CT-216 Introduction to Communication Systems

Convolution Code

Lab Group 2

April 15, 2024

Prof. Yash Vasavada

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.

1. Divyarajsinh Chundavat – 202201155
2. Meet Katharotiya -202201157
3. Kishan Patel – 202201159
4. Smit Godhani-202201162
5. Jay Goyani-202201163
6. Krisha Brahmbhatt – 202201164
7. Het Gandhi – 202201167
8. Jyot vasava – 202201169
9. Parshwa Modi – 202201165
10. Parth Vadodaria – 202201174
11. Harshvardhan Vajani – 202201413

Outline

- Problem Statement
- Objective
- Introduction to convolution
- Concept of encoding
- Modulation and AWGN simulation
- Demodulation
- Concept of decoding
- Simulation results
- Observation from plots
- Summary
- References

Problem Statement

- This project is about analyzing the effectiveness of **convolutional codes** in correcting errors in digital communication systems.
- We will also analyze how well soft and hard decision decoding for convolution performs under different SNRs.
- Plot the curves depicting the probability of detection errors for convolution codes with
  i) rate=1/2 K=3
  ii) rate=1/3 K=4
  iii) rate=1/3 K=6
  using both Viterbi hard and Viterbi soft decoding for varying values of SNR(in dB) from 0 to 10 in the steps of 0.5.
- Observe the performance of both decoding techniques and compare these simulation results with their respective analysis results.

Objective

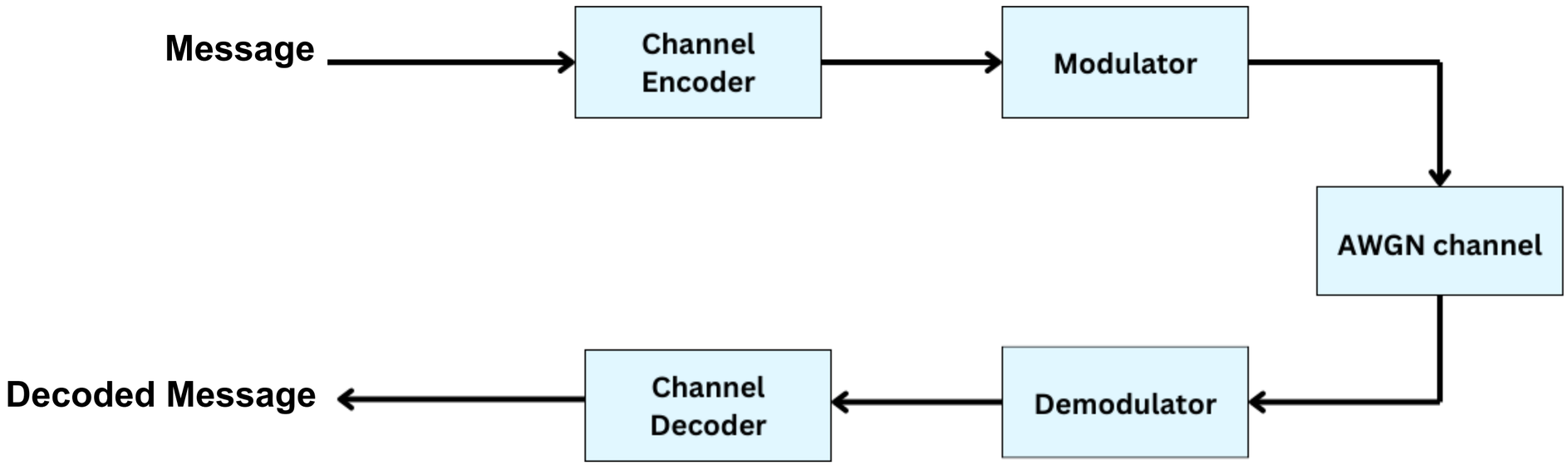
Our objective is to understand the channel coding through Convolution Codes in the digital communication. Here, the scope of the digital communication is limited to encoding, modulation, demodulation and decoding.

More Specifically:-

- Encode the transmitted message for each of the above described rate and constraint length.
- Modulation of the encoded bits using BPSK Modulator.
- Addition of the noise when the message passes through the AWGN channel for simulation purpose. In reality, when the message passes from transmitter side to receiver side some noise is introduced in it.
- Decoding the received message with two decoding techniques: viterbi hard and viterbi soft.
- Plotting the graph for probability of detection errors which demonstrates the performance of both decoding techniques for different rates and constraint length.
- Comparison of the simulation results of soft and hard decision decoding with their respective analysis results.

Introduction to Convolution Codes

- Convolutional codes were invented by *Elias* in 1955.
- Convolution codes coding method in digital communication involves the transmission of parity bits which are generated from message bits.



Concept Of Encoding

- Convolution Codes involve the transmission of the parity bits rather than transmitting the message followed by the parity bits.
- The encoder serves the purpose of generating these parity bits. The encoder uses a sliding window of size K (constraint length) to calculate r >1 parity bits by combining various subsets of the window which are determined by the generator sequences.
- 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.
- The length of the encoded message will be r*(k+K-1) where r=generated number of parity bits/number of generator polynomials, k=number of message bits, and K=constraint length.
- The rate of the convolution code will be 1/r.
- The trade-off involved in convolution code is that the longer the constraint length, the greater the resilience to bit errors but the longer the time taken to encode and decode the message.

Encoder Concept

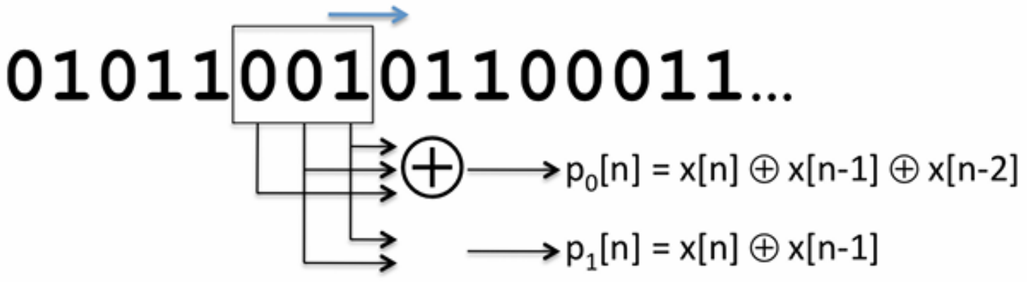


Fig-1: An example of a convolutional code with two parity bits per message bit (r =2) and constraint length K=3.

- Views of Convolution encoder

1. Block Diagram
2. State Diagram
3. Trellis Diagram

** Figure taken from Lec-8 MIT*

Encoder Concept

- Convolutional encoder has some 'k' inputs and 'n' outputs where n≥k.
- The input message is split into 'k' inputs. Each input has length equal to (length of message/k).
- The output has length equal to (length of received message/n).
- The inputs are convolved with generator sequences to give outputs.
- The block diagram uses shift registers. This diagram has two generator sequence and generates two outputs.
- In the state diagram, a new state can be obtained by shifting the current state according to the constraint length and appending 1 & 0 whichever be the next input bit. There are a total of $2^{k(K-1)}$states.

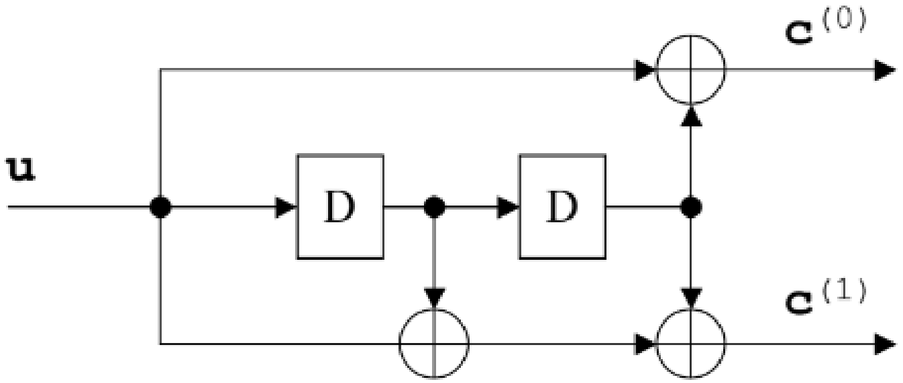


Fig-2: Block diagram view for K=3 rate=1/2

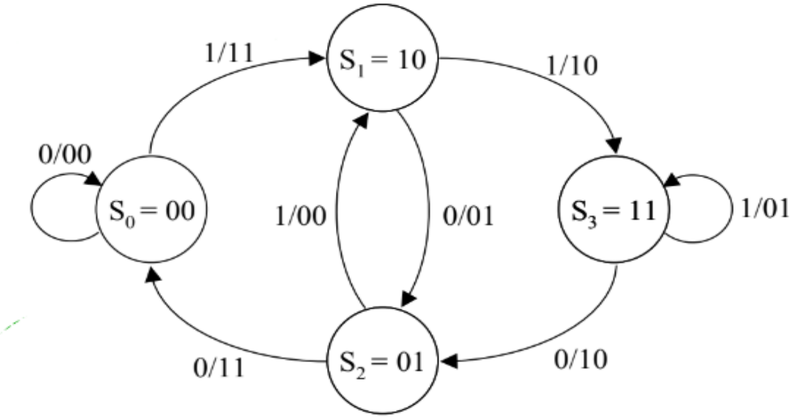


Fig-3: State diagram view for K=3 rate=1/2

** Figure taken from Lec Slide*

Encoder Concept

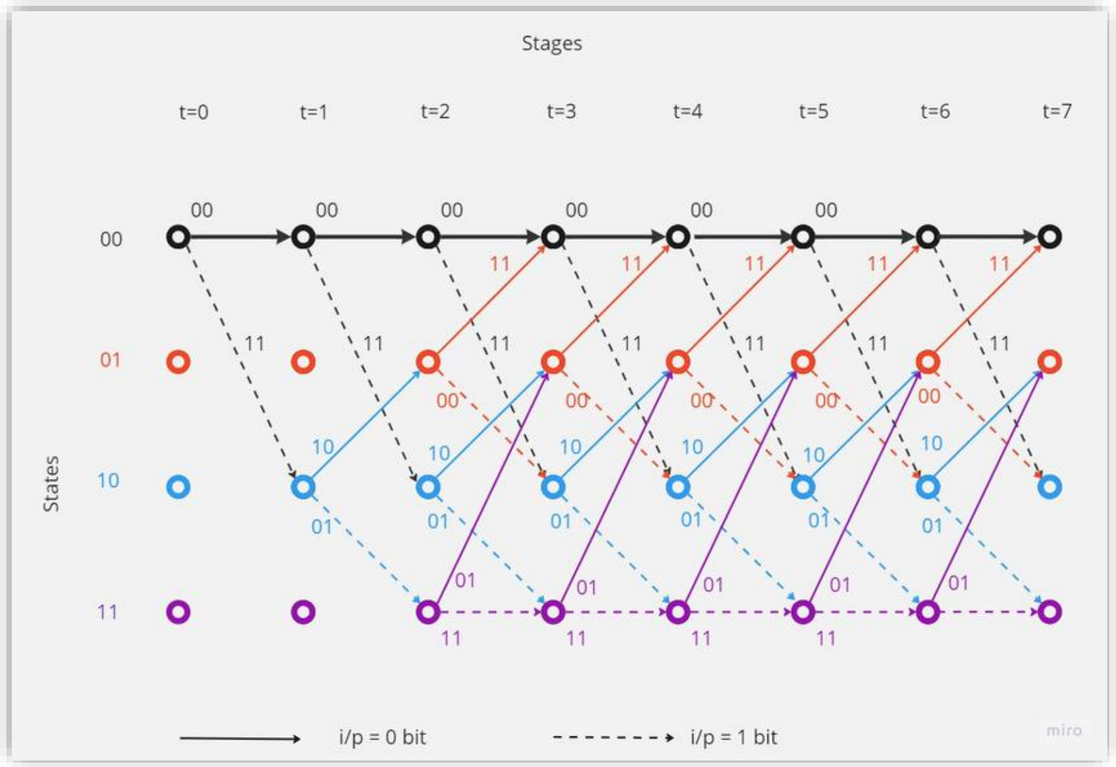


Fig-4: Trellis view for K=3 rate=1/2

- The trellis diagram is a redrawing of the state diagram. it shows all possible state transitions at each time.
- The trellis diagram can be used to decode the received parity bits which come as output of the encoding. In trellis diagram 2^kbranches enter each state and 2^kbranches leave each state.

** Image taken from research gate*

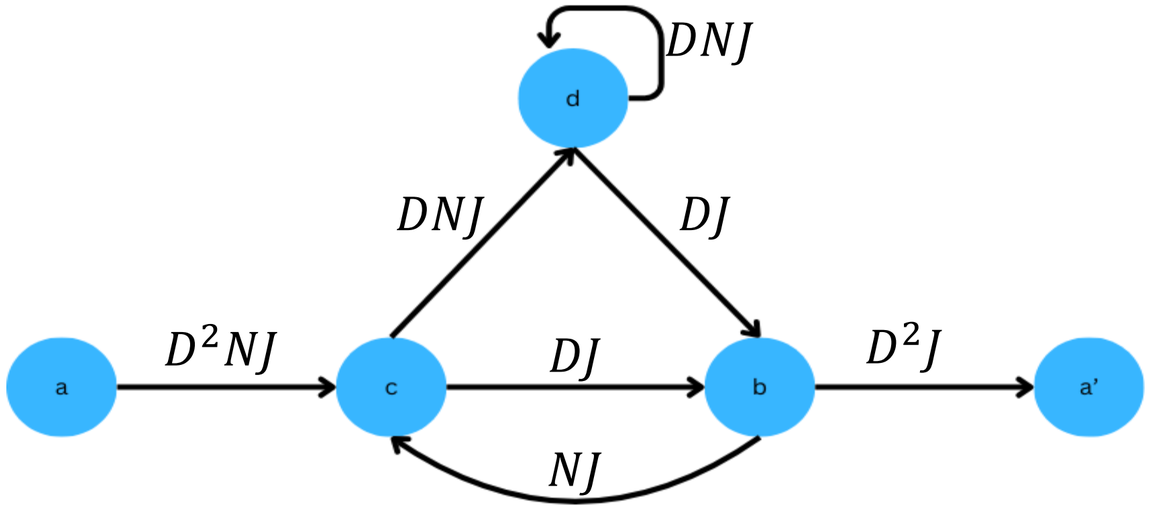
Transfer function

- A convolutional code's distance properties are examined using a transfer function.

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

- 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

- From figure 3 we obtained signal flow diagram as given below:



Transfer function

- Provides the properties of all the paths.
- Provides minimum Hamming Distance (smallest power of D of transfer function).

$$\mathrm{d} = DNJc + DNJd \quad - \ eqn\ 1$$
$$c = D^2NJa + NJb \quad - \ eqn\ 2$$
$$a' = D^2Jb \quad - \ eqn\ 3$$
$$b = DJd + DJc \quad - \ eqn\ 4$$

➢ $transfer\ function = \frac{a'}{a}$

- After solving Eqn-1,2,3 and 4 we get

$$\frac{a'}{a} = \frac{D^5NJ^3}{1 - DNJ - DNJ^2}$$

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

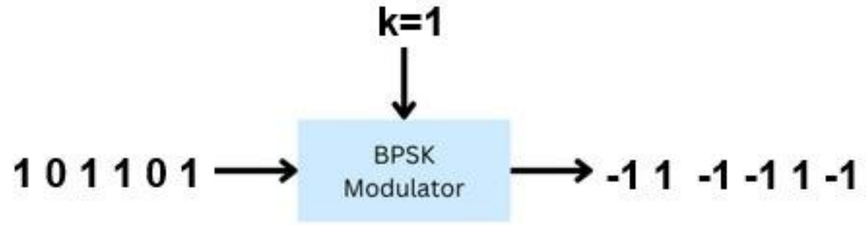
Modulation and AWGN Simulation

- Modulation involves altering certain properties of carrier signals that is amplitude, frequency and phase. It is generally required for long distance communication.
- The output of the channel encoder, which will be binary bits, b $\in\{0,1\}$, is further passed to BPSK modulator which maps the bits to symbols (voltage).

$$s = 1 - 2 * b$$

- Bits are mapped with symbols in such a way

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



- After the modulation, the information is passed through the AWGN (Additive White Gaussian Noise) channel to the receiver side.
- The channel introduces the noise to the information.
- Here, for simulation purpose let the AWGN introduce a per-symbol SNR $\gamma = \frac{E_S}{N_0}$ in linear scale. The noise power is taken as $\sigma_n = 1/\sqrt{\gamma}$. Both of them will be used to create noise which will be added to the information.

BPSK and AWGN Simulation

- BPSK Simulation

```
function
s=create_symbols(encoded_msg)
    s=1-2*encoded_msg;
 end
```

- AWGN Simulation

```
function noisy_msg=pass_msg(s,sigma_n)
    [~,col_s]=size(s);

    % Noise Creation
    us= randn([1,col_s]);
    noise=sigma_n * us;
    % sigma_n=1/sqrt(rate*k*EbNo)

    % Noise Addition
    noisy_msg=s+noise;
 end
```

Demodulation

- Demodulation is the reverse process of modulation i.e. extracting the information from the modulated signal.
- The received information is demodulated / digitized i.e. reverse mapping, symbols (voltage), s are converted to binary bits, b. Here, we have used BPSK modulator hence,

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

- Function for digitizing the received message

```
function digitized_bits = digitize(received_bits)
   digitized_bits = (received_bits<0)
 end
```

Concept of decoding

- Decoding is the process of converting the received codeword into an expected transmitted message.
- There are two commonly used decoding algorithms for convolution codes.

1. Fano algorithm
2. Viterbi algorithm

- For this project, we mainly focus on the Viterbi algorithm.
- The **Viterbi algorithm** is a Dynamic Programming Algorithm which was proposed by Andrew Viterbi in 1967 as a decoding algorithm for convolution code over noisy digital communication links.
- We are using two types of decoding techniques viz, hard decision and soft decision decoding
- For both hard and soft decision Viterbi decoding, we use two types of metrics: branch metric and path metric.

**history of Viterbi algorithm taken from Wikipedia*

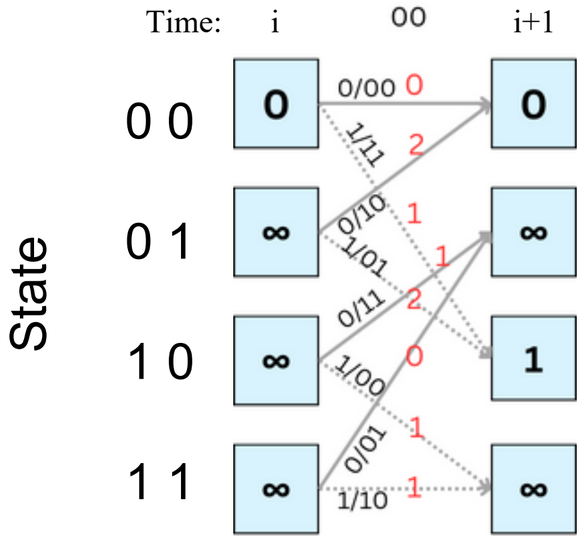
Hard Decision Decoding

- For decoding purposes in convolution codes, the trellis provides a convenient way to understand the time evolution of the state machine.
- Let us define two metrics which we discussed about in previous slide for hard decision decoding.

1. **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)
2. **Path Metric (PM):** PM is the sum of the branch metric of the branches it traversed in the path.

$$PM[s, i+1] = \min(PM[\alpha, i] + BM[\alpha \rightarrow s], PM[\beta, i] + BM[\beta \rightarrow s])$$

(Where α , β are the states at time I and s is the state reachable from α and β at time i+1)

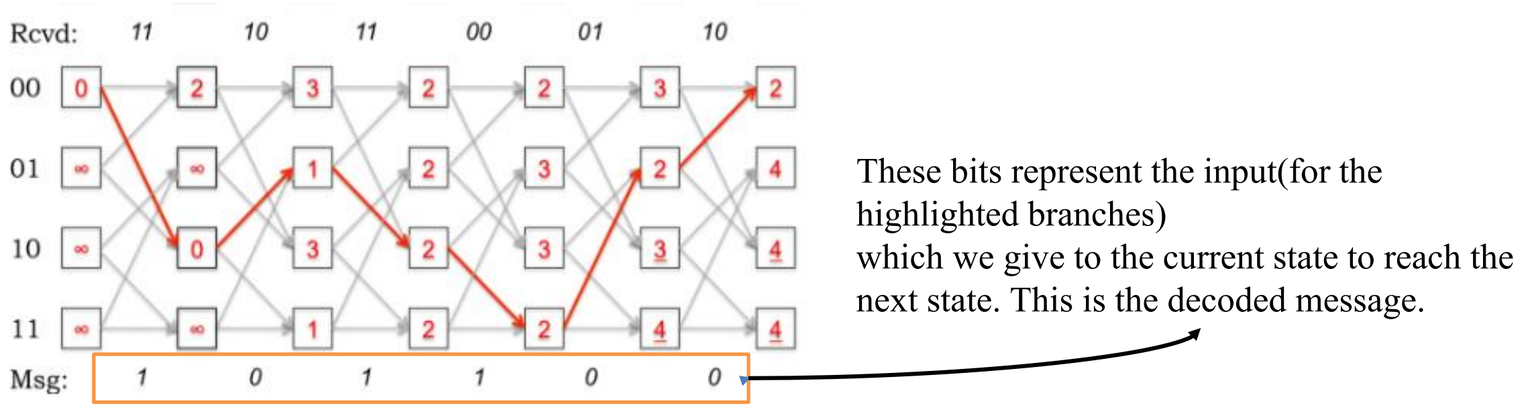


Hard Decision Decoding

- Now 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:

  1. Take the sum of the branch metric and the path metric of the current state
  2. Compare the sum that we got from two previous states α and β and assign 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.



** Figure taken from Lec-9 MIT*

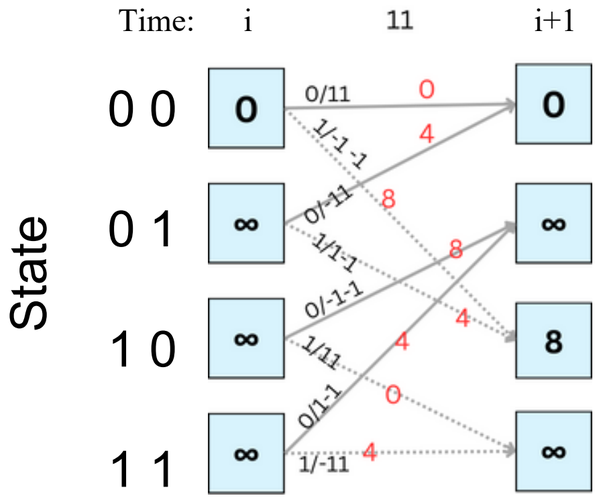
Pseudocode for Backtracking(hard)

```
function viterbi_decode(g, received_bit)
    Set dp(1,1) = 0

    Iterate over received bits:
        for each time step i:
            Extract a group of bits from received_bit

            for each state st:
                Compute Hamming distances for possible transitions
                Update dp matrix with minimum path metric for each state

    Backtrack to find decoded message:
        Initialize next_st = 0
        Iterate over time steps in reverse:
            Determine previous states with minimum path metric
            Compute Hamming distances for transitions
            Choose the previous state with the minimum path metric
            Update decoded message and next state

    Return decoded message
end

(actual Matlab function for this we have uploaded in pdf file)
```

Soft Decision Decoding

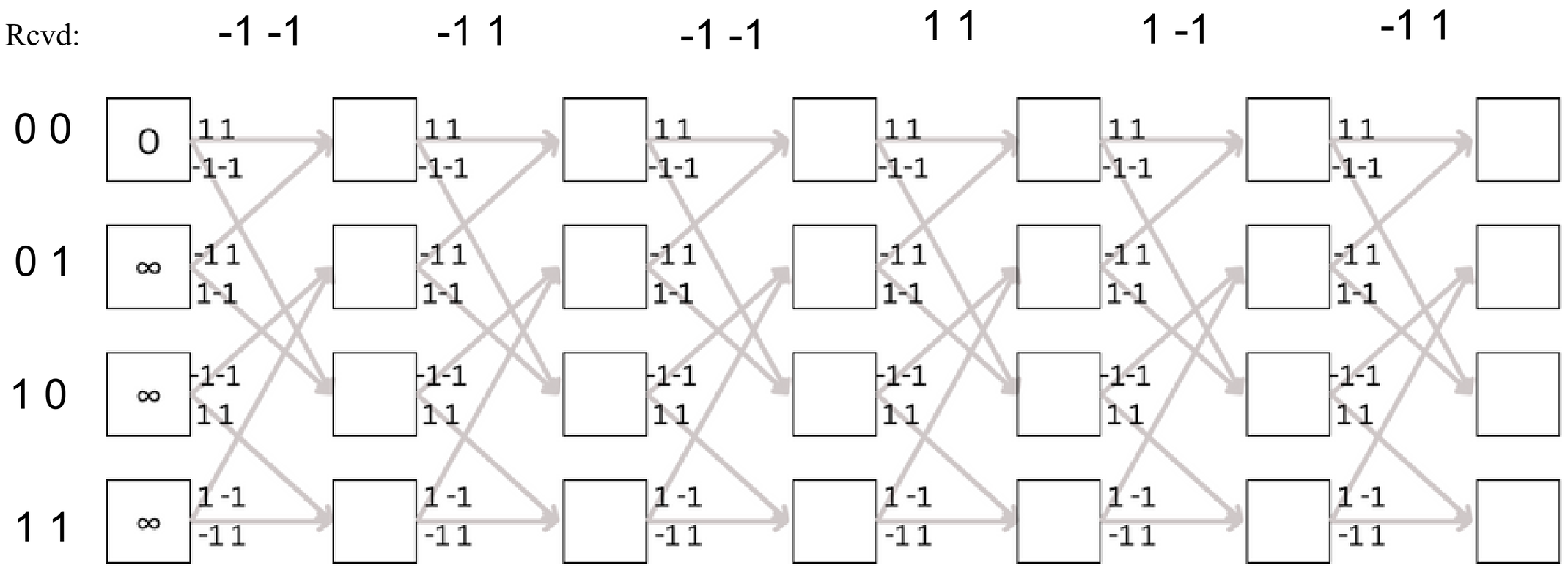
- In Soft decision decoding we do not demodulate received signal instead we take Euclidean distance with 1/-1.
- For soft decision decoding in convolution codes the computation of the path matric and the Viterbi decoding algorithm will remain the same as hard decision decoding.
- The only difference 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**.**

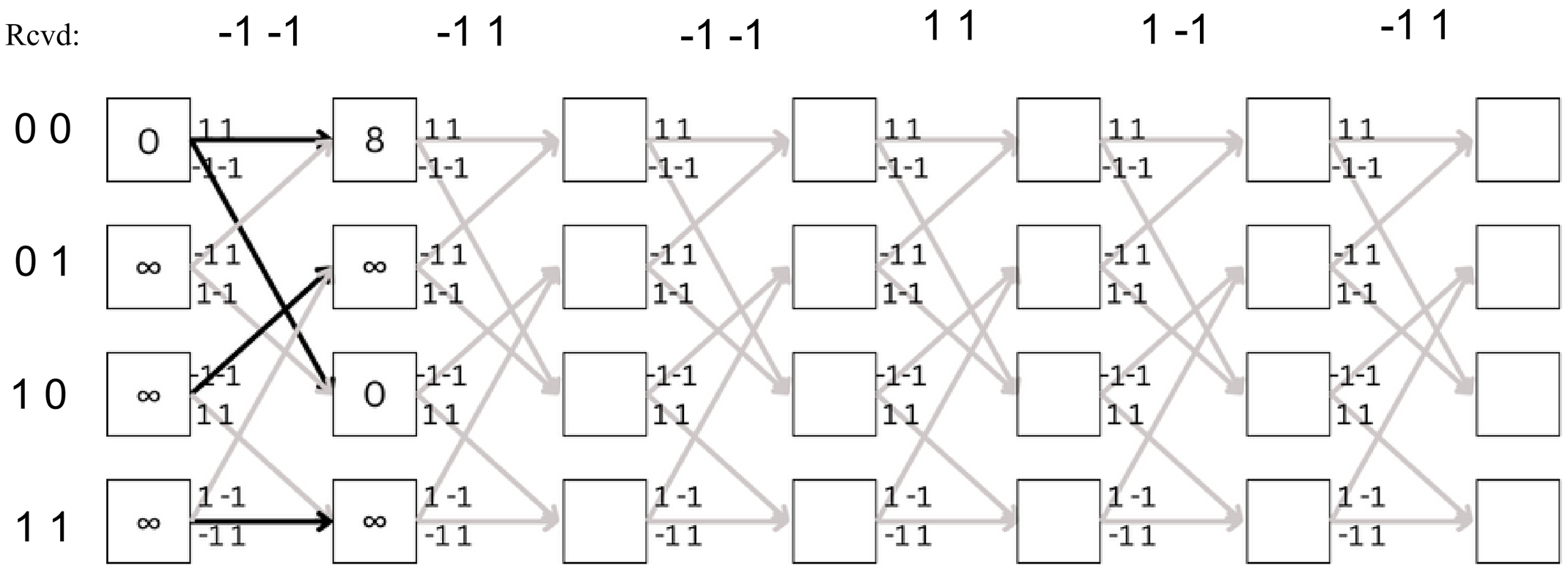


- Let’s see visualization of computing trellis and backtracking for soft decision decoding.

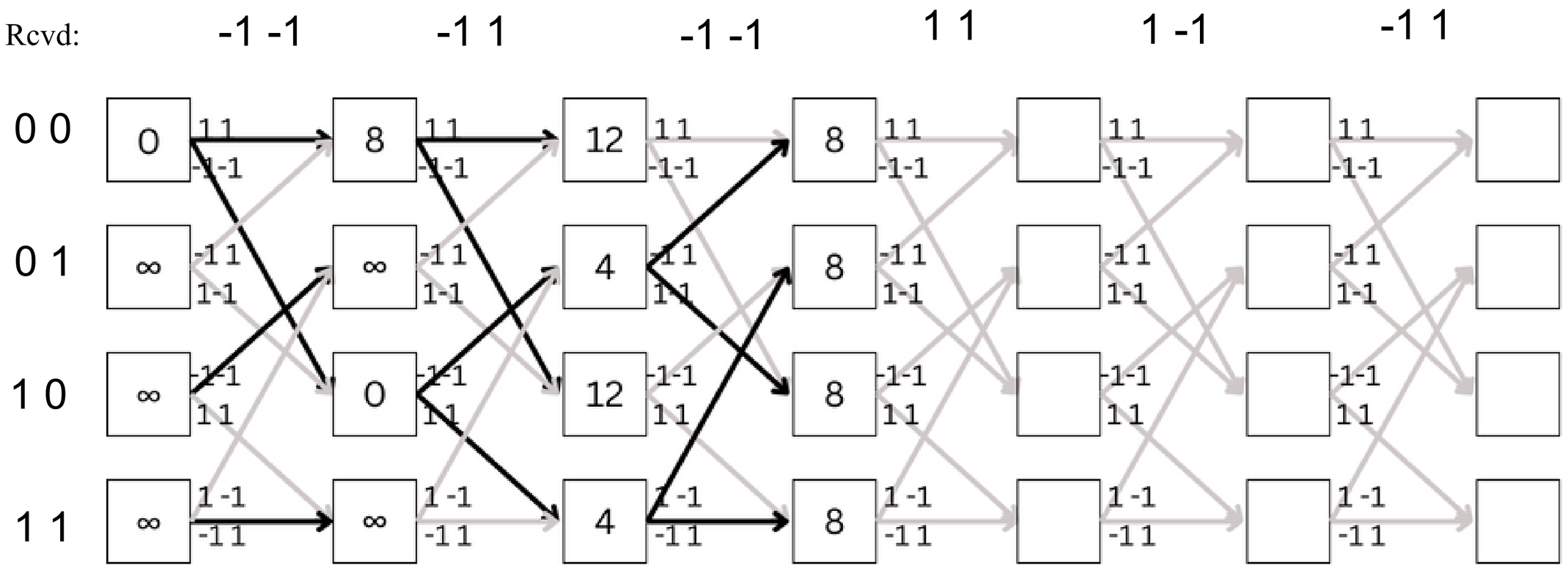
Soft Decision Decoding



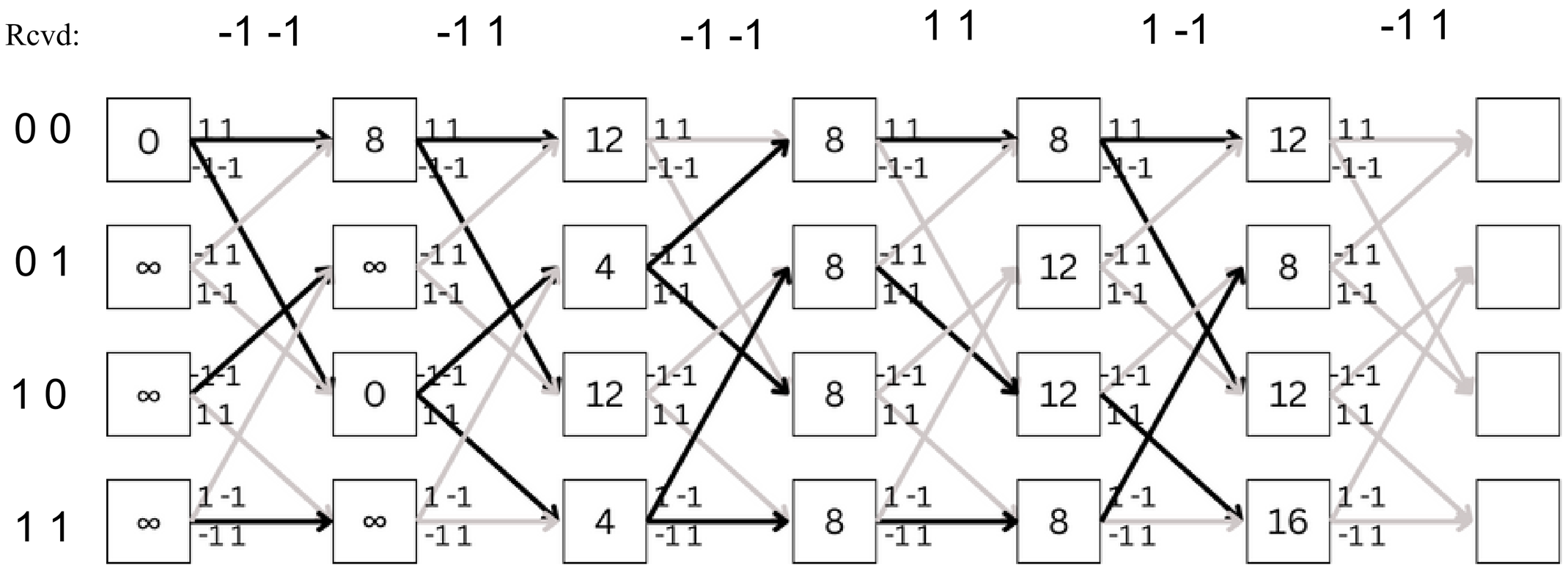
Soft Decision Decoding



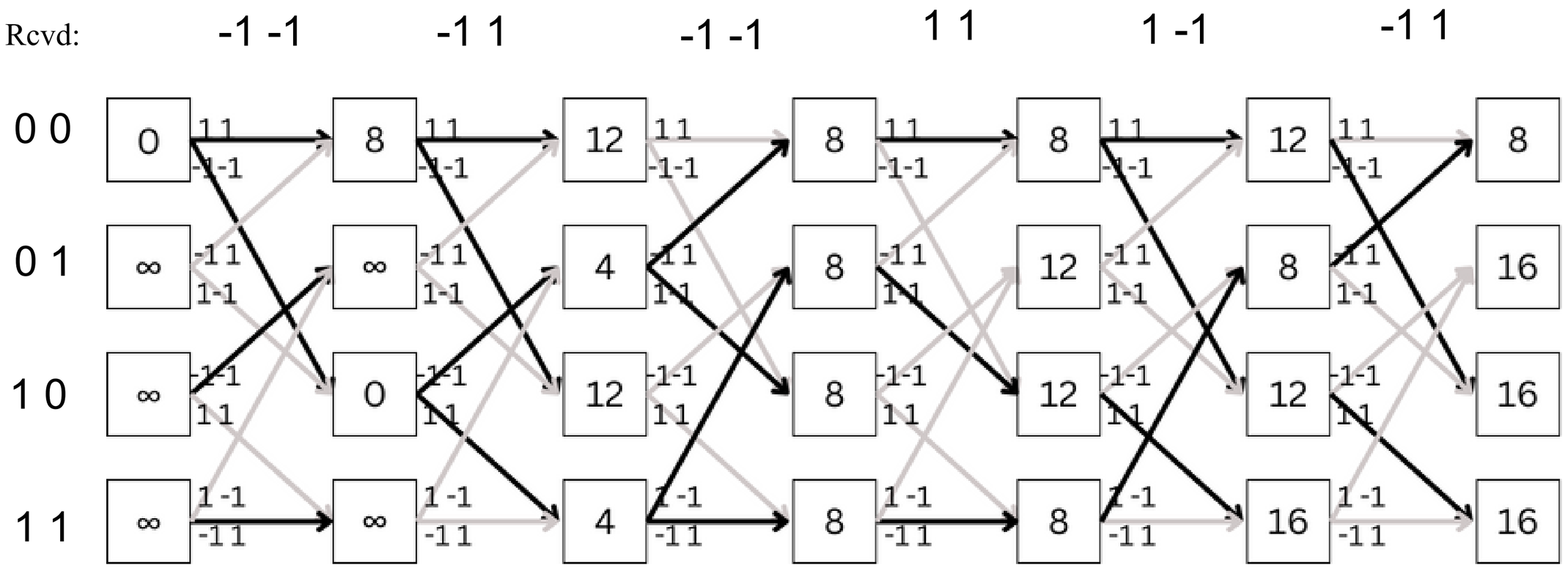
Soft Decision Decoding



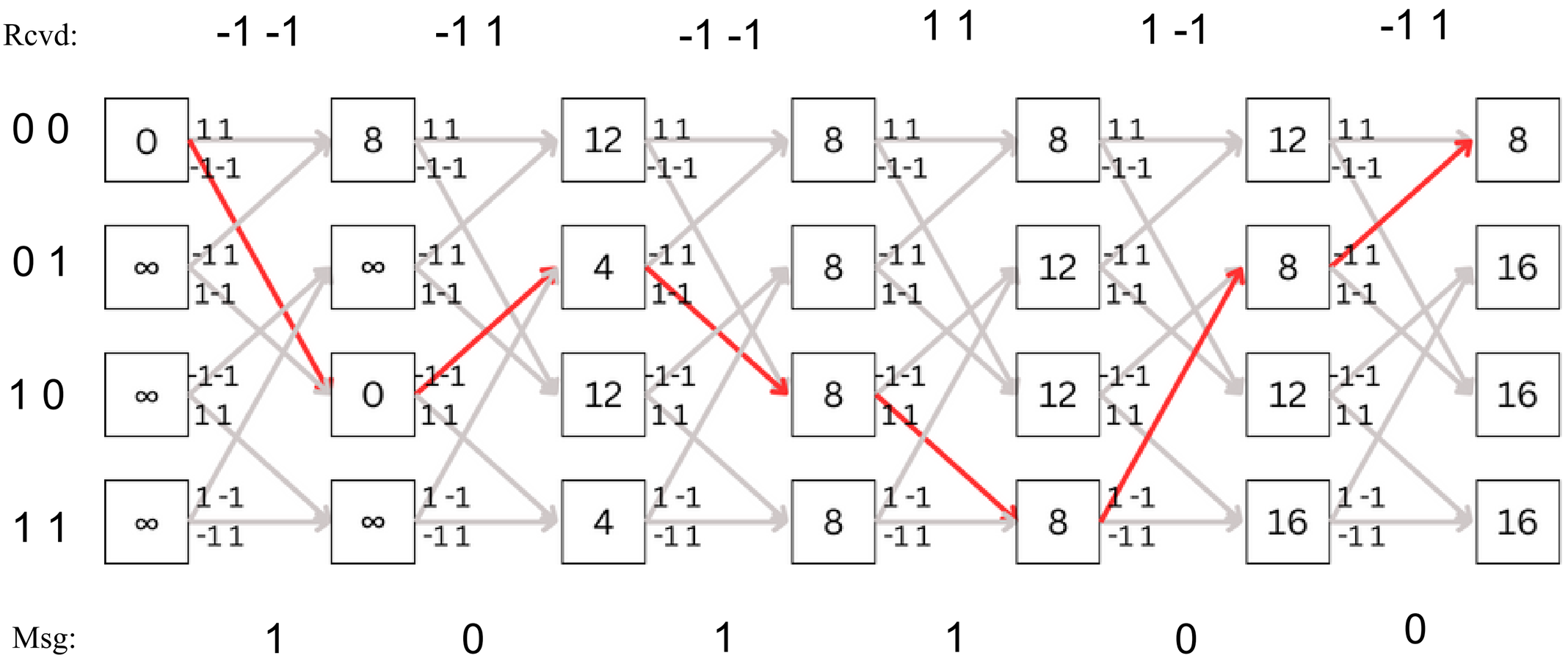
Soft Decision Decoding



Soft Decision Decoding



Soft Decision Decoding

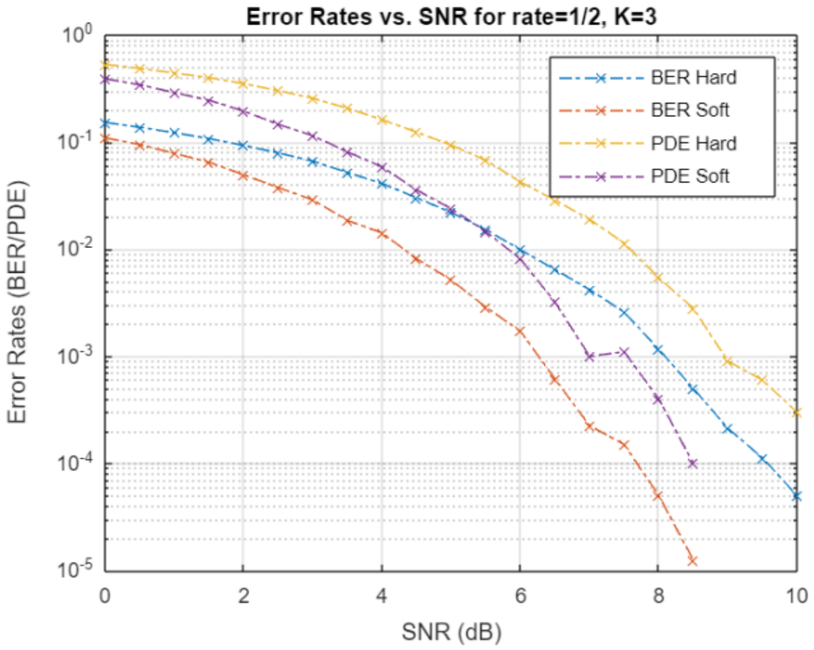


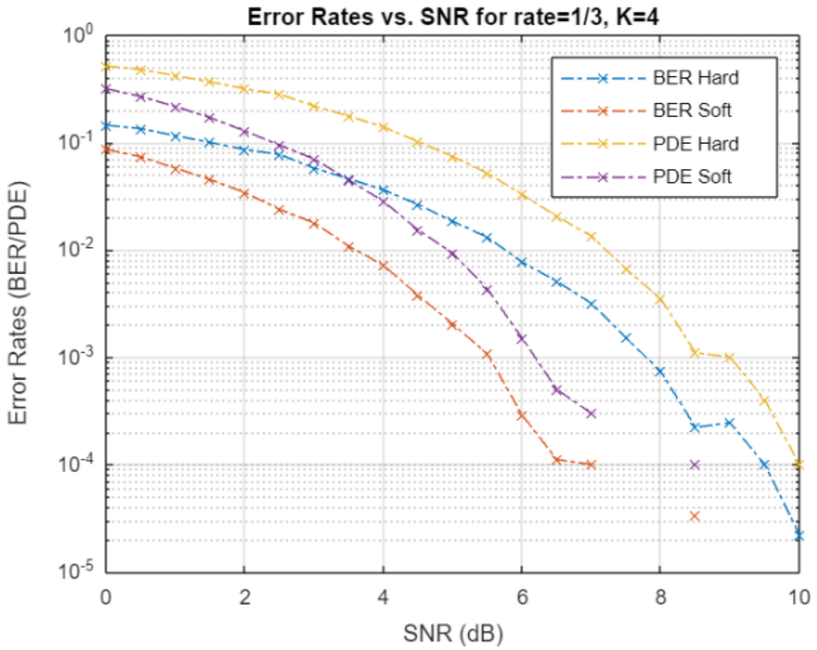
Pseudocode for Backtracking(soft)

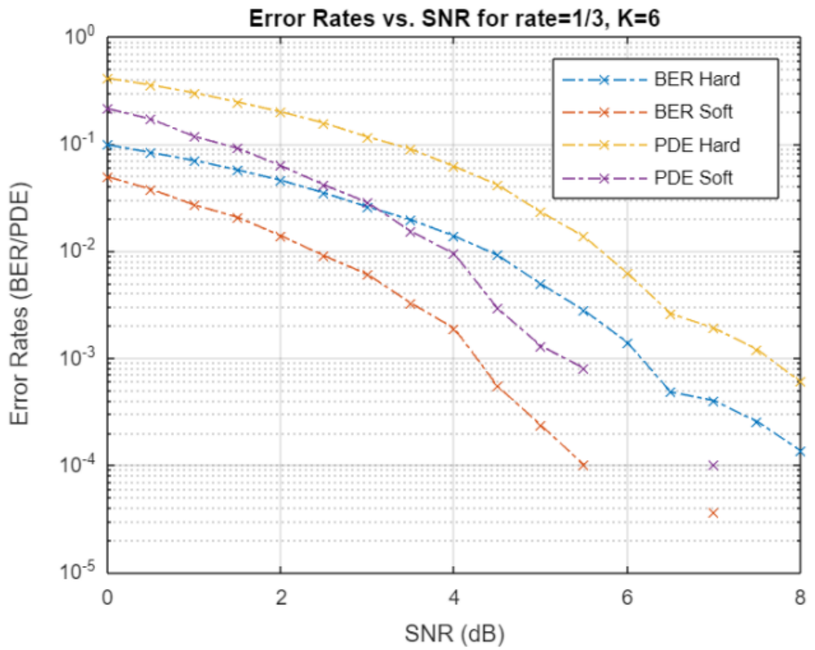
```
function viterbi_decode(g, received_bit)
    Set dp(1,1) = 0

    Iterate over received bits:
        for each time step i:
            Extract a group of bits from received_bit

            for each state st:
                Compute Euclidean distances for possible transitions
                Update dp matrix with minimum path metric for each state

    Backtrack to find decoded message:
        Initialize next_st = 0
        Iterate over time steps in reverse:
            Determine previous states with minimum path metric
            Compute Euclidean distances for transitions
            Choose the previous state with the minimum path metric
            Update decoded message and next state

    Return decoded message
end
```

Simulation results

- The plots depict the performance of the Viterbi hard and soft decoders for different rates and constraint lengths.





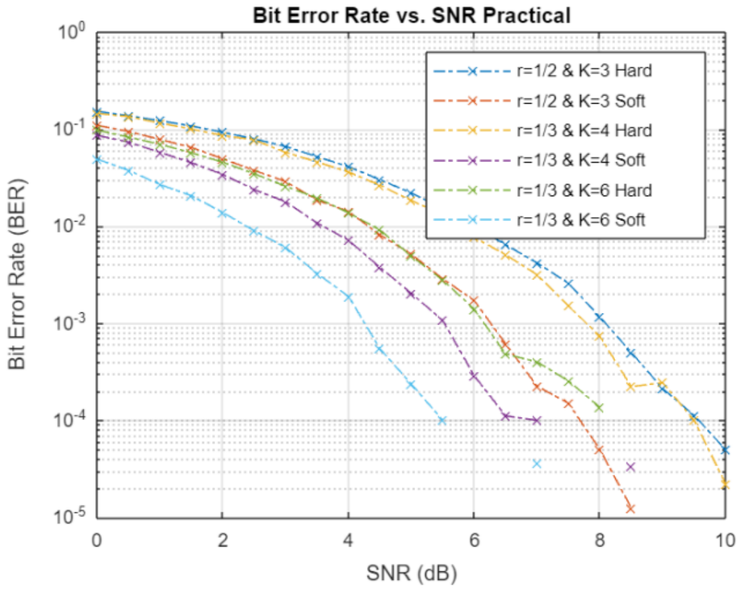


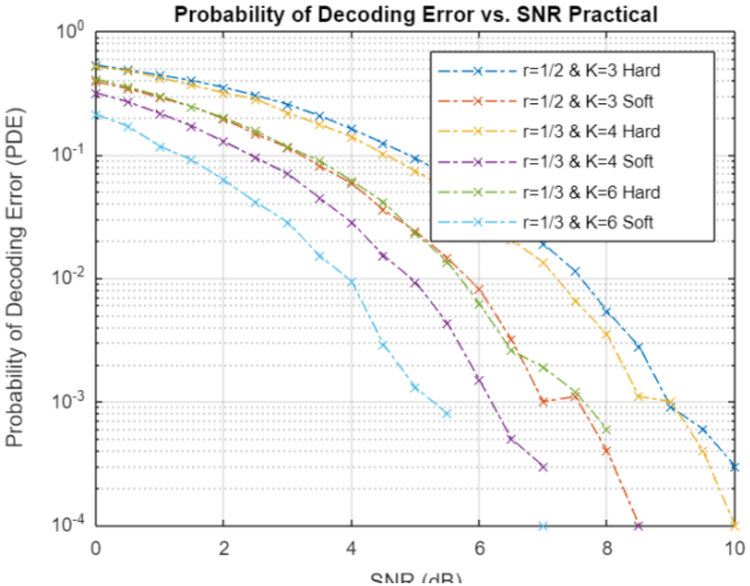
Observations from Plots

- Performance of both Viterbi hard and soft for each case increases with the increase in SNR.
- The BER and PDE is higher for lower SNR which suggests that Viterbi hard performs better for minimal noise.
- The same goes for Viterbi-soft decision decoding but it performs better than Viterbi-hard.
- Lower code rates enhance the performance of the decoders but it also introduces overhead.
- Larger constraint length enhances error correction significantly but the cost is significant increase in time due to increase in computation complexity.
- The overall performance of soft decision decoding is much better than the hard decision. The reasons can be:

>> In hard decision decoding, each of the received symbol is demodulated to bits considering a specific threshold. This leads to ambiguity because decoder may choose incorrect bits due to uncertainty.

>>Soft decision decodes using the received symbols rather than demodulating them to bits. Unlike viterbi hard, there is no room for ambiguity here.





Summary

- Convolution code is a channel coding technique which transmits parity bits rather than message bits followed by parity bits.
- Viterbi hard decision can be generalised to Viterbi soft decision just by changing the Branch Metric computation.
- The overall performance of the Viterbi Soft Decoder is better than Viterbi Hard Decoder.
- The trade off involved in convolution codes is that the errors in the decoded message decreases with the increase in constraint length but the decoding time increases significantly.
- The performance of decoder improves as the rate decreases but it also leads to increase in the parity bits generated for message which increases the computation complexity or more appropriately the decoding time.
- Convolutional codes can be used in wireless communication, satellite links, deep space communication, and digital television broadcasting.

Soft Decision Decoding

Analysis

Appendix

- Convolution Encoder is a finite state machine, And the optimum decoder is a maximum likelihood sequence estimator (MLSE), which helps us to find the most likely transmitted sequence for the received sequence.
- If the Soft-Decision Decoding is employed and the code sequence is transmitted by BPSK modulation then the input to the decoder is:

$$r_{jm} = \sqrt{\varepsilon_c}\,(2c_{jm} - 1) + n_{jm} \qquad \ldots\ldots\ldots (8\text{-}2\text{-}9)$$

❑Here ,

c_{jm} : coded symbol related to m^{th} symbol of j^{th}branch and it's components are c_{jm1}, c_{jm2} etc.

r_{jm} : output from the demodulator i.e. the Viterbi decoder inputs corresponding to m^{th} symbol of j^{th} branch in trellis.

n_{jm}: represents the additive white gaussian noise that affects the reception of m^{th} symbol of j^{th} branch in trellis.

- <u>Branch Metric :</u>

A branch metric for the j^{th} branch of the i^{th} path through the trellis is defined as the logarithm of the joint probability of the sequence conditioned on transmitted sequence

$$\mu_j{}^{(i)} = \log P\,(\{r_{jm}\}|\{c_{jm}\}) \qquad \ldots\ldots\ldots (8\text{-}2\text{-}10)$$

Appendix

❑Now, $P\ (\{r_{jm}\}|\{{c_{jm}}^{i}\}) = P\ (\ r_{jm1}r_{jm2}r_{jm3}\ldots|\{{c_{jm1}}^{i}{c_{jm2}}^{i}{c_{jm3}}^{i}\ \ldots\})$

- Since $r'_{ji}s$ are independent of each r_{jm} for any i $\neq m$,

$$P\ (\{r_{jm}\}|\{{c_{jm}}^{i}\}) = \Pi_{m=1}^{m=n}\ P\ (r_{jm}|\{{c_{jm}}^{i}\}) \qquad \ldots\ldots\ldots \text{(a)}$$

- Considering that the reception of r_{ji} is only being affected by c_{ji}

$$P\ (r_{jm}|\{{c_{jm}}^{i}\}) = P\ (r_{jm}|{c_{jm}}^{i})$$

- Now, demodulator output is described statistically by the pdf

$$P\ (r_{jm}|{c_{jm}}^{i}) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(\frac{-1}{2\sigma^2}\ [n_{jm}]^2) \qquad \ldots\ldots\ldots \text{(b)}$$

- Therefore,

$$P\ (r_{jm}|{c_{jm}}^{i}) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(\frac{-1}{2\sigma^2}\ [r_{jm}\ - \sqrt{\varepsilon_c}\ (2c_{jm}\ - 1)]^2) \qquad \ldots\ldots\ldots \text{(8-2-13)}$$

❑Where, $\sigma^2 = \frac{N_0}{2}$ is variance of Additive White Gaussian Noise.

- Using equation (a), (b) and (8-2-13) into equation (8-2-10) and by neglecting the common terms to all branch metrics then branch metric for the j^{th} branch of the i^{th} path may be expressed as

$$\mu_J{}^{(i)} = \sum_{m=1}^{m=n} r_{jm}\left(2{c_{jm}}^{i} - 1\right) \qquad \ldots\ldots\ldots \text{(8-2-14)}$$

Appendix

- Path Metric :

  A path metric for the i^{th} path containing B branches through the trellis is defined as

$$PM^{(i)} = \sum_{j=1}^{j=B} \mu_j{}^{(i)} \quad \text{..... (c)}$$

- The guidelines for choosing between two paths within trellis is to opt for the one with higher metric. This enhances the likelihood of making accurate decision or conversely if diminishes the probability of errors in the sequence of information bits.
- It is also called convolution metric of the path as well.

$$CM^{(i)} = \sum_{j=1}^{j=B} \sum_{m=1}^{m=n} r_{jm}\left(2c_{jm}{}^{i} - 1\right) \quad \text{..... (8-2-16)}$$

- This metric is nothing but essentially the summation of all branch metrics of the i^{th} path.

- Probability of error for Soft-Decision-Decoding :
- To derive the probability of error for convolutional codes, linearity property can be used to simplify the derivation.
- That is, we can assume that all-zero sequence is transmitted and we determine the probability of error in favor of other sequence.

❑Now, for all-zero path

$$CM^{(0)} = \sum_{j=1}^{j=B} \sum_{m=1}^{m=n} (-\sqrt{\varepsilon_c} + n_{jm})(-1)$$

Appendix

$$CM^{(0)} = \sqrt{\varepsilon_c}Bn - \sum_{j=1}^{j=B}\sum_{m=1}^{m=n}(n_{jm}) \quad \text{..... (8-2-17)}$$

- Now , It the incorrect path that merges with all-zero path differs from ‘d’ bits will be having it’s correlation metric $CM^{(1)}$, then the probability of error in pairwise comparison of $CM^{(0)}$ and $CM^{(1)}$ is,

$$P_2(d) = P\left(CM^{(1)} \geq CM^{(0)}\right) = P\left(CM^{(1)} - CM^{(0)} \geq 0\right)$$

- Using eq(8-2-16)

$$P_2(d) = P\left(2\sum_{j=1}^{j=B}\sum_{m=1}^{m=n} r_{jm}\left(c_{jm}{}^{(1)} - c_{jm}{}^{(0)}\right) \geq 0\right) \quad \text{..... (8-2-18)}$$

- Since the coded bits in the two paths we identical except in the ‘ d ‘ positions , this equation can be written as

$$P_2(d) = P\left(\sum_{l=1}^{l=d} r_l' \geq 0\right) \quad \text{..... (8-2-19)}$$

❑Here, r_l' represents input to the decoder for d^{th} bits.

- From equation (1)

$$r_{jm} = \sqrt{\varepsilon_c}\ (2c_{jm} - 1) + n_{jm}$$

$$\text{Where, } n_{jm} \sim N\left(0, \frac{N_0}{2}\right) \rightarrow r_{jm}/r_l' \sim N\left(-\sqrt{\varepsilon_c}, \frac{N_0}{2}\right)$$

❑Now, from Central Limit Theorem :

$$\sum_{l=1}^{l=d} r_l' \sim N\left(-d\sqrt{\varepsilon_c}, \frac{dN_0}{2}\right)$$

Appendix

- Therefore, $P_2(D) = \mathrm{P}\left[\frac{\sum_{l=1}^{d} r_l' + \mathrm{d}\sqrt{\varepsilon_c}}{\sqrt{\frac{dN_0}{2}}} \geq \frac{\mathrm{d}\sqrt{\varepsilon_c}}{\sqrt{\frac{dN_0}{2}}}\right] = Q\left(\sqrt{\frac{2d\varepsilon_c}{N_0}}\right)$ ..... (d)

$= Q(\sqrt{2\gamma_b R_c d})$ ..... (8-2-20)

❑Where, $\gamma_b = \frac{\varepsilon_b}{N_0}$ is received SNR per bit

R_c is code rate

- Now , there are many paths with different distances that merge with the all-zero path at given node. Transfer function provides (T(D)) a complex description of all such paths.
- Thus, we can sum the error probability (eq(10)) for all such possible paths which will correct into the first event error probability in the form.

$$P_e \leq \sum_{d=d_{free}}^{INF} a_d P_2(d)$$
$$\leq \sum_{d=d_{free}}^{INF} a_d \, Q(\sqrt{2\gamma_b R_c d}) \quad \text{..... (8-2-21)}$$

❑Where,

d_{free} - is smallest Hamming distance between any two particular codeword

a_d - Number of paths of distance d (which particularly is the number of 1's in that path)

Appendix

❑Now, applying chernoff's inequality we get $P(x \geq a) \leq \exp(-at)E[e^{tX}]$, which further yields to the result

$$Q(x) \leq \exp(\frac{-x^2}{2}) \quad \text{..... (e)}$$

$$\rightarrow Q\left(\sqrt{2\gamma_b R_c d}\right) \leq \exp(-\gamma_b R_c d) = D^d|_{D=e^{-\gamma_b R_c}} \quad \text{..... (8-2-22)}$$

- From equations (11) and (12)

$$P_e < \sum_{d=d_{free}}^{INF} a_d D^d \quad \text{..... (f)}$$

- Now, bit error probability can be more useful measure of performance. This probability can be upper bounded by the procedure used in bounding the first event error probability if we multiply $p2\ (d)$ by # of incorrectly decoded information bits.
- The average bit error probability is upper bounded by multiplying each $p2\ (d)$ by the corresponding # of incorrectly decoded bits for each possible incorrect path.

❑Now, $T(D,N) = \sum_{d=d_{free}}^{INF} a_d\ D^d\ N^{f(d)}$[8-2-24]. Here, exponent of N represents the number of 1's in the path with d distance.

- Therefore,

$$\frac{T(D,N)}{dN} = \sum_{d=d_{free}}^{INF} a_d \mathrm{f(d)} D^d$$

$$\frac{T(D,N)}{dN} = \sum_{d=d_{free}}^{INF} \beta_d D^d \quad \text{..... (8-2-25)}$$

Appendix

- Thus, the bit error probability for k=1 is

$$P_b < \sum_{d=d_{free}}^{INF} \beta_d P_2(d)$$

$$P_b < \sum_{d=d_{free}}^{INF} \beta_d Q\left(\sqrt{2\gamma_b R_c d}\right) \quad \text{..... (8-2-26)}$$

Hard Decision Decoding

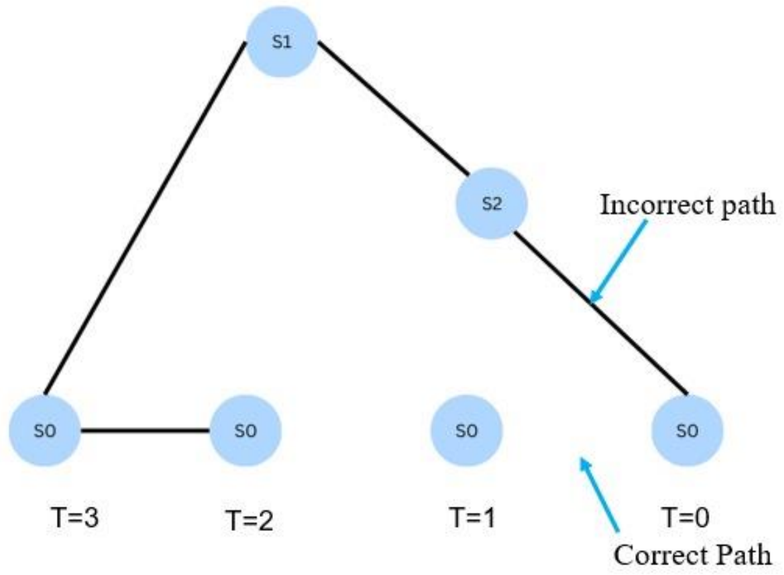
Analysis

Appendix

- We will analyze the performance of maximum likelihood decoding for a convolutional code over binary symmetric channel.
- Without loss of generality, we assume that the all zero codeword 0 is transmitted .
- A first event error happens at an arbitrary time t if the all zero path is eliminated for the first time in favor of an incorrect path.
- Assuming that the incorrect path has weight d, a first event error happens with probability

$$P_2(\mathrm{D}) = \begin{cases} \sum_{k=(d+1)/2}^{d} \binom{d}{k} p^k (1-p)^{d-k} & \text{If d is odd} \quad \text{..... (8-2-28)} \\ \sum_{k=(d+1)/2}^{d} \binom{d}{k} p^k (1-p)^{d-k} + \frac{1}{2}\binom{d}{k} p^k (1-p)^{d-k} & \text{If d is even} \quad \text{..... (8-2-29)} \end{cases}$$

Appendix



- All incorrect paths of length t branches or less can cause a first event error at time t.
- Thus the first event error probability at time t can be bounded using union bound by the sum of the error probabilities of each of these paths.

- If all incorrect paths of length greater than t are also included, then the first event error probability at any time t can be bounded by

$$P_e < \sum_{d=d_{free}}^{d} a_d \, P_2(D) \qquad \text{..... (8-2-30)}$$

- Where a_d is the number of codewords of weight d.

Appendix

- Instead of using the expressions for P2(d) given in (8-2-28) and (8-2-29), we can use the upper bound,

$$P_2(\mathrm{D}) < [4p\,(1-p)\,]^{d/2} \qquad \text{..... (8-2-31)}$$

- Use of this bound in (8-2-30) yields a looser upper bound on the first-event error probability, in the form,

$$P_e < \sum_{d=d_{free}}^{d} a_d\ [4p\,(1-p)\,]^{d/2}$$
$$< \mathrm{T(D)}\big|_{D=\sqrt{4p(1-p)}}$$

..... (8-2-32)

Appendix

For odd d,

$$P_2(d) = \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e} p^e (1-p)^{d-e}$$

$$\leq \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e} p^{d/2} (1-p)^{d/2}$$

$$= p^{d/2}(1-p)^{d/2} \sum_{e=\frac{d+1}{2}}^{d} \binom{d}{e}$$

$$= p^{d/2}(1-p)^{d/2} \sum_{e=0}^{d} \binom{d}{e}$$

$$= 2^d p^{d/2}(1-p)^{d/2} \quad \text{..... (a)}$$

For even d,

$$P_2(d) = \sum_{e=\frac{d}{2}+1}^{d} \binom{d}{e} p^e (1-p)^{d-e} + \frac{1}{2}\binom{d}{e} p^{d/2}(1-p)^{d/2}$$

$$< \sum_{e=\frac{d}{2}}^{d} \binom{d}{e} p^e (1-p)^{d-e}$$

$$< \sum_{e=\frac{d}{2}}^{d} \binom{d}{e} p^{d/2} (1-p)^{d/2}$$

$$= p^{d/2}(1-p)^{d/2} \sum_{e=0}^{d} \binom{d}{e} p^{d/2}$$

$$= 2^d p^{d/2}(1-p)^{d/2} \quad \text{..... (b)}$$

Appendix

- The bit error probability can be bounded by

$$P_b < \sum_{d=d_{free}}^{\infty} \beta_d \, P_2(d) \qquad \text{..... (8-2-33)}$$

- where β_d is the total number of nonzero information bits on all weight-d paths, divided by the number of information bits k per unit time.
- The $\{\beta_d\}$ are the coefficients in the expansion of the derivative of T(D, N), evaluated at N = 1. For P_2(d), we may use either the expressions given in (8-2-28) and (8-2-29) or the upper bound in (8-2-31). If the latter is used, the upper bound on P_b, can be expressed as,

$$P_b < \frac{dT(D,N)}{dN}\Big|_{N=1, D=\sqrt{4p(1-p)}} \qquad \text{..... (8-2-34)}$$

- When k >1, the results given in (8-2-33) and (8-2-34) for P_b, should be divided by k.

References

[1] CT216. Introduction to Communication Systems: Lecture 3 Channel Coding. DA-IICT, Winter 2024.

[2] J. G. Proakis. Digital Communications. McGraw-Hill, 4th edition, 1995.

[3] Matthew Valenti and Bibin Baby John. Introduction to Communication Systems: A Lecture on Convolution Coding and Viterbi Decoding. DA-IICT, Winter 2024.

[4] PERFORMANCE ANALYSIS OF M-QAM WITH VITERBI SOFT-DECISION DECODING by Rogerio Correa Manso

[5] MIT Lectures L8 and L9 (Convolution Codes and Viterbi Decoding)

[6] Performance Bounds for Convolutional Codes (NPTEL Lecture)

ThankYou !!!