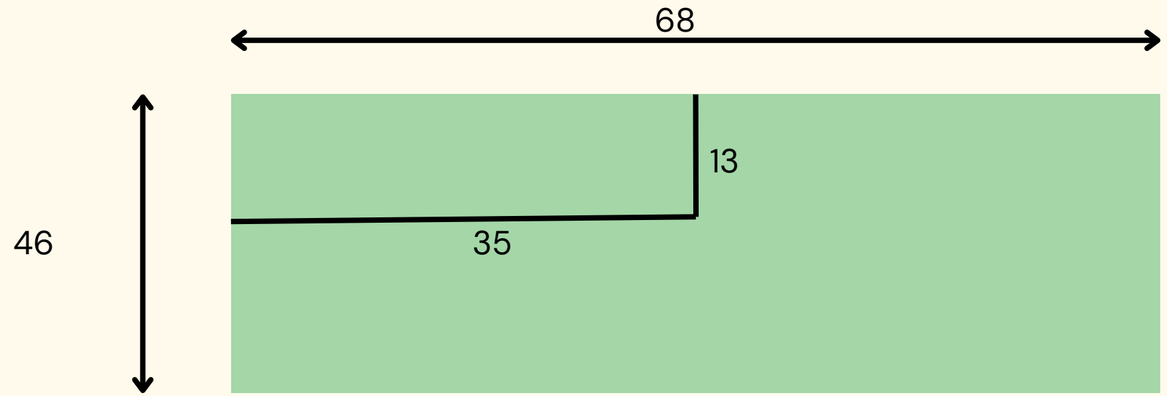
CODERATE MATCHING

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

- Rate matching is a technique used in Low-Density Parity Check (LDPC) codes to adjust the code rate of the LDPC code to match the desired transmission rate of the communication system.
- Suppose you have a base matrix of size m x n and an expansion factor of z. So the possible length of the codeword will be n*z.
- Out of these n*z bits, the first 2z are always punctured. So we have (n-2)*z bits left.
- Suppose you want to achieve an overall code rate of x/y:
  - Assume that you puncture a*z bits out of the available (n-2)*z bits.
  - Number of message bits = (n-m) * z.
  - From the equation shown, we can compute the number of parity bits required.
- So, the required size of H matrix is:
  - Number of rows = Number of parity bits which we get from the equation
  - Number of columns = Number of columns from equation + 2.
- Let’s consider one example ahead for better understanding.

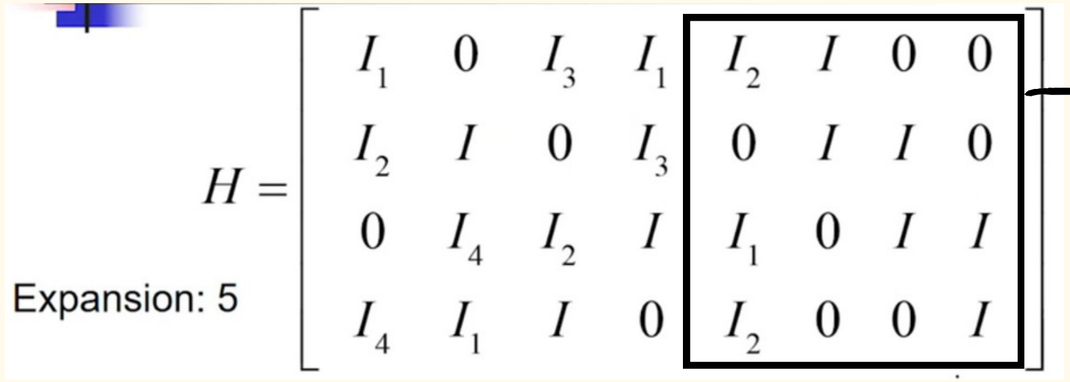
CODERATE MATCHING EXAMPLE

- Suppose we have a 46 X 68 Base matrix and we want a code rate of 2/3.
- So, n = 68, m= 46, x = 2, y = 3. Putting these values in the previous equation and solving for a, we get a = 33.
- So, now we have to transmit 66z - 33z = 33z bits in all.
- Out of these 33z bits, 20z are message bits. Thus, we send 13z parity bits.
- Number of rows of B matrix used = 13 rows
- Number of columns of B matrix used = 33 + 2 = 35
- So, for a code rate of 2/3 we use 13 X 35 entries of the B matrix.



ENCODING USING BASE MATRIX

- We exploit the structure of the base matrix to do the encoding.
- The main idea behind the decoding is that all the parity bits greater than equal to 5 can be obtained simply by solving a linear equation because of the structure of the matrix.
- To find the first four bits, we need to solve 4 equations simultaneously. This calculation is made easy due to the double-diagonal structure of the matrix.
- We will consider an example to understand the encoding:
  - We have a message vector m = [m1 m2 m3 m4 m5]
  - 4 parity bits are added to this vector which are [p1 p2 p3 p4]
  - Codeword = [c1 c2 c3 c4 c5 c6 c7 c8 c9]
- The base matrix is given as follows:



Double - Diagnol Structure

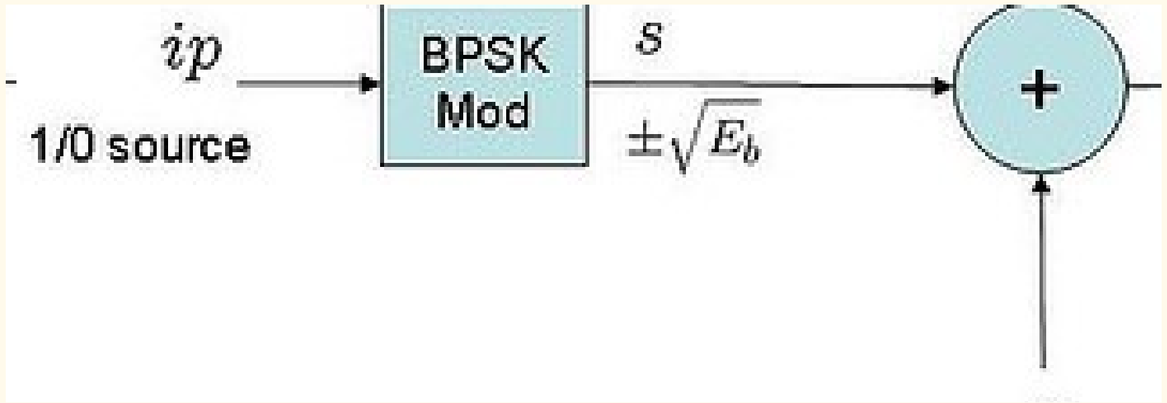
- Ik shows the 5 x 5 identity matrix right shifted k times.
- 0 represents a 5 x 5 all zeros matrix

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

- Equations obtained:
  - $I_1v_1 + I_3v_3 + I_1v_4 + I_2v_5 + Iv_6 = 0$ -Equation 1
  - $I_2v_1 + Iv_2 + I_3v_4 + Iv_6 + Iv_7 = 0$ -Equation2
  - $I_4v_2 + I_2v_3 + Iv_4 + I_1v_5 + Iv_7 + Iv_8 = 0$ -Equation 3
  - $I_4v_1 + I_1v_2 + Iv_3 + I_2v_5 + Iv_8 = 0$ -Equation 4
- Adding all four equations:
  - $I_1v_5 = I_1v_1 + I_2v_1 + I_4v_1 + Iv_2 + I_4v_2 + I_1v_2 + I_3v_3 + I_2v_3 + Iv_3 + I_1v_4 + I_3v_4 + Iv_4$
- We know that $[v_1\ v_2\ v_3\ v_4\ v_5\ v_6\ v_7\ v_8\ v_9] = [m_1\ m_2\ m_3\ m_4\ m_5\ p_1\ p_2\ p_3\ p_4]$
- So, rewriting the equation, we get:
- $I_1p_1 = I_1m_1 + I_2m_1 + I_4m1 + Im_2 + I_4m_2 + I_1m_2 + I_3m_3 + I_2m_3 + Im_3 + I_1m_4 + I_3m_4 + Im_4$
- From here we can get p_1.
- Substitute it in equation1 to get p_2 and solve so on to get all the 4 parity bits
- The encoded message must satisfy the equality H*c = 0.

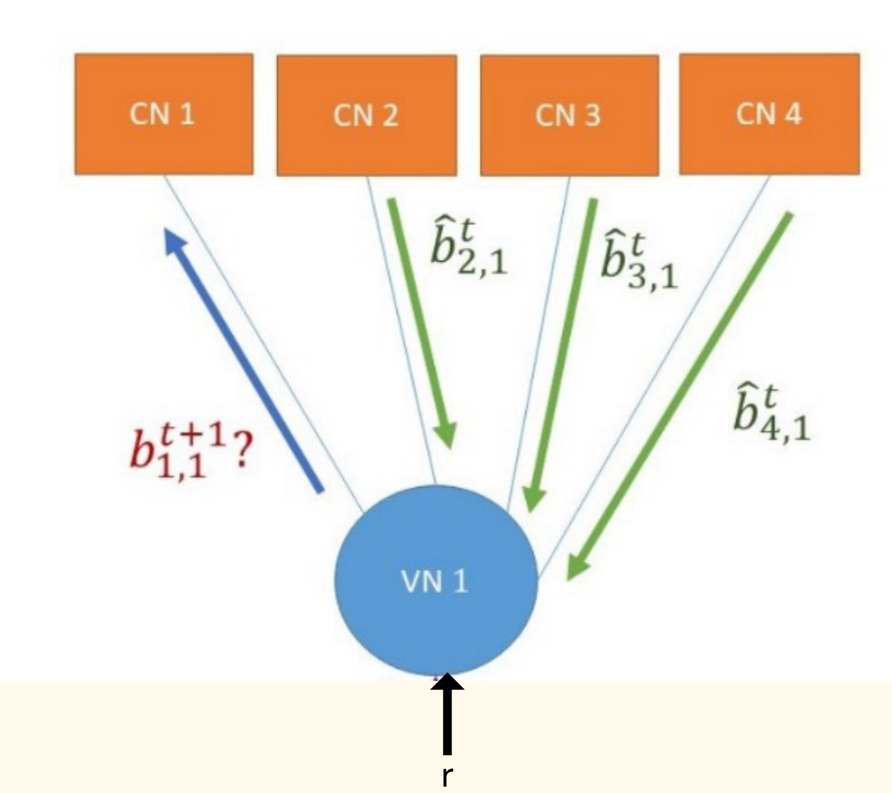
BPSK MODULATION AND NOISE

- The encoded output is then passed to the BPSK modulator which converts the matrix of encoded bits to the matrix of encoded symbols(1,-1).
  - 1 → -1
  - 0 → 1
- These encoded symbols when passed through the AWGN channel further add noise to the channel.
- Variance of noise added = σ = 1/√ (2 * Eb_No * r)



HARD DECISION DECODING

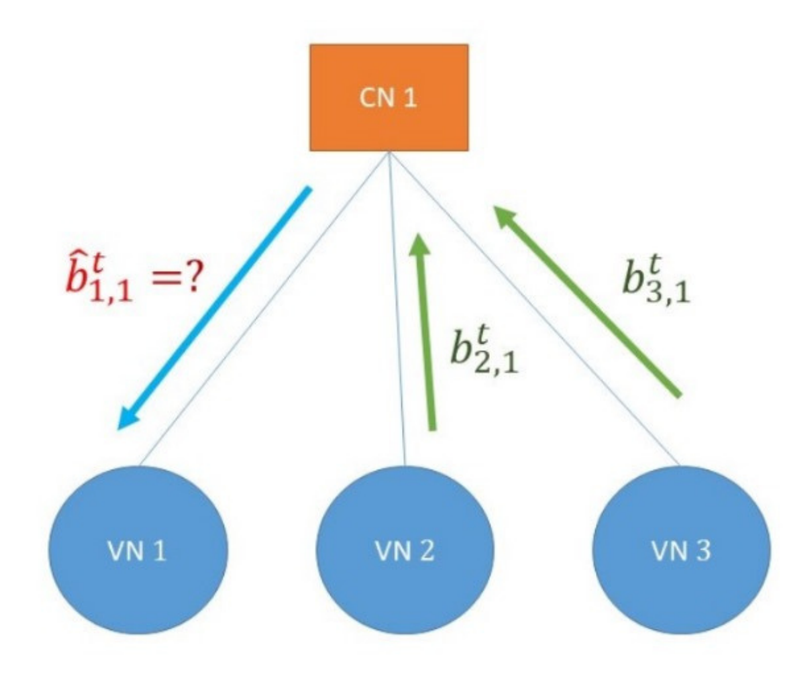
HARD DECISION DECODING ALGORITHM
VN to CN Message Passing



- 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.

HARD DECISION DECODING ALGORITHM

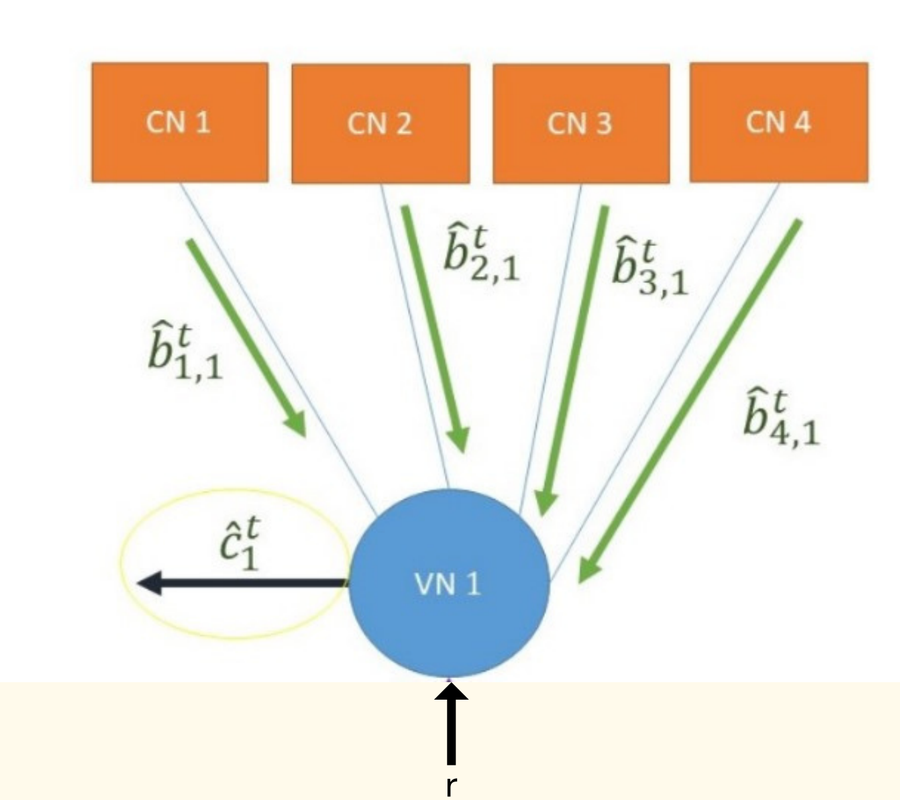
CN to VN Message Passing



- 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.

HARD DECISION DECODING ALGORITHM

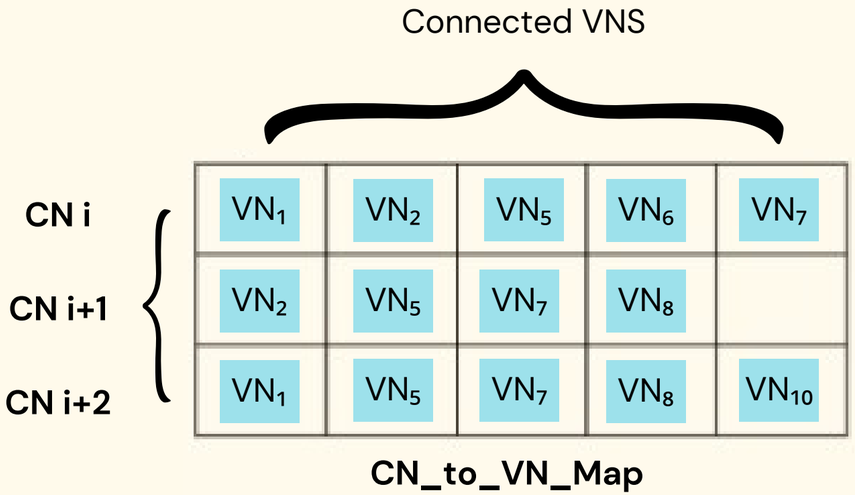
Computing ĉ

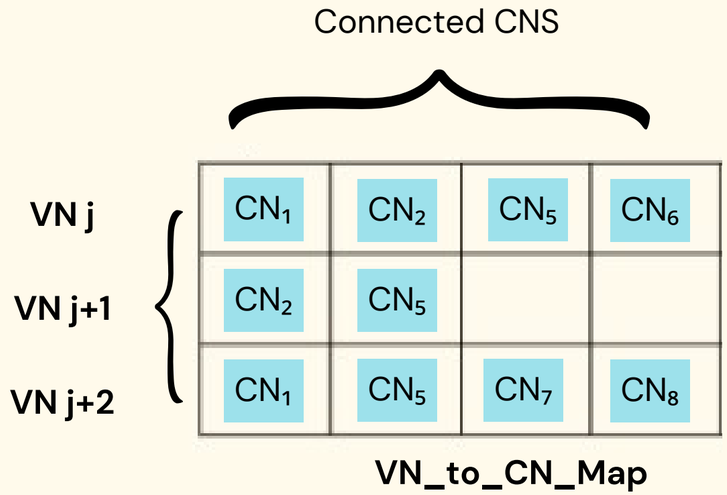


- After 1 complete iteration, we now estimate ĉ. Each bit of ĉ is determined by majority voting.
- Majority voting of all the messages received from each connected CN and the original message gives each bit of ĉ.
- We keep on doing these iterations until we reach the maximum iterations allowed or the estimated ĉ equals the encoded codeword or ĉ equals the last estimated codeword.

MAP GENERATION

- To implement the explained algorithm in MATLAB, we have made two cells named VN_to_CN_Map and CN_to_VN_Map. These maps remain same for both Soft Decision and Hard Decision Decoding.
- Following is the explanation of these maps:
- The first map shows that Cni is connected to 4 VNs - VN1, VN2, VN5, VN6 and VN 7.
- We can also see that since our H matrix is non-regular, the number of VNs connected with all CNs is not equal.
- Similarly, we have constructed a map for VN to CN.
- Now using these maps and an L matrix of the size equal to H matrix, we perform the decoding as explained before.





SOFT DECISION DECODING

SOFT DECISION DECODING ALGORITHM

- The input to the decoder is the received message from the channel which are real values which act as LLRs.

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

- In the first step these LLRs are loaded onto the variable nodes after which message transmission from the Vn to the respective connected Cn takes place. This can be calculated using the following formula.

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

Where $\alpha_{ij} \equiv sign(L(q_{i'j}))$ and $\beta_{i'j} \equiv |L(q_{i'j})|$ and $\varphi(x) \equiv log\frac{e^x+1}{e^x-1}$

- But calculating this value for every Cn for every iteration can make the algorithm complex, so we implement Min-Sum approximation for message passing from Vn to Cn.

cont.
MIN-SUM APPROXIMATION

- Due to the nature of the graph of the function ϕ(x), the sum of values of of all such values is dominated by the smallest value of x, which is:

$$\sum_{i' \in V_j/i} \varphi(\beta_{i'j}) \approx \varphi(min_{i'}(\beta_{i'j}))$$

- Thus by employing this Min-Sum approximation , the message passed from CN to VN becomes:

$$L(r_{ji}) = \prod_{i' \in V_{j/}i} a_{ij} \bullet min_{i' \in V_j/i} (\beta_{i'j}))$$

- After the CN sends message to VN, every VN also calculate the next message which it will pass to each CN it is connected to. This is given by:

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

cont.

FINAL DECISION MAKING

- After every iteration we calculate the Ĉ, which is the decoded message after every iteration. To do this we find:

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

- Through this value we can find the bits using the following method:

$$\hat{c}_i = \begin{cases} 1, & if\ L < 0 \\ 0, & else \end{cases}$$

- We have the final decoded Ĉ if either we have already completed the maximum number of iterations or when H*Ĉ=0.

ALGORITHM IMPLEMENTATION

- To implement the Soft decision decoder in MATLAB, we made use of the parity matrix H, where each row denotes a Check node and each column represents a Variable Node, and the 1's in H matrix at position h(i,j) represents and edge between the CN numbered i and VN numbered j.
- Using this model of the tanner graph, we make a copy of H and call it L, and we replace every 1 in L with the received LLR, basically loading every VN with the received LLRs.
- We then perform message passing from the CN to VN by employing the Min-Sum approximation algorithm in every row, that is replacing every non-zero value with the minimum of the every other non-zero value and then multiplying with the suitable parity.
- After this we perform message passing from the VN to CN by adding every value in each column. This gives us the total LLR which can be used to make the decision and find $\hat{C}$ and check if further iterations are required or not.

Example for Better Understanding

- Suppose we have this H matrix:

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

- The recived 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 follows:

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

Example for Better Understanding

- Calculating min1 and min2 for CN1: $\begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \end{bmatrix}$
- We calculate the minimum of the absolute non-zero values in the row. So min1 = 0.2 and min2 = 0.3. We replace all the non-zero values to min1 and the min1 to min2. Then, we calculate the overall sign and multiply it with original sign of the bit. This gives the sign of each element. $\begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \end{bmatrix}$
- Matrix obtained after all the row operations is as follows:

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

Example for Better Understanding

- Now, we calculate the total sum in each column and the received value. This gives the total belief of that bit.
- New entry in the column = Total sum - Old Entry.

$$r = \begin{bmatrix} 0.2 \\ -0.7 \\ 0 \\ -0.7 \\ -0.4 \end{bmatrix}$$

Sum = −1

- 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:

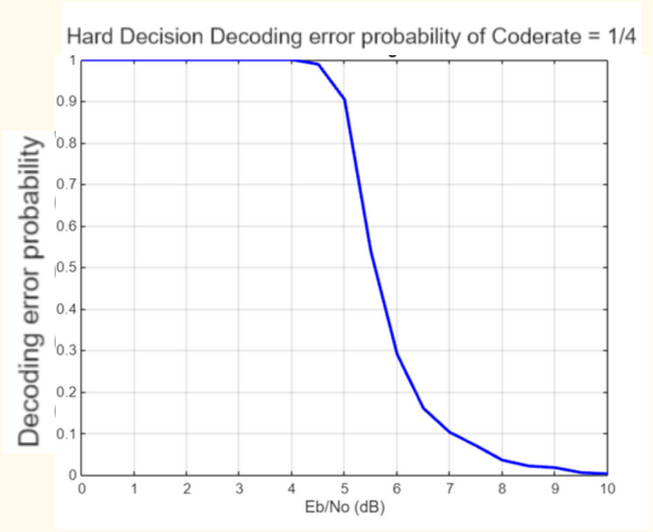
$$L = \begin{bmatrix} -0.7 & -0.6 & 1.3 & 0 & 0.6 & 0 & 0 \\ 0 & 0.1 & 0.8 & -0.3 & 0 & 0.4 & 0 \\ -0.7 & -0.6 & 0 & -0.8 & 0 & 0 & -0.9 \\ -0.4 & 0 & 1.3 & 0 & 0.6 & 0.9 & -0.9 \end{bmatrix}$$

- 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 continues in a loop.

RESULTS OF MATRIX NR_2_6_52

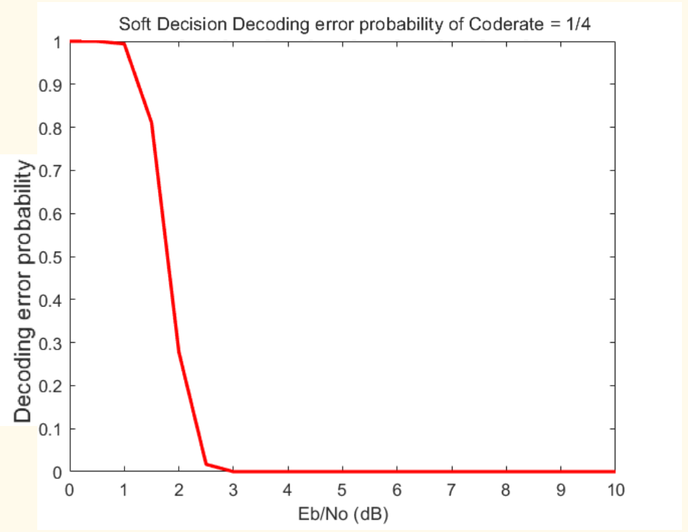
CODE RATE = 1/4

HARD DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 9 for hard decision decoding.

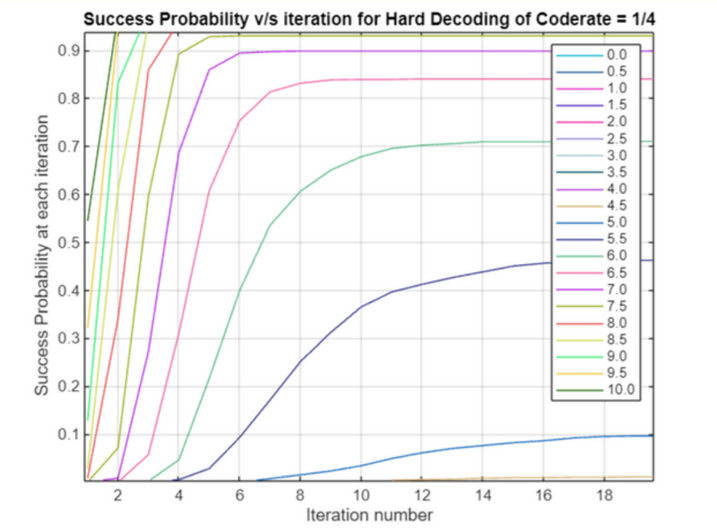
SOFT DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 3 for soft decision decoding

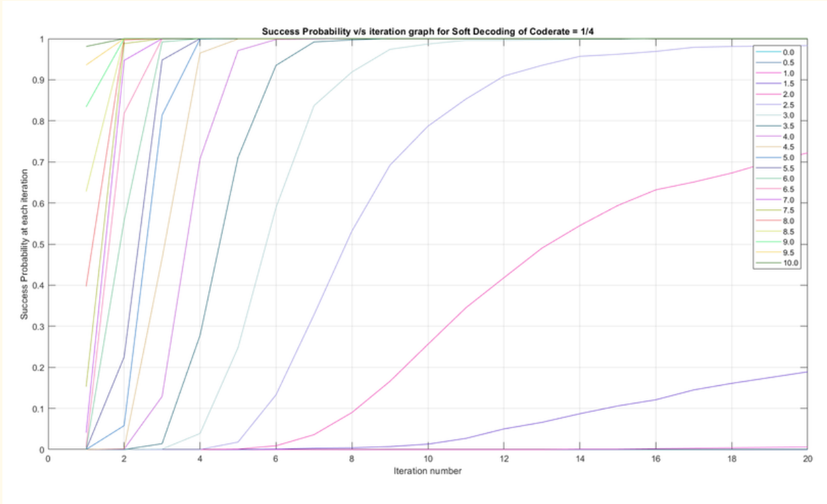
CODE RATE = 1/4

HARD DECISION DECODING



From this graph we can see that for Eb_No>= 9 db, the decoder decodes all messages successfully, though the number of iterations required differ.

SOFT DECISION DECODING

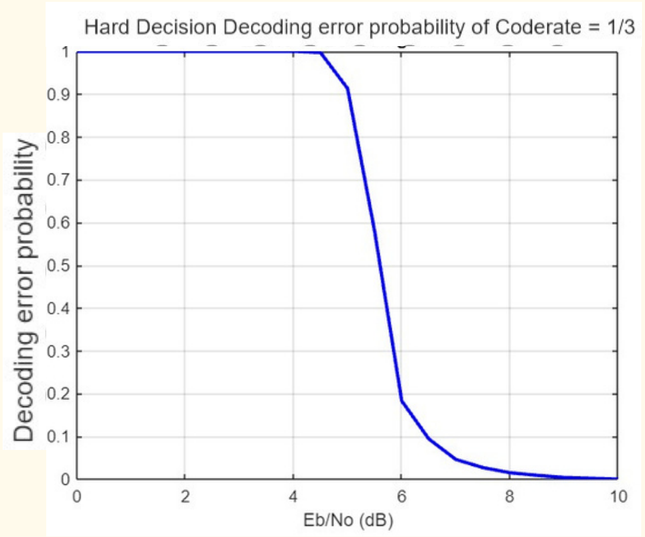


From this graph we can see that for Eb_No>= 3 db, the decoder decodes all messages successfully, though the number of iterations required differ.

When comparing hard and soft decision decoding, we can see that soft decision decoding is better, as for less iteration and less Eb_No value it gives successful results.

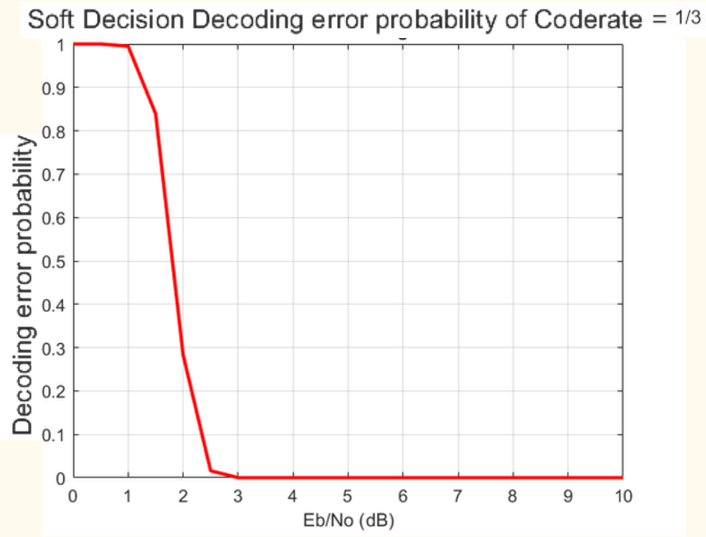
CODE RATE = 1/3

HARD DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 8.5 for hard decision decoding

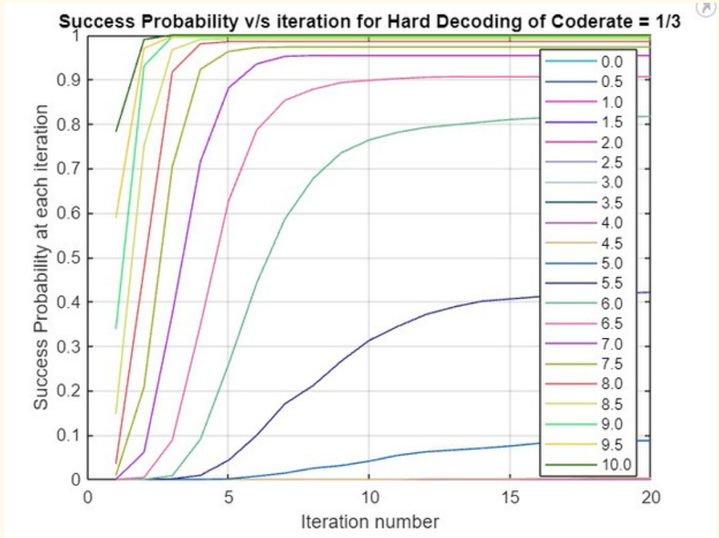
SOFT DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 3 for soft decision decoding.

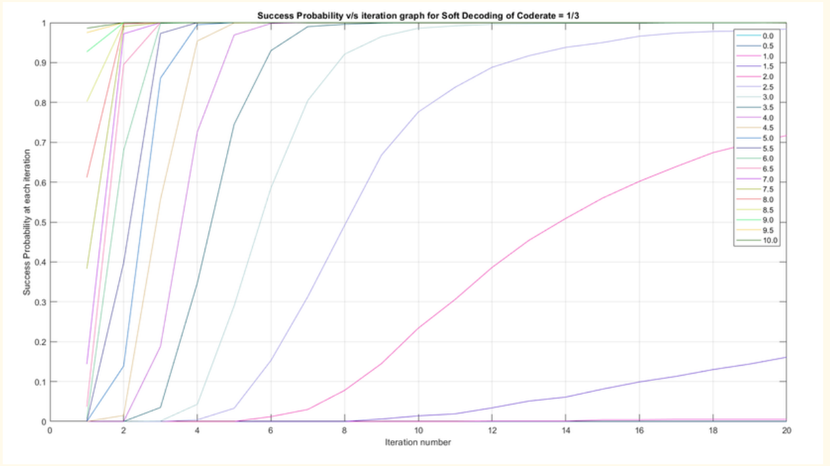
CODE RATE = 1/3

HARD DECISION DECODING



From this graph we can see that for Eb_No>= 8.5 db, the decoder decodes all messages successfully, though the number of iterations required differ.

SOFT DECISION DECODING

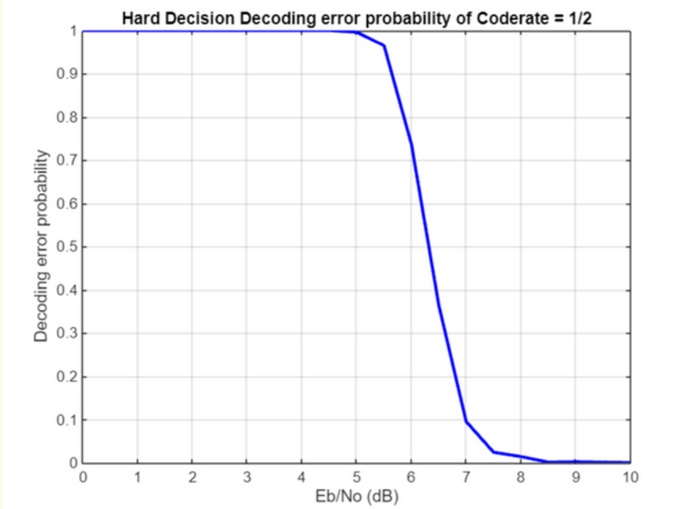


From this graph we can see that for Eb_No>= 3 db, the decoder decodes all messages successfully, though the number of iterations required differ.

When comparing hard and soft decision decoding, we can see that soft decision decoding is better, as for less iteration and less Eb_No value it gives successful results.

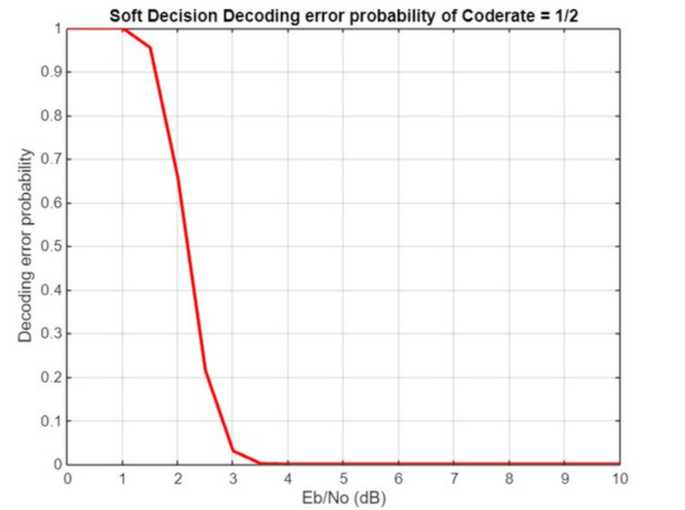
CODE RATE = 1/2

HARD DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 8.5 for hard decision decoding.

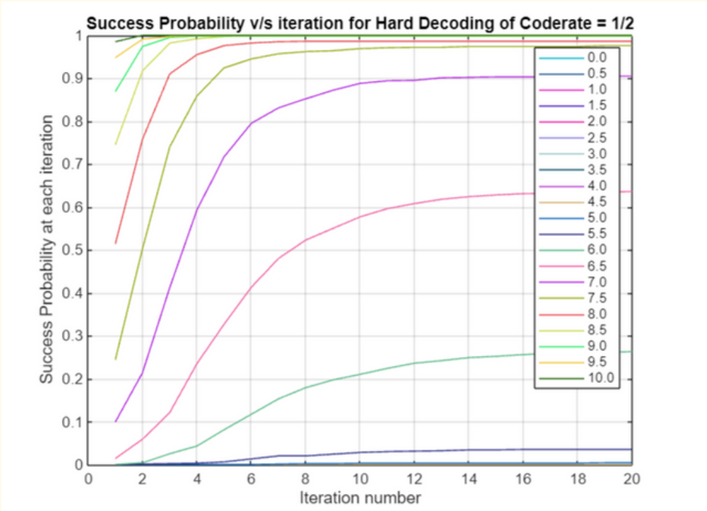
SOFT DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 3.5 for soft decision decoding.

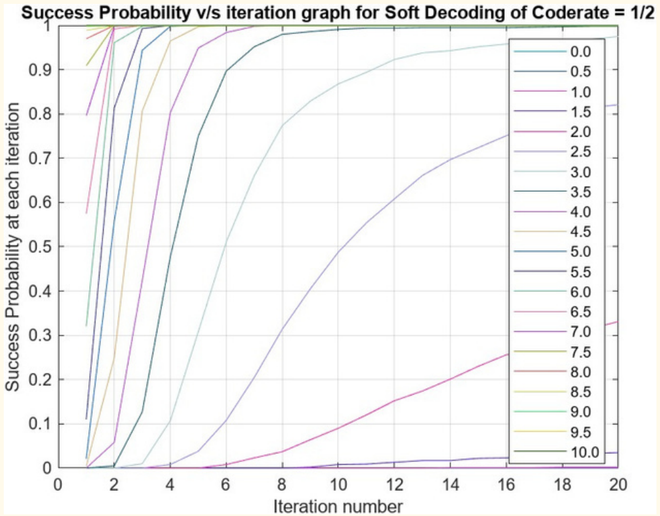
CODE RATE = 1/2

HARD DECISION DECODING



From this graph we can see that for Eb_No>= 8.5 db, the decoder decodes all messages successfully, though the number of iterations required differ.

SOFT DECISION DECODING

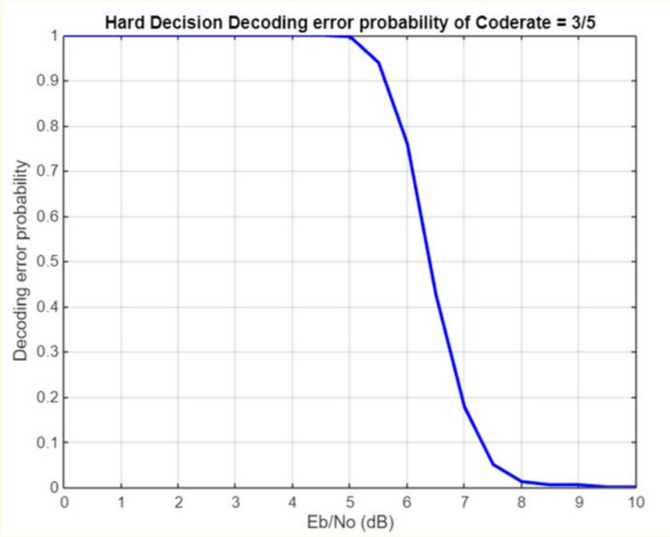


From this graph we can see that for Eb_No>= 3.5 db, the decoder decodes all messages successfully, though the number of iterations required differ.

When comparing hard and soft decision decoding, we can see that soft decision decoding is better, as for less iteration and less Eb_No value it gives successful results.

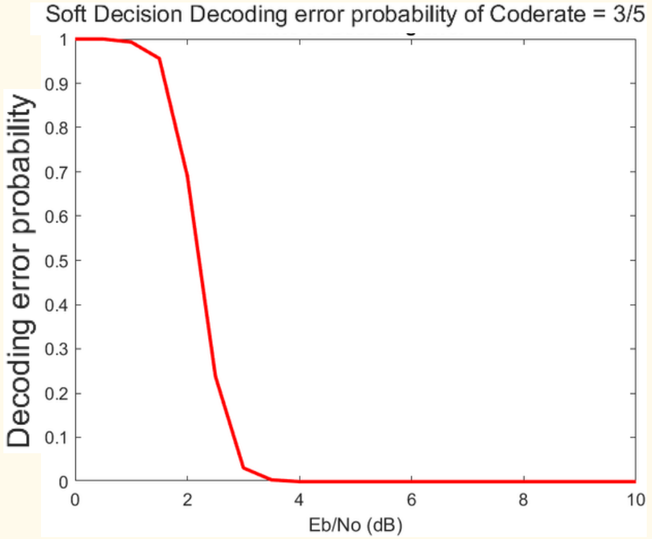
CODE RATE = 3/5

HARD DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 9 for hard decision decoding

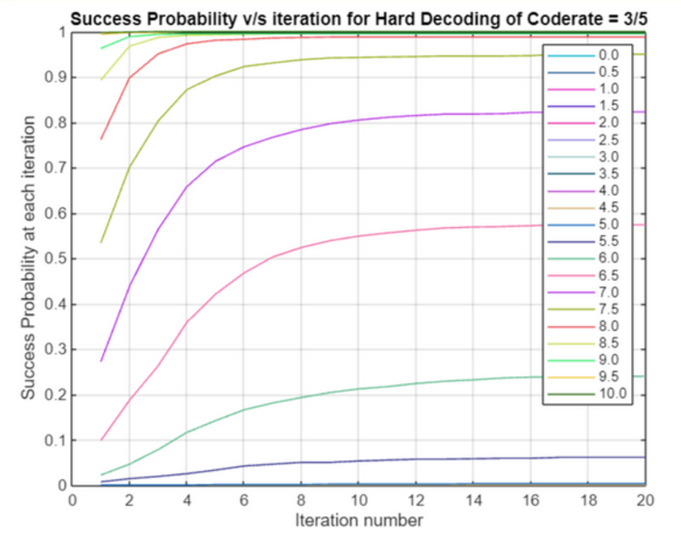
SOFT DECISION DECODING



From this graph we can see that our decoder works perfectly for values of Eb/No greater than 3 for soft decision decoding.

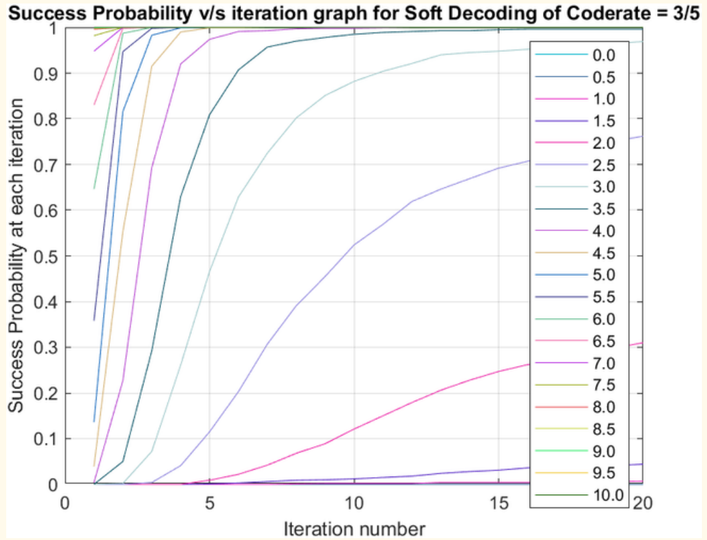
CODE RATE = 3/5

HARD DECISION DECODING



From this graph we can see that for Eb_No>= 9 db, the decoder decodes all messages successfully, though the number of iterations required differ.

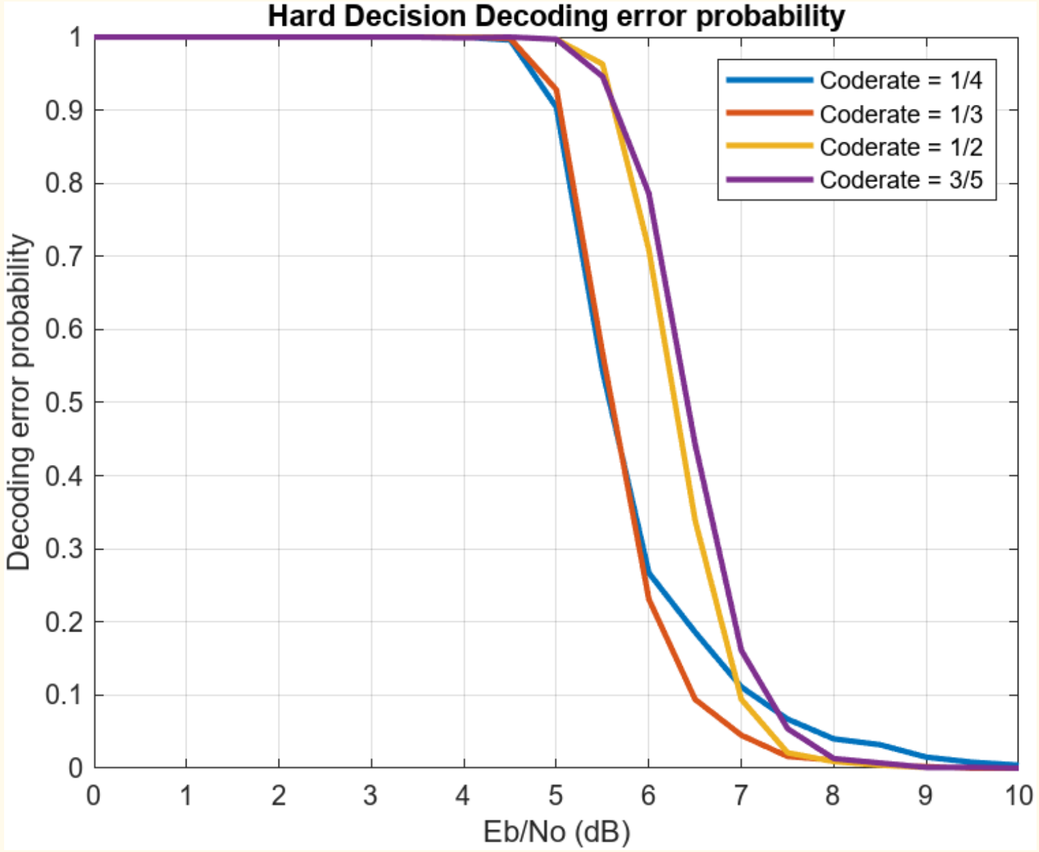
SOFT DECISION DECODING



From this graph we can see that for Eb_No>= 3 db, the decoder decodes all messages successfully, though the number of iterations required differ.

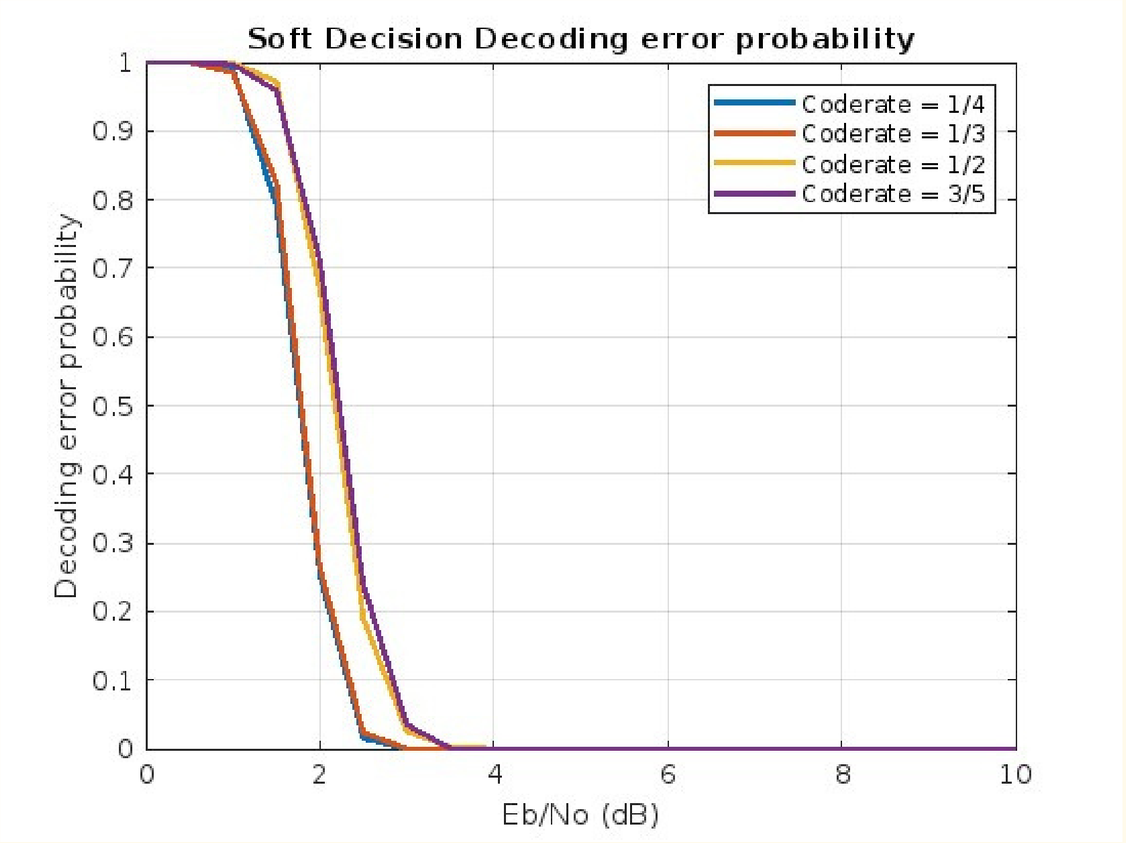
When comparing hard and soft decision decoding, we can see that soft decision decoding is better, as for less iteration and less Eb_No value it gives successful results.

HARD DECISION DECODING



From this graph we can clearly see the comparison between the various code rates. Lower the code rate, more efficient is our channel. This is so because, the number of parity bits increase for lower code rates making decoding more faster and reliable.

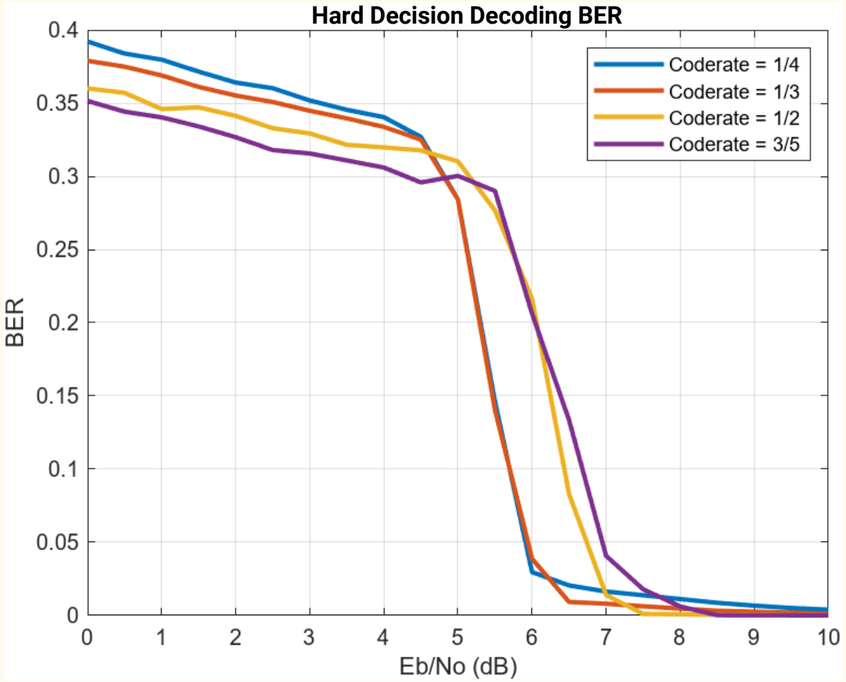
SOFT DECISION DECODING



From this graph we can clearly see the comparison between the various code rates. Lower the code rate, more efficient is our channel. This is so because, the number of parity bits increase for lower code rates making decoding more faster and reliable.

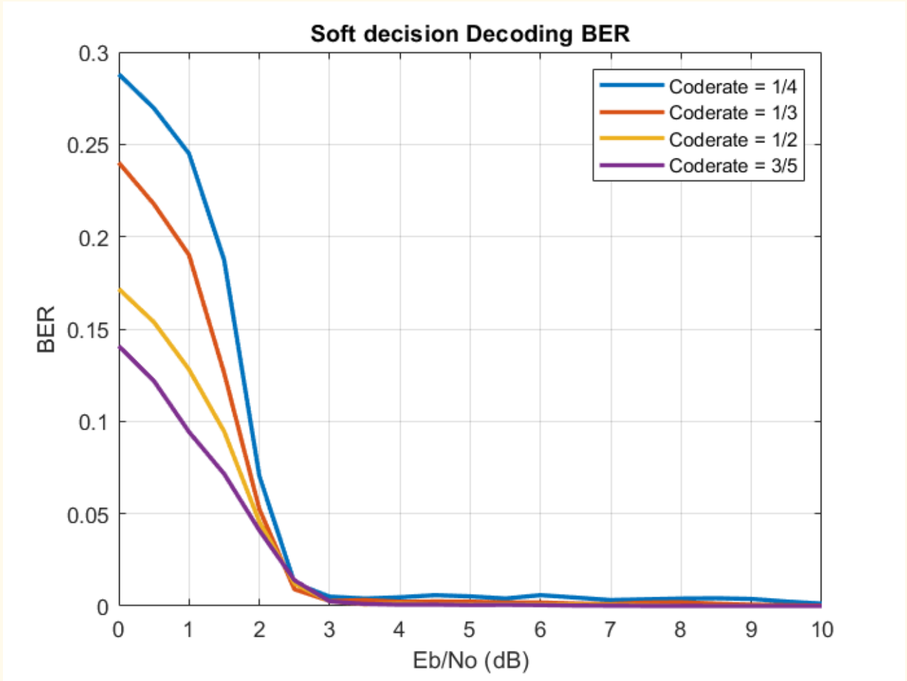
BIT ERROR RATE

HARD DECISION DECODING



The number of bits in error decreases drastically once the value of Eb/No goes above around 5 for hard decision decoder.

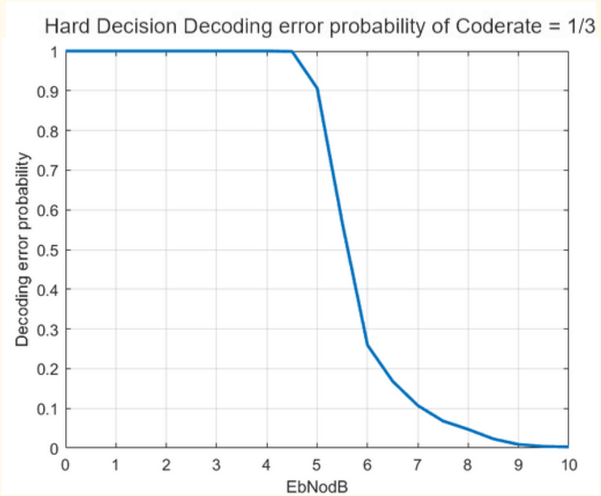
SOFT DECISION DECODING

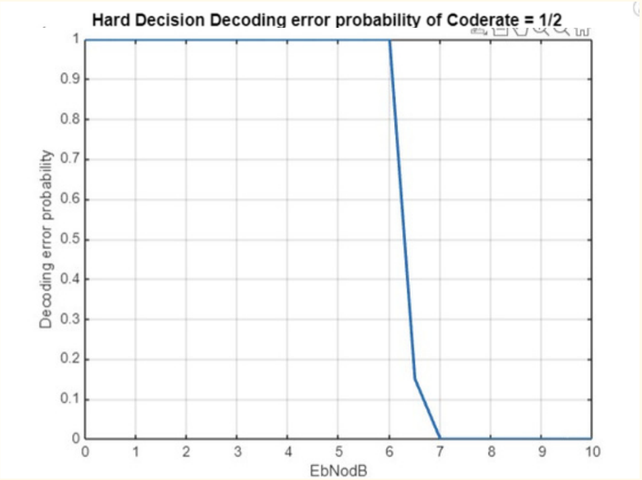


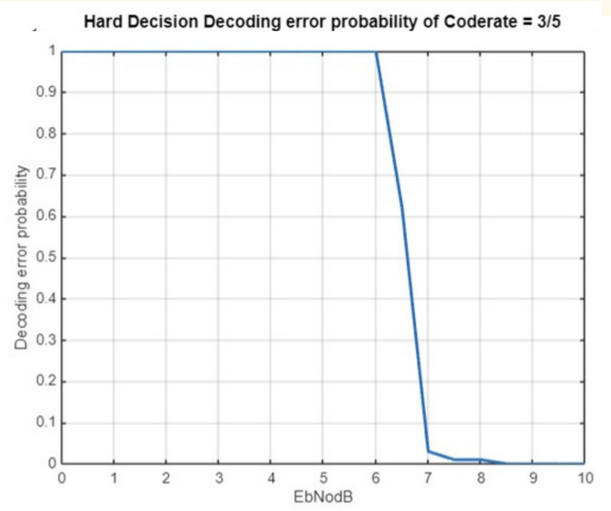
The value of BER decreases at a much lower value of Eb/No approximately 2.5 for every code rate, which suggest that Soft decision decoders work better at lower SNRs.

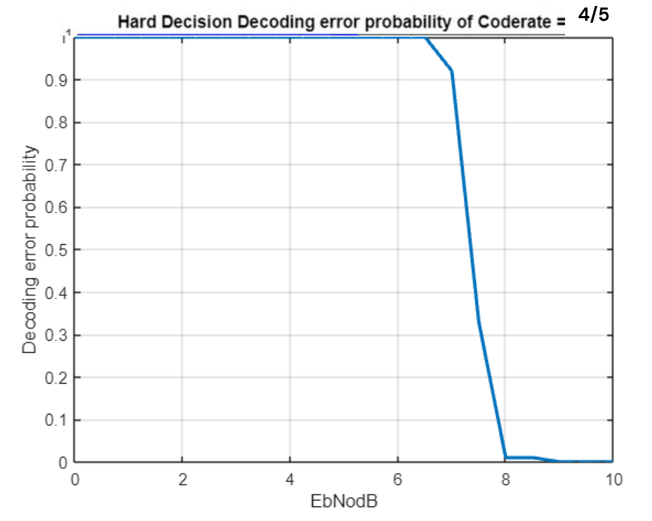
RESULTS OF MATRIX NR_1_5_352

HARD DECISION DECODING



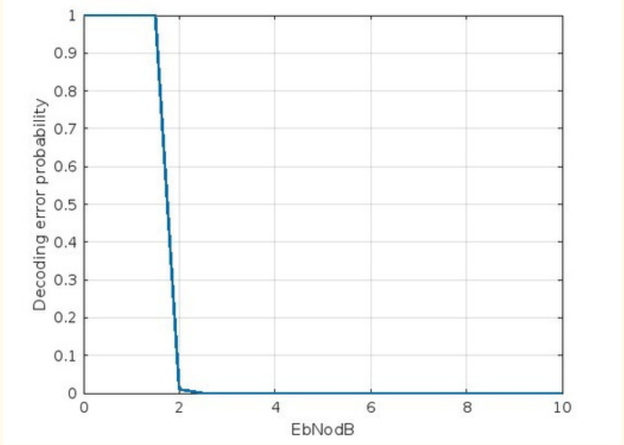




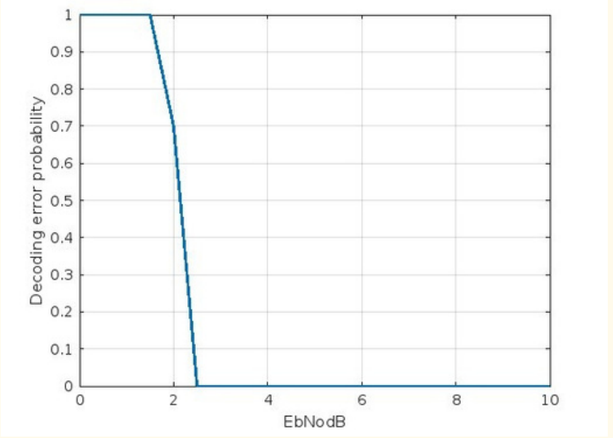


SOFT DECISION DECODING

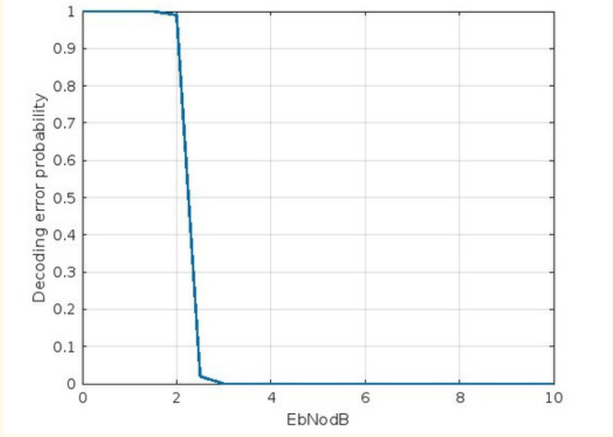
Soft Decision Decoding error probability of Coderate = 1/3



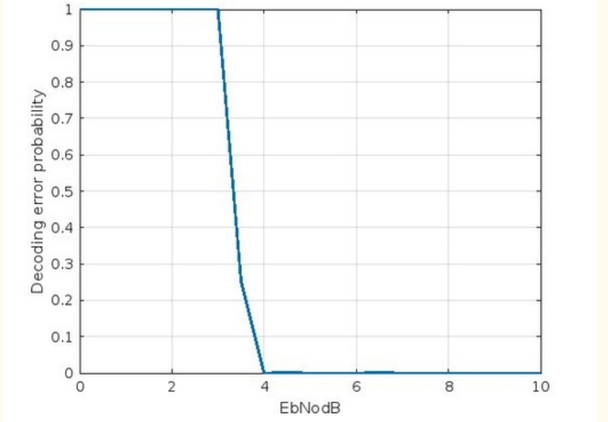
Soft Decision Decoding error probability of Coderate = 1/2



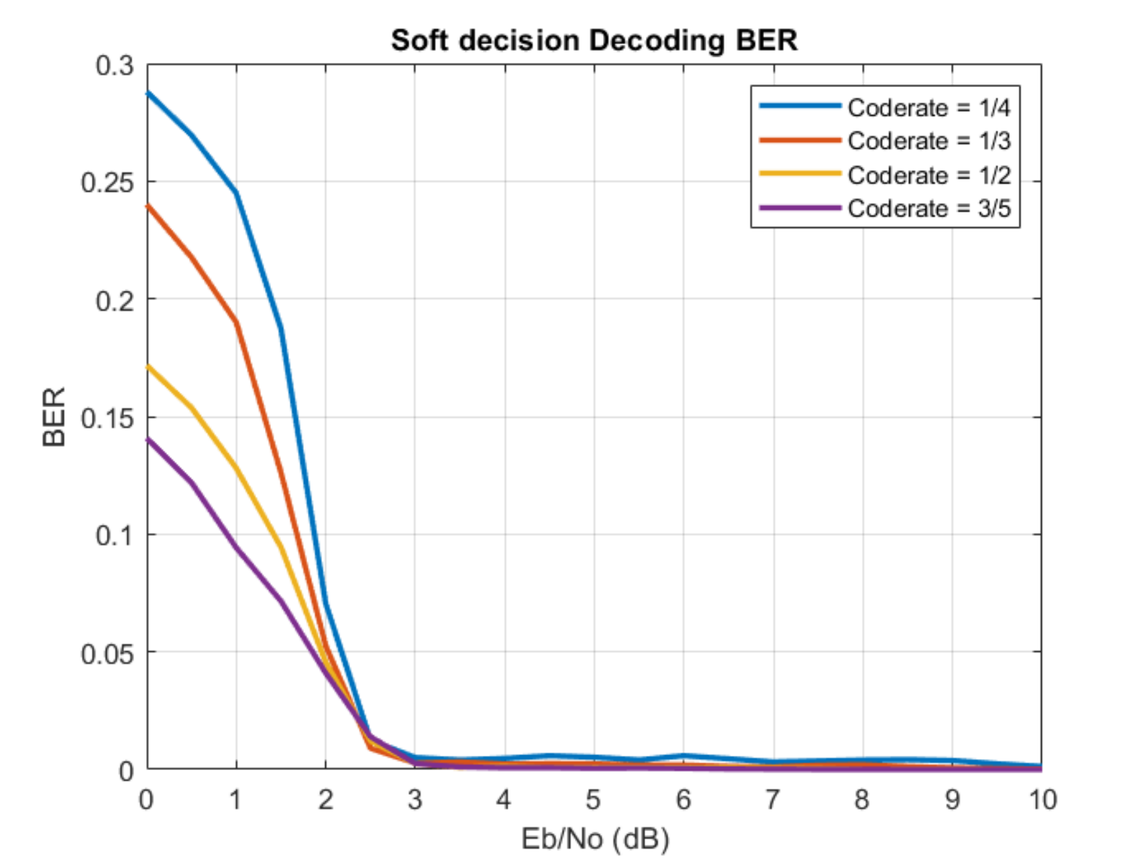
Soft Decision Decoding error probability of Coderate = 3/5



Soft Decision Decoding error probability of Coderate = 4/5



COMPARISON WITH SHANNON'S CHANNEL CAPACITY BOUND



Bit error rate is given by the ratio of number of bits in error and total number of bits. According to the Shannon's bound for Eb/No greater than or equal to 2 the BER is approximately 10^{-5}.

According to our simulations, for soft decision decoding, we found that the BER will drop to such low values only after Eb/No is greater then 2.5, which is compliant with the Shannon's channel capacity bound and is quite close to it. So, we can say that LDPC Codes have a BER quite close to that of Shannon's bound.

CONCLUSION

- From all the results obtained we can conclude that Soft decoding is much more efficient than Hard decoding as it uses the beliefs from other received bits as well.
- As the number of iterations increase, the probability of successful decoding also increases.
- The efficiency also increases with lower code rate as number of parity bits increases, facilitating faster decoding.

REFERENCES

- Lecture Slides of Channel Coding by Professor Yash Vasavada.
- 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 WANG

TEAM MEMBERS

1	202201006	PATEL KAVAN VIJAYBHAI
2	202201015	MEGHAVI GOHIL
3	202201020	SHRUTI RANJIT CHOUDHARY
4	202201024	HITARTH BHATT
5	202201025	PATEL VIVEK SANJAYKUMAR
6	202201026	TANAY JAIN
7	202201029	BHAVYA MUKESHBHAI KANTELIA
8	202201034	HARSHIT KUMAR
9	202201038	MAHAMMED JAMI
10	202201048	JOSHI SNEH VIPULKUMAR
11	202201041	DHRITI M GOENKA

THANK YOU