Dhirubhai Ambani Institute of Information and Communication Technology

Introduction to Communication Systems (CT216)

LDPC Codes

Group - 24

Mentor: Darpan Lungariya
Prof: Yash Vasavada

Honor Code

- We declare that the work presented is our own work.
- We have not copied the work of someone else.
- The concepts, understanding, and insights we will describe are our own.
- Where we have relied on existing work that is not our own, we have provided a proper reference citation.
- We know that a violation of this solemn promise can have grave consequences.

Group Members

1. 202301258 RISHIK YALAMANCHILI
2. 202301259 CHIRAG KATKORIYA
3. 202301260 MAHEK JIKKAR
4. 202301261 PATEL NAKUL JAYMITKUMAR
5. 202301262 PRIYANKA GARG
6. 202301263 YUG SAVALIA
7. 202301264 KRISH PATEL
8. 202301265 JALU RISHABH
9. 202301266 VANSH VORA
10. 202301267 ARAV VAITHA
11. 202301268 SIDDHANT SHEKHAR

Contents

1 Brief about LDPC Codes **4**

2 Encoding **4**
2.1 Importance of Base Matrix and Expansion Factor 4
2.1.1 Base Matrix 4
2.1.2 Expansion Factor 4
2.1.3 Encoding Example with $Z = 5$ 5
2.1.4 Rate Matching 6

3 Brief about BPSK Modulation, AWGN Noise Added in the Channel Encoding **6**

4 Hard Decoding **7**
4.1 Algorithm 7
4.2 Results 7
4.2.1 Error Probability of different code rates 7
4.2.2 Hard Decision Decoding Error Probability for Different Eb/No and Code Rates in Log Scale 8
4.3 Problems with Hard Decoding 9

5 Soft Decoding **9**
5.1 Algorithm (Min-Sum) 10
5.2 Steps of the Min-Sum Algorithm 10
5.2.1 Initialization 10
5.2.2 Check Node Update 10
5.2.3 Variable Node Update 10
5.2.4 Decision 11
5.3 Channel LLR (Intrinsic LLR) 11
5.4 Extrinsic LLR in Min-Sum Algorithm 12
5.4.1 Derivation of Min-Sum Approximation 12
5.5 Results 14
5.5.1 Success Probability vs Iteration Number 14
5.5.2 Success Probability vs Eb/No for Different Code Rates 15
5.5.3 BER Performance with Normal Approximation and Shannon Limits in log scale 16
5.6 An Improved Algorithm - Sum-Product Algorithm 17
5.6.1 Initialization 17
5.6.2 Check Node Update 17
5.6.3 Variable Node Update 18
5.6.4 Decision 18
5.7 Results for sum-product and comparison with min-sum 18
5.8 Conclusion And Possible Explanation of Unexpected behaviour Of NA . . . 19

1. Brief about LDPC Codes

LDPC codes were originally conceived by Robert G. Gallager (and are thus also known as Gallager codes). These codes are a class of linear block codes known for their excellent error-correcting capabilities and near-capacity performance. They have a sparse parity-check matrix — a matrix mostly filled with zeros — which allows for efficient decoding using iterative algorithms like belief propagation.

Key properties:

- **High performance near Shannon limit:** LDPC codes can operate very close to the theoretical maximum data rate (capacity) of a noisy channel.
- **Iterative decoding:** Uses message-passing between variable nodes and check nodes in a bipartite Tanner graph, allowing rapid convergence to the correct message.
- **Scalability:** They support flexible code lengths and rates, which is why they're used in 5G NR.

2. Encoding

2.1 Importance of Base Matrix and Expansion Factor

2.1.1 Base Matrix

The base matrix is a small, abstract representation of the LDPC code's structure. It is typically a matrix of integers where:

- A value of -1 indicates a zero submatrix (no connection).
- A non-negative integer k represents a circulant permutation matrix that shifts an identity matrix by k positions.

This base matrix defines the connectivity pattern between variable nodes and check nodes, and it's the blueprint used for generating the full-size parity-check matrix. For example, in 5G NR, there are two standardized base matrices that we have used in the project:

- Base Graph 1 (BG1) for larger block sizes and higher code rates (46×68).
- Base Graph 2 (BG2) for smaller block sizes and lower code rates (42×52).

2.1.2 Expansion Factor

Expansion factor (sometimes called the lifting factor) is the number by which we "blow up" the Base Matrix to get the real LDPC code's parity-check matrix. If the Expansion Factor is Z, each entry in the Base Matrix becomes a $Z \times Z$ submatrix:

- A "-1" turns into a $Z \times Z$ all-zero block.

- A "0" becomes a single permutation of the $Z \times Z$ identity matrix (one copy of each connection).
- A higher integer, say "r," becomes a $Z \times Z$ identity matrix with "r" columns shifted.

Thus, the full parity-check matrix H has a size of $(m \times Z) \times (n \times Z)$ if the base matrix is of size $m \times n$. For encoding, the E block in the Base Matrix is extremely important, which has a double diagonal structure. We can easily determine the parity bits based on the double diagonal structure by calculating simple equations and then back substituting.

2.1.3 Encoding Example with $Z = 5$

To illustrate the encoding process, consider a base matrix expanded with an expansion factor $Z = 5$. Let the message be $m = [m_1, m_2, m_3, m_4]$, where each m_i has 5 bits. The codeword is $c = [m_1, m_2, m_3, m_4, p_1, p_2, p_3, p_4]$, where p_1, p_2, p_3, p_4 are the parity bits to be determined, each also having 5 bits. The parity-check matrix H is given by:

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

where I_k denotes a 5×5 identity matrix shifted right by k positions, and I is the unshifted 5×5 identity matrix.

The parity-check equation is $H[m_1, m_2, m_3, m_4, p_1, p_2, p_3, p_4] = 0$, which gives the following system of equations:

$$I_1 m_1 + I_2 m_3 + I_1 m_4 + I_2 p_1 + I p_2 = 0 \quad (1)$$

$$I_2 m_1 + I m_2 + I_3 m_4 + I p_2 + I p_3 = 0 \quad (2)$$

$$I_4 m_2 + I_2 m_3 + I m_4 + I_1 p_1 + I p_3 + I p_4 = 0 \quad (3)$$

$$I_4 m_1 + I_1 m_2 + I m_3 + I_2 p_1 + I p_4 = 0 \quad (4)$$

Adding all four equations, we obtain:

$$I_1 p_1 = I_1 m_1 + I_2 m_3 + I_1 m_4 + I_2 m_1 + I m_2 + I_3 m_4 + I_4 m_2 + I_2 m_3 + I m_4 + I_4 m_1 + I_1 m_2 + I m_3$$

From this, p_1 can be computed. Then:

- Use p_1 in Equation (1) to find p_2.
- Use p_2 in Equation (2) to find p_3.
- Use p_3 in Equation (3) to find p_4.

For larger matrices with an E block, the remaining parity bits can be determined using the parity bits found from the double diagonal structure.

2.1.4 Rate Matching

Rate matching adjusts the LDPC code rate to fit channel constraints using techniques like puncturing:

- **Purpose**: Matches the encoded data to the channel's fixed resources by altering the code rate.
- **Puncturing**: Removes specific bits (usually parity bits) from the codeword to increase the code rate.
- **Puncturing Process**: Bits are systematically omitted (e.g., from the end or per a pattern); at the receiver, these bits are treated as erasures (set to 0 for decoding).
- **Example (BG1, rate** $1/3$ **to** $1/2$**)**:
  - BG1 has $22Z$ message bits, $44Z$ parity bits (excluding first $2Z$ bits), total $66Z$ bits.
  - Target rate 1/2: Transmit $44Z$ bits ($22Z$ message + $22Z$ parity), puncturing $22Z$ parity bits.
  - Resulting H matrix: Reduced to $24Z \times 46Z$ (24 check nodes, 46 variable nodes).
- **Impact**: Puncturing reduces redundancy, which may affect error correction if not carefully designed.

3. Brief about BPSK Modulation, AWGN Noise Added in the Channel Encoding

BPSK Modulation converts each bit into a carrier wave with one of two phases:

- Bit $0 \rightarrow +1$ (or $0°$ phase) using the formula of $(1 - 2 \times c)$ where c is the codeword.
- Bit $1 \rightarrow -1$ (or $180°$ phase).

The modulated signal is then transmitted over a noisy channel—typically modeled as an AWGN (Additive White Gaussian Noise) channel.

AWGN Channel: This channel adds Gaussian noise to the transmitted signal. So if a symbol $x \in \{+1, -1\}$ is transmitted, the received signal is:

$$y = x + n$$

where n is a random value drawn from a normal distribution with mean zero and variance σ^2. The noise is independent across symbols.

Demodulation and Decoding: The receiver uses coherent detection, comparing the noisy value y to 0:

- $y > 0 \rightarrow$ decide bit 0.
- $y < 0 \rightarrow$ decide bit 1.

The detected bitstream is passed through a channel decoder to correct errors using the added redundancy.

4. Hard Decoding

In this, we use hard thresholding with a boundary decision at $r = 0$, and then pass the vector for hard decoding which follows the VN to CN message-passing algorithm. Since it only deals with binary values, hard decoding is computationally simpler and faster, but it sacrifices performance because of simplicity. Because it ignores confidence levels (e.g., how close a signal is to 1 or 0), hard decoding generally results in a higher bit error rate (BER) compared to soft decoding, especially in noisy channels.

In the first iteration, we directly pass the values from Variable Nodes (VNs) to Check Nodes (CNs), and then for passing the message back to VNs from CNs, parity is checked at each CN, excluding the bit of the VN to which we will send the bit, to avoid a positive feedback loop. Then at each VN, if it is connected to 5 CNs for example, it gets 5 different bits from 5 CNs for a single bit, and hence it is like repetition coding, and thus a majority voting method is used at the VN to pass the message back to the CN.

The iterations happen until we get the original message back, or if we get the same codeword in two successive iterations, then we break out of the loop.

4.1 Algorithm

The hard decoding algorithm follows these steps:

- Initial values are passed from Variable Nodes (VNs) to Check Nodes (CNs).
- At each CN, parity is checked excluding the VN it's responding to (to avoid feedback).
- The CN sends messages back to VNs.
- At each VN, messages from all connected CNs are combined using majority voting.

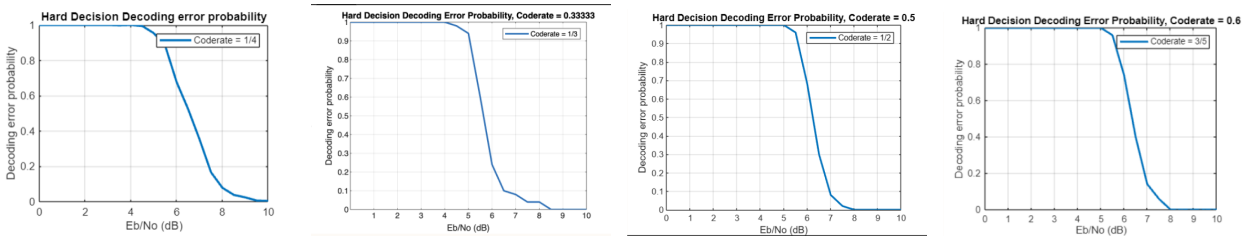
The iterations continue until:

- The decoded word matches the original message, or
- The codeword remains unchanged across two iterations.

4.2 Results

4.2.1 Error Probability of different code rates

From the simulation results, we observe that as E_b/N_0 increases, the error probability decreases for all code rates. This trend indicates improved performance of the LDPC code under better channel conditions, as higher E_b/N_0 values correspond to a lower noise level, facilitating more reliable decoding.



(a) Rate - 1/4 (b) Rate - 1/3 (c) Rate - 1/2 (d) Rate - 3/5

Figure 1: Decoding error probability vs E_b/N_0.

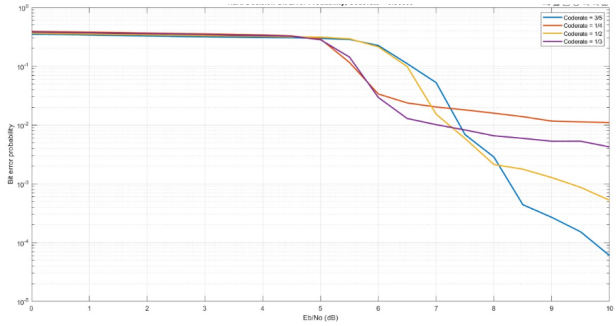


Figure 2: Bit error probability vs. E_b/N_0 for different code rates in log scale.

4.2.2 Hard Decision Decoding Error Probability for Different Eb/No and Code Rates in Log Scale

From the plot, we observe that as E_b/N_0 increases, the bit error probability decreases for all code rates, reflecting improved decoding performance under better channel conditions. Additionally, lower code rates (e.g., 1/4) exhibit significantly lower error probabilities compared to higher code rates (e.g., 4/5) at the same E_b/N_0, due to the increased redundancy providing stronger error correction capabilities. This reults plotted in log scale provides a more precise explanation of the situation reflected in the previous results.

4.3 Problems with Hard Decoding

Hard Decision decoding relies on an absolute threshold of received bits, which can lead to errors. For example, suppose we want to send a message $m = [1\,0\,1\,0\,1]$. According to the modulation implemented during encoding:

- 0 maps to +1.
- 1 maps to -1.

Let us suppose that the bits received are $r = s + n = [0.2\,0.5\,-1.3\,-0.1\,-0.3]$. After implementing Hard Decoding, we would decode the received bits as $[0\,0\,1\,1\,1]$. As visible, there are errors at 2 places.

Hard Decision decoding ignores the confidence levels of the received signals (e.g., a received value of -1.3 is a stronger indication of 1 than -0.1). It relies on minimum Hamming distance, which is less effective than Euclidean distance used in soft decoding, leading to a higher error rate.

5. Soft Decoding

Soft Decision Decoding relies on the actual received values to make the decoding more accurate and effective, using log likelihood ratios (LLRs) which reflect the belief that a bit is 0 or 1.

Each VN is initially assigned a channel Log-Likelihood Ratio (LLR), which is the likelihood of a received bit being a 0 or a 1. These LLRs are the initial messages sent from VNs to CNs.

In the first iteration, each VN sends its channel LLR to all the CNs it is connected to. In subsequent iterations, a VN sends a new message to each CN, which is the sum of its channel LLR and the incoming messages from all other CNs (excluding the CN it's currently sending a message to).

Each CN then performs Single Parity Check (SPC) Soft-Input Soft-Output (SISO) decoding using the messages received from the VNs. To update the message that a CN sends to a VN, the tanh rule is used (often approximated using the min-sum rule for efficiency). Specifically, for each CN, the sign of the message is computed as the product of the signs of all incoming messages in that row. The magnitude is the minimum of the absolute values of the incoming messages, excluding the one being sent back.

After CNs send these updated messages, each VN updates its belief by summing its original channel LLR with all the incoming messages from CNs. This process is repeated over several iterations.

Finally, once decoding has converged or a set number of iterations is reached, the final decision for each bit is made. For each VN, the final value is calculated by summing its channel LLR with all incoming CN messages. If the result is greater than zero, the decoded bit is assumed to be 0; otherwise, it is 1.

5.1 Algorithm (Min-Sum)

The min-sum algorithm is an efficient method used to decode Low-Density Parity-Check (LDPC) codes, which are error-correcting codes widely applied in communication systems, including 5G New Radio (NR). This algorithm operates by passing messages between nodes in a Tanner graph—a diagram where variable nodes represent the bits of the codeword and check nodes represent the parity-check constraints. Through iterative message passing, the algorithm determines the most likely values of the transmitted bits based on the received data. The min-sum algorithm is particularly suitable for hardware implementation due to its low computational complexity, achieved by simplifying the check node update operation compared to more complex decoding methods.

5.2 Steps of the Min-Sum Algorithm

The min-sum algorithm follows these steps to decode the LDPC code:

5.2.1 Initialization

- Each variable node, corresponding to a bit in the codeword and indexed by i, receives a log-likelihood ratio (LLR) from the communication channel, known as the intrinsic LLR. This LLR, denoted $L_i^{(0)}$, indicates the initial probability of the bit being 0 or 1.
- The message from each variable node i to each connected check node j, denoted $L_{i\to j}$, is initialized to the channel LLR: $L_{i\to j} = L_i^{(0)}$.

5.2.2 Check Node Update

- For each check node j, the algorithm computes the message to send to each connected variable node i, denoted $L_{j\to i}$, known as the extrinsic LLR. This message is based on the messages received from all other variable nodes connected to j, excluding i. The formula is:

$$L_{j\to i} = \left(\prod_{i' \in \mathcal{N}(j)\setminus\{i\}} \operatorname{sign}(L_{i'\to j}) \right) \cdot \min_{i' \in \mathcal{N}(j)\setminus\{i\}} |L_{i'\to j}|$$

  Here, $\mathcal{N}(j)$ is the set of variable nodes connected to check node j, and $\setminus\{i\}$ indicates that variable node i is excluded. The sign product determines the overall sign of the message, while the minimum magnitude approximates the reliability of the parity-check constraint.

5.2.3 Variable Node Update

- For each variable node i, the algorithm computes the message to send to each connected check node j, denoted $L_{i\to j}$. This message is the sum of the channel LLR and the messages from all other check nodes connected to i, excluding j. In the min-sum algorithm, the LLRs are simply added:

$$L_{i\to j} = L_i^{(0)} + \sum_{j' \in \mathcal{M}(i)\setminus\{j\}} L_{j'\to i}$$

Here, $\mathcal{M}(i)$ is the set of check nodes connected to variable node i.

5.2.4 Decision

- After several iterations or when a stopping condition (e.g., all parity checks satisfied or maximum iterations reached) is met, the algorithm calculates the total LLR for each variable node i:
$$L_i = L_i^{(0)} + \sum_{j \in \mathcal{M}(i)} L_{j \to i}$$
- The decoded bit value is then determined:
  - If $L_i \geq 0$, the bit is decoded as 0.
  - If $L_i < 0$, the bit is decoded as 1.

5.3 Channel LLR (Intrinsic LLR)

The channel LLR, or intrinsic LLR, $L_i^{(0)}$, represents the initial reliability of bit i based on the received signal from the channel. For Binary Phase-Shift Keying (BPSK) modulation over an Additive White Gaussian Noise (AWGN) channel, the LLR is derived as follows:

- The transmitted bit $c_i \in \{0, 1\}$ is mapped to $x_i = 1 - 2c_i$, so $c_i = 0 \to x_i = +1$, $c_i = 1 \to x_i = -1$.
- The received signal is $y_i = x_i + n_i$, where $n_i \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise with variance σ^2.
- The LLR is:
$$L_i^{(0)} = \log\left(\frac{P(c_i = 0|y_i)}{P(c_i = 1|y_i)}\right)$$
- The conditional probabilities are:
$$P(y_i|c_i = 0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - 1)^2}{2\sigma^2}\right), \quad P(y_i|c_i = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i + 1)^2}{2\sigma^2}\right)$$
- Thus:
$$L_i^{(0)} = \log\left(\frac{\exp\left(-\frac{(y_i-1)^2}{2\sigma^2}\right)}{\exp\left(-\frac{(y_i+1)^2}{2\sigma^2}\right)}\right) = \frac{-(y_i - 1)^2 + (y_i + 1)^2}{2\sigma^2}$$
$$= \frac{-y_i^2 + 2y_i - 1 + y_i^2 + 2y_i + 1}{2\sigma^2} = \frac{4y_i}{2\sigma^2} = \frac{2y_i}{\sigma^2}$$
- The intrinsic LLR $L_i^{(0)} = \frac{2y_i}{\sigma^2}$ quantifies the likelihood that bit c_i is 0 versus 1 based on the channel observation y_i. A positive value suggests $c_i = 0$, and a negative value suggests $c_i = 1$.

5.4 Extrinsic LLR in Min-Sum Algorithm

The extrinsic LLR, $L_{j\to i}$, is the message computed by check node j for variable node i, representing information about bit i derived from other variable nodes connected to j. In the min-sum algorithm, it is computed using the sign product and minimum magnitude of incoming messages.

5.4.1 Derivation of Min-Sum Approximation

- Consider a check node j connected to variable nodes $i_1, i_2, \ldots, i_k \in \mathcal{N}(j)$. We compute $L_{j\to i_1}$, the message to variable node i_1, based on messages from $i_2, i_3, \ldots, i_k$.
- Define $P_{i'} = \log\left(\frac{P(c_{i'}=0|r_{i'})}{P(c_{i'}=1|r_{i'})}\right) = L_{i'\to j}$ for each variable node $i' \in \mathcal{N}(j)$. Here, $r_{i'}$ is the received signal for bit i'.
- The parity-check constraint requires $c_{i_1} \oplus c_{i_2} \oplus \cdots \oplus c_{i_k} = 0$. Thus:

  $$P(c_{i_1}=0|r_2,r_3,\ldots,r_k) = P(c_{i_2}\oplus\cdots\oplus c_{i_k}=0) = P_{i_2}P_{i_3}\cdots P_{i_k} + (1-P_{i_2})(1-P_{i_3})\cdots(1-P_{i_k})$$

  $$P(c_{i_1}=1|r_2,r_3,\ldots,r_k) = P(c_{i_2}\oplus\cdots\oplus c_{i_k}=1) = P_{i_2}(1-P_{i_3})\cdots(1-P_{i_k}) + (1-P_{i_2})P_{i_3}\cdots P_{i_k} + \cdots$$

- The extrinsic LLR is:

  $$L_{j\to i_1} = \log\left(\frac{P(c_{i_1}=0|r_2,r_3,\ldots,r_k)}{P(c_{i_1}=1|r_2,r_3,\ldots,r_k)}\right) = \log\left(\frac{P_{i_2}P_{i_3}\cdots P_{i_k} + (1-P_{i_2})(1-P_{i_3})\cdots(1-P_{i_k})}{1-[P_{i_2}P_{i_3}\cdots P_{i_k} + (1-P_{i_2})(1-P_{i_3})\cdots(1-P_{i_k})]}\right)$$

- Simplify using the transformation $P_{i'} = \frac{1}{1+e^{-L_{i'\to j}}}$, so $1-P_{i'} = \frac{e^{-L_{i'\to j}}}{1+e^{-L_{i'\to j}}}$. Thus:

  $$\frac{P_{i'}}{1-P_{i'}} = e^{L_{i'\to j}}, \quad \frac{1-P_{i'}}{P_{i'}} = e^{-L_{i'\to j}}$$

- This leads to:

  $$\frac{P_{i'}}{1-P_{i'}} = \tanh\left(\frac{L_{i'\to j}}{2}\right), \quad \log\left(\tanh\left(\frac{L_{i'\to j}}{2}\right)\right) = \log\left(\frac{1-e^{-L_{i'\to j}}}{1+e^{-L_{i'\to j}}}\right)$$

- For $k=3$ (variable nodes i_2, i_3):

  $$L_{j\to i_1} = \log\left(\frac{\frac{1-e^{-L_{i_2\to j}}}{1+e^{-L_{i_2\to j}}}\cdot\frac{1-e^{-L_{i_3\to j}}}{1+e^{-L_{i_3\to j}}}}{1-\frac{1-e^{-L_{i_2\to j}}}{1+e^{-L_{i_2\to j}}}\cdot\frac{1-e^{-L_{i_3\to j}}}{1+e^{-L_{i_3\to j}}}}\right)$$

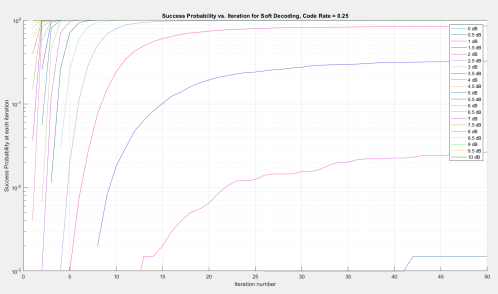
- Define $f(x) = \log\left(\tanh\left(\frac{|x|}{2}\right)\right)$. Then:

  $$\log\left(\tanh\left(\frac{|L_{j\to i_1}|}{2}\right)\right) = \log\left(\tanh\left(\frac{|L_{i_2\to j}|}{2}\right)\right) + \log\left(\tanh\left(\frac{|L_{i_3\to j}|}{2}\right)\right)$$

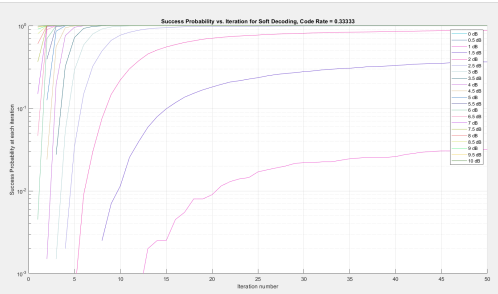
- The sign is:
$$\text{sign}(L_{j\to i_1}) = \text{sign}(L_{i_2\to j}) \cdot \text{sign}(L_{i_3\to j})$$
since tanh is an odd function.
- The function f is self-inverse (i.e., $f(f(x)) = x$), which can be shown by substituting $y = \tanh\left(\frac{x}{2}\right)$, so $f(x) = \log(y)$, and $f(\log(y)) = \log(\tanh(y/2))$, which simplifies back to x.
- Thus:
$$|L_{j\to i_1}| = f\left(f(|L_{i_2\to j}|) + f(|L_{i_3\to j}|)\right)$$
- It can be proven that:
$$f(|L_{i_2\to j}|) + f(|L_{i_3\to j}|) \approx f(\min(|L_{i_2\to j}|, |L_{i_3\to j}|))$$
because f is a decreasing function, and the sum is dominated by the smaller term.
- Therefore:
$$|L_{j\to i_1}| \approx f\left(f(\min(|L_{i_2\to j}|, |L_{i_3\to j}|))\right) = \min(|L_{i_2\to j}|, |L_{i_3\to j}|)$$
- Combining the sign and magnitude:
$$L_{j\to i_1} \approx (\text{sign}(L_{i_2\to j}) \cdot \text{sign}(L_{i_3\to j})) \cdot \min(|L_{i_2\to j}|, |L_{i_3\to j}|)$$
- Generalizing to all $i' \in \mathcal{N}(j) \setminus \{i\}$, we obtain the min-sum update formula:
$$L_{j\to i} = \left(\prod_{i'\in\mathcal{N}(j)\setminus\{i\}} \text{sign}(L_{i'\to j})\right) \cdot \min_{i'\in\mathcal{N}(j)\setminus\{i\}} |L_{i'\to j}|$$

5.5 Results

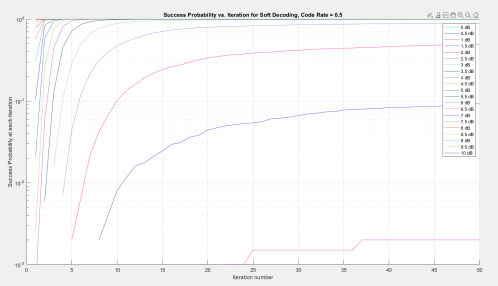
5.5.1 Success Probability vs Iteration Number



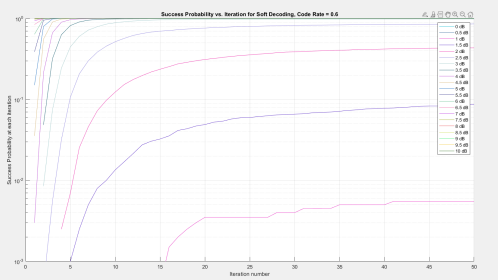
(a) Rate - 1/4



(b) Rate - 1/3



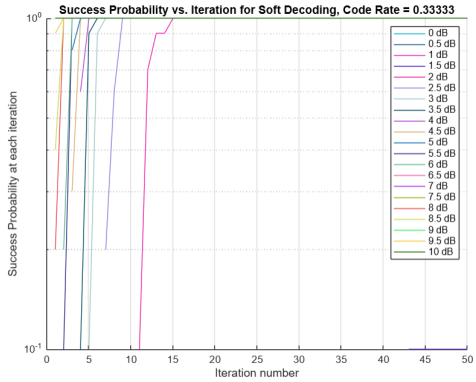
(c) Rate - 1/2



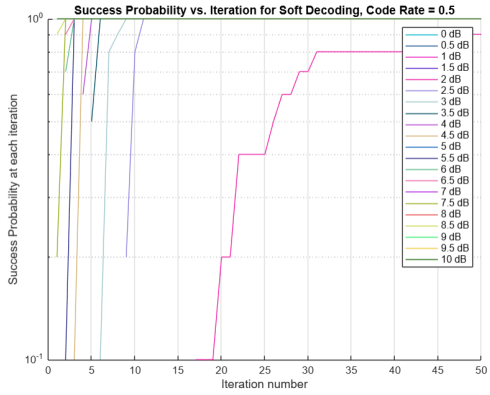
(d) Rate - 3/5

Figure 3: Success probability vs iteration number for NR_2_6_52.

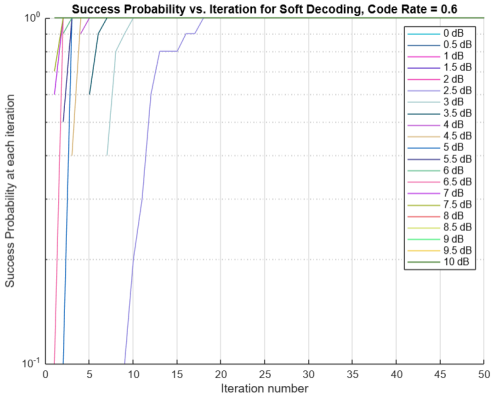
The graphs show that as we increase iterations, the chance of successful decoding changes with E_b/N_0. At higher E_b/N_0, success happens sooner because the signal is stronger than the noise. At lower E_b/N_0, success often doesn't happen, meaning the signal is too weak compared to the noise for proper decoding.



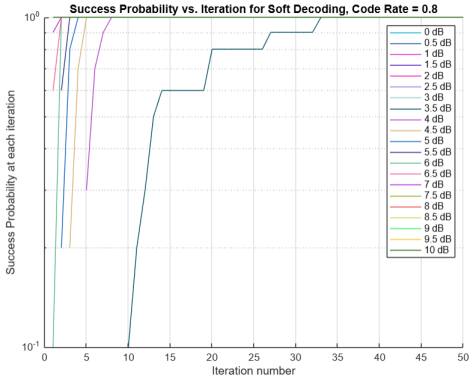
(a) Rate - 1/3



(b) Rate - 1/2



(c) Rate - 3/5

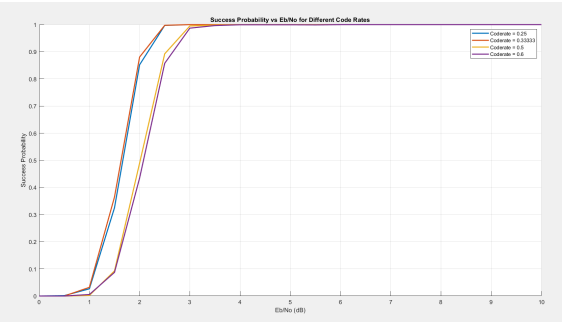


(d) Rate - 4/5

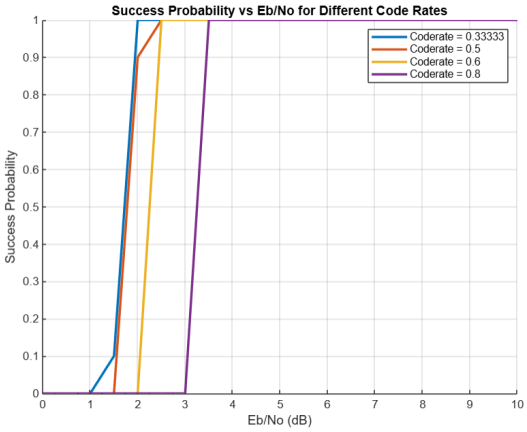
Figure 4: Success probability vs iteration number for NR_1_5_352.

5.5.2 Success Probability vs Eb/No for Different Code Rates

The graphs illustrate the bit error rate (BER) as a function of E_b/N_0 for different LDPC code rates. It is observed that higher E_b/N_0 values lead to a lower BER, indicating better decoding performance due to the stronger signal relative to the noise. Additionally, lower code rates generally achieve a reduced BER compared to higher code rates at the same E_b/N_0, as they provide greater error correction capability through increased redundancy.



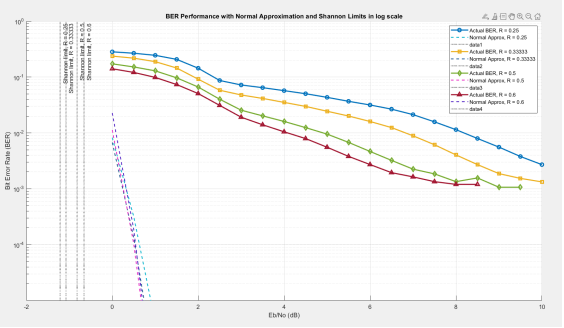
(a) NR_2_6_52



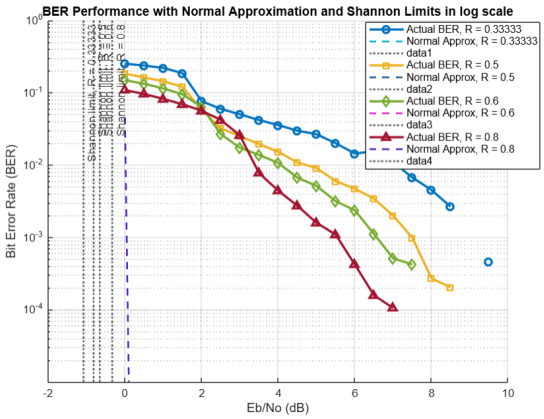
(b) NR_1_5_352

Figure 5: Success Probability vs E_b/N_0 for different Code rates.

5.5.3 BER Performance with Normal Approximation and Shannon Limits in log scale



(a) NR_2_6_52



(b) NR_1_5_352

Figure 6: Comparison of Shannon limit, normalization approximation (NA), and simulated BER vs E_b/N_0 in log scale.

The graph compares the Shannon limit, the normalization approximation (NA), and the simulated bit error rate (BER) against E_b/N_0 for different code rates. The key observations are as follows:

- The Shannon limit, shown as a vertical line, indicates the lowest E_b/N_0 for lossless transmission at a given code rate, as derived in Shannon's 1948 paper. This limit assumes an infinite code length.
- Our simulation uses a finite code length, so we apply the normalization approximation (NA), based on a formula from the instruction manual, to estimate the theoretical lowest BER for a given E_b/N_0.

- The simulated BER decreases steadily as E_b/N_0 increases, showing better decoding with a stronger signal compared to noise. However, the NA's BER drops much faster and becomes negligible for $E_b/N_0 > 1$.In NR_1_6_352 this is even steeper.
- We hypothesize that this rapid drop in NA's BER may be due to the large code length N, causing the NA's bit error rate (BER) to decrease exponentially to very small values. Additionally, factors like rate matching issues—where we used the calculated rate k/n instead of the theoretical rate—and noise scaling might contribute to the higher simulated BER.
- To address the higher BER, we have now implemented the sum-product algorithm, which aims to reduce the simulated BER by improving the decoding accuracy compared to the previous min-sum approach.

5.6 An Improved Algorithm - Sum-Product Algorithm

The sum-product algorithm is a key method used to decode Low-Density Parity-Check (LDPC) codes, which are error-correcting codes widely applied in communication systems. This algorithm operates by passing messages between nodes in a Tanner graph—a diagram where variable nodes represent the bits of the codeword and check nodes represent the parity-check constraints. Through iterative message passing, the algorithm determines the most likely values of the transmitted bits based on the received data.

Steps of the Sum-Product Algorithm. The sum-product algorithm follows these steps to decode the LDPC code:

5.6.1 Initialization

- Each variable node, corresponding to a bit in the codeword and indexed by i, receives a log-likelihood ratio (LLR) from the communication channel. This LLR, denoted $L_i^{(0)}$, indicates the initial probability of the bit being 0 or 1.
- The message from each variable node i to each connected check node j, denoted $L_{i\to j}$, is initialized to the channel LLR: $L_{i\to j} = L_i^{(0)}$.

5.6.2 Check Node Update

- For each check node j, the algorithm computes the message to send to each connected variable node i, denoted $L_{j\to i}$. This message is based on the messages received from all other variable nodes connected to j, excluding i. The formula is:

$$L_{j\to i} = 2\tanh^{-1}\left(\prod_{i'\in\mathcal{N}(j)\setminus\{i\}} \tanh\left(\frac{L_{i'\to j}}{2}\right)\right)$$

  Here, $\mathcal{N}(j)$ is the set of variable nodes connected to check node j, and $\setminus\{i\}$ indicates that variable node i is excluded from the product.

5.6.3 Variable Node Update

- For each variable node i, the algorithm computes the message to send to each connected check node j, denoted $L_{i\to j}$. This message is the sum of the channel LLR and the messages from all other check nodes connected to i, excluding j:

$$L_{i\to j} = L_i^{(0)} + \sum_{j'\in\mathcal{M}(i)\setminus\{j\}} L_{j'\to i}$$

  Here, $\mathcal{M}(i)$ is the set of check nodes connected to variable node i.

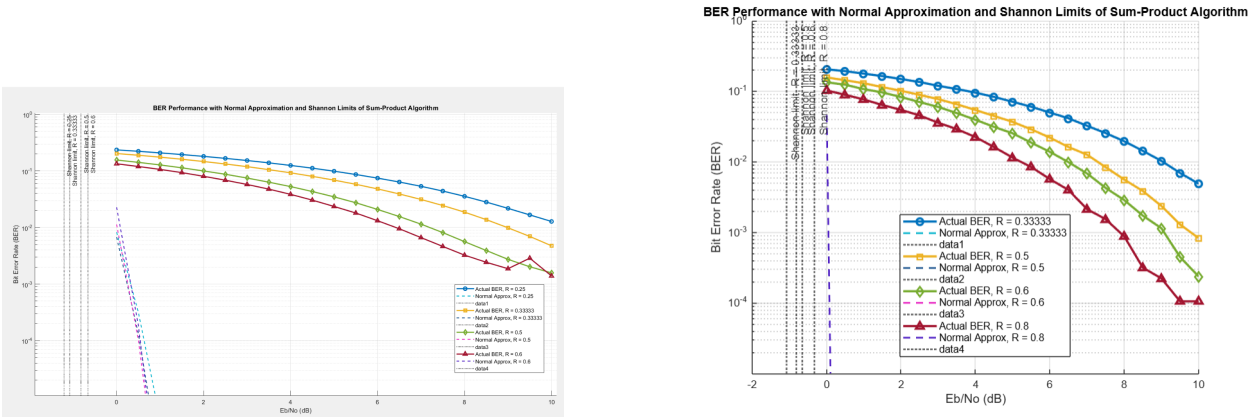
5.6.4 Decision

- After several iterations (or when a stopping condition is met), the algorithm calculates the total LLR for each variable node i:

$$L_i = L_i^{(0)} + \sum_{j\in\mathcal{M}(i)} L_{j\to i}$$

- The decoded bit value is then determined:
  - If $L_i \geq 0$, the bit is decoded as 0.
  - If $L_i < 0$, the bit is decoded as 1.

5.7 Results for sum-product and comparison with min-sum



(a) NR_2_6_52 (b) NR_1_5_352

Figure 7: Comparison of BER using sum-product algorithm and normalization approximation (NA) vs E_b/N_0 for an LDPC code.

The graph compares the bit error rate (BER) using the sum-product algorithm with the normalization approximation (NA) against E_b/N_0 for an LDPC code with a specific rate, alongside theoretical calculations to explain the NA's behavior. The observations are as follows:

- The BER with the sum-product algorithm is a bit better compared to the previous min-sum approach and it appears much smoother, indicating more consistent decoding performance across E_b/N_0 values.
- The NA's BER remains unchanged, decreasing rapidly with higher E_b/N_0, and becomes negligible for $E_b/N_0 > 1$, consistent with previous observations.

5.8 Conclusion And Possible Explanation of Unexpected behaviour Of NA

The soft decoding analysis of LDPC codes for 5G NR, using the Min-Sum and Sum-Product algorithms, reveals the following key insights:

- The Min-Sum algorithm decodes efficiently and succeeds faster at higher E_b/N_0, but it struggles at lower values due to weak signal strength.
- The Sum-Product algorithm lowers the bit error rate (BER) and provides smoother performance, thus improving decoding reliability.
- The Normal Approximation (NA) shows a rapid BER drop with increasing E_b/N_0, becoming negligible past $E_b/N_0 > 1$, driven by the large code length.
- Practical challenges, such as rate matching and noise scaling, cause the simulated BER to remain higher than the NA's predictions.

In summary, soft decoding enhances LDPC performance, with the Sum-Product algorithm outperforming the Min-Sum. However, practical limitations prevent reaching the theoretical bounds.

To understand the NA's rapid decrease, we calculate the theoretical block error rate (BLER) for the NA at $E_b/N_0 = 8\,\text{dB}$, with a theoretical code rate $r = 1/4$, code length $N = 1768$, and other parameters:

- Compute $E_b/N_0 = 10^{8/10} \approx 6.31$.
- Calculate the power ratio $P = r \times E_b/N_0 = 0.25 \times 6.31 \approx 1.578$.
- Determine the channel capacity $C = \log_2(1 + P) = \log_2(1 + 1.578) \approx 1.32$.
- Compute the variance term $V = (\log_2(e))^2 \times \frac{P(P+2)}{2(P+1)^2} \approx (1.4427)^2 \times \frac{1.578 \times 3.578}{2 \times (2.578)^2} \approx 0.885$, where $\log_2(e) \approx 1.4427$.
- Evaluate the finite length correction term $\frac{\log_2(N)}{2N} = \frac{\log_2(1768)}{2 \times 1768} \approx 0.0033$.
- Calculate the adjusted capacity $C - r + \frac{\log_2(N)}{2N} = 1.32 - 0.25 + 0.0033 \approx 1.0733$.
- Compute the scaling factor $\sqrt{\frac{N}{V}} = \sqrt{\frac{1768}{0.885}} \approx 44.7$.

- Determine $x = 44.7 \times 1.0733 \approx 48$.
- Estimate the NA's BLER as $P_{N,c} = Q(48) \approx 10^{-500}$, which is practically zero, explaining the negligible BER.

We hypothesize that the NA's rapid BER drop is due to the large code length $N = 1768$, causing the BLER to decrease exponentially, as shown by the Q-function's extremely small value.

A discrepancy in rate matching may also contribute to the higher simulated BER:

- The theoretical code rate is $r = 1/4 = 0.25$.
- The actual rate, based on $k = 416$ and $n = 1768$, is $r = k/n = 416/1768 \approx 0.2352$.

This difference, along with potential noise scaling issues, might explain why the simulated BER remains higher than the NA's theoretical BER.

Bibliography

References

[1] Vasavada, Yash. *Lecture Slides on Channel Coding*. Course Notes, 2023.

[2] Thangaraj, Andrew. *Video Lectures on LDPC Codes*. NPTEL-NOC IIT Madras, 2023.

[3] Wang, Lifang. *Implementation of Low-Density Parity-Check Codes for 5G NR Shared Channels*. Technical Report, 2022.

[4] Gallager, Robert G. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.

[5] Communications Engineering Lab, KIT. *Sum-Product Algorithm for LDPC Decoding*. YouTube Channel, 2023.