Information Theory and Channel Coding

Prof. Rodrigo C. de Lamare
CETUC, DEE, PUC-Rio, Brazil
delamare@puc-rio.br

IX. Convolutional codes

A. Introduction

B. Encoding

C. Structure and design of convolutional codes

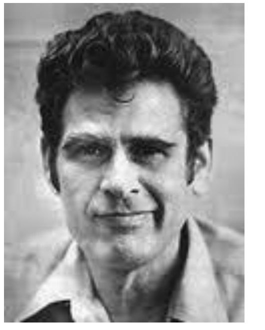
D. Maximum likelihood decoding

E. Error correction capability

F. Performance

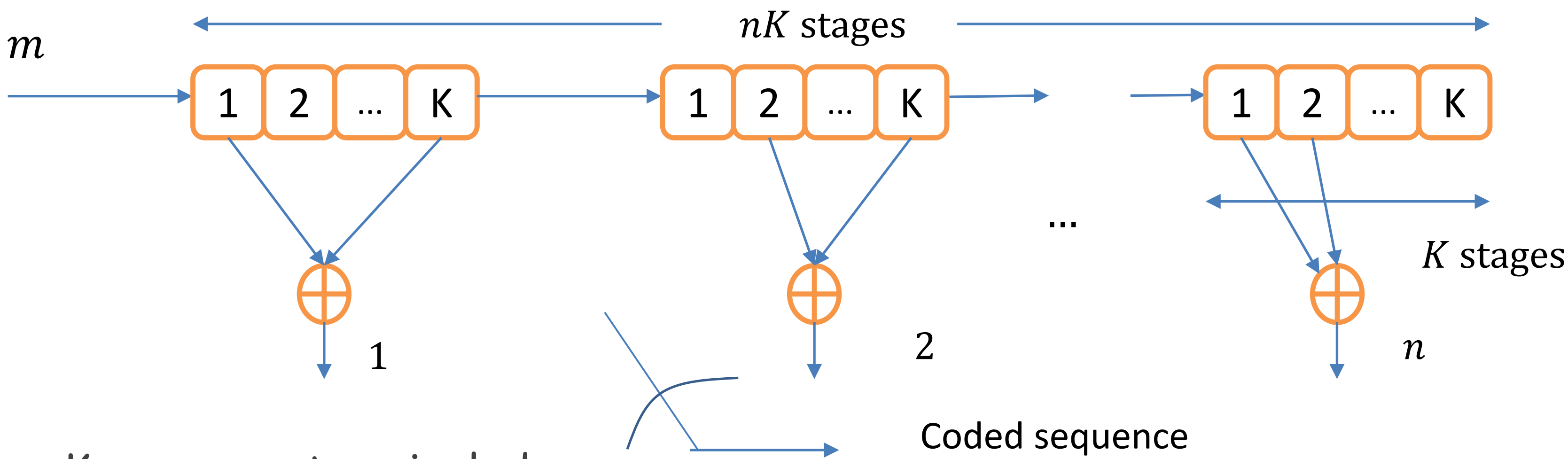
A. Introduction

- Convolutional codes are an important alternative to linear block codes that were invented by Peter Elias, an MIT professor, in 1955.



- Convolutional codes can approach Shannon's theoretical limit by using maximum likelihood decoding.
- The basic idea consists of processing messages sequentially rather than in blocks using shift registers and adders.
- Decoding convolutional codes is carried out by a maximum likelihood decoding strategy known as Viterbi algorithm.

- Let us consider a general convolutional coding scheme:



- Key parameters include:
  - The code rate: $R = \frac{k}{n}$
  - The constraint length (K): the number of bit shifts required to modify the output.
  - The memory (M): $K = M + 1$

- Convolutional encoders can be categorized as:
  - systematic or non systematic,
  - recursive or non recursive.

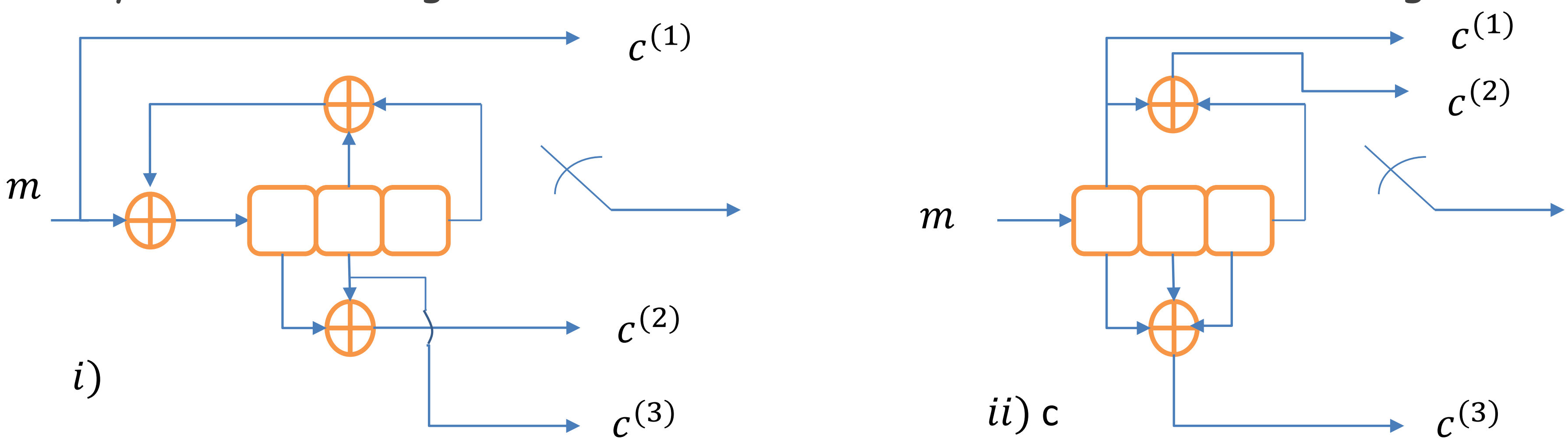
- Systematic encoders:
  - cannot be catastrophic-> when a finite number of errors result in an infinite number of errors in the decoding.
  - the message is explicitly shown.

- Recursive encoders:
  - employ a configuration with feedback.
  - can be implemented as IIR filters.

- Non-recursive encoders:
  - can be implemented as FIR filters

Example 1

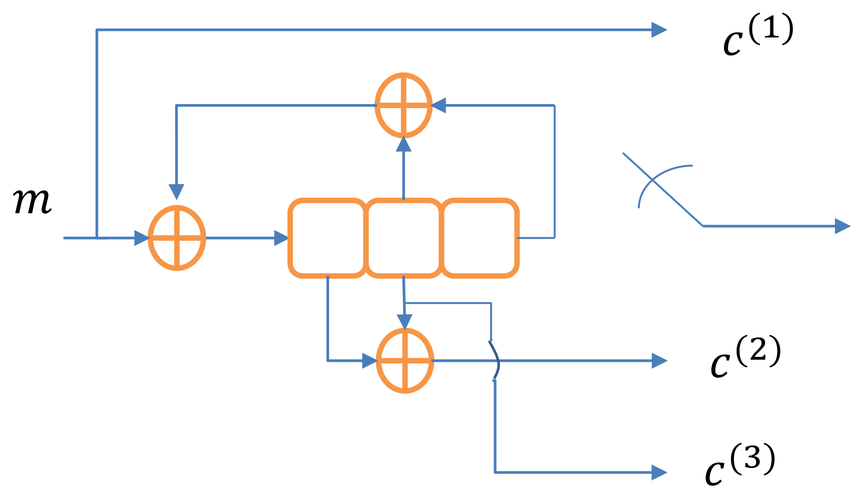
Analyze the following convolutional encoders and describe the following:



a) Rate
b) Constraint length and memory
c) Are they systematic or non systematic?
d) Are they recursive or non recursive?

Solution:

For encoder $i)$, we have



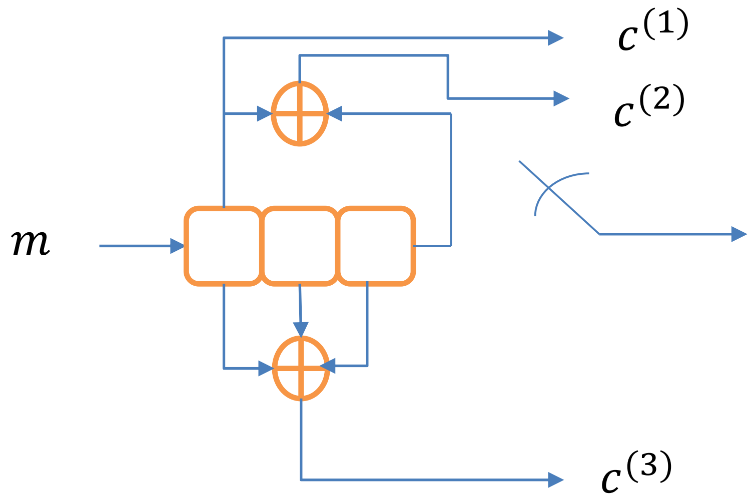
The code rate is $R = \frac{1}{3}$

The contraint length is $K = 3$ and the memory is $M = 2$

This is a systematic encoder because the message is explicitly shown.

The encoder is recursive because there is a feedback loop that affects encoding.

For encoder $ii)$, we have



The code rate is $R = \frac{1}{3}$

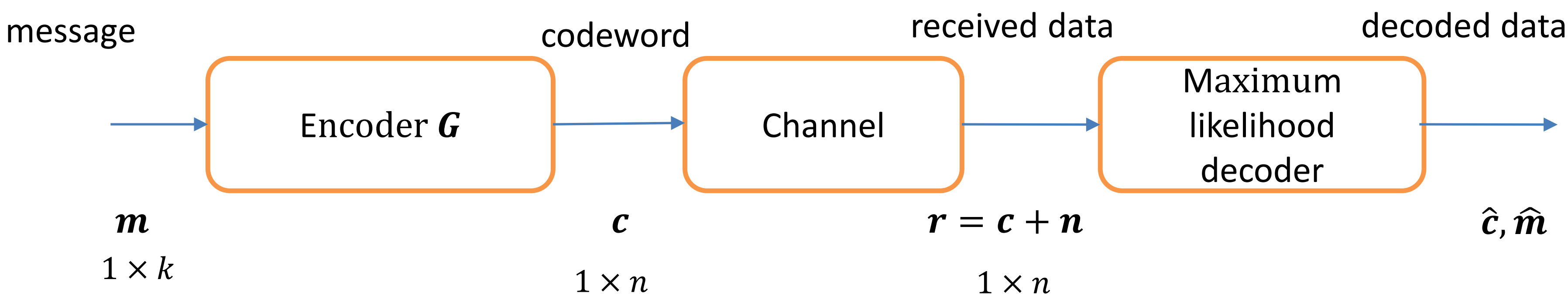
The contraint length is $K = 3$ and the memory is $M = 2$

This is a non systematic encoder because the message is not explicitly shown.

The encoder is non recursive because there is no feedback loop that affects encoding.

B. Encoding

- Let us consider a convolutional coding system with the block diagram.



- We will assume that the convolutional encoder has $\mathrm{k} = 1$ inputs and n outputs in our exposition.

- The D −transform will be adopted for the description of the message and code sequences as polynomials, where D refers to a delay.

Message sequence $\boldsymbol{m}$:

- The message sequence $\boldsymbol{m} = [m_0\ m_1\ \dots m_{k-1}]$ at the input of the encoder can be described as a polynomial:

$$m(D) = m_0 + m_1 D + \cdots + m_{k-1} D^{k-1},$$

where $m_d \in \{0,1\}$, $d = 0,1,\dots,k-1$.

Encoder structure:

- The generator polynomial $g^{(j)}(D)$ is given by

$$g^{(j)}(D) = {g^{(j)}}_0 + g_1^{(j)}\, D + \cdots + g_M^{(j)}\, D^M, \qquad j = 1,2, \dots, n$$

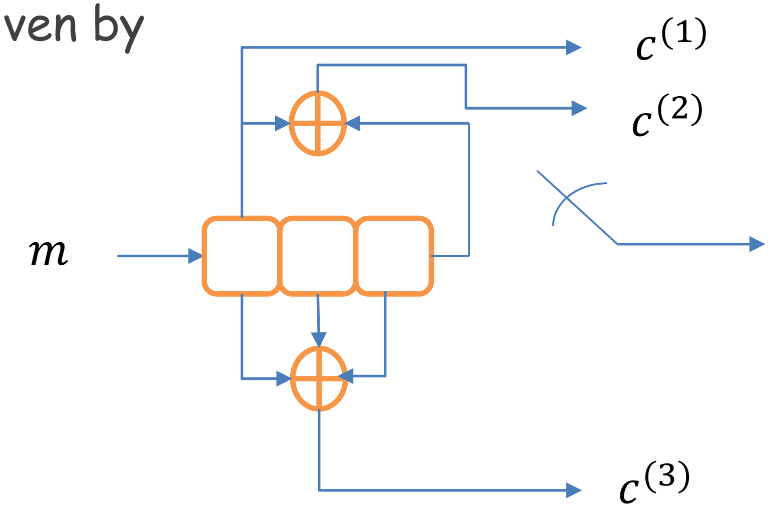
where $g_m^{(j)} \in \{0,1\}$, $m = 0,1, \dots, M$ and M is the memory of the encoder polynomial and n is the number of outputs.

- The impulse response of the encoder is equivalent to the generator polynomial and is given by

$$\boldsymbol{g}^{(j)} = \left[{g^{(j)}}_0 \;\; g_1^{(j)} \dots g_M^{(j)}\right]$$

Example 2

Consider a convolutional encoder given by



a) Compute the impulse response of each output of the encoder.
b) Compute the generator polynomial of each output of the encoder and express it in octal form.

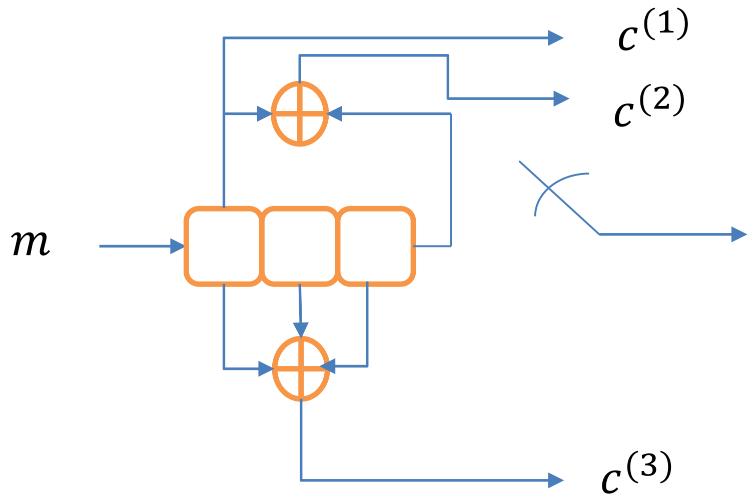
Solution:

a) We obtain the impulse response by inspection as follows:

$\boldsymbol{g}^{(1)} = [1 \;\; 0 \; 0]$

$\boldsymbol{g}^{(2)} = [1 \;\; 0 \; 1]$

$\boldsymbol{g}^{(3)} = [1 \;\; 1 \; 1]$

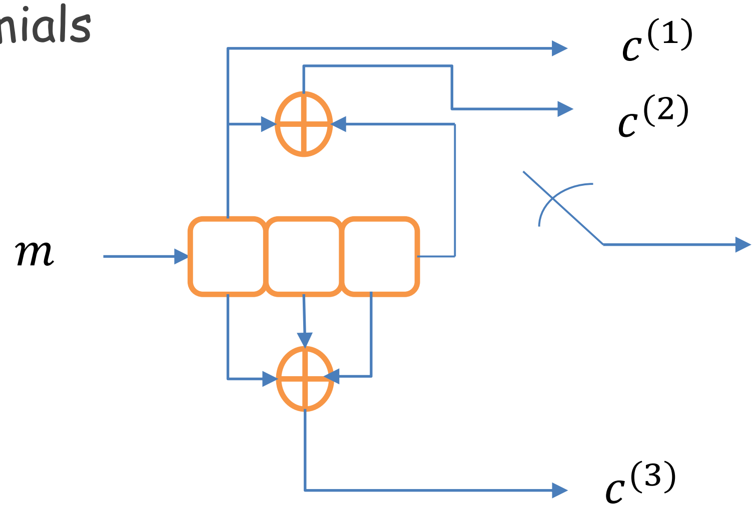


b) We obtain the generator polynomials by inspection as follows:

$$g^{(1)}(D) = 1$$
$$g^{(2)}(D) = 1 + D^2$$
$$g^{(3)}(D) = 1 + D + D^2$$

In octal, we have

$$g^{(1)}(D) = 4$$
$$g^{(2)}(D) = 5$$
$$g^{(3)}(D) = 7$$



Convolutional codes can be generated using different strategies, namely:

i) Convolutional sum

The convolutional code at time i is given by

$$c_i = \sum_{l=0}^{M} m_{i-l} g_l^{(j)}, \qquad i = 0,1, \dots, k + M - 1,$$

where $\boldsymbol{m} = [m_0\ m_1\ \dots m_{k-1}]$ is the message, M is the memory and $\boldsymbol{g}^{(j)} = [\ g_0^{(j)}\ g_1^{(j)}\ \dots .\ g_M^{(j)}]$ is the impulse response of path j of the encoder.

ii) Discrete convolution in time

For a message $\boldsymbol{m} = [m_0\, m_1\, \ldots m_{k-1}]$ and impulse responses of the encoder given by $\boldsymbol{g}^{(j)} = [\, g_0^{(j)}\; g_1^{(j)}\; \ldots .\, g_M^{(j)}]$, the code is described by

$$\boldsymbol{c}^{(j)} = [c_0\, c_1\, \ldots c_{k+M-1}] = \boldsymbol{m} * \boldsymbol{g}^{(j)},$$

where $*$ refers to convolution.

The convolution can be represented by

$$\boldsymbol{c}^{(j)} = \boldsymbol{m}\boldsymbol{G}^{(j)},$$

where $\boldsymbol{G}^{(j)} = \begin{bmatrix} g_0^{(j)}\; g_1^{(j)}\; \ldots .\, g_M^{(j)} & & \\ & \ddots & \\ & & g_0^{(j)}\; g_1^{(j)}\; \ldots .\, g_M^{(j)} \end{bmatrix} \in \mathrm{F}^{k \times (k+M)}$

iii) Polynomial multiplication:

The convolutional code can also be described in polynomial form:

$$c^{(j)}(D) = g^{(j)}(D)m(D), \qquad j = 1,2, \dots, n,$$

where the degree of $c^{(j)}(D)$, i.e., $\deg\left(c^{(j)}(D)\right)$, is the sum of the degrees of $g^{(j)}(D)$ and $m(D)$, that is,

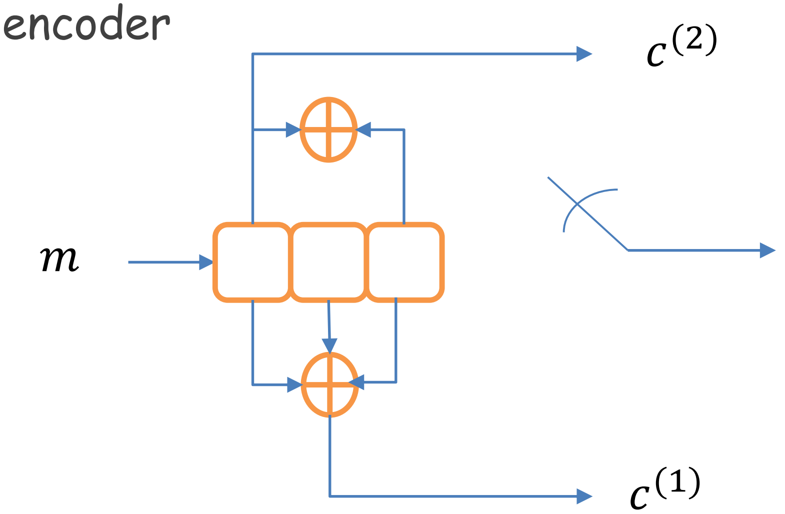
$$\deg\left(c^{(j)}(D)\right) = \deg\left(g^{(j)}(D)\right) + \deg\big(m(D)\big).$$

The output of the code is given by

$$\begin{aligned} c(D) &= c^{(1)}(D)D^1 + c^{(2)}(D)D^2 + \cdots + c^{(n)}(D)D^n \\ &= \sum_{j=1}^{n} c^{(j)}(D)D^j \\ &= \sum_{j=1}^{n} \left(g^{(j)}(D)m(D)\right)(D)D^j \end{aligned}$$

Example 3

Consider the following convolutional encoder

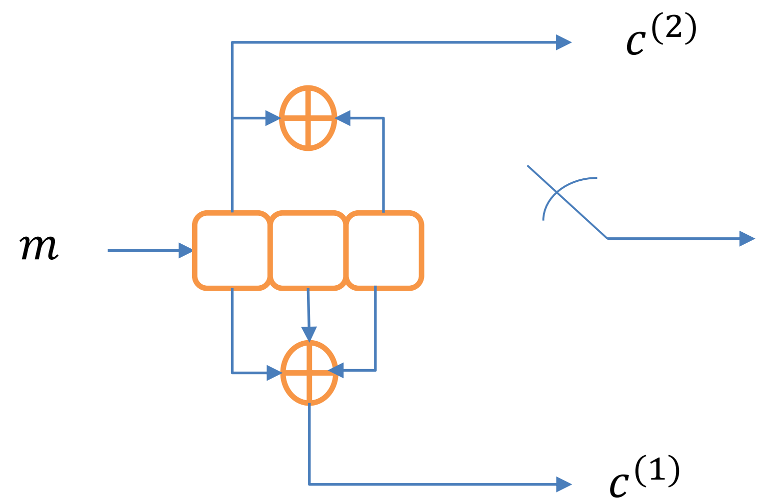


a) Write down the rate, impulse response and the generator polynomials of the encoder.

b) Compute the code for the message $m = [1\ 0\ 0\ 1\ 1]$

Solution:

a) The rate of this encoder is $R = \frac{1}{2}$.



We obtain the impulse response by inspection as follows:

$$\boldsymbol{g}^{(1)} = [1 \;\; 1 \; 1] = 7$$
$$\boldsymbol{g}^{(2)} = [1 \;\; 0 \; 1] = 5$$

The polynomials are

$$g^{(1)}(D) = 1 + D + D^2$$
$$g^{(2)}(D) = 1 + \; D^2$$

b) The message $\boldsymbol{m} = [1\ 0\ 0\ 1\ 1]$ can be written in the form of a polynomial

$m(D) = 1 + D^3 + D^4$

The code polynomials of each path are:

$$c^{(1)}(D) = g^{(1)}(D)m(D) = 1 + D + D^2 + D^3 + D^6$$
$$c^{(2)}(D) = g^{(2)}(D)m(D) = 1 + D^2 + D^3 + D^4 + D^5 + D^6$$

The code is then

$$c(D) = 1 + D + D^2 + D^4 + D^5 + D^6 + D^7 + D^9 + D^{11} + D^{12} + D^{13}$$

$$\boldsymbol{c} = [1\ 1|\ 1\ 0\ |\ 1\ 1\ |\ 1\ 1\ |\ 0\ 1\ |\ 0\ 1\ |\ 1\ 1\ ]$$

iv) Recursive convolutional codes:

We define the generator matrix $\boldsymbol{G}(D)$ as

$$\boldsymbol{G}(D) = \begin{bmatrix} g_1^{(1)}(D) & \dots & g_1^{(n)}(D) \\ \vdots & \ddots & \vdots \\ g_k^{(1)}(D) & \dots & g_k^{(n)}(D) \end{bmatrix} \in \mathrm{F}^{k \times n}$$

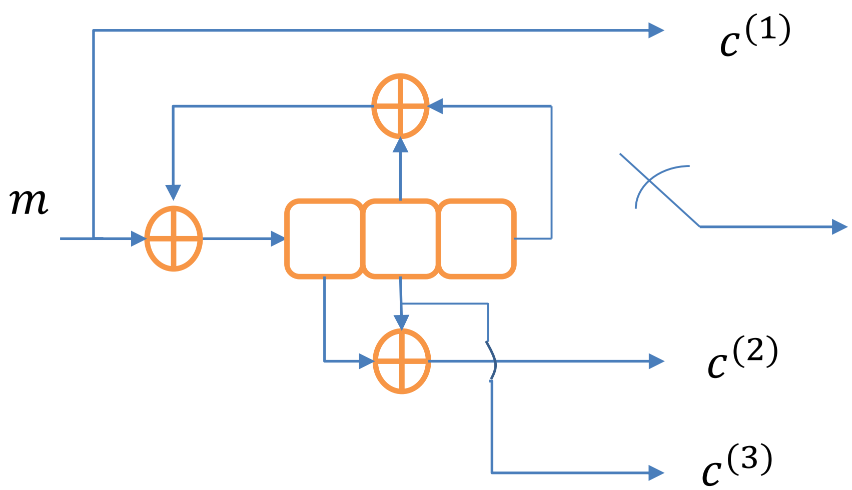
In general, a systematic recursive convolutional code has a generator matrix in the form

$$\boldsymbol{G}(D) = [\boldsymbol{I}_k \mid \boldsymbol{P}(D)] \in \mathrm{F}^{k \times n},$$

where $\boldsymbol{P}(D) = \frac{\boldsymbol{B}(D)}{\boldsymbol{M}(D)}$ describes the feedback part of the encoder.

Example 4

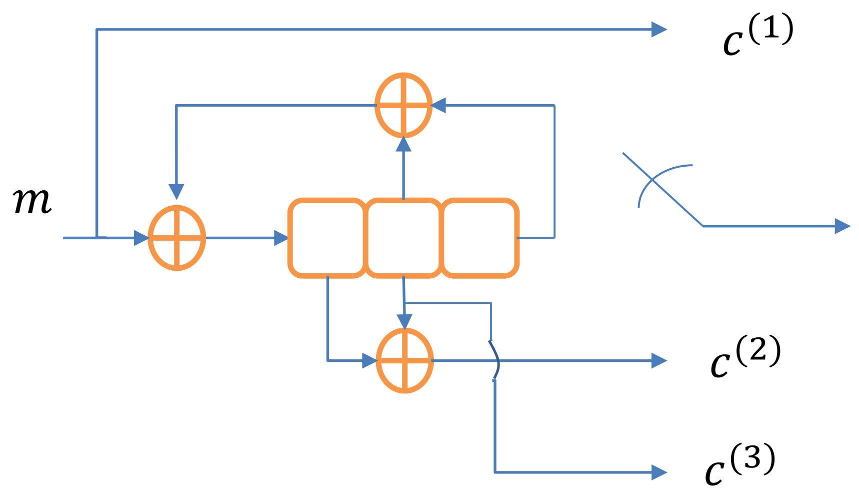
Write down the transfer function known as generator matrix of the encoder



Solution:

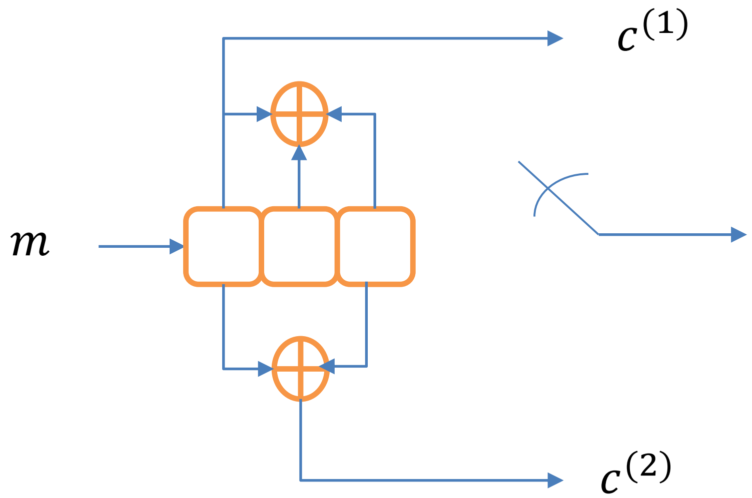
By inspection, we have

$$G(D) = [\boldsymbol{I}_k \mid \boldsymbol{P}(D)]$$
$$= \left[1 \;\middle|\; \frac{1+D}{1+D+D^2} \;\middle|\; \frac{D}{1+D+D^2}\right]$$



C. Structural properties of convolutional codes

- Let us consider the following convolutional encoder:



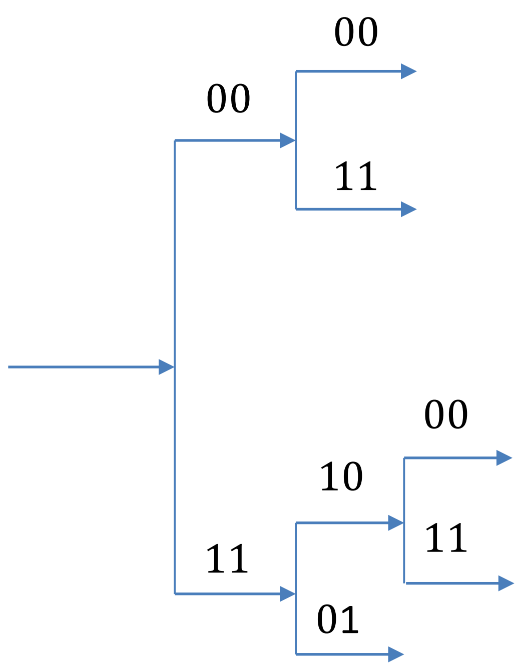
- This encoder has $R = \frac{1}{2}, K = 3$ and $M = 2$, which results in $2^M = 4$ states.

- In what follows, we will examine the states and transitions of the encoders using the code tree, the trellis and the state diagram.

Code tree

Notation:
0 ↑
1 ↓

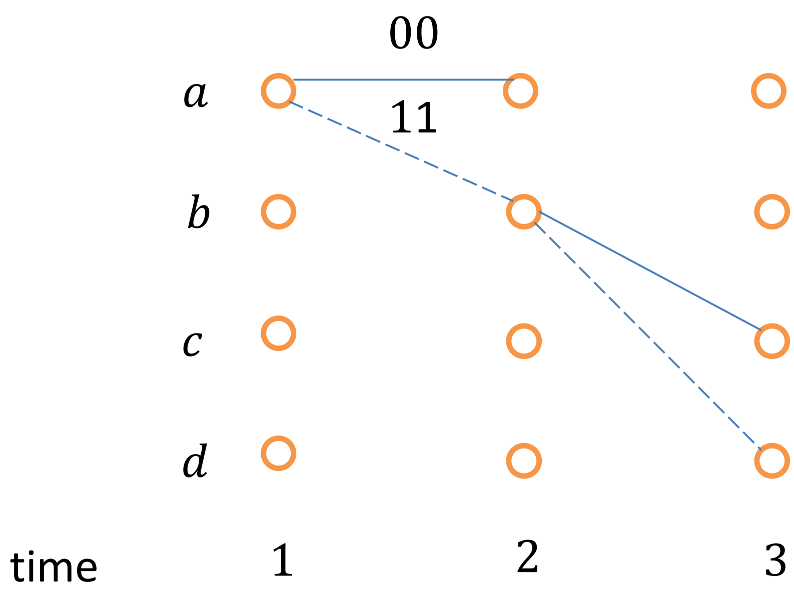
- The code tree is a diagram that allows us to visualize the transitions between the states and the code output.



Trellis diagram

Notation:
0 ____
1 – –

- The trellis diagram allows us to visualize the transitions between the states and the code output without growing vertically.



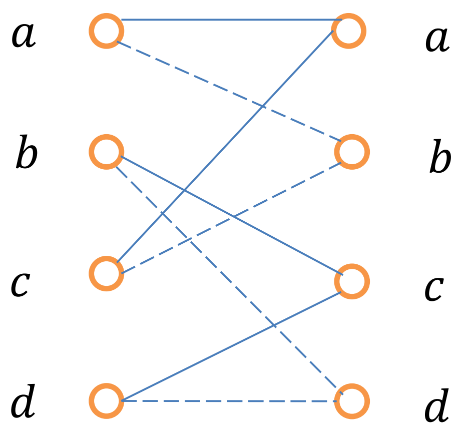
State diagram

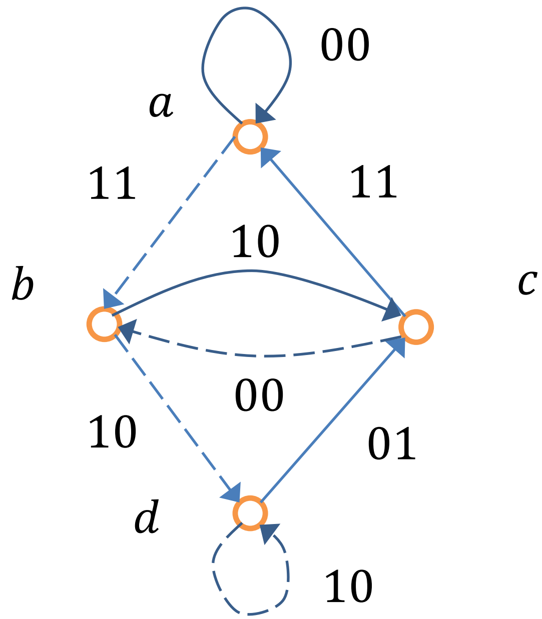
Notation:

0 ____

1 – –

- The state diagram allows us to visualize the transitions between the states and the code output without growing vertically and horizontally.





D. Maximum likelihood decoding

- Consider the message $\boldsymbol{m} = [m_0\ m_1\ \ \ldots m_{k-1}]$ and the code $\boldsymbol{c} = [c_0\ c_1\ \ \ldots c_{n-1}]$ obtained by the encoder as described by the scheme below.

- The received vector is given by

$$\boldsymbol{r} = [r_0\ r_1\ \ \ldots r_{n-1}]$$

- The ML decoder observes $\boldsymbol{r}$ and computes

$$\hat{\boldsymbol{c}} = \arg\max_{\mathbf{c}}\ \log p(\boldsymbol{r}|\boldsymbol{c}),$$

where $\log p(\boldsymbol{r}|\boldsymbol{c})$ is the log likelihood function or ratio (LLR).

- For a BSC channel, we have

$$p(\boldsymbol{r}|\boldsymbol{c}) = \prod_{i=1}^{n} p(r_i|c_i),$$

where $p(r_i|c_i)$ is the conditional transition probability for each bit.

- Computing the $\log$ of $p(\boldsymbol{r}|\boldsymbol{c})$, we obtain

$$\log p(\boldsymbol{r}|\boldsymbol{c}) = \sum_{i=1}^{n} \log p(r_i|c_i)$$

- Defining the transition probability as

$$p(r_i|c_i) = \begin{cases} p, & \text{if } r_i \neq c_i \\ 1-p, & \text{if } r_i = c_i \end{cases}$$

- Suppose that $\boldsymbol{r}$ differs from $\boldsymbol{c}$ in d positions, then the LLR is given by

$$\begin{aligned} \log p(\boldsymbol{r}|\boldsymbol{c}) &= \sum_{i=1}^{n} \log p(r_i|c_i) \\ &= d \log p + (n-d)\log(1-p) \\ &= d \log\left(\frac{p}{1-p}\right) + \underbrace{n\log(1-p)}_{\text{constant}} \end{aligned}$$

- Since $p \ll \frac{1}{2}$ the ML decoder for a BSC minimizes the Hamming distance between $\boldsymbol{r}$ and $\boldsymbol{c}$.

- ML decoding strategies:
  - Hard decoding: use of the Hamming distance $d(\boldsymbol{r}, \boldsymbol{c})$
  - Soft decoding; use of the Euclidean distance $d_E = ||\boldsymbol{r} - \boldsymbol{c}||$
- Efficient decoding approaches for convolutional codes include:
  - The Viterbi algorithm
  - List decoding
  - Sequential decoding

E. The Viterbi algorithm

- The Viterbi algorithm is a recursive and efficient strategy to perform ML decoding of convolutional codes invented by Andrew Viterbi.

- Consider a code sequence described by $\boldsymbol{c}^{(j-1)} = [\boldsymbol{c}_0, \boldsymbol{c}_1, \dots, \boldsymbol{c}_{j-1}]$, which leaves state s_j at time j.

- This sequence determines a sequence of states given by $\boldsymbol{\pi}_j = [s_0, s_1, \dots, s_j]$ through a trellis.

- Consider also the LLR given by

$$\log p\left(\boldsymbol{r}^{(j-1)} \middle| \boldsymbol{c}^{(j-1)}\right) = \sum_{i=0}^{j-1} \log\ p(\boldsymbol{r}_i|\boldsymbol{c}_i)$$

- Let us define the path metric given for s_j by

$$M_{j-1}(s_j) = -\log p\left(\boldsymbol{r}^{(j-1)} \middle| \boldsymbol{c}^{(j-1)}\right)$$

- Consider now the sequence $\boldsymbol{c}^{(j)} = [\boldsymbol{c}_0, \boldsymbol{c}_1, \dots, \boldsymbol{c}_{j-1}, \boldsymbol{c}_j]$ and its path metric

$$\begin{aligned}
M_j(s_{j+1}) &= -\sum_{i=0}^{j} \log p(\boldsymbol{r}_i|\boldsymbol{c}_i) \\
&= -\sum_{i=0}^{j-1} \log p(\boldsymbol{r}_i|\boldsymbol{c}_i) - \log p(\boldsymbol{r}_j|\boldsymbol{c}_j) \\
&= M_{j-1}(s_j) - \underbrace{\log p(\boldsymbol{r}_j|\boldsymbol{c}_j)}_{\mu(\boldsymbol{r}_j|\boldsymbol{c}_j)-\text{branch metric}} \\
&= M_{j-1}(s_j) + \mu(\boldsymbol{r}_j|\boldsymbol{c}_j)
\end{aligned}$$

- Since $\boldsymbol{c}^{(j)}$ moves on the trellis from state s_j to state s_{j+1} we can write

$$M_j(s_{j+1}) = \sum_{i=0}^{j-1} \mu(\boldsymbol{r}_i|\boldsymbol{c}_i) + \mu_j(\boldsymbol{r}_j|\boldsymbol{c}_j)$$
$$= M_{j-1}(s_j) + \mu_j(\boldsymbol{r}_j|\boldsymbol{c}_j)$$

- Assuming that 2 paths with metrics $M_{j-1}(s_j, 1)$ and $M_{j-1}(s_j, 2)$ arrive at state s_{j+1}, we have

$$M_1\colon\ M_{j-1}(s_j, 1) + \mu_j(\boldsymbol{r}_j|\boldsymbol{c}_j)$$
$$M_2\colon\ M_{j-1}(s_j, 2) + \mu_j(\boldsymbol{r}_j|\boldsymbol{c}_j)$$

- The path with smallest metric is retained and called survivor:

$$M_j(s_{j+1}) = \min(M_1, M_2)$$

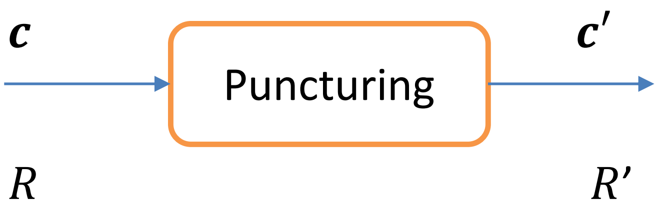
- In summary, the Viterbi algorithm performs the following operations:

$$M_j(s_{j+1}) = \underbrace{\min_{s_j}\left[\underbrace{M_{j-1}(s_j) + \mu_j(\boldsymbol{r}_j|\boldsymbol{c}_j)}_{\text{extension of paths}}\right]}_{\text{choice of smallest metric}}$$

- It only requires the observation of states over a window of time that is appropriate for the computations.

Puncturing

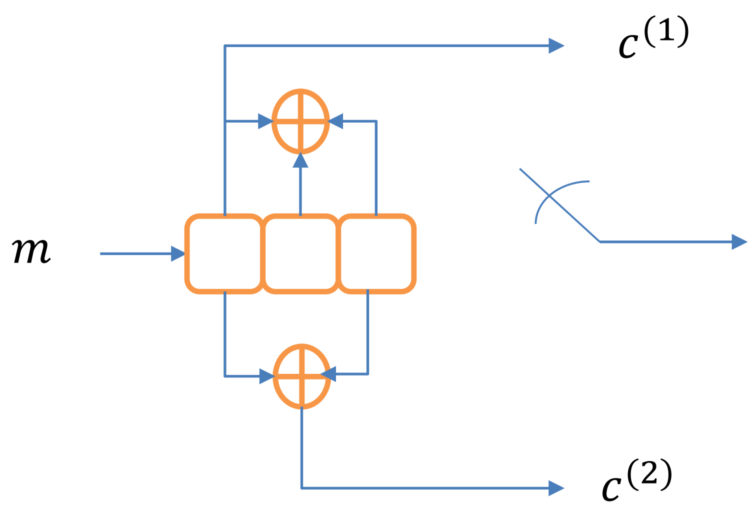
- The use of puncturing consists of removing coded bits to increase the code rate from R to R', where $R' > R$.

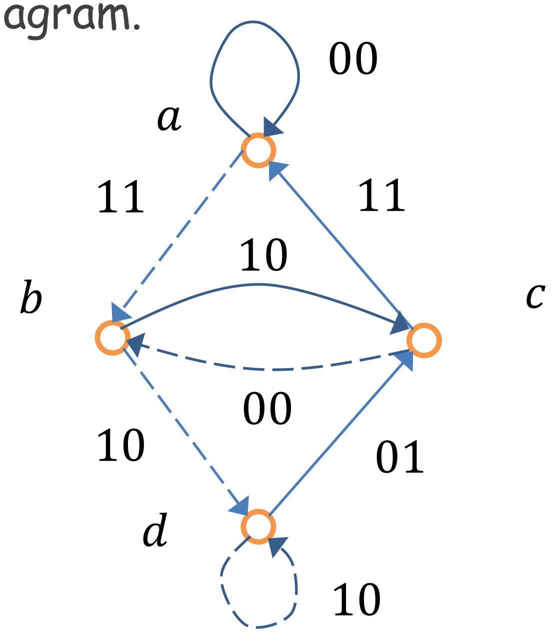


- Decoding:
    - The same trellis is used.
    - The branch metric of the punctured bit does not need to be computed, resulting in computational savings.

Example 5

- Consider the following encoder and its state diagram.





- This encoder has $R = \frac{1}{2}, K = 3$ and $M = 2$, which results in $2^M = 4$ states.

- Suppose that the encoder generates an all-zero sequence that is sent over the BSC channel.

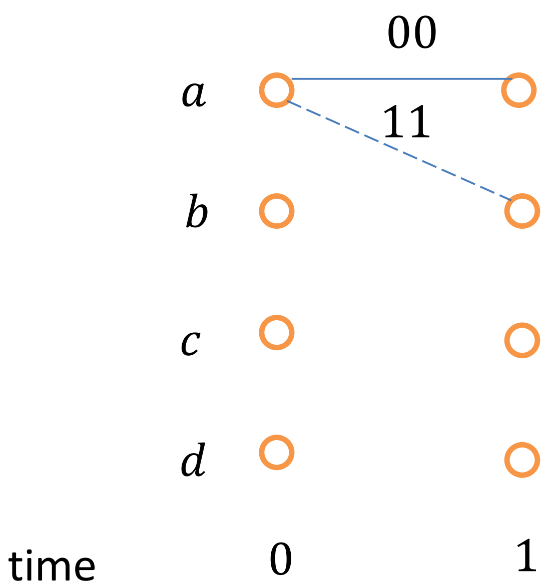
- The received sequence is 0100010000 and contains 2 errors. Decode it.

$$M_j(s_{j+1}) = M_{j-1}(s_j) + \mu_j(\boldsymbol{r}_j|\boldsymbol{c}_j)$$

Solution:

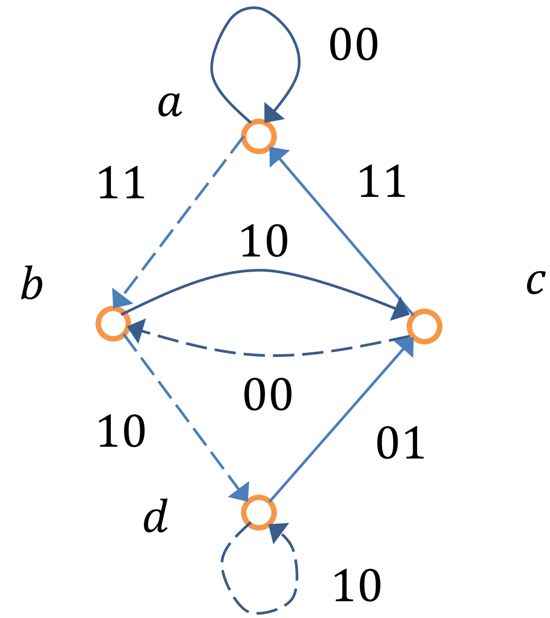
$\boldsymbol{c} = [0000000000]$

$\boldsymbol{r} = [0100010000]$



$M_1(a_2) = M_0(a_1) + \mu(\boldsymbol{r}_1|\boldsymbol{c}_1) = 1$

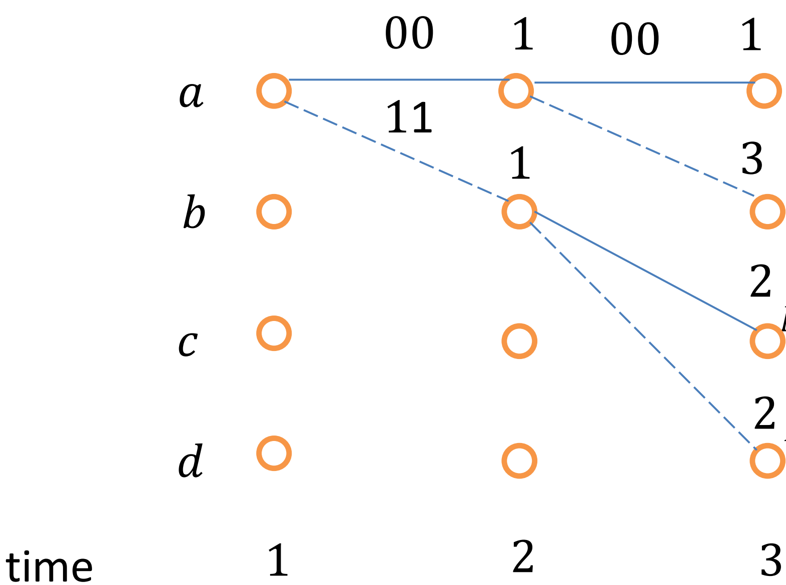
$M_1(b_2) = M_0(a_1) + \mu(\boldsymbol{r}_1|\boldsymbol{c}_1) = 1$



$$M_j(s_{j+1}) = M_{j-1}(s_j) + \mu_j(\boldsymbol{r}_j|\boldsymbol{c}_j)$$

$\boldsymbol{c} = [0000000000]$

$\boldsymbol{r} = [0100010000]$

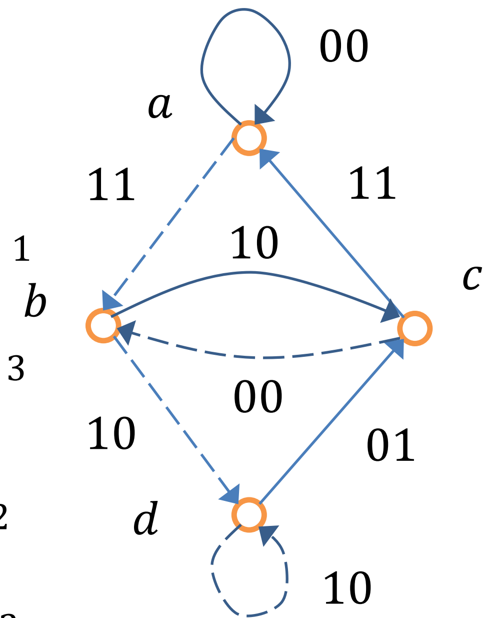


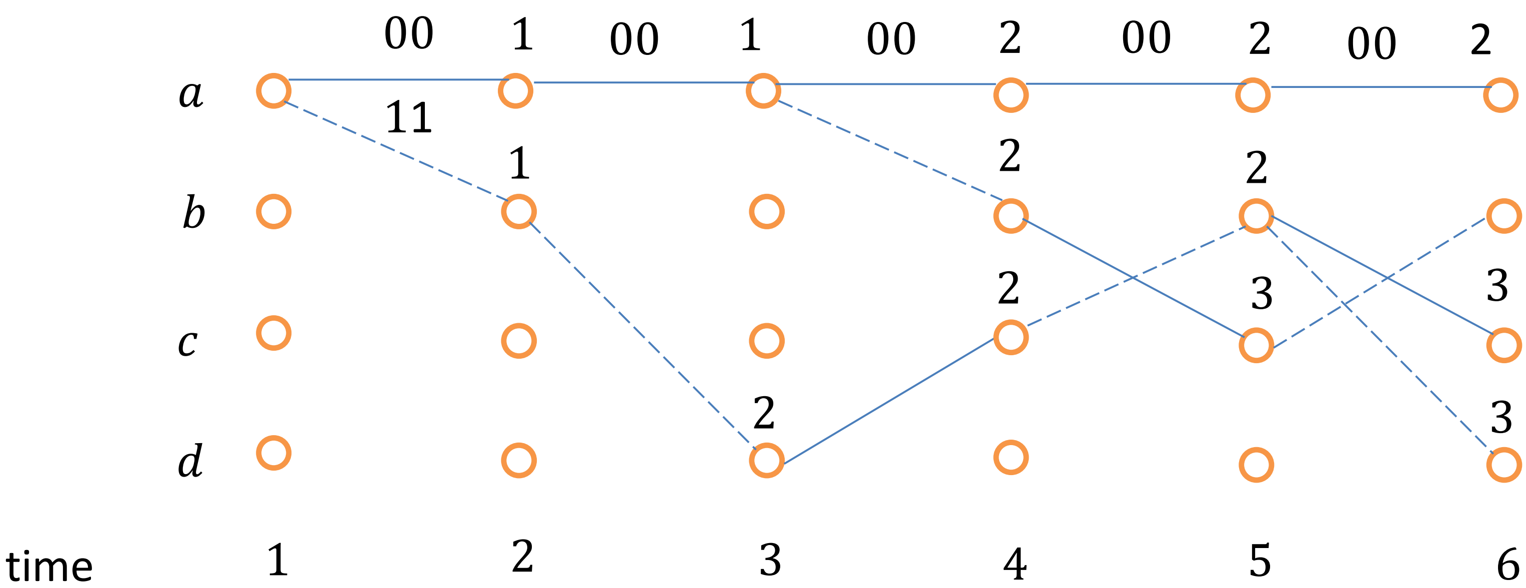
$M_2(a_3) = M_1(a_2) + \mu(\boldsymbol{r_2}|\boldsymbol{c_2}) = 1$

$M_2(b_3) = M_1(a_2) + \mu(\boldsymbol{r_2}|\boldsymbol{c_2}) = 3$

$M_2(c_3) = M_1(b_2) + \mu(\boldsymbol{r_2}|\boldsymbol{c_2}) = 2$

$M_2(d_3) = M_1(b_2) + \mu(\boldsymbol{r_2}|\boldsymbol{c_2}) = 2$





00
1
00
1
00
2
00
2
00
2
a
11
1
2
2
b
2
3
3
c
2
3
d
time
1
2
3
4
5
6

F. Error correction capability

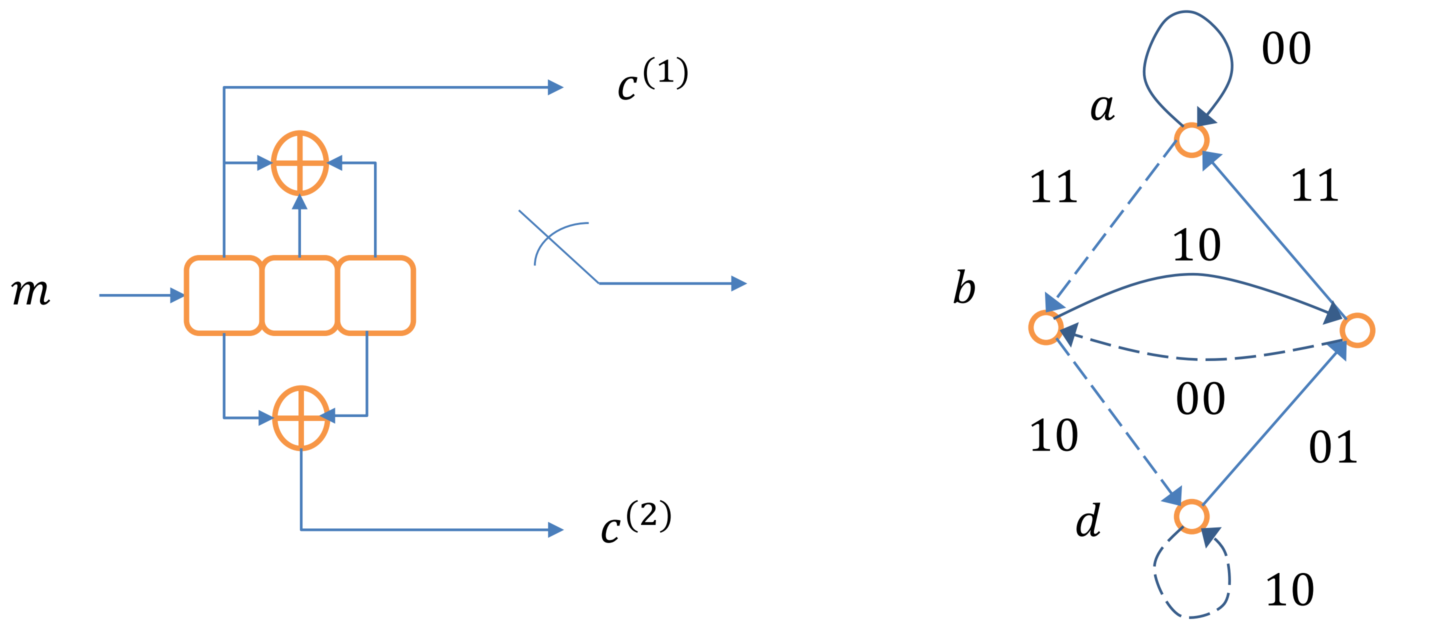
Free distance (d_{free}) :

- It is the minimum Hamming distance of any pair of codewords.
- A convolutional code can correct up to t errors if

$$d_{\text{free}} > 2t$$

- The free distance d_{free} can be obtained by the state diagram and the transfer function of the encoder.

- Consider the following encoder and its state diagram.



- This encoder has $R = \frac{1}{2}, K = 3$ and $M = 2$, which results in $2^M = 4$ states.

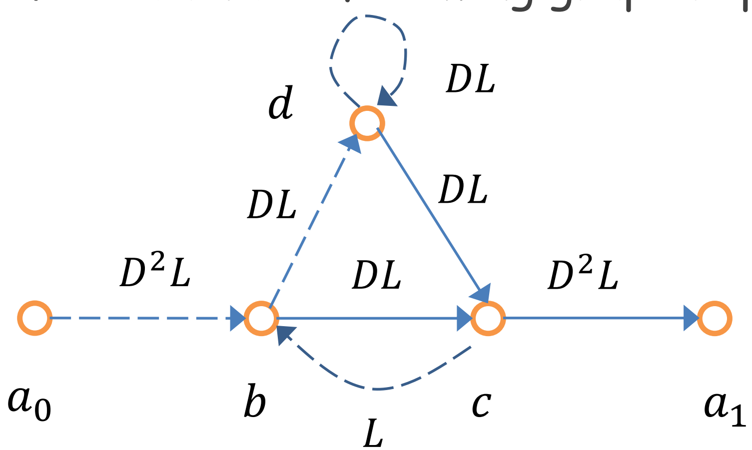
- In what follows, we will show how to obtain its transfer function.

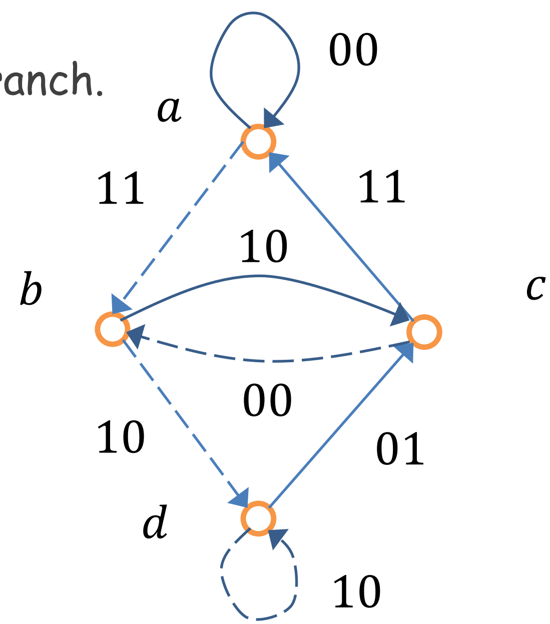
- Starting from its state diagram, the transfer function of a convolutional encoder can be obtained by the following rules:

  - The exponent of D corresponds to the Hamming distance to an all zero codeword.

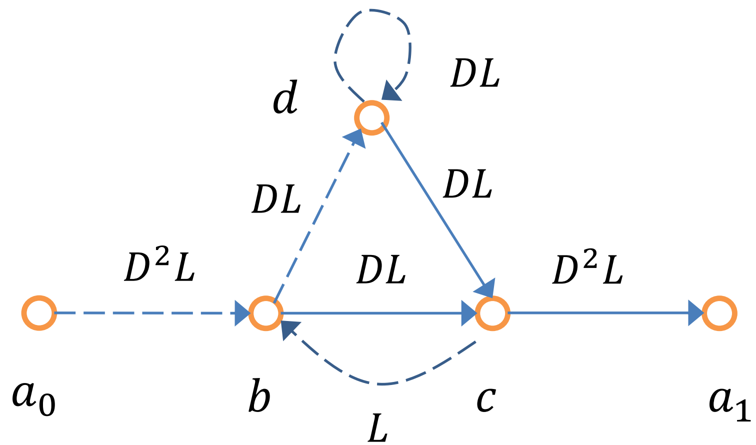
  - The exponent of L corresponds to the length of the branch.

- We obtain the following graph representation:





- Let $T(D,L)$ be the transfer function of the graph with D and L as unknowns.



- Using the rules of the graph and the connections, we obtain a system of equations:

$$b = D^2La_0 + Lc$$
$$c = DLd + DLb$$
$$d = DLb + DLd$$
$$a_1 = D^2Lc,$$

where a_0, b, c, d and a_1 are the graph signals.

- The solution of the previous system of equations yields the transfer function given by

$$T(D,L) = \frac{a_1}{a_0} = \frac{D^5 L^3}{1 - DL(1+L)}$$

- Using the binomial expansion on the transfer function, we obtain

$$T(D,L) = D^5 L^3 \sum_{i=1}^{\infty} (DL(1+L))^i$$

- Setting $L = 1$, we obtain the following power series

$$T(D,L) = D^5 + 2D^6 + 4D^7 + \cdots$$

- The free distance d_{free} corresponds to the smallest degree of $T(D,L)$, which yields

$$d_{free} = 5 \rightarrow \ 2t < 5 \ \rightarrow t = 2 \text{ errors}$$

- The transfer function $T(D, L)$ enumerates the number of codewords with a given Hamming distance.

- It also provides insight into the mathematical structure of the encoder.

- The power series that arises from $T(D, L)$ can be of two types:

    - Convergent

    - Divergent -> leads to catastrophic codes

G. Performance

- In this section, we study the performance in terms of error rates of convolutional codes.
- Assumptions:
  - An all-zero codeword is transmitted.
  - If soft decoding is adopted then the metric is the Euclidean distance.
  - If hard decoding is adopted then the metric is the Hamming distance.

Probability of word error with soft decoding

- The probability of codeword error for soft decoding and for the case in which 2 paths differ by d bits is given by

$$P_w(d) = Q\left(\sqrt{\frac{2E_c d}{N_0}}\right) = Q\left(\sqrt{\frac{2E_b R d}{N_0}}\right) = Q\left(\sqrt{2\gamma_b R d}\right),$$

where $\gamma_b = \frac{E_b}{N_0}$ is the SNR per bit and R is the code rate.

- To compute an upper bound, P_e, on $P_w(d)$ we take into account all possible distances:

$$P_e \leq \sum_{d=d_{free}}^{\infty} a_d P_w(d) = \sum_{d=d_{free}}^{\infty} a_d Q\left(\sqrt{2\gamma_b R d}\right),$$

where a_d is the number of paths with distance d that reach an all-zero codeword.

Probability of bit error with soft decoding

- Consider the transfer function given by

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

- Computing the derivative of $T(D,N)$ with respect to N and setting $N = 1$, we obtain the number of errors for each path:

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

- The probability of bit error is then given by

$$P_b \leq \sum_{d=d_{free}}^{\infty} \beta_d P_w(d) = \sum_{d=d_{free}}^{\infty} \beta_d Q\left(\sqrt{2\gamma_b R d}\right)$$

Performance of bit error with hard decoding

- If d is odd then the path associated with an all-zero codeword will be selected if the number of errors is less than $\frac{1}{2}(d+1)$.

- The probability of selecting the incorrect path is given by

$$P_c(d) = \sum_{k=\frac{d+1}{2}}^{d} \binom{d}{k} p^k (1-p)^{d-k},$$

where p is the probability of error of the BSC.

- If d is even then we have

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

- By summing all the events associated with all distances, we obtain the probability of codeword error given by

$$P_e \leq \sum_{d=d_{free}}^{\infty} a_d P_c(d)$$

Performance of bit error with hard decoding

- To compute the probability of bit error we consider the transfer function given by

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

- We then compute the derivative of $T(D,N)$ with respect to N and set $N = 1$, we obtain the number of errors for each path:

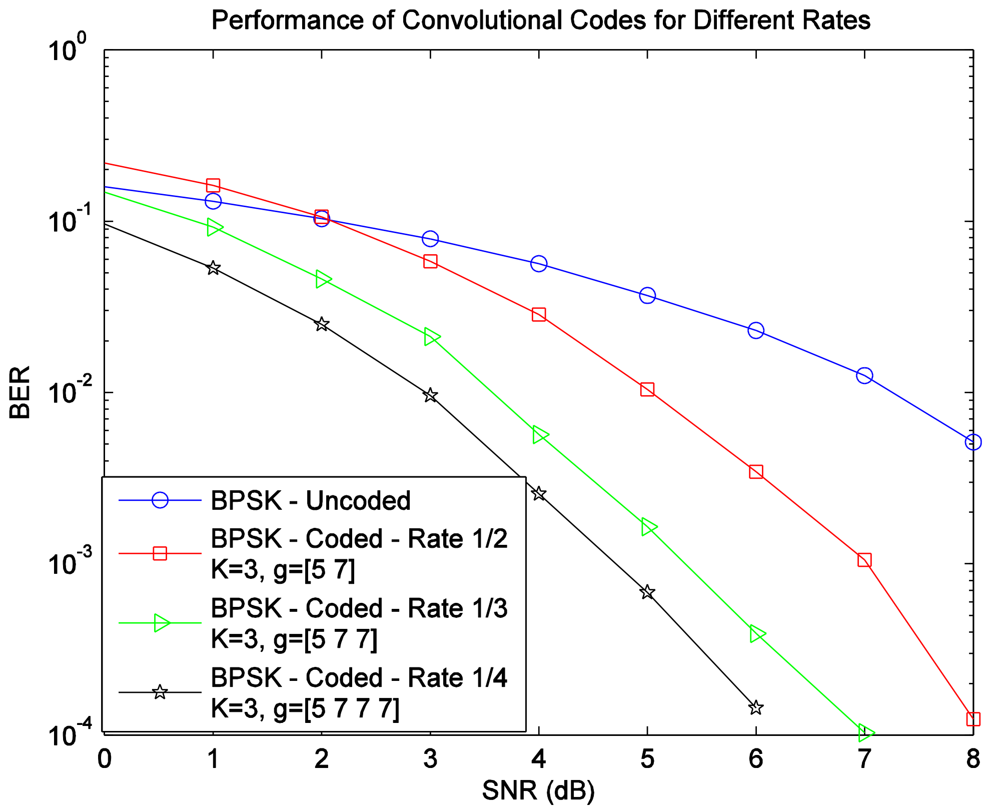
$$\mathrm{d}\frac{T(D,N)}{dN} = \sum_{d=d_{free}}^{\infty} a_d f(d) D^d = \sum_{d=d_{free}}^{\infty} \beta_d D^d$$

- The probability of bit error is then given by

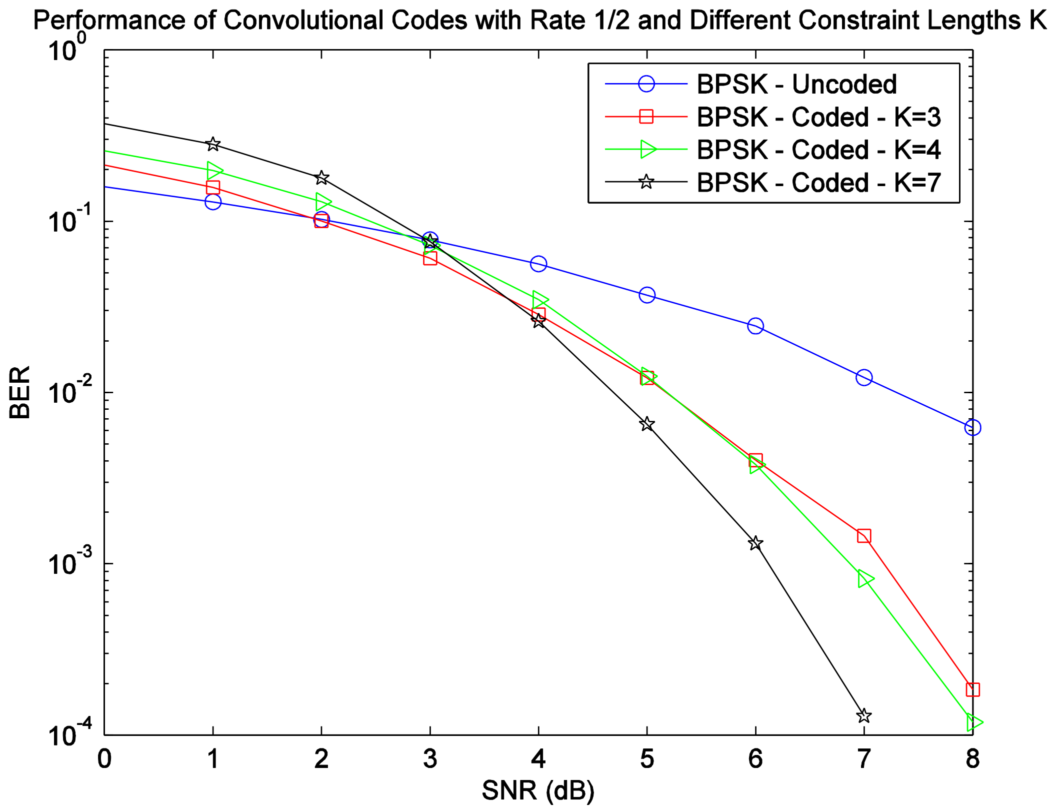
$$P_b \leq \sum_{d=d_{free}}^{\infty} \beta_d P_c(d)$$

Example 6

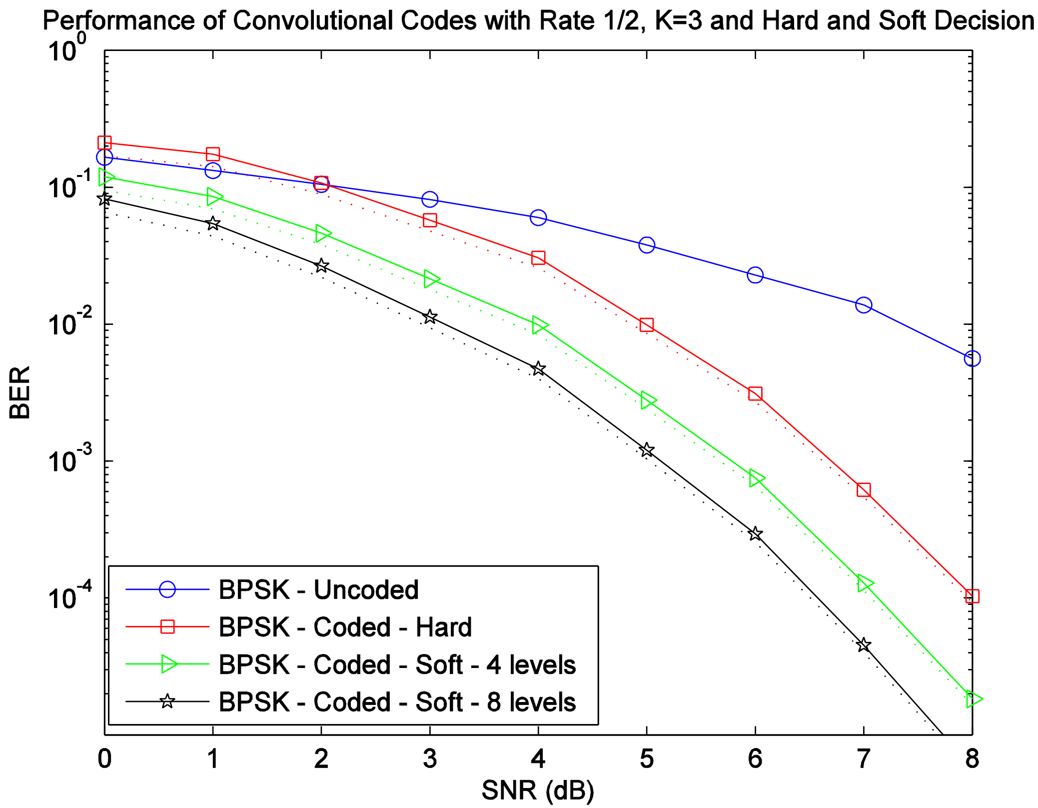
In this example, we consider the performance of convolutional codes with different constraint lengths, rates and decoding strategies.



Performance of Convolutional Codes for Different Rates
BER
SNR (dB)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
6
7
8
BPSK - Uncoded
BPSK - Coded - Rate 1/2
K=3, g=[5 7]
BPSK - Coded - Rate 1/3
K=3, g=[5 7 7]
BPSK - Coded - Rate 1/4
K=3, g=[5 7 7 7]



Performance of Convolutional Codes with Rate 1/2 and Different Constraint Lengths K
BER
SNR (dB)
BPSK - Uncoded
BPSK - Coded - K=3
BPSK - Coded - K=4
BPSK - Coded - K=7



Performance of Convolutional Codes with Rate 1/2, K=3 and Hard and Soft Decision
BER
SNR (dB)
BPSK - Uncoded
BPSK - Coded - Hard
BPSK - Coded - Soft - 4 levels
BPSK - Coded - Soft - 8 levels