Convolution Code

CT-216 Introduction to Communication Systems

Project Group 9 (Lab Group 2)

Prof. Yash Vasavada

April 19, 2025

Honor Code

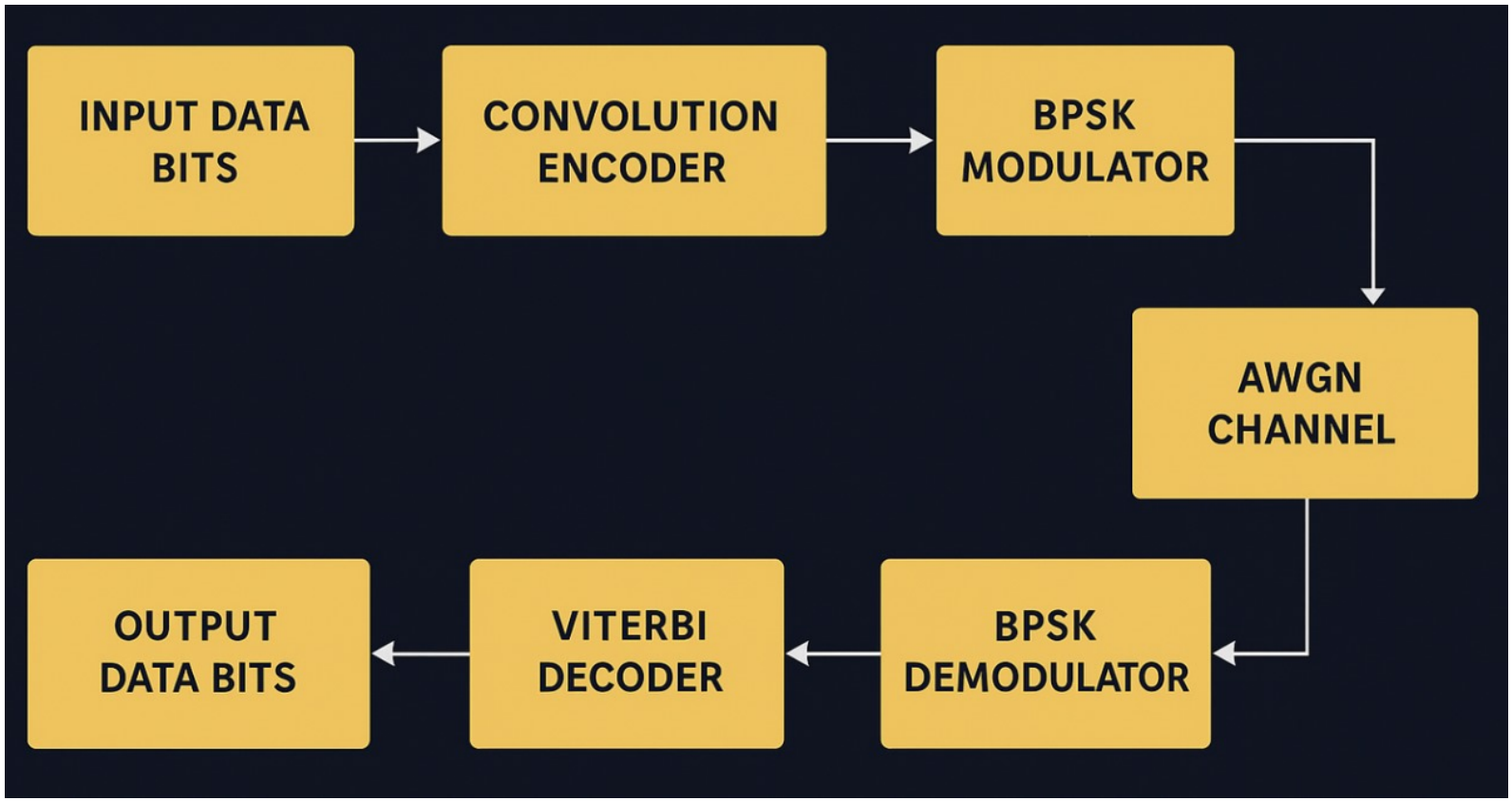
- The work we are presenting is our own.
- We have not copied work from others.
- Concepts and insights are our own.
- We make this pledge truthfully.
- Group Members:

  - Yaksh Patel 202301089
  - Jhil Patel 202301090
  - Meet Patel 202301091
  - Rajdeep Patel 202301092
  - Neel Shah 202301093
  - Yash Panchal 202301094
  - Sutariya Om 202301096
  - Deep Kakadiya 202301097
  - Kirtan Chhatbar 202301098
  - Krish Malhotra 202301099

Introduction

- Convolutional coding is a popular error-correcting coding method used to improve the reliability of communication systems.
- This project focuses on simulating the transmission of convolutionally encoded data over an AWGN channel and decoding the received data using two versions of the Viterbi decoding algorithm:
  - Hard Decision Decoding (HDD)
  - Soft Decision Decoding (SDD)
- Our goal is to analyze Bit Error Rate (BER) vs. Signal-to-Noise Ratio (SNR) and show that SDD outperforms HDD through simulation results.

Flow Diagram



Problem Statement

- Analyze aspects of Convolution codes.
- Plot detection error probabilities for:
  - Rate $= 1/2$, K $= 3$
  - Rate $= 1/3$, K $= 4$
  - Rate $= 1/3$, K $= 6$
- Use Viterbi hard and soft decoding for SNR 0–10 dB (step $=$ 0.5)

Definitions

- Rate : Ratio of the number of input bits to the number of output bits. generally, rate=k/n where k is information bits and n is encoded bits.
- Constraint length (K): The number of delay elements in the convolutional coding.
- Generator polynomial : Wiring of the input sequence with the delay elements to form the output.

Encoder

- Convolutional encoders map a continuous stream of input bits to output bits, with each output depending on current and past inputs using shift registers.
- They offer reliable performance at low cost, making them suitable for applications with limited hardware resources.
- The parity bits are generated by the convolution of the message bits and the generator polynomial, g.

$$p_i[n] = \left(\sum_{j=0}^{k-1} g_i[j]\, x[n-j] \right) \mod 2$$

- The above equation is used to encode the message.

Encoder Concept

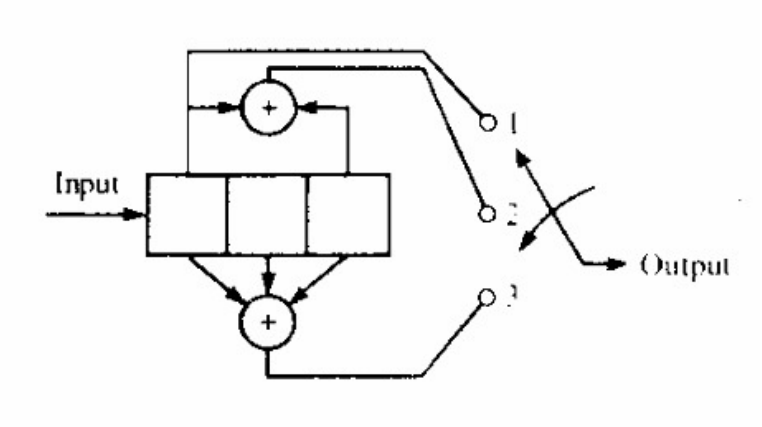


Figure: *

Block diagram for $K = 3$, rate $= 1/3$

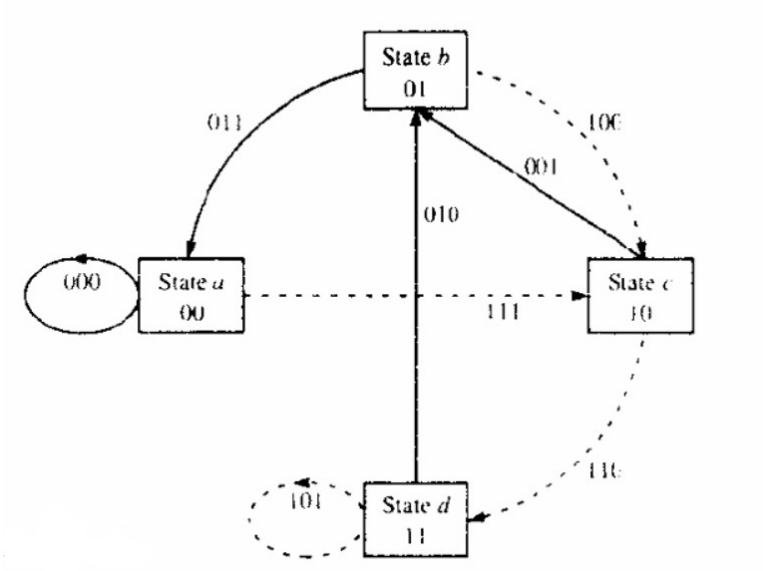


Figure: *

State diagram for $K = 3$, rate $= 1/3$

Encoder Concept

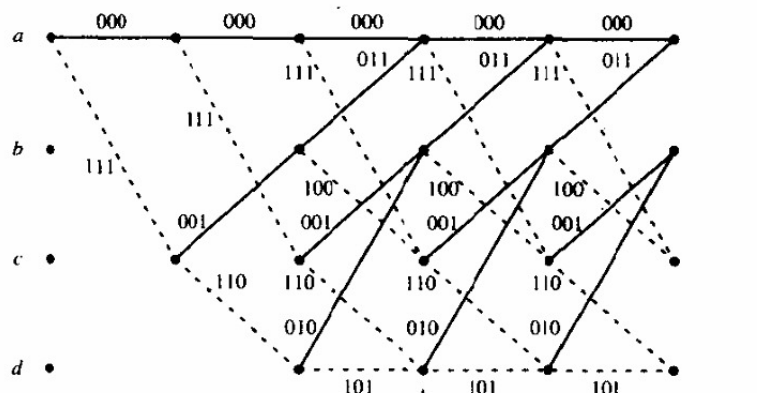


Figure: *

Trellis diagram for $K = 3$,
rate $= 1/3$

Pseudocode for encoding

```
for each random message trial:
    generate message bits msg[1...k]
    padded_bits = append zeros to msg
    shift_reg = [0, 0, ..., 0]
    encoded = empty list
    for bit in padded_bits:
        shift_reg = [bit] + shift_reg[1...K{1]
        for i = 1 to num_output_branches:
            out_i = modulo_2( sum( taps_i[j] * shift_reg[j] for j=1...K ) )
        append [out_1, out_2, ...] to encoded
    end
end
```

BPSK Modultor

- Binary Phase Shift Keying (BPSK) is a modulation technique employed in communication systems to transmit information via a communication channel.
- Binary Phase Shift Keying maps each bit to one of two signal levels:
  - Bit 0 $\rightarrow +1$
  - Bit 1 $\rightarrow$ -1
- This is done using the formula: BPSK signal = 1 - 2 * bit stream.

AWGN Noise Channel

- The AWGN channel adds Gaussian noise to the transmitted signal. This noise has zero mean and variance based on the signal's energy and the SNR.
- Received signal:
  - r = s + n, where n is Gaussian noise
  - Noise variance = 1 / (r * Eb/N0), where r is the code rate

Pseudocode for BPSK and AWGN Channel

```
for each SNR value EbNO_dB:
    EbNO_lin = 10^(EbNO_dB/10)
    tx_symbols = [ 1 - 2 * bit  for bit in encoded ]
    sigma = 1 / sqrt(EbNO_lin)
 rx_values = [ tx + sigma * randn()  for tx in tx_symbols]
```

BPSK Demodulator

- At the receiver, the received signal undergoes demodulation to recover the original binary data. This process typically involves:
- Coherent Detection: Multiplying the received signal with a local oscillator at the carrier frequency to extract the phase information.
- Decision Making: Comparing the phase of the received signal with a threshold value to determine the transmitted symbol. If the phase is closer to one phase state, it's interpreted as '0'; if it's closer to the other phase state, it's interpreted as '1'. For soft decision decoding
- If the convolutional code produces p parity bits, and the p corresponding analog samples are v = v1, v2,...,vp.
- These values are analog in nature as we do not perform demodulation of the received signal we directly pass it to the decoder.

Viterbi Decoding

- The Viterbi algorithm is a dynamic programming technique used to decode convolutionally encoded messages. It traverses a trellis diagram to find the most likely transmitted sequence.

  ## Hard Decision Decoding (HDD)

  - HDD converts received analog values into binary bits using thresholding, then computes the Hamming distance to choose the best path through the trellis.

  ## Soft Decision Decoding (SDD)

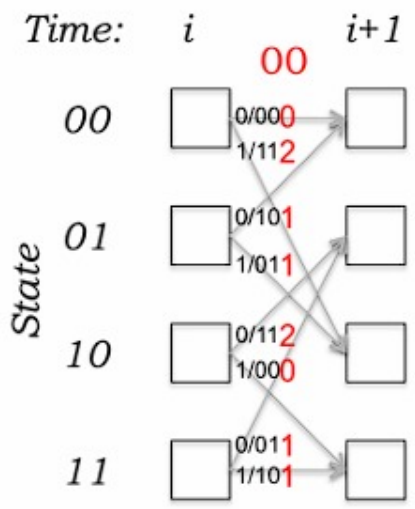
  - SDD directly uses the received real-valued signal and calculates the Euclidean distance between received and expected values. This provides more accuracy and generally results in a lower BER.
  - According to Proakis, the BER performance improves significantly with SDD, especially in low SNR regimes.

Hard Decision Decoding

- **Branch Metric (BM)**: BM is the measure of the hamming distance between the received codeword and the expected parity bits (Hamming distance: No of places where corresponding bits are different)
- **Path Metric (PM)**: PM is the sum of the branch metric of the branches it traversed in the path.

  $\text{PM}[s, i+1] = \min\left(\text{PM}[\alpha, i] + \text{BM}[\alpha \rightarrow s], \text{PM}[\beta, i] + \text{BM}[\beta \rightarrow s]\right)$

  where α and β are two predecessor states.



Hard Decision Decoding

- We are describing how the decoder finds the most likely path.
- Initially, the cost of state 00 will be 0 and the cost of the rest $2^{(k-1)} - 1$ states will be infinity.
- Further, the main loop consists of the computation of the branch metric for the next set of parity bits and the path metric for the next time step. Path metric computation consists of two steps:
  - Take the sum of the branch metric and the path metric of the current state
  - Compare the sum that we got from two previous states α and β and assign the minimum from them.
- Further, to retrieve the transmitted message we will perform backtracking. We will consider the path which has the smallest value and the decoded message that we will get at the end will have the fewest errors.

Setup for Viterbi

```
build trellis from (K, generator_polynomials)
num_states = 2^(K-1)
num_input_symbols = 2
num_output_symbols = 2^num_output_branches
initialize path_metric[state=0, time=0] = 0
initialize other path_metric[*] = ∞
```

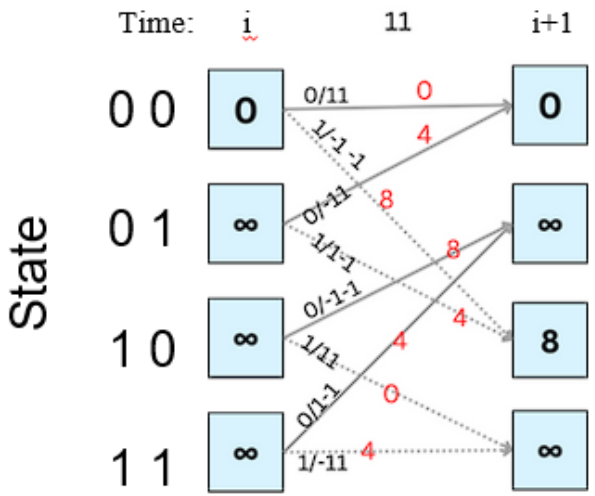
Pseudocode for HDD

```
for time t = 1 ... num_steps:
    rxt = rx_hard_bits[ (t-1)*num_output_branches + 1...t*num_output_branches]
    for previous_state in 0...num_states - 1:
        for input_bit in {0,1}:
            next_state = trellis.nextState[previous_state, input_bit]
            expected_bits = trellis.outputBits[previous_state, input_bit]
            branch_metric = sum(rxt[j] != expected_bits[j]
            for j=1...num_output_branches)
            metric = path_metric[previous_state,t-1] + branch_metric
            if metric < path_metric[next_state, t]:
                path_metric[next_state, t] = metric
                survivor[next_state, t] = previous_state
                in_rec[next_state, t] = input_bit
end

decoded_bits = traceback survivors & inputs back to t=1
```

Soft Decision Decoding

- In Soft decision decoding we do not demodulate received signal instead we take Euclidean distance with 1/-1.
- The only difference in soft and hard decision decoding is the computation of the branch metric.
- Let us define the branch metric for soft decision decoding.
- **Branch Metric (BM)**:The branch metric for soft decision decoding is the Euclidean distance between the received message and the expected message.



Pseudocode for SDD

```
    for time t = 1 ... num_steps:
    rxt = rx_values[(t-1)*num_output_branches + 1...t*num_output_branches]
    for previous_state in 0...num_states{1:
        for input_bit in {0,1}:
            next_state = trellis.nextState[previous_state, input_bit]
            expected_bits = trellis.outputBits[previous_state, input_bit]
            expected_sym = [1 - 2 * expected_bits[j]
            for j=1...num_output_branches]
            branch_metric = sum((rxt[j] - expected_sym[j])^2)
            for j=1...num_output_branches)
            metric = path_metric[previous_state,t-1] + branch_metric
            if metric < path_metric[next_state, t]:
                path_metric[next_state, t] = metric
                survivor[next_state, t] = previous_state
                in_rec[next_state, t] = input_bit
end

decoded_bits = traceback survivors & inputs back to t=1
```

Transfer Function

- Transfer Function is an algebraic representation of all paths that starts and end at all zero state.

$$T(D,N,J) = \sum_{d=dfree}^{\infty} a_d D^d N^{f(d)} J^{g(d)}$$

- Here, d = no. of ones in output codeword
- $f(d)$ = no. of ones in input (k-bits)
- $g(d)$ = no. of branches spanned by the path
- The transfer function = (Output at end zero state / Input at zero start state)
- Provides the properties of all the paths.
- Minimum non-zero exponent of D gives the minimum hamming distance.

Simulation results

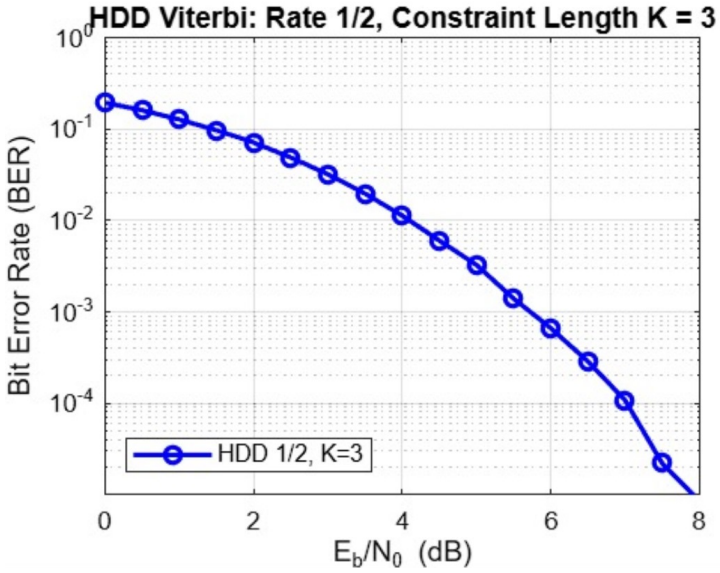


Figure: *

MATLAB Simulated image

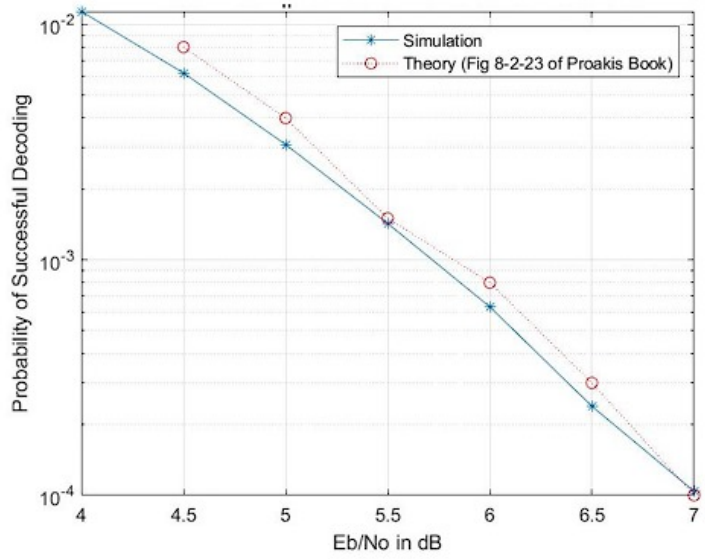
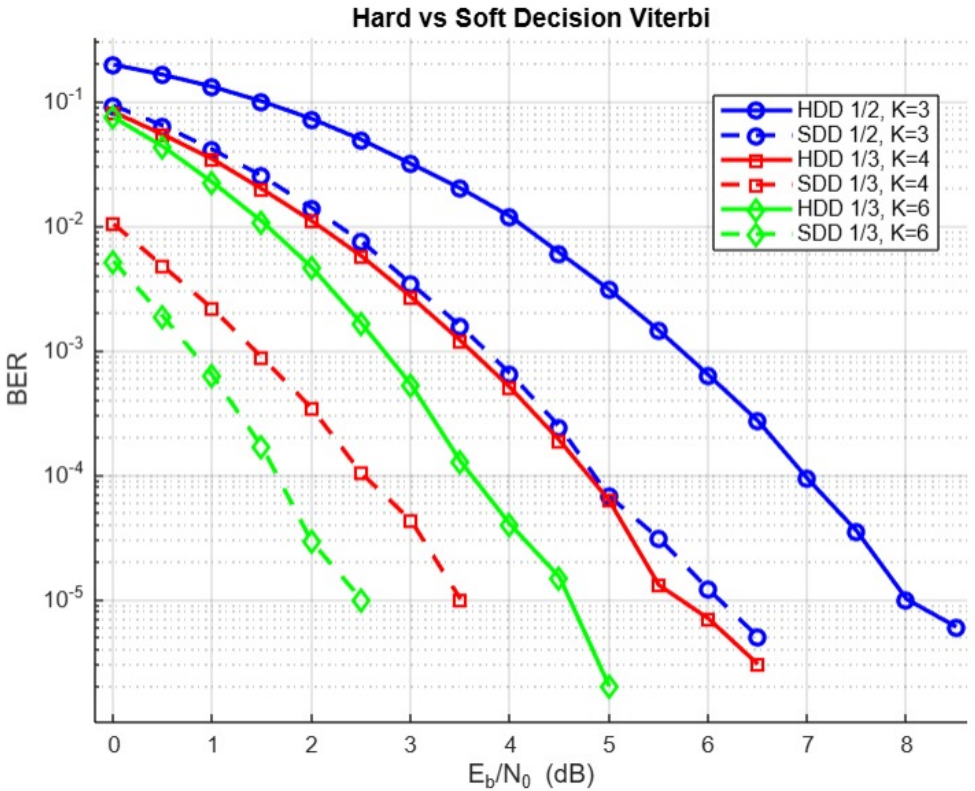


Figure: *

Image from manual

Simulation results



Observations

- BER decreases with increasing SNR.
- Soft decoding outperforms hard decoding.
- Lower code rates improve error correction.
- Longer constraint length = better performance but more computation.

Summary

- Convolution codes use parity instead of message bits.
- Viterbi soft decision is superior in performance.
- Tradeoff: error correction vs decoding complexity.

Thank You