CONVOLUTION CODE

Group 10

Prof. Yash Vasavada

Mentor - Heman Chauhan

HONOUR CODE

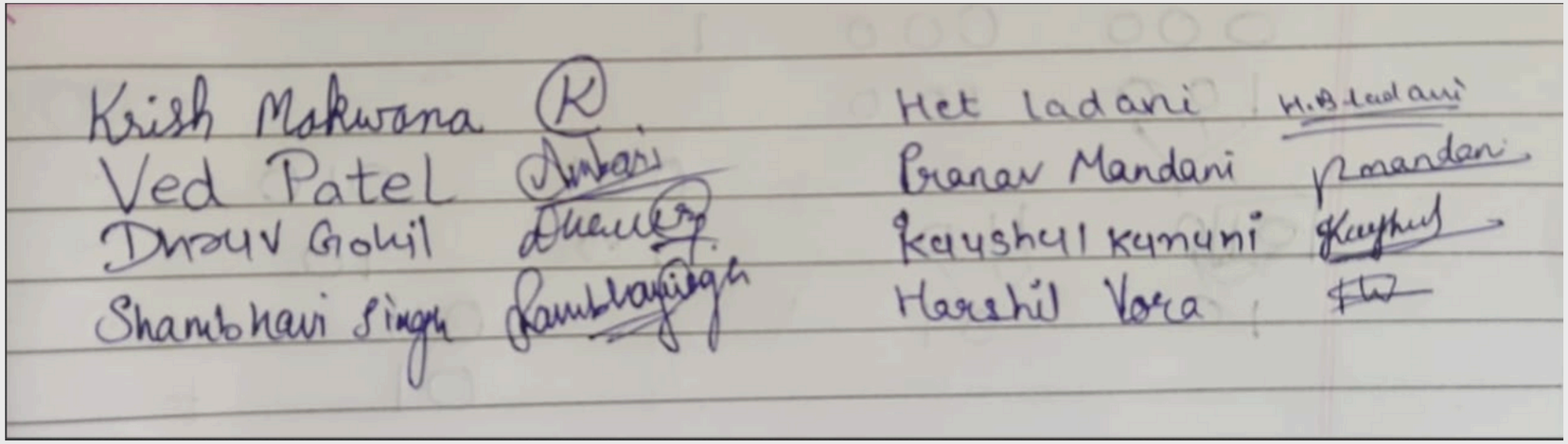
We declare that

→ 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.



INDEX

INTRODUCTION

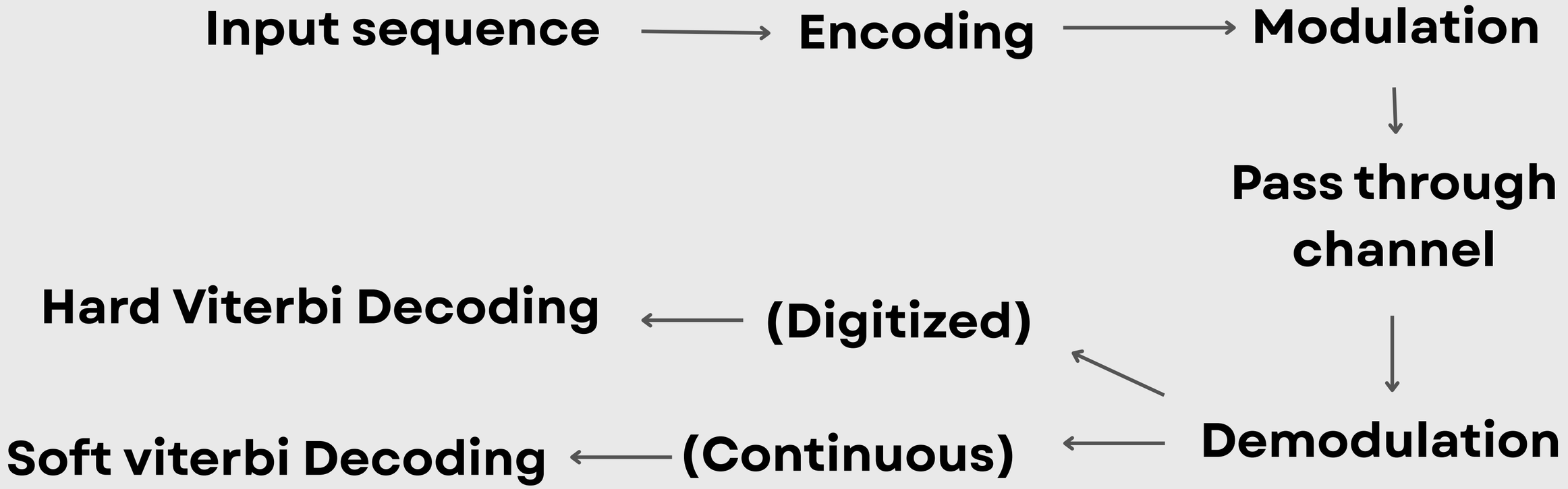
Convolutional codes were introduced in 1955 by Peter Elias.

Unlike block codes, which process data in fixed-length blocks, convolutional codes treat data as a continuous stream. The output bits of a convolutional code are determined by logical operations performed on the current input bits and a limited number of previous input bits.

OBJECTIVE

- Our goal is to comprehend digital communication's channel coding using convolution codes. In this case, only encoding, modulation, demodulation, and decoding are included in the digital communication scope.

PROCEDURE


Input sequence
Encoding
Modulation
Pass through channel
Demodulation
(Digitized)
Hard Viterbi Decoding
(Continuous)
Soft viterbi Decoding

PROCEDURE

1. The encoding of transmitted messages is done according to given rates and constraint length.
2. Then the modulation of the encoded bit using the BPSK Modulator.
3. For transmission of the encoded bits the AWGN channel is used and during the transmission the noise is introduced in the message.
4. Then decoding is performed by 2 techniques:
   - Viterbi hard decoding
   - Viterbi soft decoding
5. Now at the end plotting the graphs for probability of error detection to analyse the performance of both the decoding techniques for different rates and constraint length and.

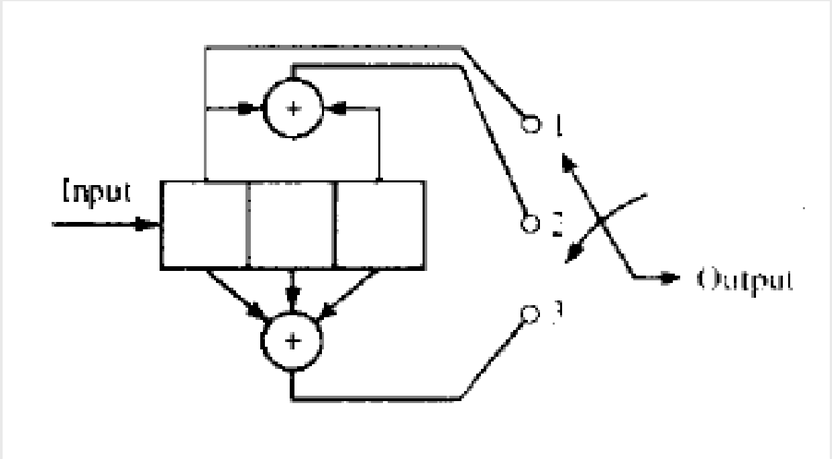
ENCODING

The central idea behind convolution encoding is to encode data by taking EXOR with a limited number of previous inputs.

Generator Polynomials

The generator polynomials are key to an efficient encoder.
We must choose generator matrices that have a high dfree the difference in paths starting and ending at the same points.

The image is of an example generator polynomials.
Few generator polynomial are
[1, 1, 1]; [1, 0, 1];
We can also write it as [7, 5].



PSEUDOCODE (ENCODER)

For better understanding purposes I am writing the pseudocode in the ppt please find the matlab code in a seperate pdf

padded_input = {(k-1)*zeros} + {input}

Pad the input bits with zeros at the start .

```
for i = glen:len
     for j = 1:nglen
     jtr = 1;
    sum = 0;
    for k = i-glen+1: i
        if(gen(j, jtr) == 1)
            sum = sum + ip(1, k);
        end
     jtr = jtr +1;
    end
    out(1, itr) = mod(sum, 2);
     itr = itr +1;
     end
```

Iterate over all input bits.

Iterate over all generators

For each bit calculate the output of generators.

and insert them in the output

MODULATION

- Modulation is the process of modifying a carrier signal (usually a high-frequency sine wave) with the information from a lower-frequency message signal so that it can be transmitted efficiently over a medium like air, cable, or fiber.
- Here we used the BPSK modulator to modulate the which maps the bits b $\in\{0,1\}$, to symbols.

$$s = 1 - 2*b$$

- Bits are mapped according to:

$$s = \begin{cases} -1, & b = 1 \\ 1, & b = 0 \end{cases}$$

AWGN

(Additive White Gaussian Noise)

- After the modulation is done the information is transmitted through the AWGN channel.
- During the transmission the noise is introduced the information.
- Here, for simulation purpose let the AWGN introduce a per symbol SNR : $\gamma = E_s / N_o$. The noise power is taken as $\sigma_n = 1 / \sqrt{\gamma}$. Because signal power is taken to be 1.

The fact that we know our noise is AWGN let's us conduct soft decision decoding if we did not know the nature of our noise hard decision decoding would be our go to choice

BPSK AND AWGN SIMULATION CODES:

BPSK

```
function out = bpsk(encodedbits)
        out = zeros(1, numel(encoded_bits_2));
        for i = 1:numel(encodedbits)
            out(i) = 1-2*encodedbits(i);
        end
end
```

AWGN

```
function noisy_msg=pass_msg(s,sigma_n)
            [~,col_s]=size(s);
            %creation
             us= randn([1,col_s]);
             noise=sigma_n * us;
             % Addition
             noisy_msg=s+noise;
 end
```

DEMODULATION

- Demodulation is the process of extracting the original message signal from a modulated carrier wave at the receiver end of a communication system.
- The received signal undergoes demodulation and digitization, where the process of reverse mapping is applied – the received symbols (voltages), denoted as sss, are converted back into binary bits, b

$$b = \begin{cases} 1, & s < 0 \\ 0, & else \end{cases}$$

DECODING

- Convolutional code decoding is the process of recovering the original binary message from the encoded bitstream.
- In convolution codes decoding can be performed by two algorithms:
    1. Fano Algorithm: It is also called sequential decoding. It uses a tree search strategy.
    2. Viterbi Algorithm: Finds the most likely sequence of input bits based on the received output.
- For this project, we mainly focus on the Viterbi Algorithm.

VITERBI ALGORITHM

- There are two techniques to perform the Viterbi algorithm:
    i. Hard Decision Decoding
    ii. Soft Decision Decoding
- In both hard and soft decision Viterbi decoding, two key metrics are used: the branch metric and the path metric.
- Branch Metric: The branch metric is a value that measures how closely a received symbol matches a possible encoded symbol.
- Path Metric: The path metric represents the cumulative cost of a path through the trellis – essentially, how well a particular sequence of state transitions matches the received signal.

- Difference between Hard and Soft Decision Decoding:
  - In Hard Decision Decoding Convert the received analog signal into binary and then convert it into binary using some threshold .This approach simplifies the decoding by reducing the noisy values into a binary stream but it discards the valuable information.
  - In contrast in soft decision decoding It uses the actual received signal without making the hard 0/1 decision. This makes the decoding more complex and intense but it leads to significantly better error performance.

HARD DECISION DECODING(PSEUDOCODE)

Steps:-

- obtain the received sequence.
- initialize state 0 with distance zero and rest states with infinite.
- calculate the distance adding the branch metric which the difference of received codeword and the required codeword for transition.
- we select the branch with the least error for each state.
- Keep repeating the above step till we reach the end of the sequence.
- Upon reaching the end of sequence we choose the element with least metric and backtrack to get our answer

SOFT DECISION DECODING(PSEUDOCODE)

Steps:-

- obtain the received sequence (non-digitized).
- initialize state 0 with distance zero and rest states with infinite.
- The only critical difference comes here where the branch metric is calculated as the Euclidean distance.
- we select the branch with the least error for each state.
- Keep repeating the above step till we reach the end of the sequence.
- Upon reaching the end of sequence we choose the element with least branch metric .
- Then backtrack to get the minimum path metric

EXAMPLE:

Example:- input string = {10110}

R = ½ ; (Rate) **Generator = {7 , 5}(in octal)**

k = 1 ; (No. of input bits per time step) **= { [111] , [101] } (in binary)**

n = 2; (No. of output bits per time step) **g1 g2**

K = 3; (Constraint length) **Generator polynomials= {ut⊕ut-1⊕ut-2 , ut⊕ut-2}**

Input	Shift reg.	g1	g2	Output
1	100	1 ⊕ 0 ⊕ 0 = 1	1 ⊕ 0 = 1	11
0	010	0 ⊕ 1 ⊕ 0 = 1	0 ⊕ 0 = 0	10
1	101	1 ⊕ 0 ⊕ 1 = 0	1 ⊕ 1 = 0	00
1	110	1 ⊕ 1 ⊕ 0 = 0	1 ⊕ 0 = 1	01
0	011	0 ⊕ 1 ⊕ 1 = 0	0 ⊕ 1 = 1	01

- After the Encoding of the input bit stream the encoded bits are passed through the BPSK Modulator.
  such that:
- Encoded Bits: {1110000101}
- Now, Each bit is mapped to the symbols using the below formula:

$$s = 1 - 2 * b;$$

s - symbols; b - bits{0,1}

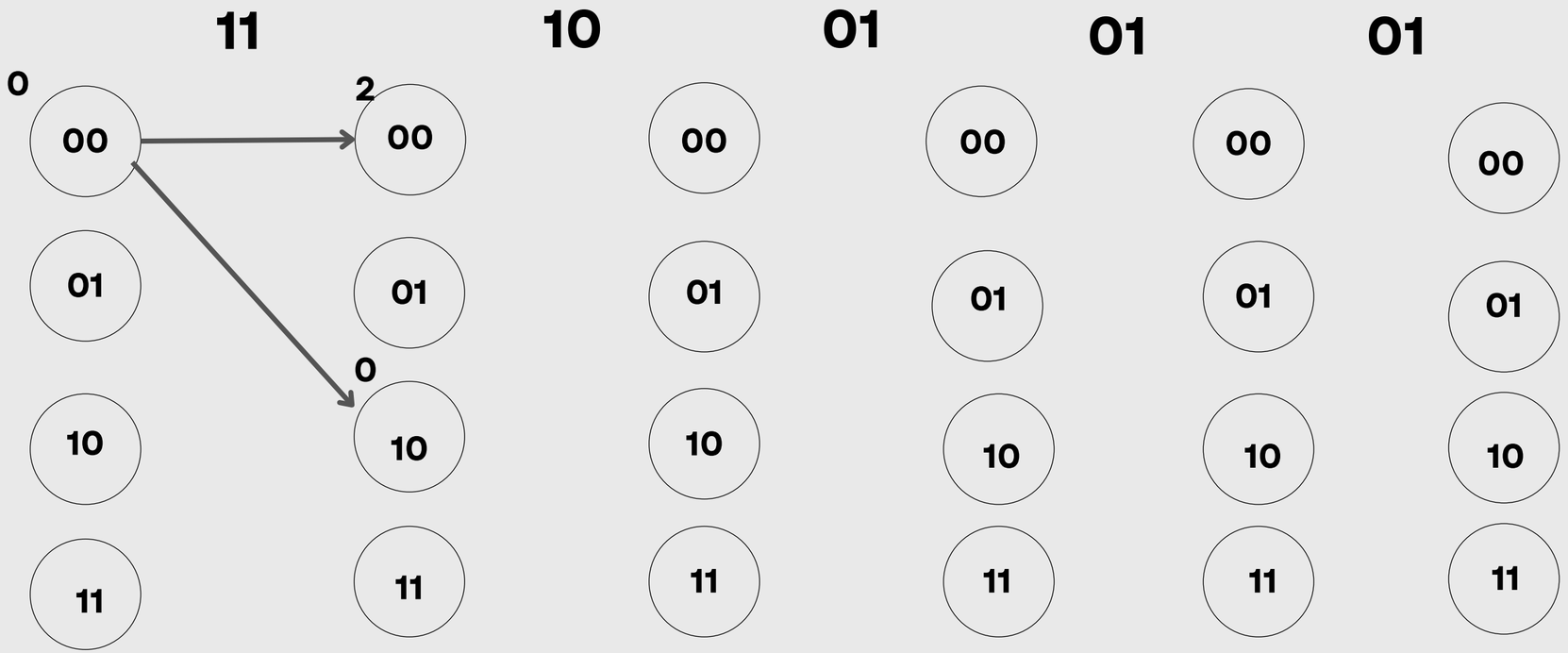
- So the BPSK Modulated Signal = {-1 -1 -1 1 1 1 1 -1 1 -1}

- After this the for the transmission of the signal takes place with the help of AWGN(Additive White Gaussian Noise) channel. Due to which the noise is introduced in the transmitted signal.
- Noise: {-0.1, +0.8, -0.3, +0.1, -0.2, -1.1, -0.4, +0.2, +0.2, -0.3}
- We get the noisy received signal as = Noise + BPSK
- The noisy signal is as follow: {-1.1, -0.2, -1.3, 1.1, 0.8, -0.1, 0.6, -0.8, 1.2, -1.3}

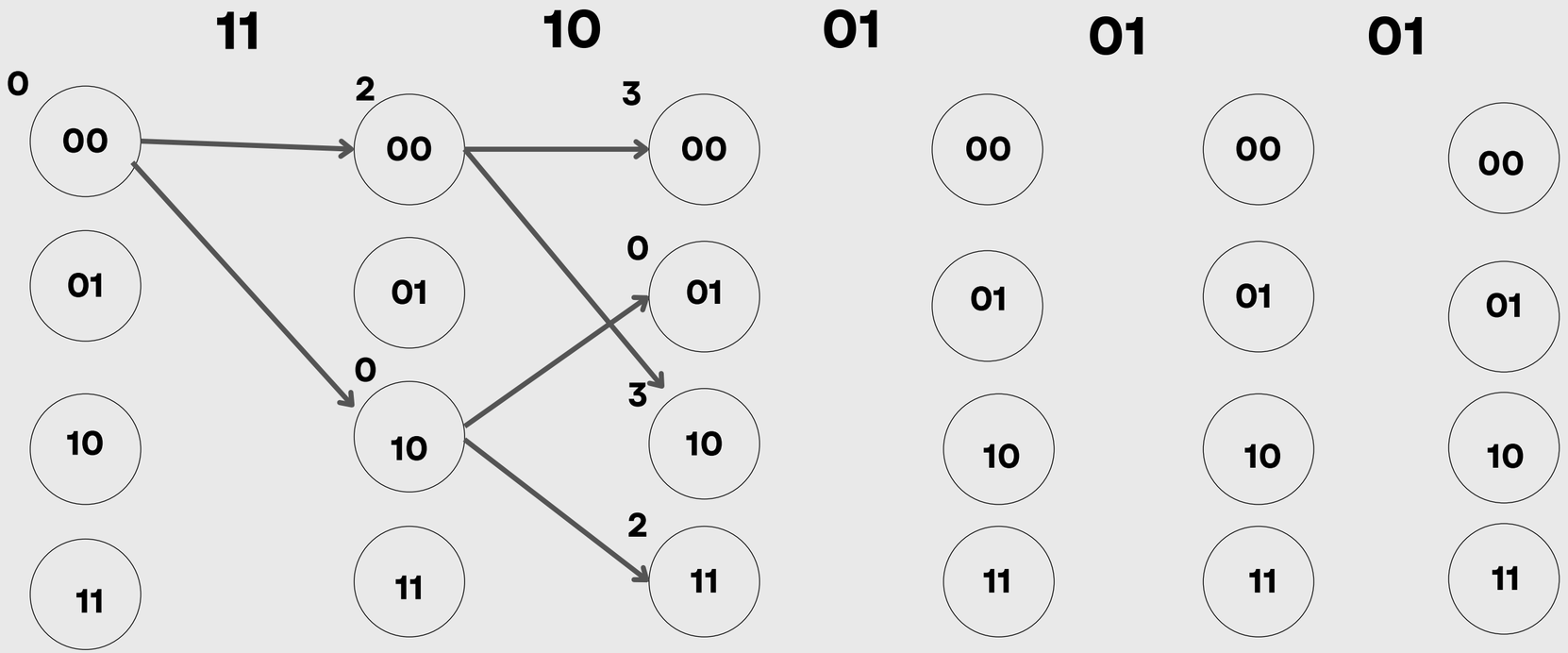
- For the decoding part, we can decode it in two techniques hard and soft decision decoding.
- here, we are using hard decision decoding.
- In hard decision decoding first of all the noisy signal is in converted in to binary by demodulation. And then with the help of trellis diagram the decoded codeword is found.
- here the demodulated codeword of the noisy signal is:

Demodulated bits = {1110010101}

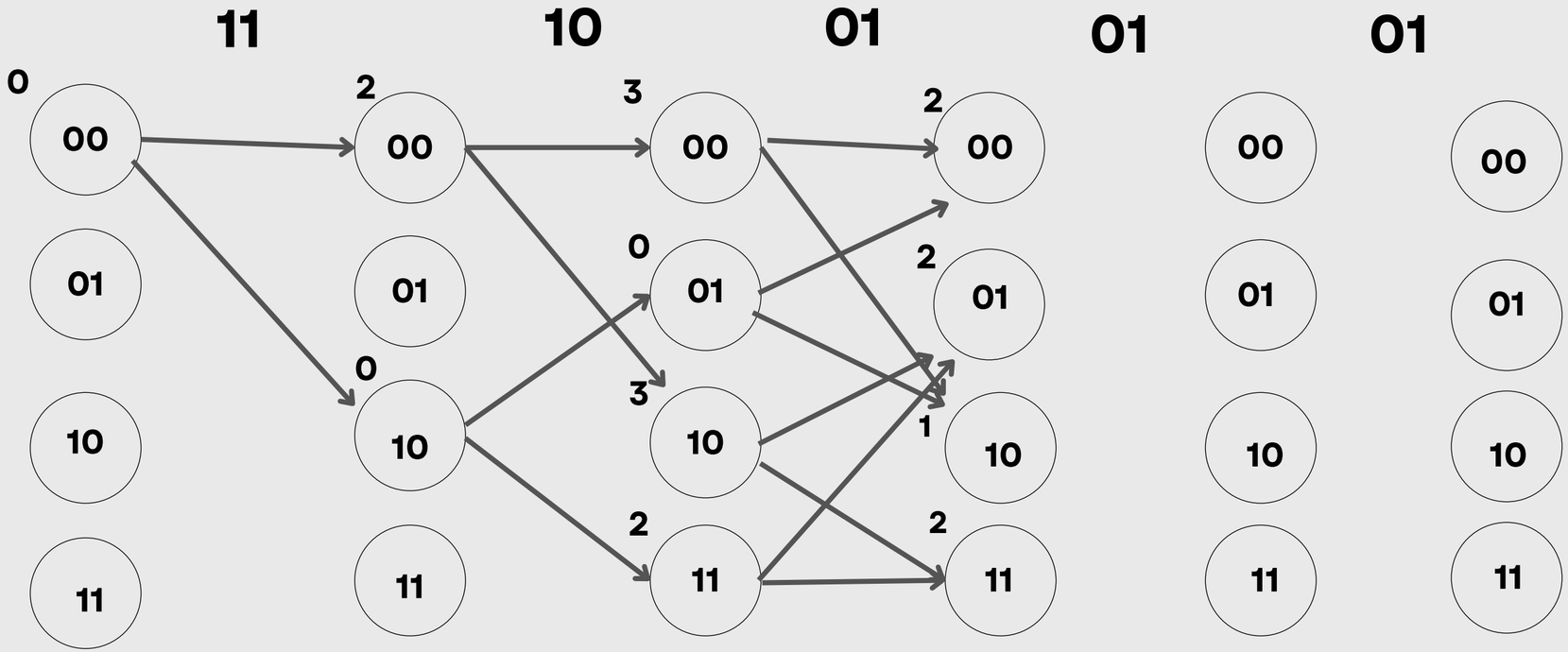
HARD DECISION DECODING



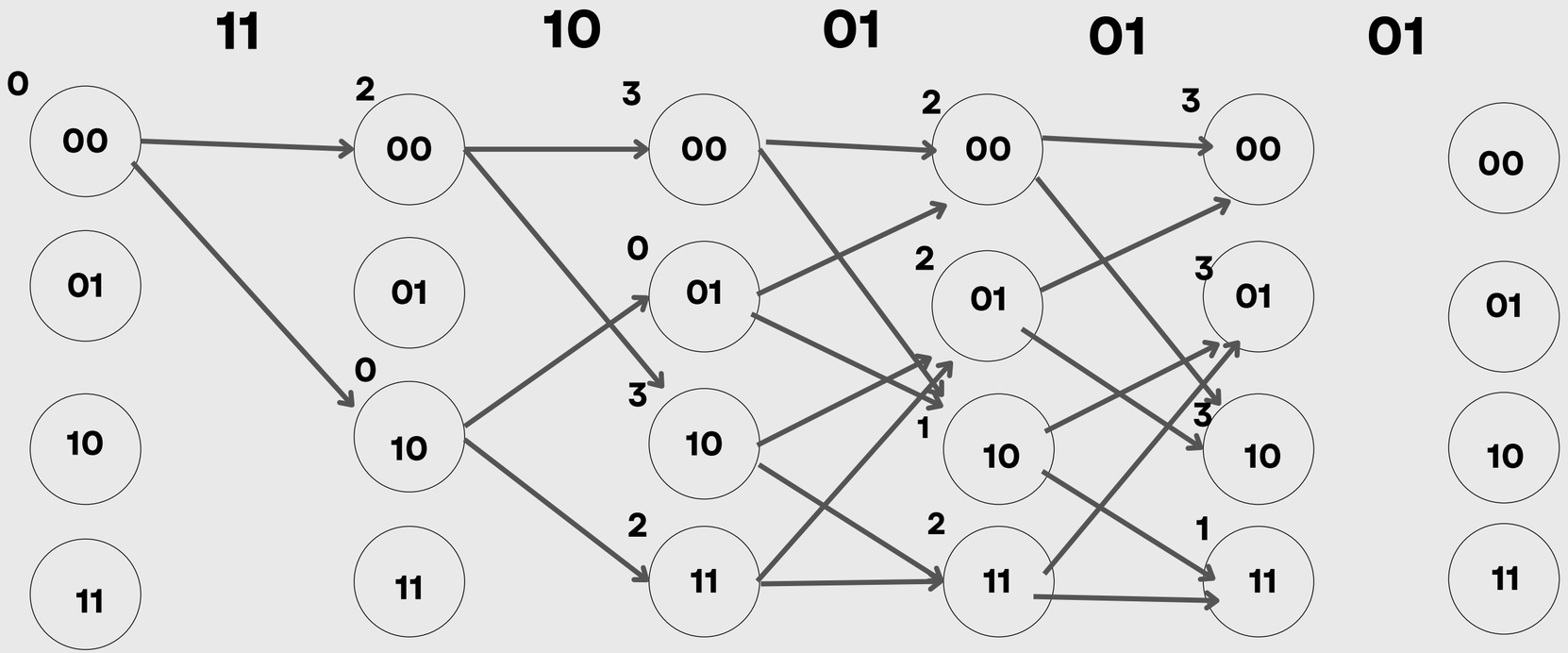
HARD DECISION DECODING



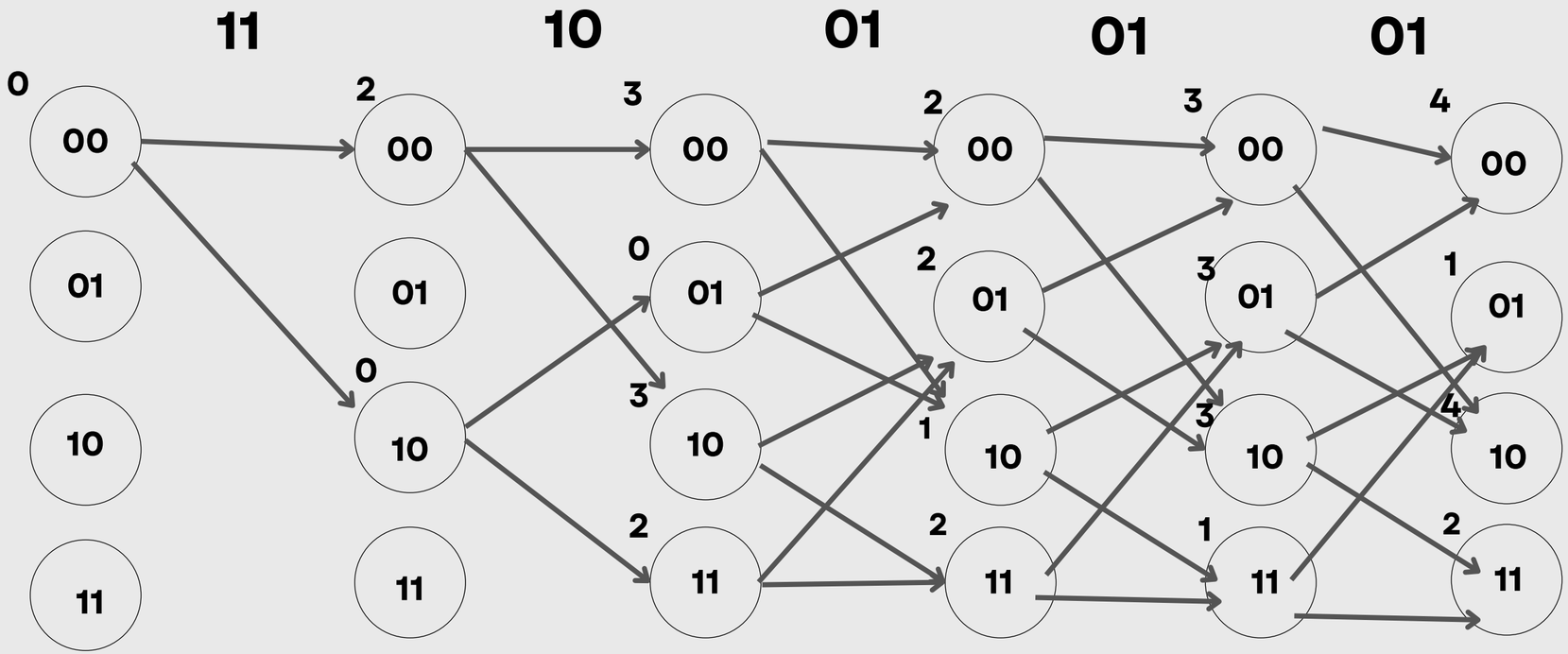
HARD DECISION DECODING



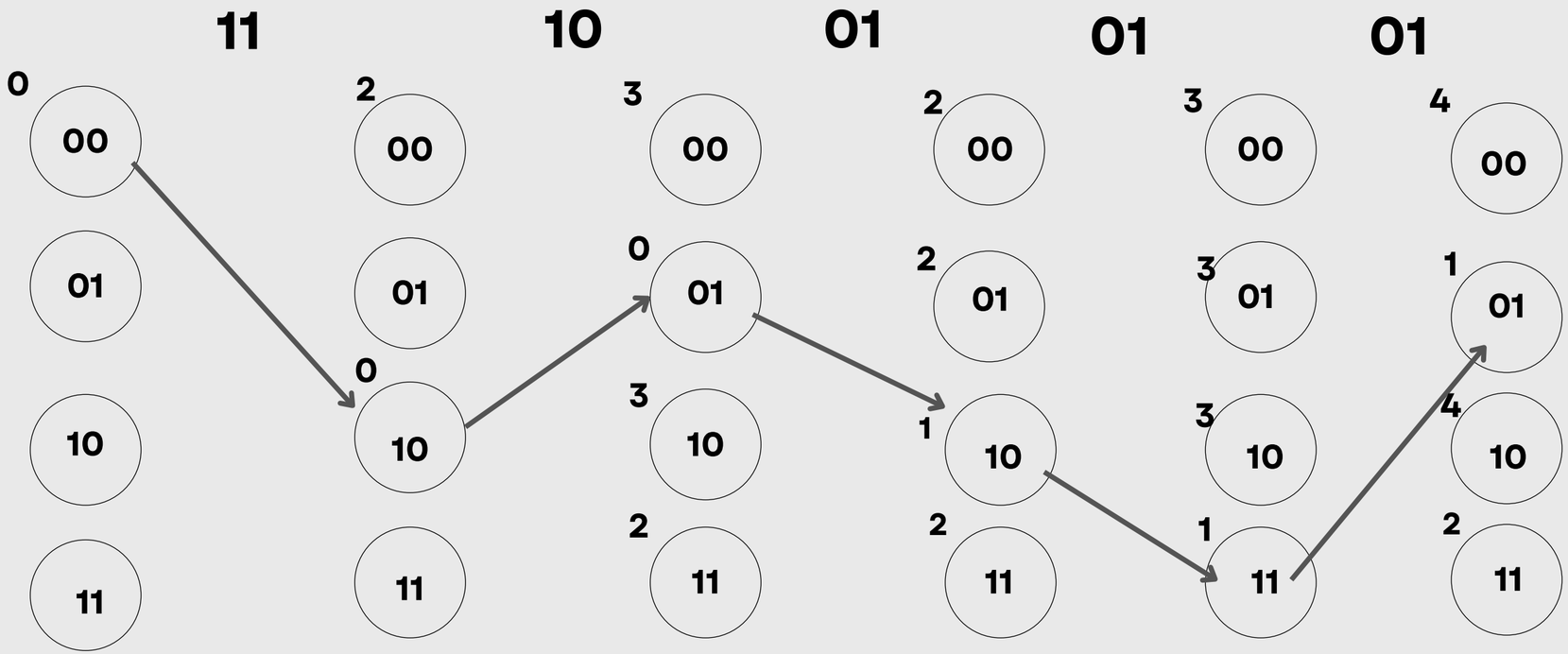
HARD DECISION DECODING



HARD DECISION DECODING

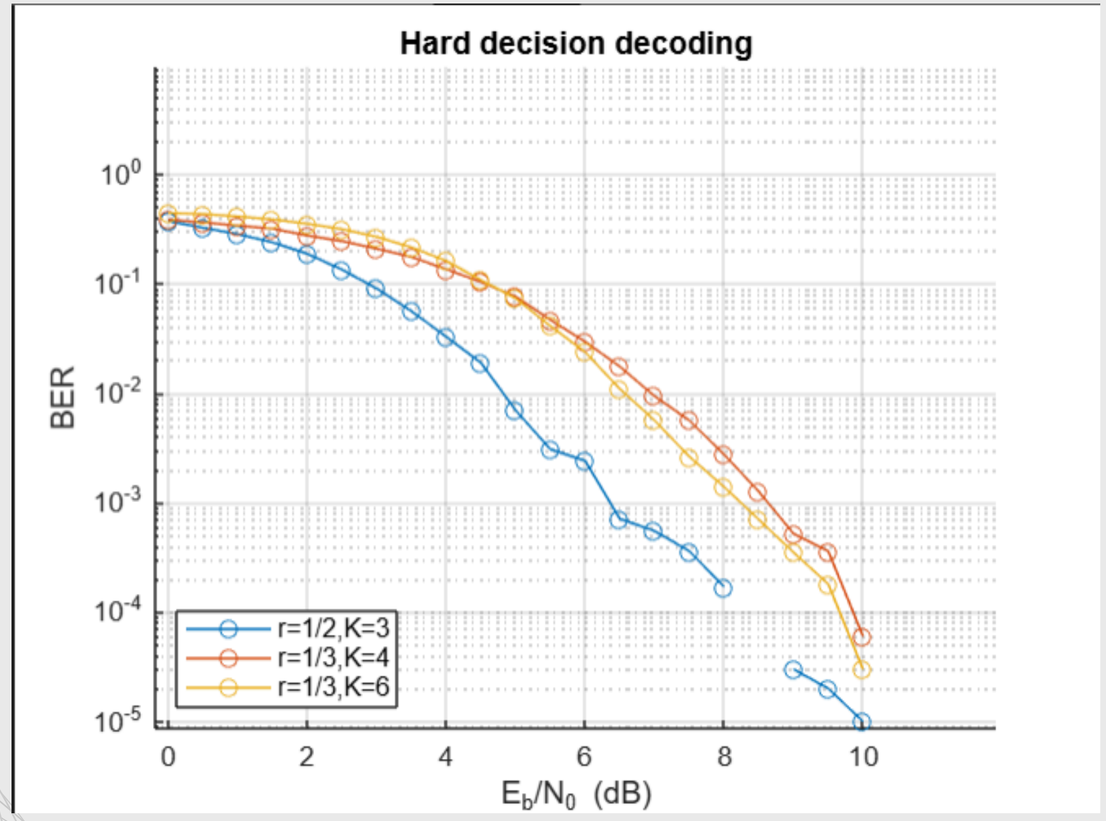


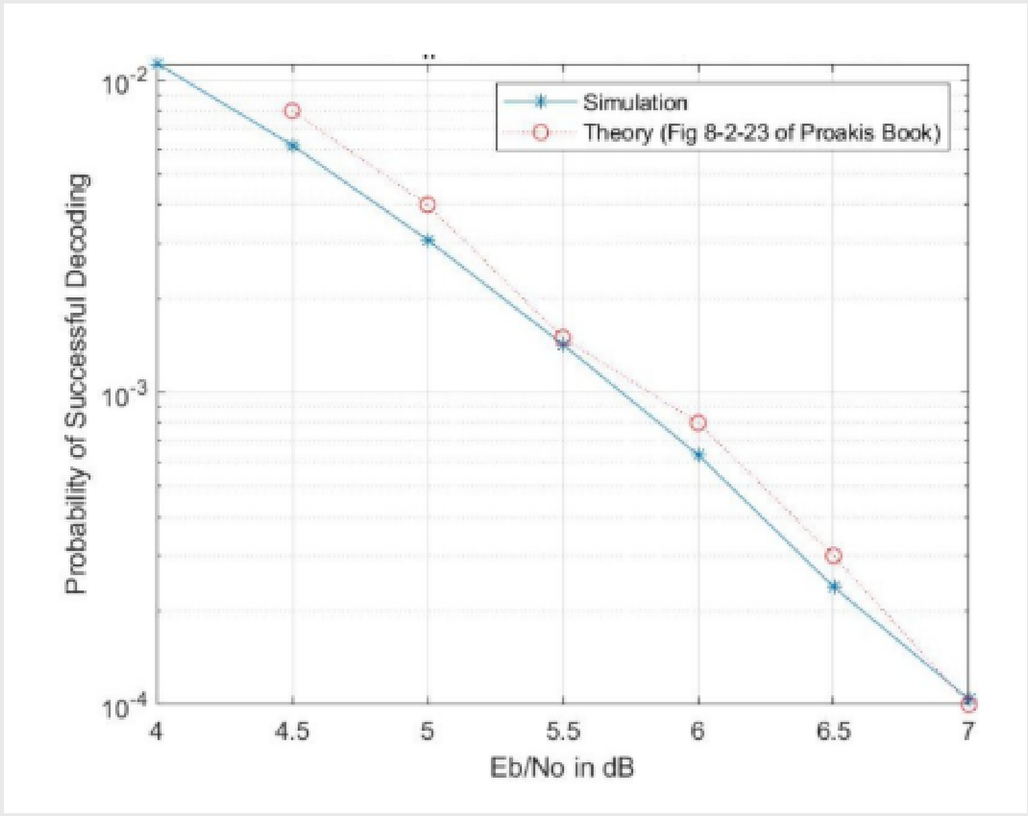
HARD DECISION DECODING



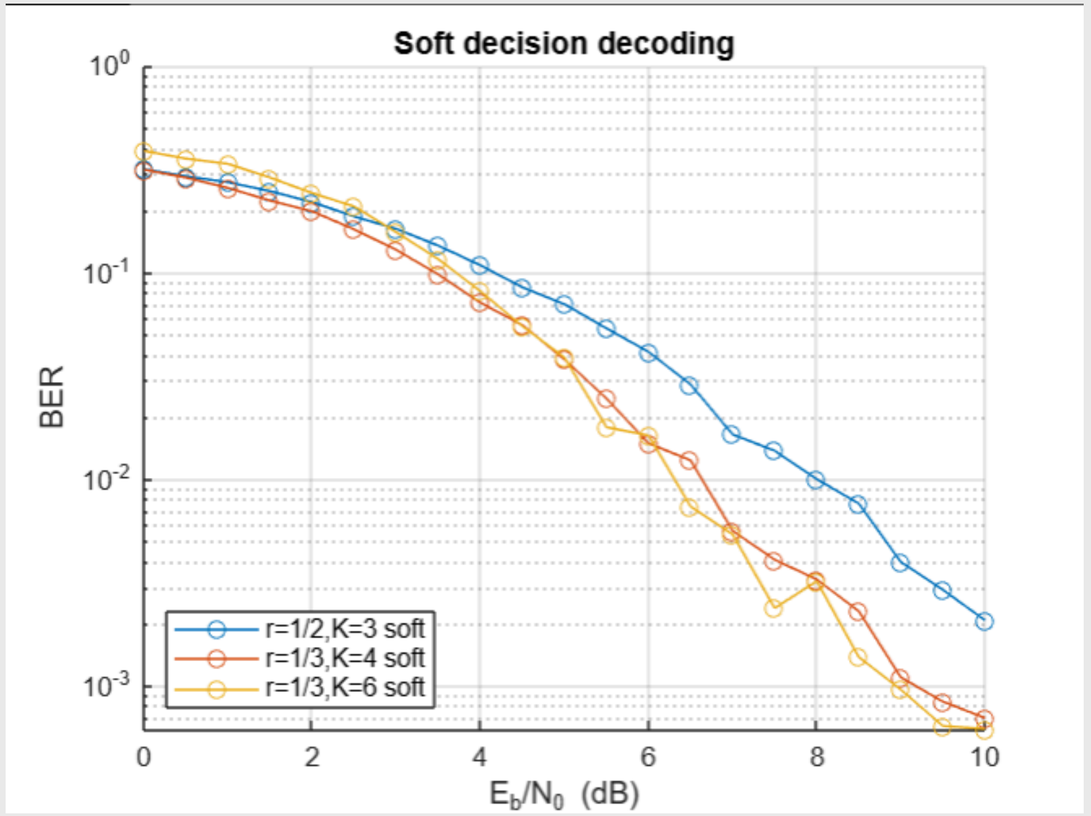
Decoded sequence :- 10110

SIMULATION


Hard decision decoding
BER
E_b/N_0 (dB)
r=1/2,K=3
r=1/3,K=4
r=1/3,K=6


Probability of Successful Decoding
Eb/No in dB
Simulation
Theory (Fig 8-2-23 of Proakis Book)

SIMULATION


Soft decision decoding
BER
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
2
4
6
8
10
E_b/N_0 (dB)
r=1/2,K=3 soft
r=1/3,K=4 soft
r=1/3,K=6 soft

Observations

For hard decision decoding:

- From the plots in the previous slide the we can observe that the BER(bit error rate) is decreases as the Eb/N0 increases.
- This indicates that it shows better performance at higher SNRs. As it should because our signal power is constant at 1 and noise power is decreasing.
- At lower SNR, the BER is high, indicating more decoding errors.

For soft decision decoding:

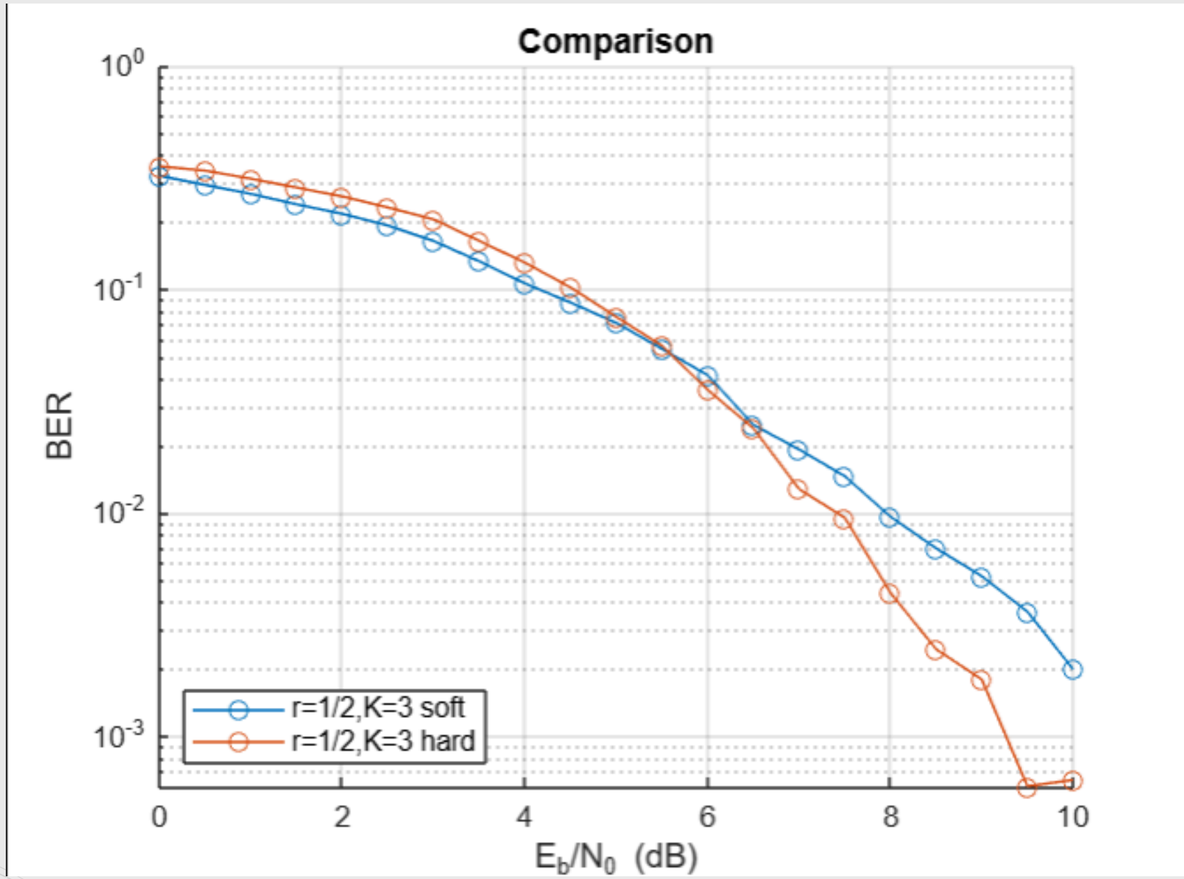
- BER are is consistently lower than the hard decision case across all Eb/N0 values.

Observations

Comparing both hard and soft decision decoding:

- According to the observations soft decision decoding outperforms hard decision decoding.
- We can say that soft decision decoding is more effective than hard decision decoding for convolutional codes over AWGN channels due to better use of channel information.

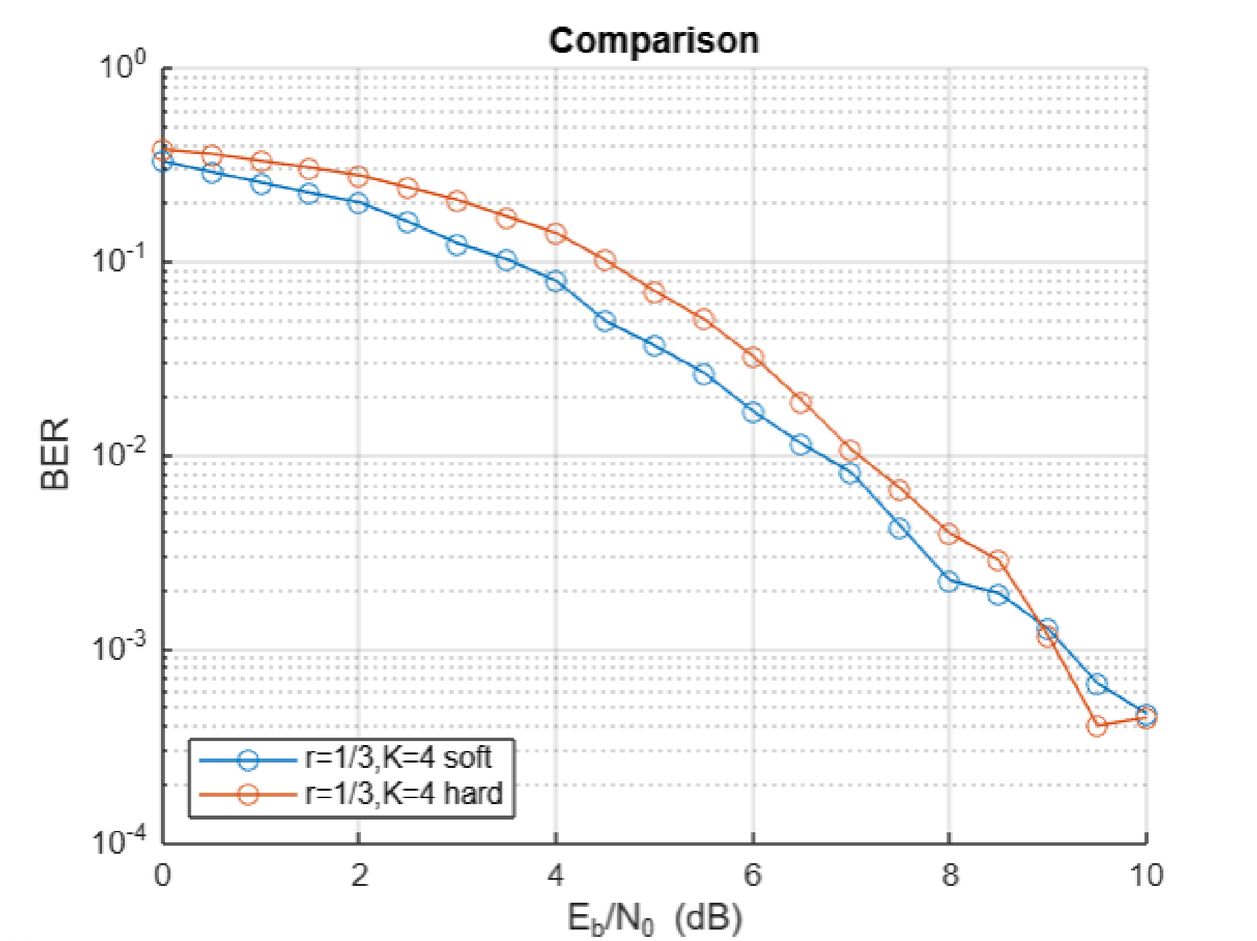
SIMULATION



Observation

Soft decision decoding performing better than hard decision for
r = ½ & k = 3
till SNR = 5
then hard decision starts to perform better

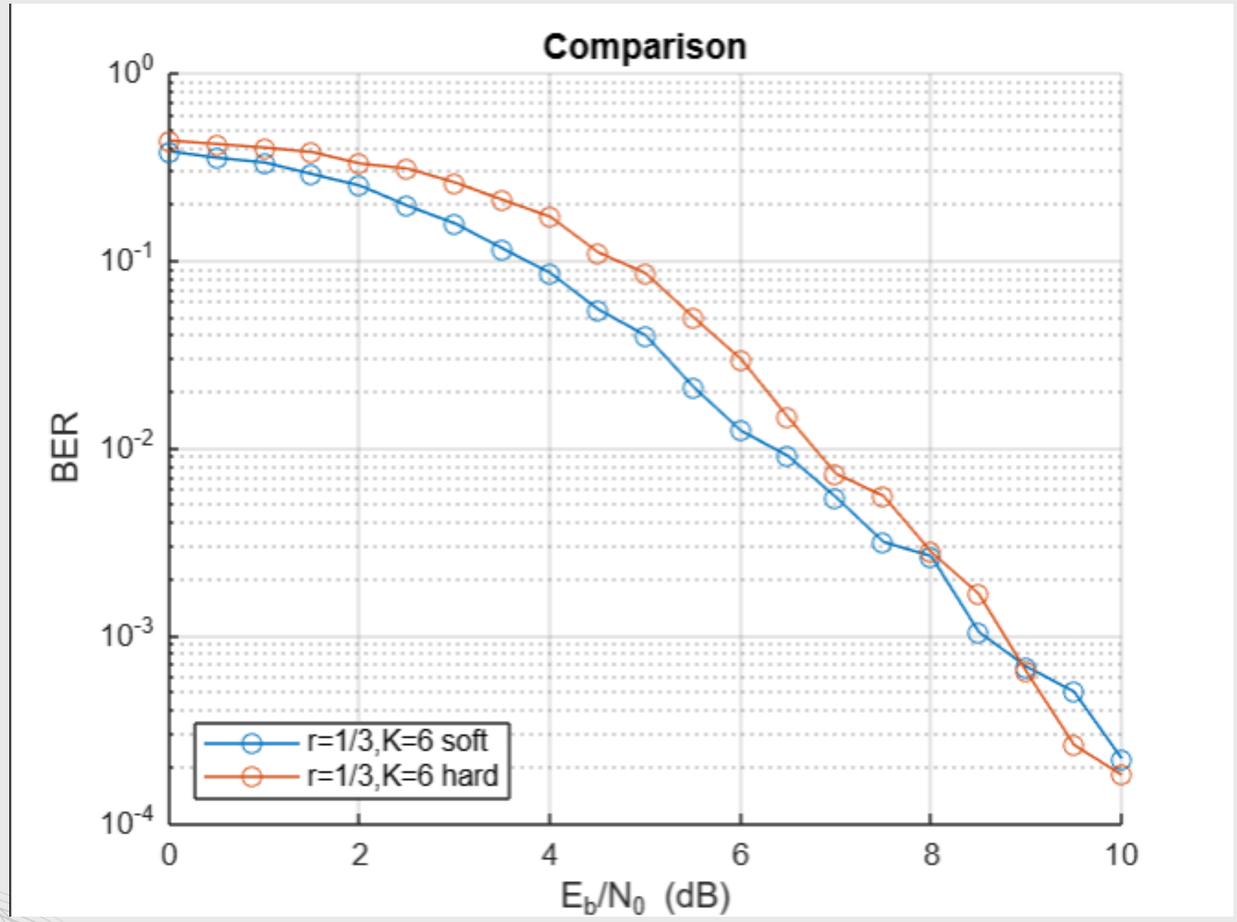
SIMULATION



Observation

Soft decision decoding performing better than hard decision for r = ⅓ & k = 4

SIMULATION



Observation

Soft decision decoding performing better than hard decision for r = ⅓ & k = 6

Analysis of soft decision decoding

The idea behind soft decision decoding is to make use of the fact that our noise is gaussian and using this information we can obtain information that how much a signal is likely to be 1 or 0.

For example :- if we are using BPSK and we receive two signals 0.2 and 0.8.

Hard decision decoding we say that both are closer to 0 we simply convert them to 0, 0 symbols.

But we know that since the noise we are encountering is **additive white gaussian noise** we can say that 0.8 is more likely to be one as compared to 0.2.

Analysis of soft decision decoding

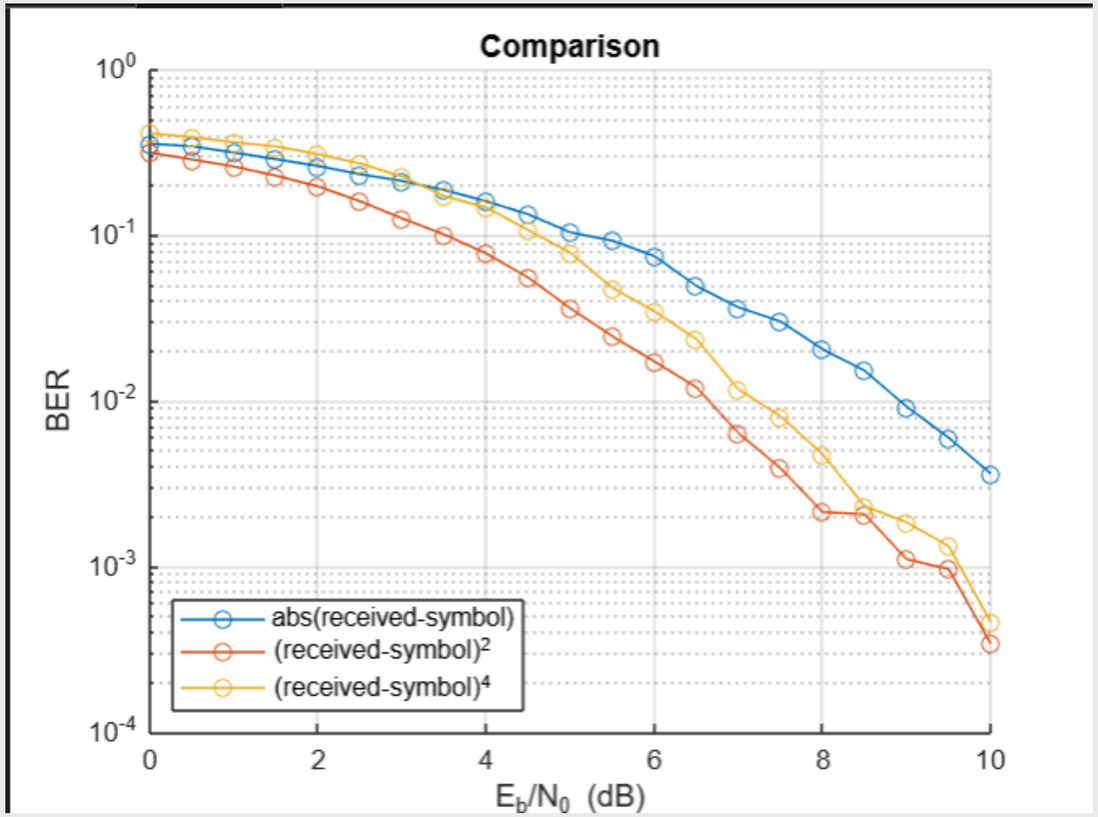
The penalty calculation for soft decision could be done using any of these

- abs(received-symbol)
- $(\text{received-symbol})^2$
- $(\text{received-symbol})^e$ where e is even

I have tried a few different penalty calculations in the following slide and their results when we have AWGN channel in convolution code. We can choose the one with the lowest BER.

We cannot use odd powers because negative and positive errors can cancel each other out to give a lower error than actually present.

PENALTY FUNCTIONS FOR SOFT DECISION DECODING



The (received-symbol)^2 penalty function performs the best.
Here I have kept the generator polynomial to be
[1,1,1]; [1,0,1]
r= ½ k = 3
across all the three plots varying the penalty in soft decision decoding.

TRANSFER FUNCTION

The transfer function of a convolution function that:

- Counts all possible error paths in the trellis.
- Helps in analyzing the distance properties of the code.

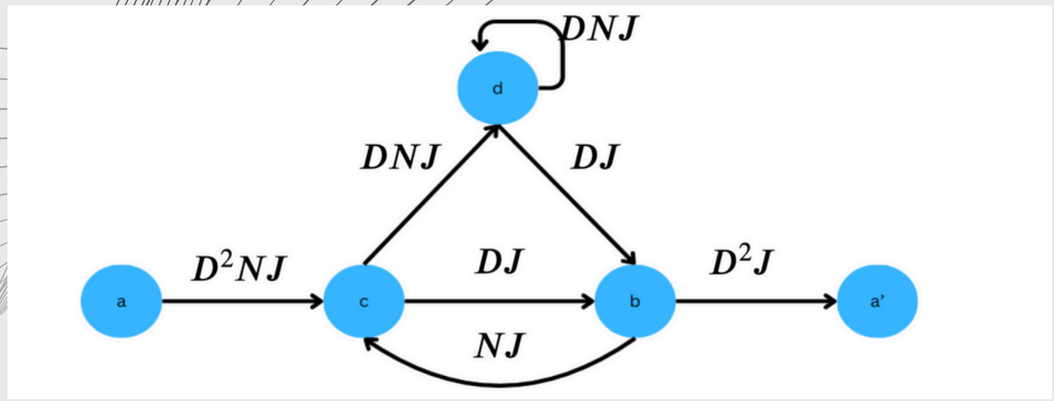
Use:

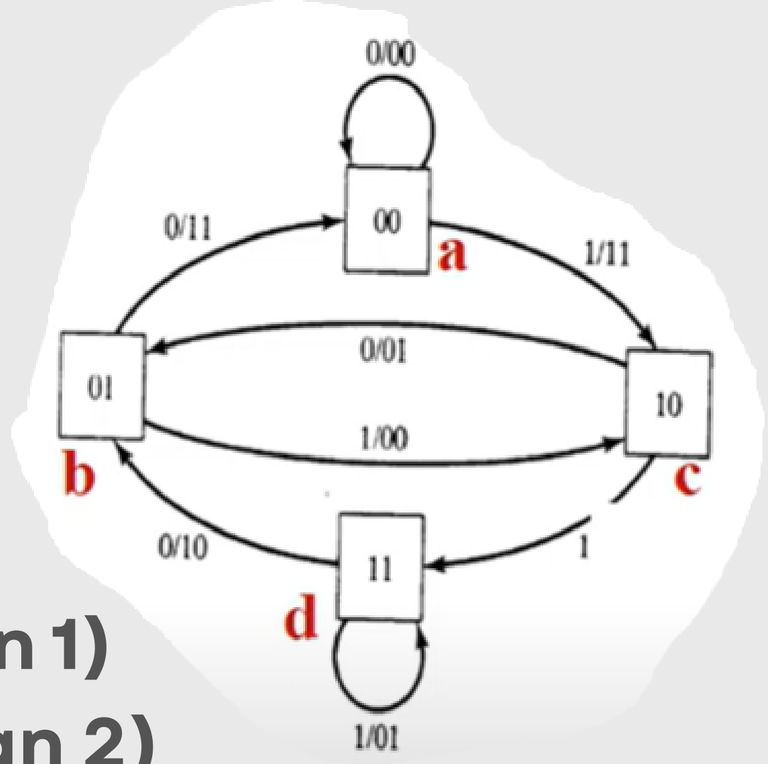
- You can extract the minimum Hamming distance (d_{min}) from the transfer function.
- d_{min} is the smallest weight path (excluding the all-zero path) – this indicates how good the code is at error detection and correction.

TRANSFER FUNCTION FORMULA

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

- d = number of 1s in the output codeword (Hamming weight),
- f(d) = number of 1s in the input (k-bits),
- g(d) = number of branches (or transitions) spanned by the path,
- D,N,J = dummy variables (used for counting purposes)
- a_d = number of paths with weight d.





- **System of Equations**

  1.State d: d = DNJc + DNJd **(eqn 1)**

  2.State c: c = D2NJa + NJb **(eqn 2)**

  3.Output(a′): $a' = D^2Jb$ **(eqn 3)**

  4.State b: b = DJd + DJc **(eqn 4)**

  Transfer Function = **a′/a**

  After solving equations 1, 2, 3 and 4,

  T(D,N,J) = $D^5NJ^3 + D^6N^2J^4 + D^6N^2J^5 + D^7N^3J^5 + \ldots$

USES OF CONVOLUTION CODES

- GSM and 3G cellular networks
- NASA’s Voyager spacecraft
- Wi-Fi (older standards like 802.11a/b/g)
- Satellite and deep space communication systems.

Convolution coding is good and intuitive for streaming data where length is not fixed. As convolution codes also do not require to get the whole data simultaneously. This may also significantly decrease the memory requirements.

BIBLIOGRAPHY

- WIKIPEDIA
- Convolution Code at MIT
- Teaching Convolution Code by Davoud Arasteh

THANK YOU