CT216

20 April 2025

Convolutional Code G-19

TECHNOLOGY

Presented by : G19

Table of *Contents*

Honor Code

Introduction

Process

Example

Simulation Results

Performance Analysis

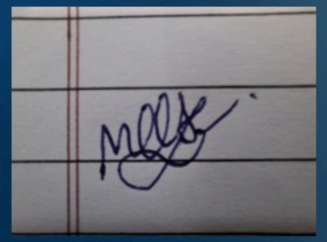
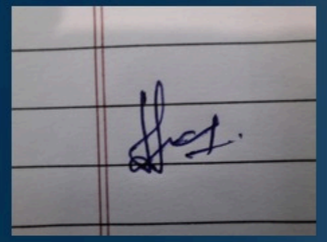
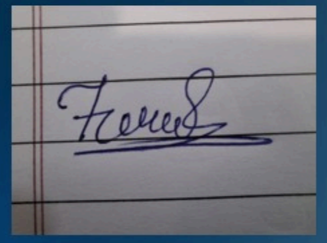
References

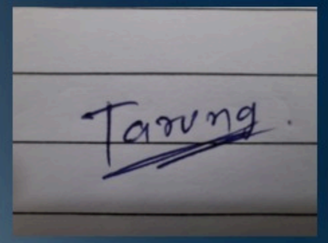
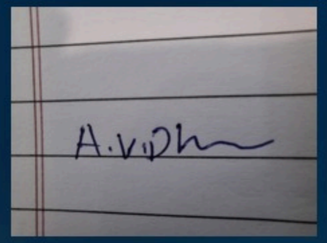
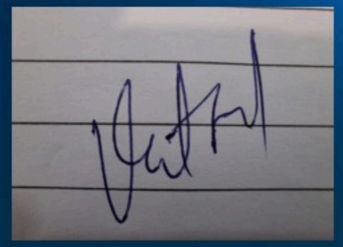
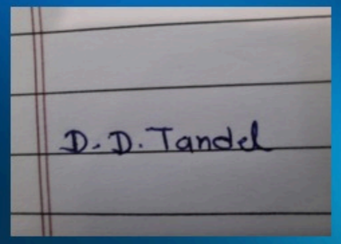
Our Team

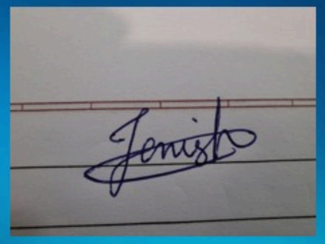
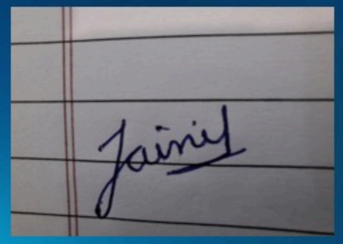
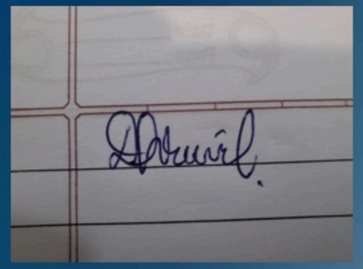
TECHNOLOGY

We declare that

- the work that We are presenting is our own work
- We have not copied the work (Matlab code, results, etc.) that someone else has done
- Concepts, understanding and insights We will be describing are our own
- Wherever We have relied on an existing work that is not our own, We have provided a proper reference citation
- We make this pledge truthfully. We know that violation of this solemn pledge can carry grave consequences

Convolutional Codes are a type of linear error-correcting code. What makes them special is that they have a structured generator matrix according to octal values, which allows the encoding process to work like a filter.

This means the encoded output is produced by sliding a window over the input bits and applying a fixed rule(like filtering), rather than just multiplying by a big matrix like in block codes.

Unlike block codes in systematic form, however, the sender does not send the message bits followed by the parity bits, in a convolutional code, the sender sends only parity bits.

They were among the earliest codes for which effective soft-decision decoding algorithms were developed.

Key Terms:

1. Constraint Length (K_c): The total number of bits that generates the output at any time.

$$K_c = t + n$$

where, t = number of input bits at a time

n = number of memory registers storing previous input bits

2. Registers: Memory elements used to store the previous input bits.

3. Code Rate(r) : The ratio of the number of input bits to the number of encoded bits.

4. Generator Matrix & Octal Values : Each convolutional code is defined by a set of generators represented in octal form. For e.g. With $K_c = 3$ and $r = ½$.

$$g_1 = 5_8 = 101_2$$

$$g_2 = 7_8 = 111_2$$

These binary patterns determine how input and memory bits are combined to produce encoded bits

Encoding

BPSK Modulation

AWGN Noise Adding

Demodulation

Decoding

1] Encoding:

In Convolutional Coding, input bits are encoded into parity bits to enable accurate decoding of the original message at the receiver using the Viterbi decoding

The Convolutional_Encoder function takes the input message, generator matrix (converted from octal to binary), code rate, and constraint length.

- Zero Padding : To handle the initial empty states of the shift registers, $K_c - 1$ zeros are prepended to the input message.
- Bits Matrix Formation : For each time step, a window of K_c bits (current input bit + past register value) is extracted to form a matrix, where each column corresponds to a time step.
- Encoding : Each column of the input matrix is multiplies with the binary generator matrix. The result is taken modulo 2 to generate parity bits, which further are reshaped into a single row vector, which forms the final encoded message.

2] BPSK Modulation:

Binary Phase Shift Keying Maps:

Bit 0 ⟶ +1 Symbol (Phase 0 degree)
Bit 1 ⟶ -1 Symbol (Phase 180 degree)

This is done using:

modulated = 1 - 2*codeword

The symbols are symmetric around zero, making the average symbol energy equal to 1

3] AWGN Noise Adding:

First, convert E_b/N_0 from dB to linear scale:

$$\left(\frac{E_b}{N_0}\right)_{linear} = 10^{\left(\frac{E_b}{N_0}\right)_{db}/10}$$

r is code rate and r=1 for BPSK $E_s = r * E_b$

Since the signal power is 1, the noise power becomes

$$\sigma^2 = \frac{1}{\left(\frac{E_s}{N_0}\right)_{linear}}$$

Gaussian noise is generated as:

$$n = \sigma \cdot N(0,1)$$

The received (corrupted) signal is:

$$r = s + n$$

4] Demodulation:

The received(corrupted) signal is demodulated by thresholding: Values < 0 are mapped to bit 1, and values >= 0 to bit 0, reversing the BPSK modulation process.

5] Decoding:

For decoding, the receiver should determine the "best possible" sequence of transmitted states. Here are many ways of defining "best", but one that is especially appealing is the most likely sequence of states that must have been traversed by the true transmitter. A decoder that is able to infer the most likely sequence is also called a maximum likelihood decoder.

Hard Decision : In this, decoder assume that the received bits are either 0 or 1. It compares the received sequence (r) with the possible encodes codewords and chooses the one with the least number of bits mismatches using hamming distance. It's a simplified ML decoder because it operates on filtered decisions.

Soft Decision : In this, decoder uses the actual received signal soft values (like voltage levels) instead of just 0s and 1s. It calculates the Euclidean distance or log-likelihood ratio between the received values and the expected ones, making it a more accurate ML decoder.

log-likelihood ratio = $2\, a_i\, r_i / \sigma^2$ where $a_i = +1$ or -1,

r_i = received bit

We want to maximize this for soft decision decoding

$$\sum 2\, a_i\, r_i / \sigma^2 \qquad \text{where } i = 1, 2, 3, \ldots, n$$

For Euclidean method , we want to minimize

$$\sum (r_i - a_i)^2 \qquad \text{where } i = 1, 2, 3, \ldots, n$$

5] Decoding:

1. State Table Generation (state_table_generator function) :

   The decoder uses a state transition table (state_data) generated by the state_table_generator(K_c, r, gen) function. This table contains all possible transitions for a given constraint length K_c and code rate r. The table has four columns :

   Current state, Input bit (0 or 1), Next state, Corresponding output bits

   for e.g. $K_c = 3$, r = ½

   For each state and input bit, it gives the next state and the encoded output. This forms the foundation for traversing the trellis.

2. Input Preprocessing:

   The received bit stream ip is reshaped into time-step chunks based on the code rate (r).

   ip = reshape(ip, [ ], s / nop);

   This aligns the input format with trellis steps for comparison.

3. Dynamic Programming Table:

   A 3D matrix data(level, [metric, input, parent_state], state) is created to store three values : Minimum metric (Hamming or Euclidean), Input bit that led to this state, Parent state.

5] Decoding:

4. Trellis Diagram:

The trellis is a structured way of tracking how the convolutional encoder could have moved from one state to another over time. Each path through the trellis corresponds to a possible input sequence. Each node in the trellis corresponds to a state at a particular time step. Each edge represents a transition due to input 0 or 1, and carries the output bits The decoder uses this structure to efficiently identify the most likely transmitted sequence by comparing paths – like navigating a maze and picking the most probable route.

5. Viterbi Decoding (DP Logic) :

For each time step (level in trellis):Loop through all possible transitions from the state table. Calculate the distance metric:

Hard decision: Hamming distance using bit xor.

Soft decision: Euclidean distance using norm.

If multiple paths reach the same state, only the minimum metric path is kept using a tmpstr buffer.

Indexing is adjusted : For early levels (before 2^{K_c}), I = I + 2. After that, full connections → I = I + 1.

5] Decoding:

6. Backtracking for Message Recovery :

After filling the table, the decoder:

Starts from the final state with the minimum total metric.

Backtracks through the data table using stored parent states.

Recovers the input bits in reverse to get the decoded message.

7. Unified Decoder (Hard & Soft):

The same code supports both hard and soft decision decoding with a flag H_or_S:

'H' for Hamming distance

'S' for Euclidean distance

For the constraint length 3 and code rate ½

Output sequence $X_1\ X_2$

$$X_1 = u_l^t + u_{l-1}^t + u_{l-2}^t \qquad X_2 = u_l^t + u_{l-2}^t$$

Linear Shift Register

$g_1 = 5_8 = 101_2$

$g_2 = 7_8 = 111_2$

Encoding of Input Message – [101]

Extra zero bits were added to the input to flush the shift registers and generate the final output bits.

Input Message	1	0	1	0	0					
Input bit and Registers state	1 →	0 →	0							
Encoded message state	1	1	-	-	-	-	-	-	-	-
Input bit and Registers state	0 →	1 →	0							
Encoded message state	1	1	0	1	-	-	-	-	-	-
Input bit and Registers state	1 →	0 →	1							
Encoded message state	1	1	0	1	0	0	-	-	-	-

Example

Input bit and Registers state	0 →	1 →	0							
Encoded message state	1	1	0	1	0	0	0	1	-	-
Input bit and Registers state	0 →	0 →	1							
Final Encoded Message	1	1	0	1	0	0	0	1	1	1

Modulated Sequence:

-1	-1	1	-1	1	1	1	-1	-1	–1

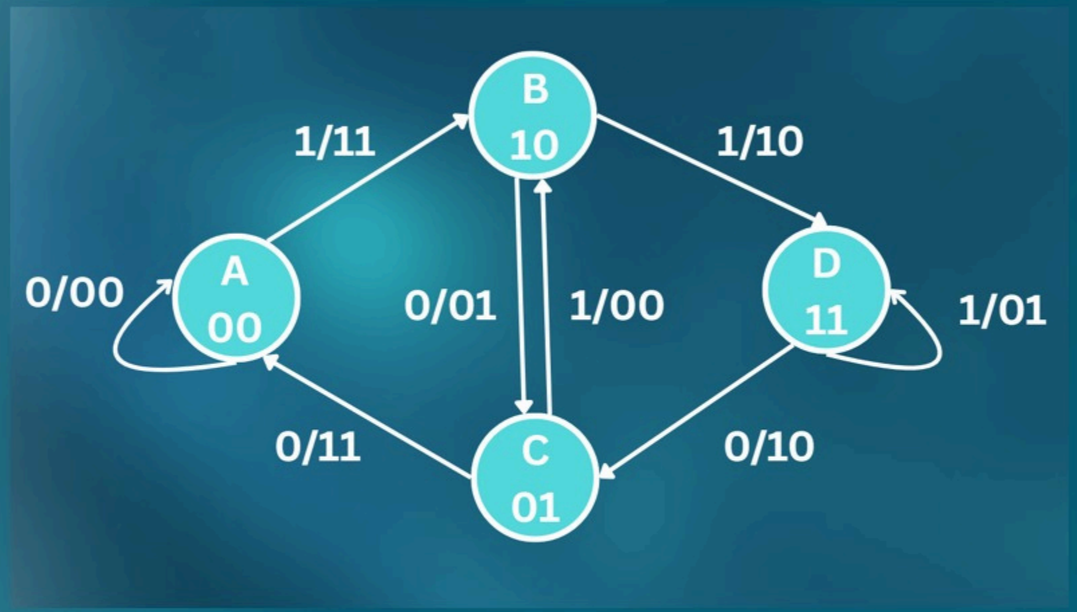
AWGN Noise Adding:

Suppose now received sequence is:

-0.3	-0.1	1.3	0.5	1.9	0.3	-1.5	1.2	-1.1	-1.9

Demodulated Sequence:

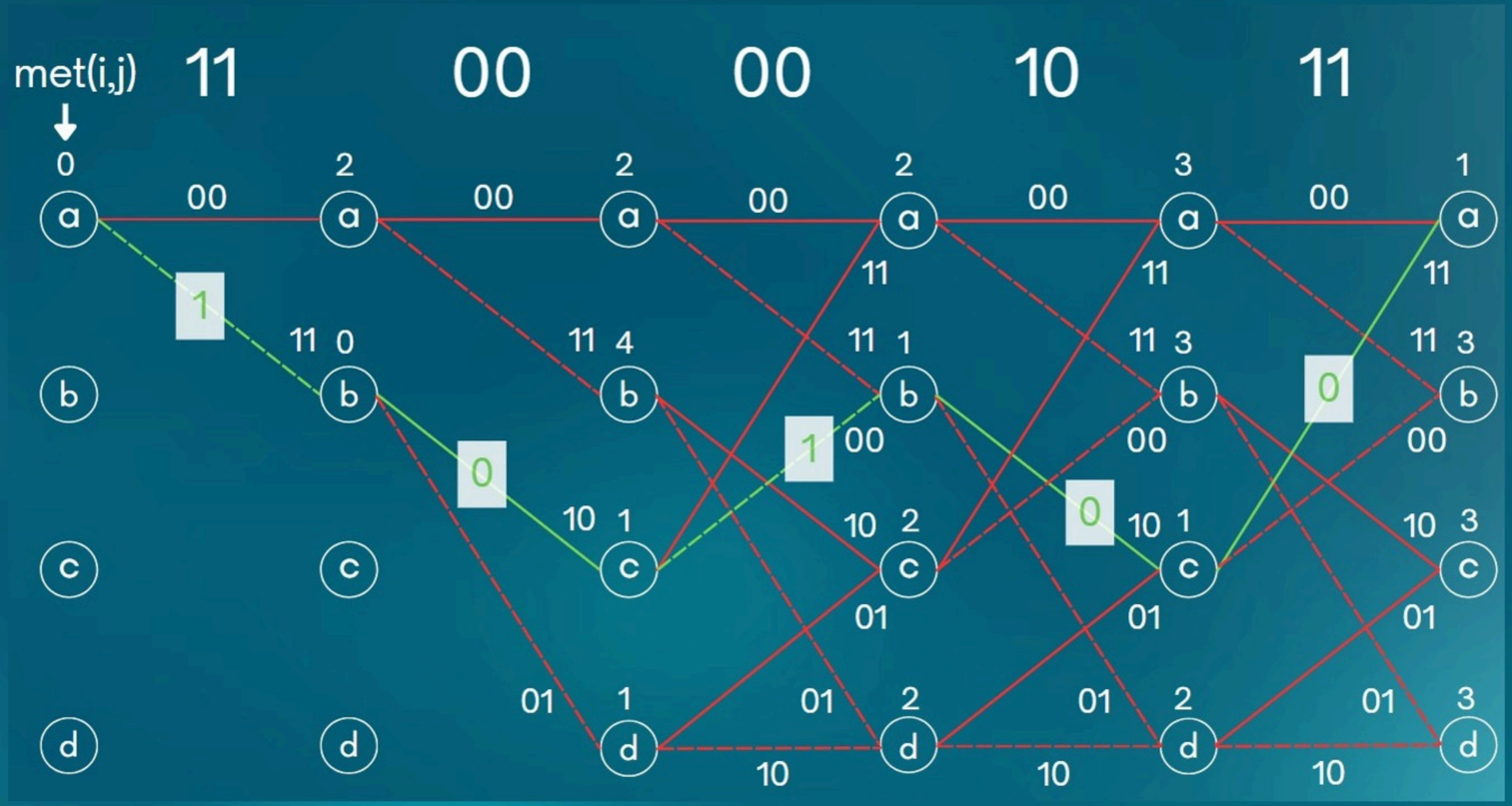
1	1	0	0	0	0	1	0	1	1



State diagram to traverse trellis diagram

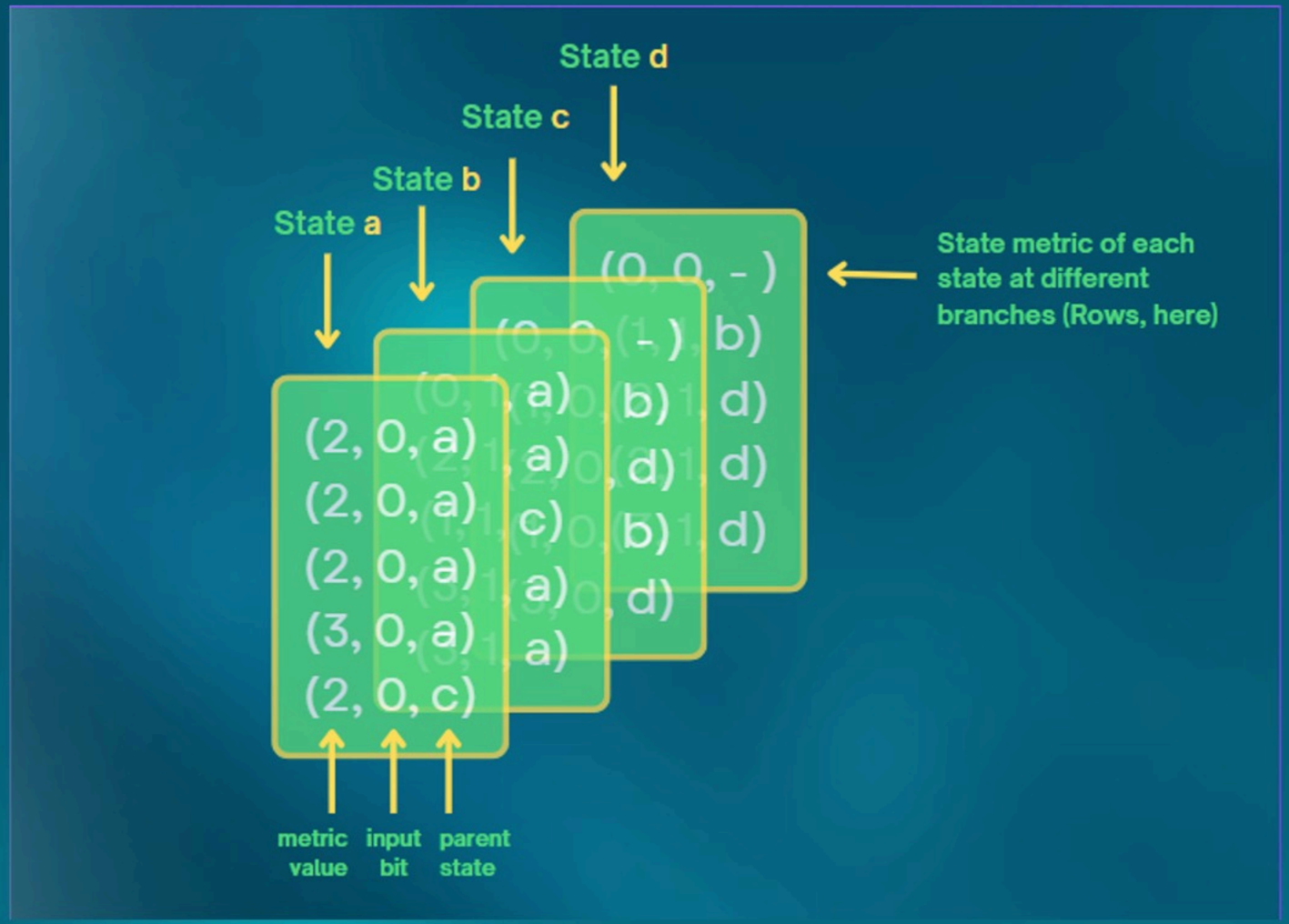
Current state	input bit	Next state	output
A	0	A	‘00’
C	0	A	‘11’
A	1	B	‘11’
C	1	B	‘00’
B	0	C	‘01’
D	0	C	‘10’
B	1	D	‘10’
D	1	D	’01’

State Table

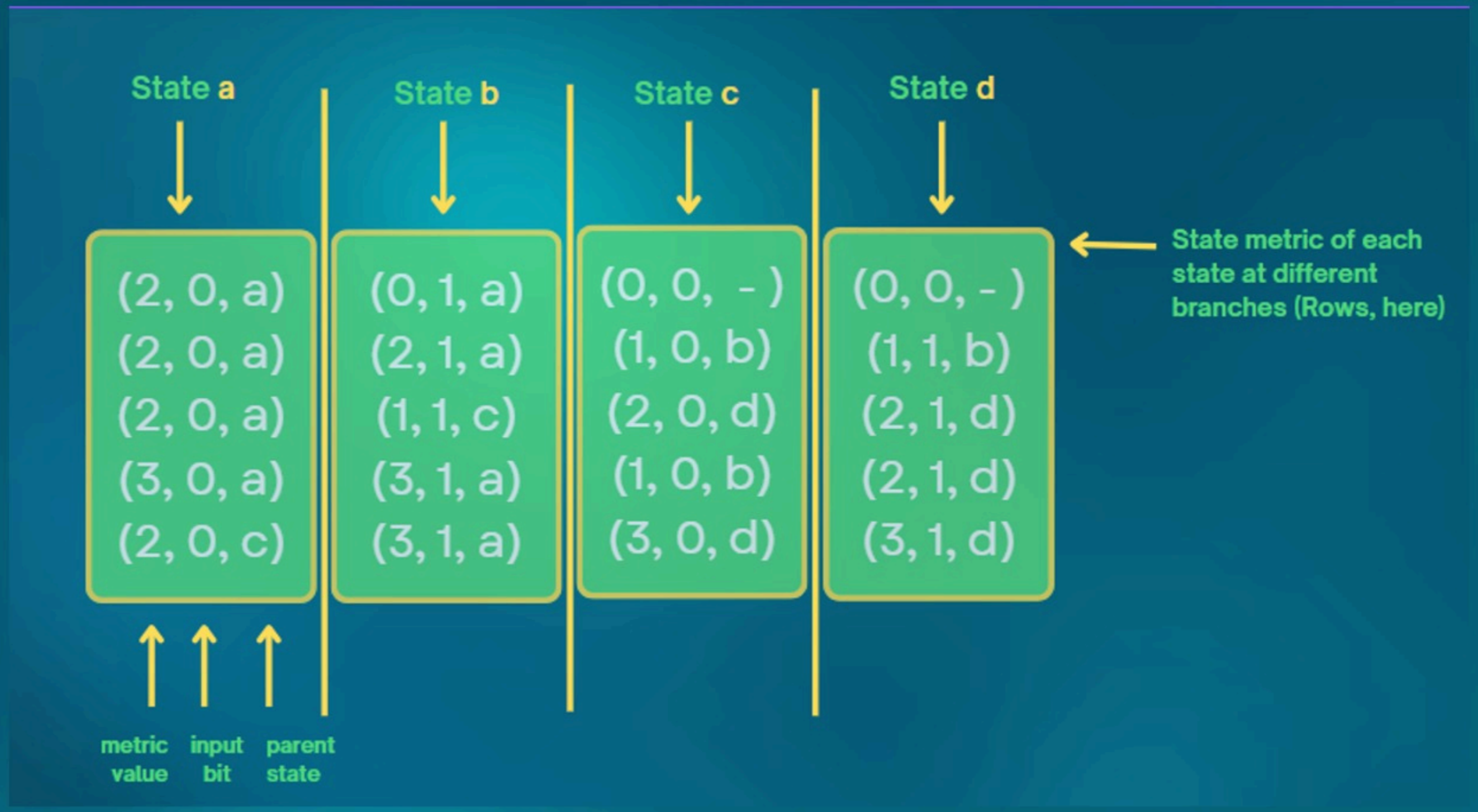


As shown in the trellis diagram, the green path represents the minimum Hamming distance for the received sequence. By backtracking this path, the final decoded sequence is: [1 0 1 0 0].

Example

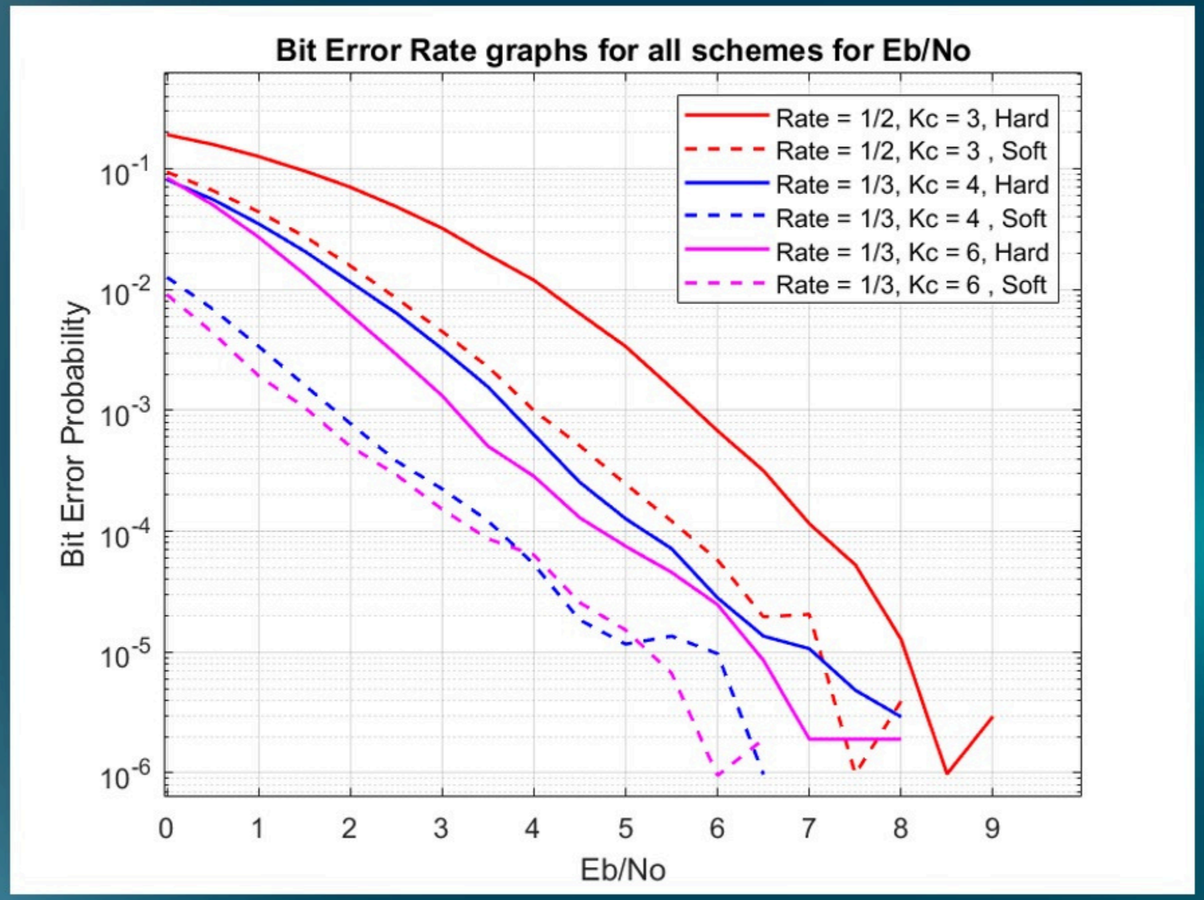

State d
State c
State b
State a
State metric of each state at different branches (Rows, here)
(2, 0, a)
(2, 0, a)
(2, 0, a)
(3, 0, a)
(2, 0, c)
metric value
input bit
parent state

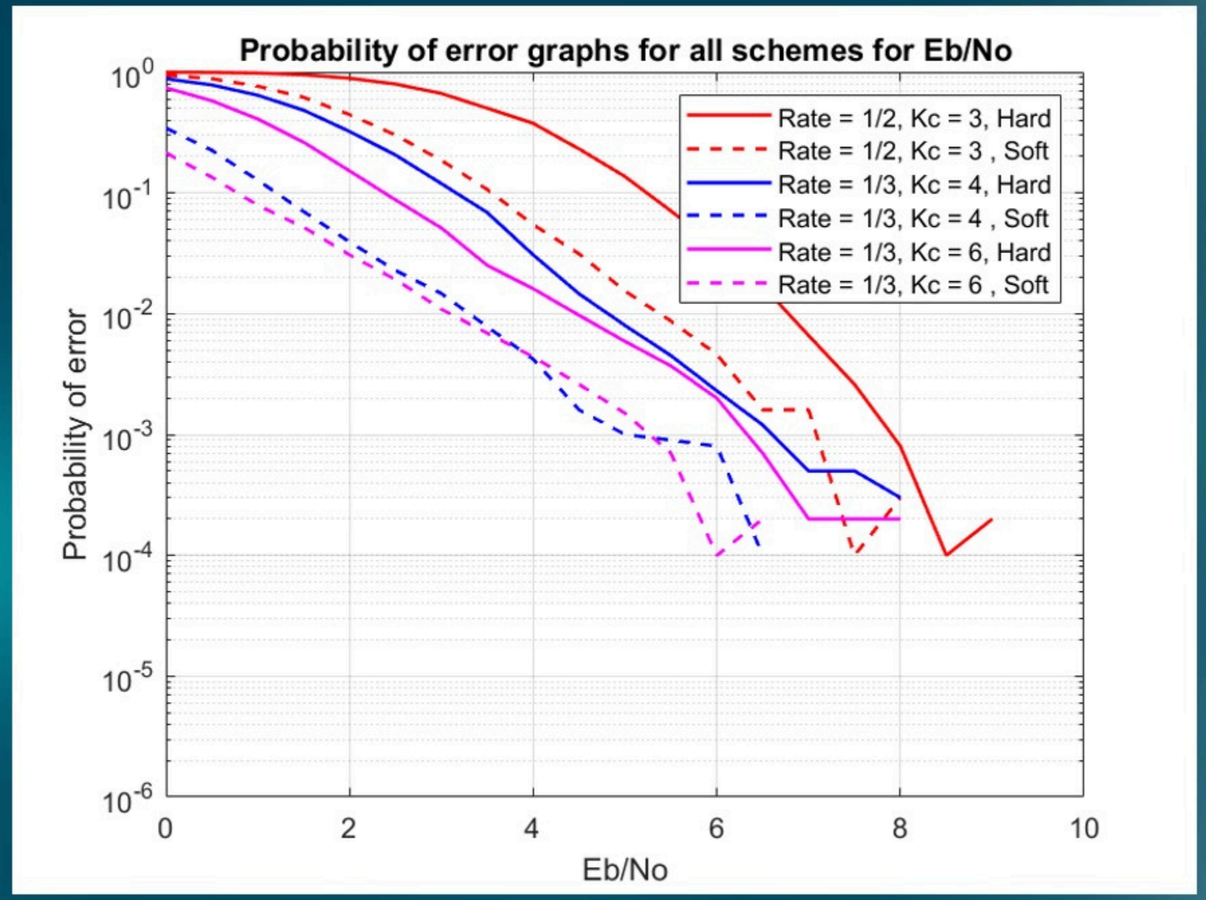
Example



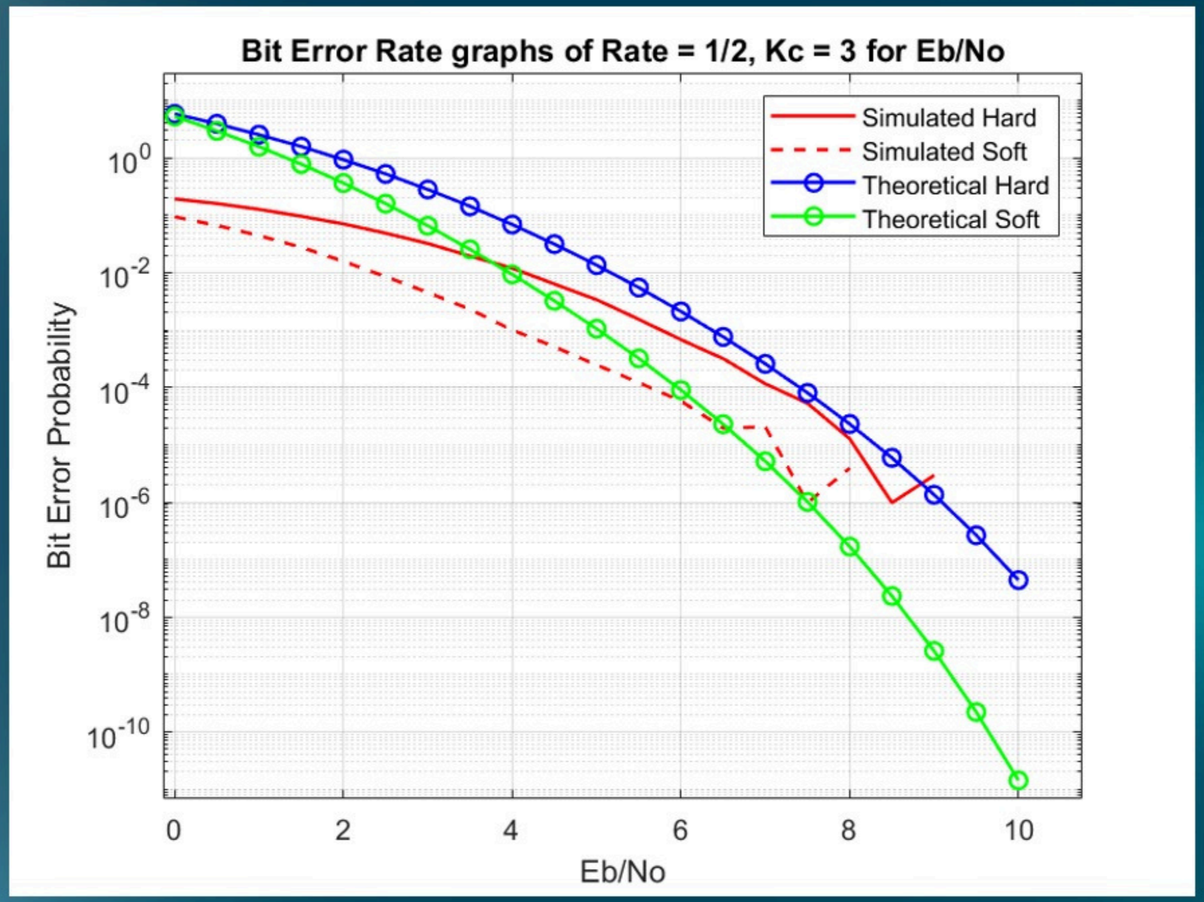
To get the decoded sequence from the 3D matrix, we start by finding the state in the last row with the minimum metric value. Using the stored parent states, we backtrack through the path, and by collecting the corresponding input bits in reverse order, we obtain the decoded sequence: [1 0 1 0 0].

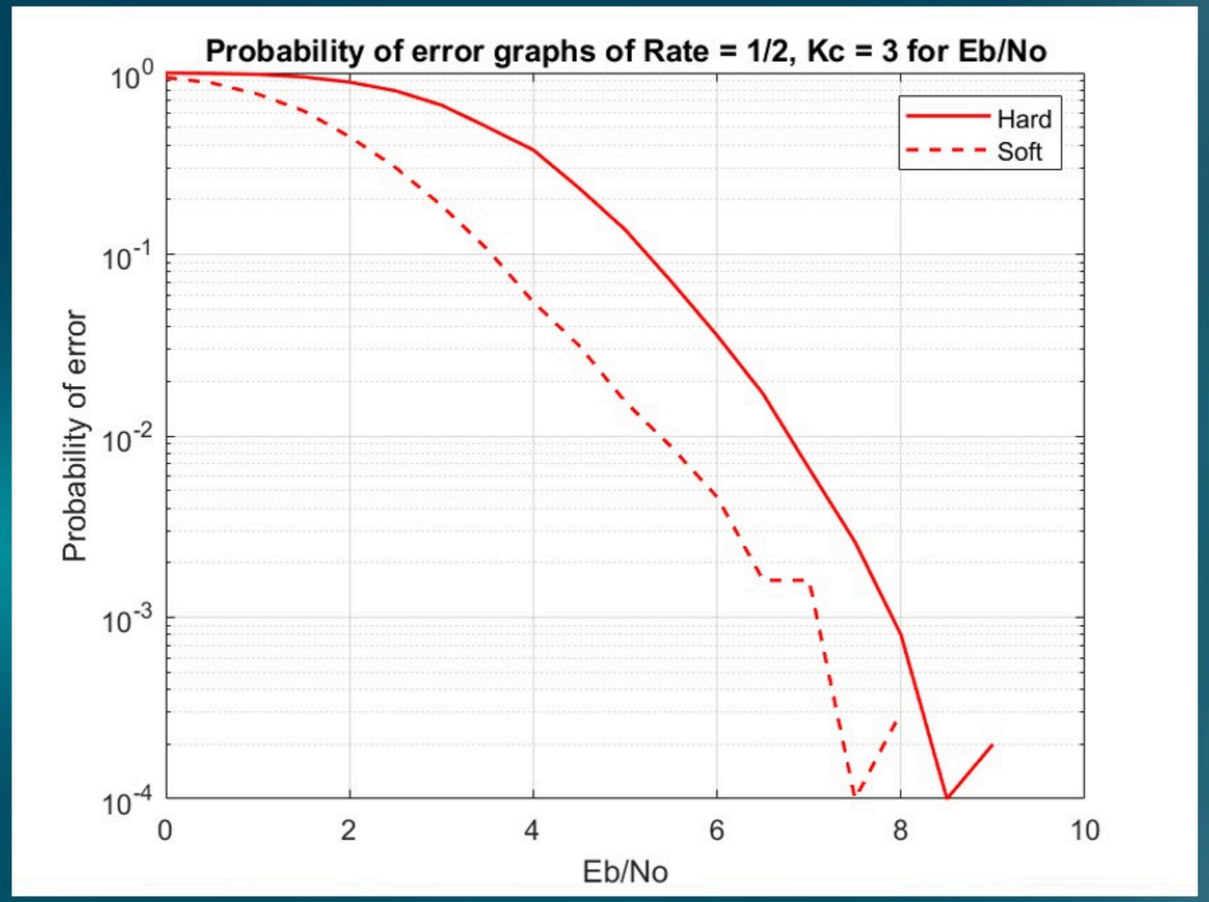
Simulation Results



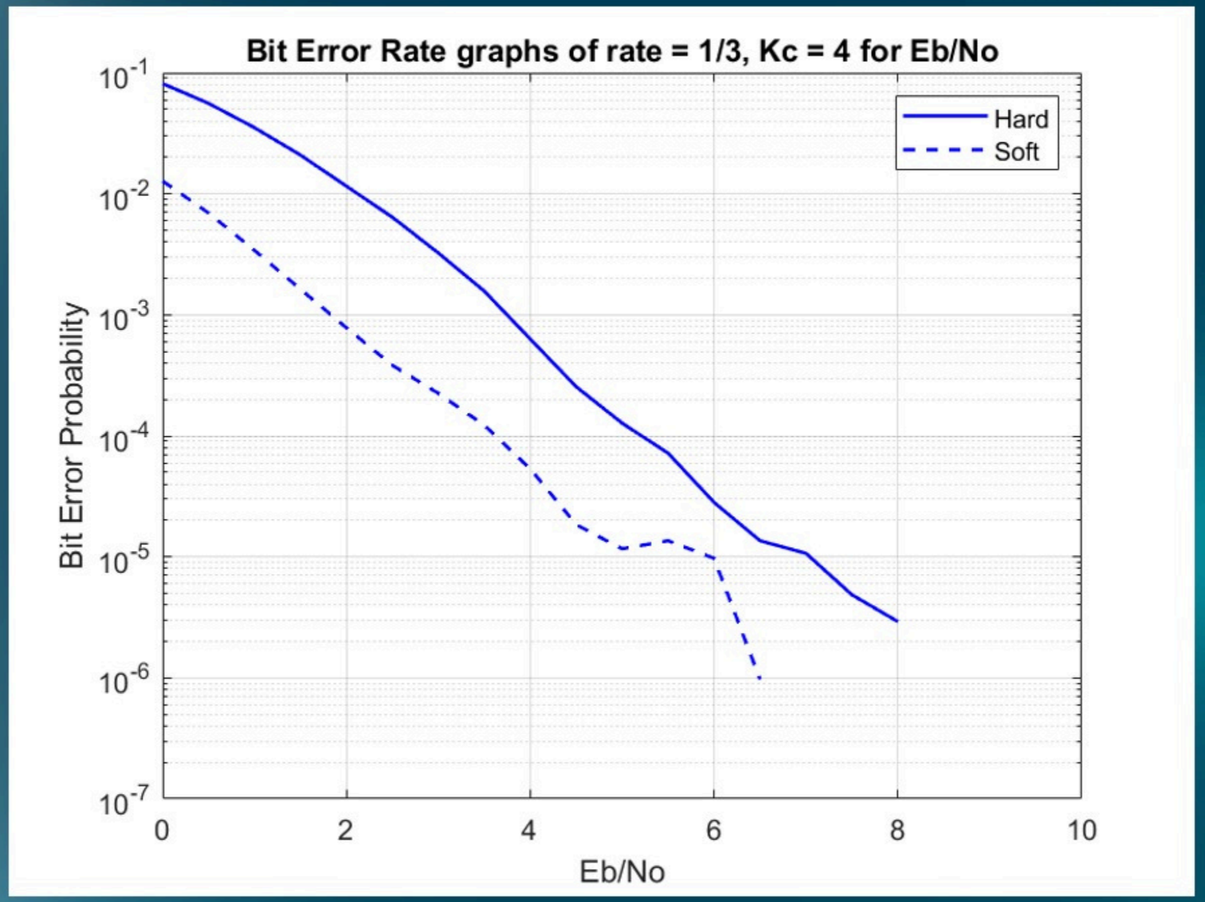


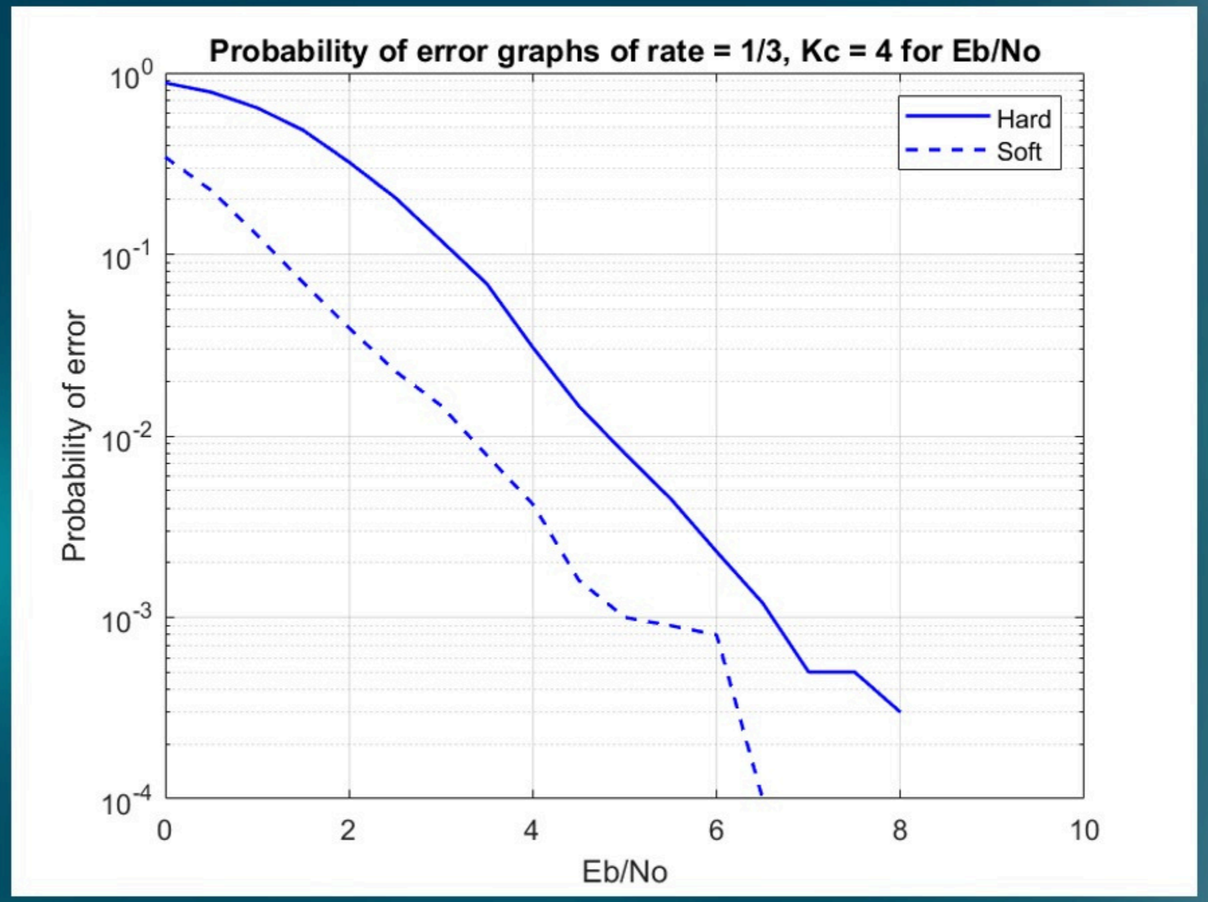
Simulation Results



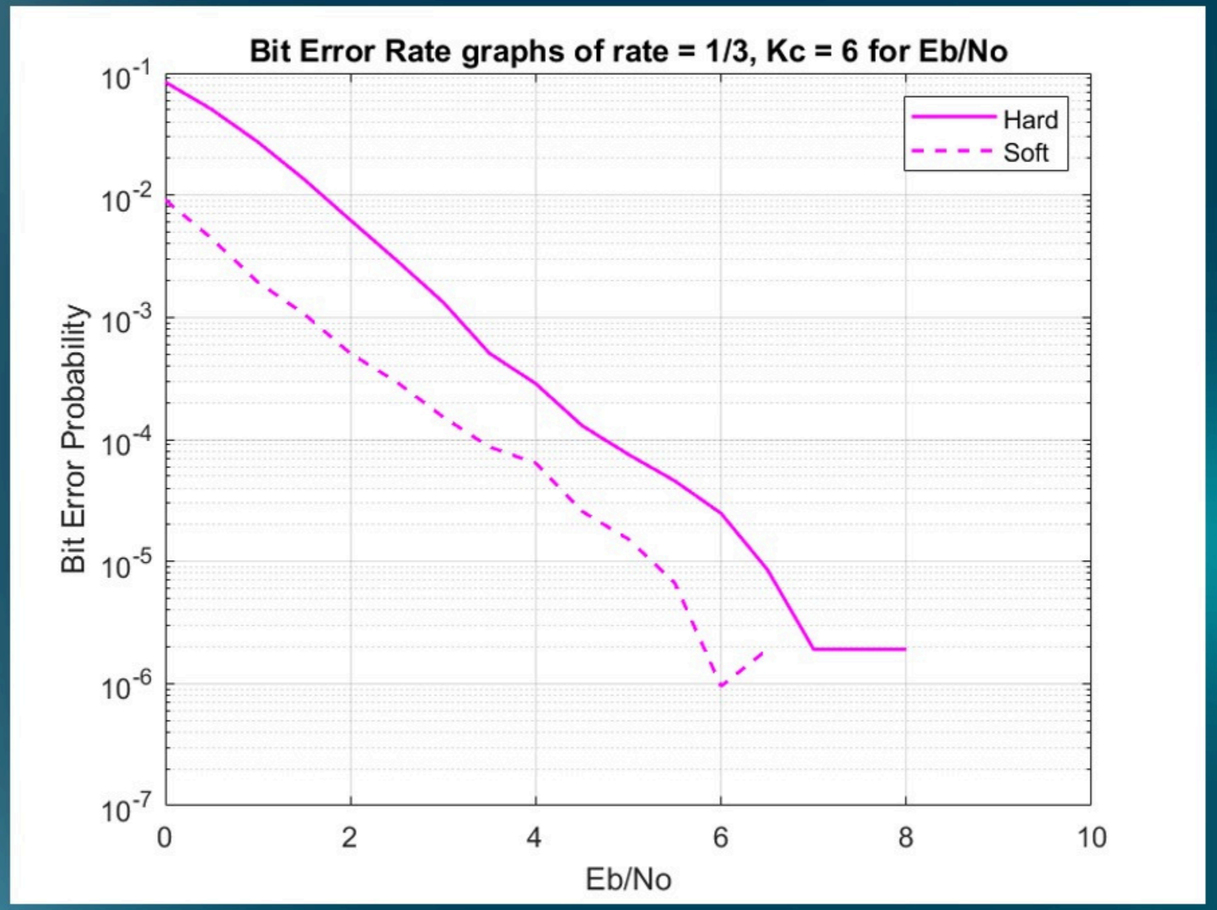


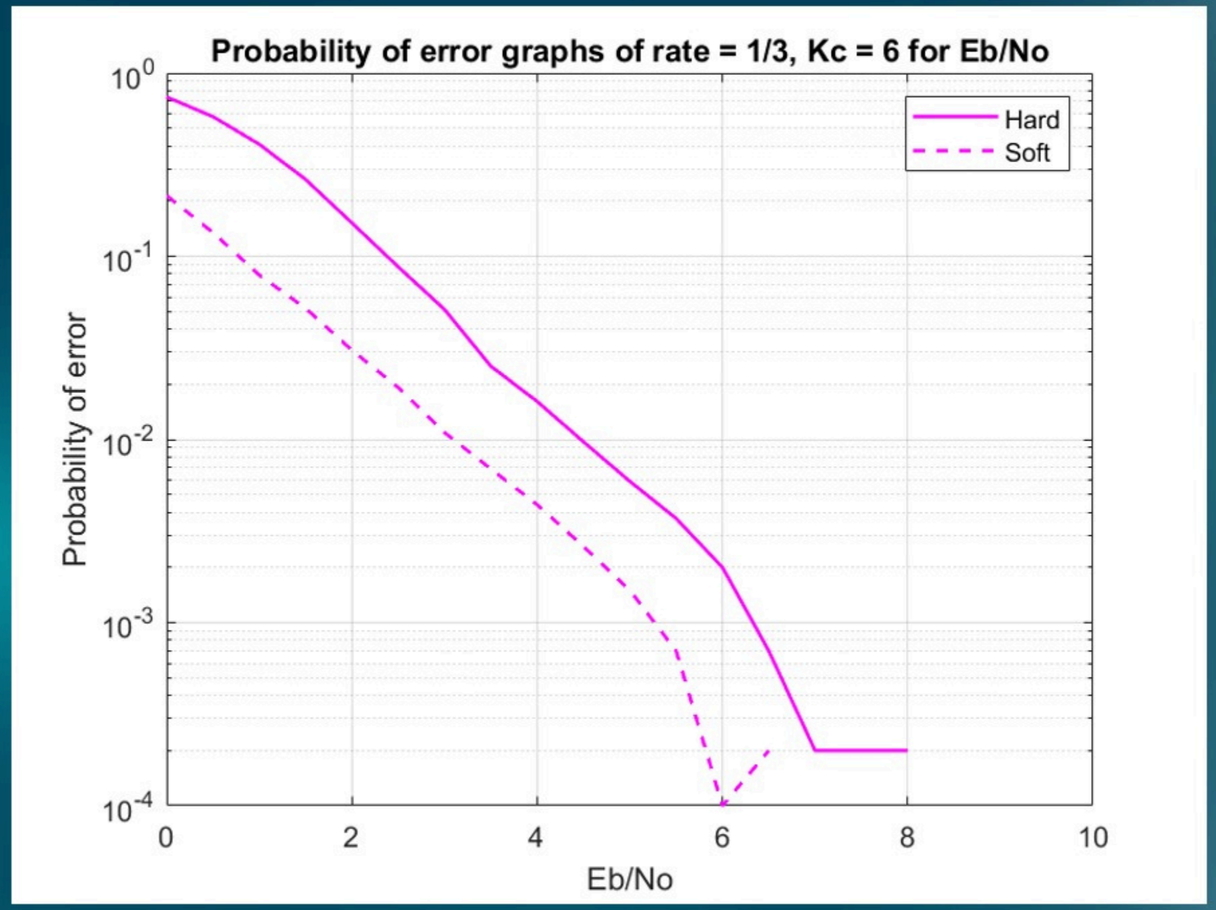
Simulation Results





Simulation Results





The Bit Error Rate (BER) decreases as the Signal-to-Noise Ratio (SNR) increases because higher SNR means lower relative noise in the communication channel. Since SNR=Signal Power/Noise Power, an increase in SNR typically indicates a reduction in noise power. This results in fewer bit errors during transmission, allowing the decoder to more accurately reconstruct the original message. As a result, system performance improves with increasing SNR due to reduced bit error probability.

Increasing the constraint length in convolutional coding enhances the encoder's memory, allowing it to consider more previous input bits when generating output. A larger constraint length creates a more complex trellis, enabling the Viterbi decoder to use a longer path history when making decoding decisions. This improves its ability to distinguish between correct and incorrect bit sequences, especially under noisy conditions. Therefore, even with the same code rate, increasing the constraint length leads to better error correction and reduced BER.

Reducing the code rate introduces more redundancy in the encoded sequence by transmitting more output bits per input bit. For example, a rate of 1/3 sends three output bits for each input bit, compared to two in a 1/2 rate. This longer codeword allows the effects of noise to be distributed across more bits, giving the decoder a better chance to detect and correct errors. As a result, systems with lower code rates generally achieve lower BER, even when the constraint length remains the same.

Hard and soft decision decoders use the same trellis but interpret the received signal differently. Hard decision decoding simply maps the received analog values to binary (0 or 1) using a threshold, losing all amplitude information. In contrast, soft decision decoding retains the actual received values (or their quantized versions), preserving the confidence level of each bit.

Because soft decoding uses more detailed information (soft values), it can more accurately distinguish between likely and unlikely paths in the trellis. This results in better error correction performance, especially in noisy environments, as the decoder makes decisions based on both the bit value and its reliability.

Soft decision decoding provides a coding gain of approximately 1.2 to 1.8 dB compared to hard decoding. This means that for the same target BER, soft decoding requires less transmission power, making it more power-efficient.

Comparison with Theoretically Derived BER:

The theoretical BER for hard decision decoding can be obtained from the formula:

$$P_b \leq \sum_{d=d_{free}}^{\infty} \beta_d P_2(d) = \left.\frac{dT(D,N)}{dN}\right|_{N=1, D=P_2(d)}$$

Where:

$$P_2(d) = \sum_{k=\frac{d+1}{2}}^{d} \binom{d}{k} p^k (1-p)^{d-k}, \quad p = Q\left(\sqrt{\frac{2RE_b}{N_0}}\right)$$

R is the code rate

β_d is the weight of ones for distance d in the transfer function

Comparison with Theoretically Derived BER:

The theoretical BER for soft decision decoding can be obtained from the formula:

$$P_b \leq \sum_{d=d_{free}}^{\infty} \beta_d e^{-\gamma_b R}$$

Where:

$$\gamma_b = \frac{E_b}{N_0}$$

R is the code rate

β_d is the weight of ones for distance d in the transfer function

The above formulas for both methods are explained in detail in the project report. The comparison of simulated and theoretical results is also included in the project report.

Digital Communications, John Proakis, Fourth Edition.

6.02 MIT Lecture on Convolutional Codes (also Viterbi Algorithm Lectures).

Channel Coding for IEEE 802.16e Mobile WiMax Lecture - Matthew C. Valenti

Error Control Coding: Fundamentals and Applications - Shu Lin, Daniel J. Costello Jr.

Error Correction Coding : Mathematical Methods and Algorithms - Todd K. Moon

Youtube Videos : Engineering Funda
nptelhrd
Arksquad

Our Team

PATEL DHRUVIL DARSHITBHAI	-	202301201
VADODARIYA JENISH KIRTIKUMAR	-	202301202
TANDEL DHRUVINEE DINESHKUMAR	-	202301203
DHAMECHA AYUSH VIJAYBHAI	-	202301204
JAINIL UMESHKUMAR PATEL	-	202301205
PATEL MEET BHARATBHAI	-	202301206
HIRAPARA TARANG RAKESHBHAI	-	202301207
VASAVA HARSHILKUMAR RAMESHBHAI	-	202301208
GORI FARAN FIROZBHAI	-	202301209
SOMALIYA VATSAL	-	202301210