Experiment (1)

Introduction to Probability of error calculation using MATLAB

Objective:

(1) Investigate the systematic procedure of evaluating the BER in communication Systems.
(2) Investigate the performance of digital communication system.

Theoretical Background:

(1) Evaluating BER for wireless communication systems:

To perform Monte-Carlo simulation (i.e. calculating the BER via simulations using MATLAB) you must perform the following

a. Generate an array of random bits.
b. Add noise (based on SNR).
c. Detect received bits from the noisy received signal
d. Count number of errors.
e. Previous steps are repeated large number of iterations and the BER is found by averaging.
f. Steps "a" to "f" is repeated per SNR.

(2)AWGN channel modeling:

To model the AWGN by its baseband equivalent you will generate a Gaussian distributed signal with total power (i.e. variance) equals to signal power/SNR.
For simplicity we always normalize the signal to unity so you can model the AWGN channel by the following equation

$$noise = \frac{1}{\sqrt{SNR}} * randn$$

Procedure:

(1) Simulation parameters:
    a. Number of bits/SNR=1e6 bits
    b. Signal to noise ratio range=0 to 30 dB with 2 dB steps.
(2) Generate random binary data vector (you can make use of `randint` or `randi`).
(3) Apply noise to bits (Hint: you must calculate the signal power in this case because it is not unity)
    `Rx_sequence=bits+noise.`
    Or
    `Rx_sequence=awgn(bits,snr,'measured')`
(4) Decide whether the Rx_sequence is '1' or '0' by comparing the samples with threshold=1/2 (Hint: try to use relational operators and indexing to make the code more efficient)
(5) Compare the original bits with the detected bits and calculate number of errors (you can make use of `xor` or `biterr`).

(6) Save the probability of error of each SNR in matrix , BER
```
BER=[BER  new prob. of error]
```
(7) Plot the BER curve against SNR (use `semilogy`)

Report requirement:

(1) Well commented M-file.
(2) Softcopy report containing required figure
(3) Calculation of transmitted signal power
(4) Identifying meaning of 'measured' field?
(5) At which value of SNR the system is nearly without error (for the given frame)?

Experiment (2)

Performance of Matched filters and correlators

Objective:

(1) Investigate the matched filter and correlator receivers
(2) Modify the system in experiment (1) to use a matched filter receiver instead of simple sampler.
(3) Compare the performance of simple detector and matched filter receiver.

Theoretical Background:

(1) Matched filter receivers and correlators:

Matched filter is baseband filter which is designed as function of transmitted waveforms to maximize the received SNR at the designed sampling instant and hence minimizes the resultant probability of error and considered the optimal receiver in case of AWGN channel.

The matched filter is designed in case of AWGN channel as

$$h_{MF}(t) = c(s_1(T-t) - s_2(T-t))$$

And the output of the matched filter is the convolution between $h_{MF}(t)$ and received signal to have a MF output that will be further sampled to be the input of the simple detector.

To ease the design of implementing of the MF, the correlator is proposed. The correlator consists of multiplier and integrator , it will multiply by $g(t)$ which equals to

$$g(t) = s_1(t) - s_2(t)$$

The received signal is multiplied by this $g(t)$ and then accumulated to have a value that will be compared with the threshold.
The correlator receiver output equals the MF output at the sampling instant.The threshold at this case can be calculated as

$$V_{th} = \frac{s_1(T)+s_2(T)}{2} = c\frac{E_1-E_2}{2}$$

(3) Modeling Matched filter receiver

To model the Matched filter receiver using Matlab , you should use the sampled MF version , That will result in sampling $h_{MF}(t)$ by specified number of samples (example:10 samples) , the transmitted signal itself should be represented by same number of samples , then you can perform convolution at the receiver and choose the sample in the middle which represents the sampling process and then you can repeat experiment (1)

Procedure:

(8) Simulation parameters:
   a. Number of bits/SNR=1e6 bits
   b. Signal to noise ratio range=0 to 30 dB with 2 dB steps.
   c. Number of samples that represents waveform $m = 10$
   d. Sampling instant =10
   e. Receiver type : Matched filter and correlator.
   f. $s_1(t)$ is rectangular signal with Amp=1 and $s_2(t)$ is zero signal

(9) Generate random binary data vector (you can make use of `randint` or `randi`).

(10) Represent each bit with proper waveform (hint: use concatenation , notice that the resultant vector will be 10e6 samples)

(11) Apply noise to samples (Hint:For fair comparison between MF and simple detector you should equate the average power in the two cases , and hence you should notice that the power of signal in case of MF is calculated by adding the squares of all samples that represents $s_1(t)$ and hence SNR will change)

```
Rx_sequence=bits+noise.
```
Or
```
Rx_sequence=awgn(bits,snr,'measured')
```

(12) Apply convolution process in the receiver (Hint: the convolution process will be performed in bit by bit basis i.e. 10 samples by 10 samples) You may use `conv`
In case of correlator you will apply element by element multiplication and then add the resultant samples together.

(13) Sample the output of the Matched filter (i.e. choose the middle sample , you may use indexing tools)

(14) Decide whether the Rx_sequence is '1' or '0' by comparing the samples with threshold (Hint: try to use relational operators and indexing to make the code more efficient)

(15) Compare the original bits with the detected bits and calculate number of errors (you can make use of `xor` or `biterr`).

(16) Save the probability of error of each SNR in matrix , BER

(17) Plot the BER curve against SNR (use `semilogy`)

Report requirement:

(6) Well commented M-file.

(7) Softcopy report containing required figures (MF and correlator and its comparison to simple detector) , comment.

(8) Calculation of transmitted signal power

(9) At which value of SNR the system is nearly without error (for the given frame)?

(10) The code should run in a reasonable time (around few minutes) (Hint: try to reduce the number of loops as much as possible and try to consider preallocation)

Bonus:

(1) Generalize the program to acquire any number of samples that represents $s_1(t)$,and any sampling instant, compare performance of having 10,20,100 samples
(2) Generalize program to have $s_1(t)$ and $s_2(t)$ as general waveforms or user defined .

Experiment (3)

Performance of Different Modulation types

Objective:

Compare the performance of the different modulation schemes(ASK – FSK – PSK).

Theoretical Background:

(1) ASK has many cases, but the case we are interested in here is the special case which is called OOK.
(2) PSK has many cases, but the case we are interested in here is the special case which is called PRK.
(3) FSK has many cases, but the case we are interested in here is the special case which is called orthogonal-FSK, in which the 2 transmitted bits are sent on 2 orthogonal carriers.

Procedure:

(4) Simulation parameters:
    a. Number of bits/SNR=1e6 bits
    b. Signal to noise ratio range=0 to 30 dB with 2 dB steps.
(5) Generate random binary data vector (you can make use of `randint` or `randi`).
(6) Modulate the signal according to the type of modulation you want, ex :
    OOK : No change in the bits will be required
    PRK: You will have to represent the 1 by 1 and the 0 bit by -1 (i.e you can use this formula : 2*vector_bits-1)
    FSK : You will have to modulate the first bit of the bit stream on a certain carrier and the other bit on a carrier orthogonal on it and so on (it can be done by matlab as : if bit to send=0 send 1 else send i, where i:is the complex number)
(7) Apply noise to bits(or symbols in case of FSK) (Hint: you must calculate the signal power in this case because it is not unity)
    `Rx_sequence=bits+noise.`
    Or
    `Rx_sequence=awgn(bits,snr,'measured')`
(8) Decide whether the Rx_sequence is '1' or '0' (Hint: try to use relational operators and indexing to make the code more efficient)
(9) Compare the original bits with the detected bits and calculate number of errors (you can make use of `xor` or `biterr`).
(10) Save the probability of error of each SNR in matrix , BER
    `BER=[BER new prob. of error]`
(11) Plot the BER curve against SNR (use `semilogy`)

Report requirement:

(1) Well commented M-file.
(2) Softcopy report containing required figures (BER figure for all 3 types of modulation on the same figure).
(3) Which type of modulation has the best performance?Why?
(4) At which value of SNR the system is nearly without error (for each type of modulation)?

Bonus:

(1) Evaluate the same curves using the MATLAB built-in function `modem.pskmod`, `modem.pammod` ,……
(2) Evaluate the probability of error of the 16QAM modulation.

Experiment (4)

Pulse code modulation (PCM)

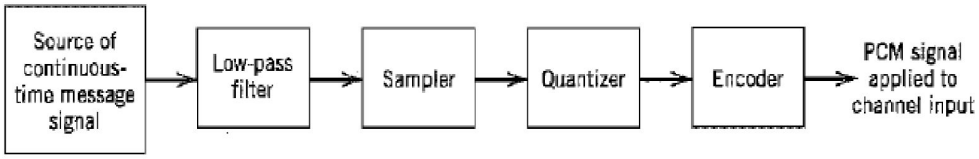
Objective:

(1) Investigate the PCM system components.
(2) Investigate the effect of changing the number of levels.
(3) Investigate the oversampling, critical sampling and undersampling cases.
(4) Calculate the quantization error of the transmitted signal.

Theoretical Background:

(2) PCM overview

PCM , pulse code modulation is the traditional baseband analog to digital converter which consists of sampling , quantization and coding processes.



The sampling process is transformation of analog signal into discrete signal, the signal must be sampled with at least Nyquist rate $f_s = 2f_m$. At this case the sampling would be critical sampling which requires a sharp ideal reconstruction filter.if the sampling is done with $f_s \gg 2f_m$, the sampling would be oversampling and a smoother filter can be used . if $f_s < 2f_m$, the signal cannot be reconstructed due to aliasing.

Quantization process is approximation process of the output samples from the sampling process to discrete levels which equals 2^n where n is the number of bits that defines the sample. Increasing number of bits will result in enhancing the SQNR and hence better QoS.

The encoding process is transforming the quantized bits into proper line code

Calculation mean square quantization error

The quantization error is calculating by averaging the quantization error over all the transmitted samples

$$MSE_q = \frac{1}{N}\sum_{n=0}^{N-1}[x_q(n) - x(n)]^2$$

Generate any signal , quantize it.Draw the quantization error on the Y axis versus n bits on the horizontal axis

Procedure:

Quantization error:

(1) Generate a sinusoidal wave of the following parameters:
   a. Amplitude=1V.
   b. Frequency= 2Hz.
   c. Sampling frequency=4000Hz.

(2) Quantize the sampled signal by m bits where $m = 2n + 1$ and n is the number of bits that will represents the integer value and the fraction part and the last bit is the sign bit
(Hint : use `fi` command)
`a = double(fi(v,s,m,n))`

(3) Convert the quantized samples to binary (you may use `de2bi` or any other suitable method)

(4) Calculate the mean square quantization error as equation for $n = 3, 4, 5, 10$

Sampling distortion:

```
% this part below must be copied to your m file and complete the       %
required


clear
clc

% reconstruction from oversampling
t=0:0.001:1;                    % time signal
y=2*cos(2*pi*5*t);

[B,A] = BUTTER(3,1000/100000,'low');       % butter fly filter
zero_added_signal=zeros(1,length(y)*10);
for i=1:length(y)
    zero_added_signal(i*10)=y(i);
end

zero_added_signal(1:9)=[];
% Adding zeros enhances the signal display and don't change the
%spectrum,it changes sampling freq. only
t=linspace(0,1,length(zero_added_signal));
filtered_signal = filter(B,A,zero_added_signal);
plot(t,filtered_signal,'r')
XLABEL('time')
YLABEL('oversampled signals')


% construction from minimum sampling
figure
t=0:??:1;  % replace ?? with the suitable number
```

```
y=2*cos(2*pi*5*t);
[B,A] = BUTTER(10,0.1,'low');
zero_added_signal=zeros(1,length(y)*10);
for i=1:length(y)
    zero_added_signal(i*10)=y(i);
end

zero_added_signal(1:9)=[];
t=linspace(0,1,length(zero_added_signal));
filtered_signal = filter(B,A,zero_added_signal);
plot(t,filtered_signal,'r')
XLABEL('time')
YLABEL('minimum sampled signals')

s=fft(filtered_signal);
s=fftshift(s);
fs=100;   % why 100?? Write your comments in the m file
freq=linspace(-fs/2,fs/2,length(s));
figure
plot(freq,abs(s))
XLABEL('freq')
YLABEL('magnitude of minimum sampled signals')

% construction from undersampling sampling

figure
t=0:0.2:1;
y=2*cos(2*pi*5*t);
[B,A] = BUTTER(10,0.2,'low');
% complete this part as shown in the construction from minimum sampling
%and do the necessary changes , you have to do low pass filtering and  %
displays the spectrum
```

For this code Complete the attached file to see the difference between the oversampling, critical sampling and undersampling, you need to understand all the used commands.

Report requirement:

(11) Well commented M-file.
(12) Softcopy report containing required figures , comment.

Bonus:

(3) Repeat the program for changing fraction and integer parts , find the best resolution
(4) Repeat the program using a quantizer of your choice using `quantize` command
(5) Repeat the program with non-uniform quantizer (you can use `compand`)

Experiment (5)

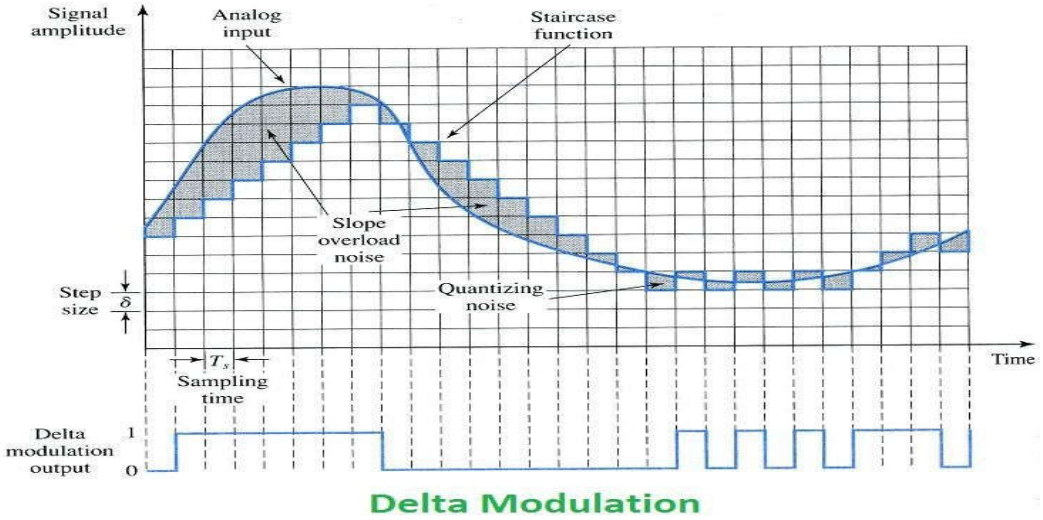
Delta - Modulation types

Objective:

Investigate the performance of the delta modulation.

Theoretical Background:

(1) Delta modulation is a technique to reduce the number of bits sent from the transmitter, than the case of PCM.
(2) Delta- modulation is based on just sending the difference occurring in the signal, and not the exact value.
(3) The delta modulation has two most important parameters, which are the step width and step height.
(4) There are 2 main types of quantization errors in the delta-modulated signals:
   i- Slope overload :which is when the slope of the input signal is higher than the slope of the stairs of the delta-modulation.
   ii- Granular noise : it is when the signal is a constant value, also called quantizing noise.



Delta Modulation

Procedure:

(1) Simulation parameters:
   a. Use a sine wave signal of frequency =500 Hz (use high sampling frequency >>2*fm).
   b. Find a suitable Ts and Step size (δ) for the delta modulation.
   c. Plot the output signal versus the delta-modulated signal (as figure above).
   d. Restore the delta- modulated signal using a suitable low-pass filter.
   e. Calculate the value of the square error (original signal - reconstructed signal)^2.

(2) Repeat the previous steps (b-e) when the input is (for the same Ts and δ):
   i- DC voltage.
   ii- Square wave(500 Hz).
(3) Repeat 1 and 2 when
   i- δ_{new}=.1* δ_{old} , while Ts is the same as in part 1.
   ii- Ts_{new}= .1*Ts_{old}, while δ is the same as in part 1.

Report requirement:

(1) Well commented M-file.
(2) Softcopy report containing required figures types (you have to put them on the same figure using subplot).
(3) The square error in each type.
(4) Which type of signal has the lowest error? Comment
(5) Explain a modified delta modulation technique to reduce the slope overload and granular noise (use schematic block diagrams).

Bonus:

(1) Modify the δ modulation in the experiment to have a variable slope δ modulation ex. The modulator will increase by δ in case of low slope signals and 2δ in case of high slope signals. Calculate the error in this case.
(2) Modify the δ modulation to be a DPCM system with 4 level quantizer and predictor of

$$\hat{x}[n] = 2x[n-1] - x[n-2]$$

Calculate the error in this case

Experiment (6)
Line Codes

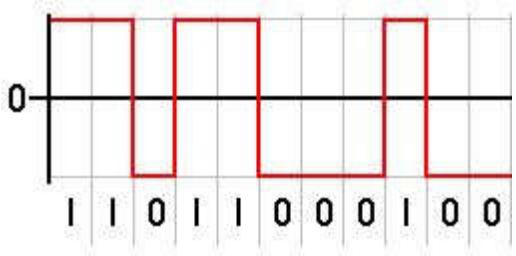
Objective:

(1) Compare the different types of line codes used in digital communications.

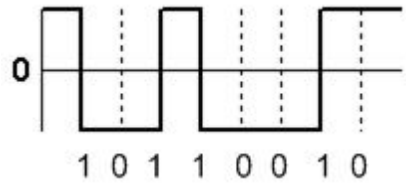
Theoretical Background:

(1) Line codes are also called **digital baseband modulation**.
(2) There are many different types of line codes and each one of them has an advantage from a certain point of view.
(3) The main types are :

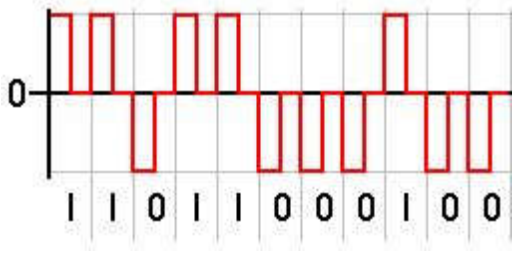
i- Non-return to zero.



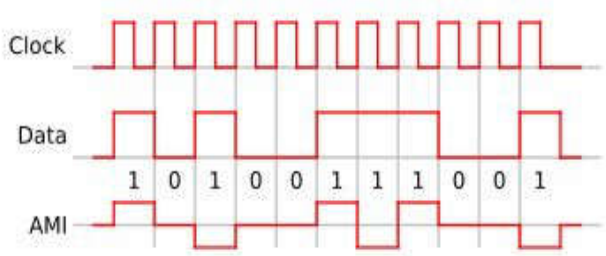
ii- Non-return to zero inverted.



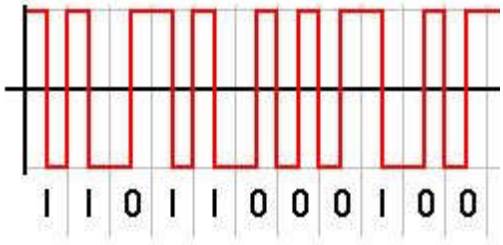
iii- Return to zero.



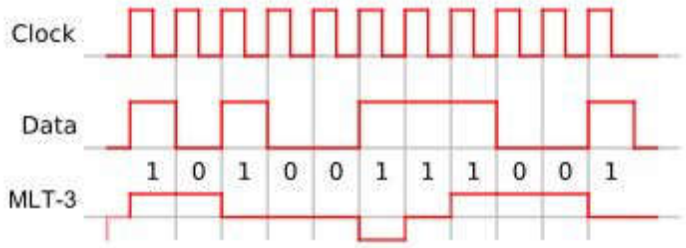
iv- Alternative mark inversion (AMI)



v- Manchester coding



vi- Multi-level transmission 3



Procedure:

(1) Generate random bits of zeros and ones.
(2) Modulate this same vector using the different types of line codes.
(3) Plot a sample of all previous line code modulation and plot them under each other on the same figure using subplot, ex subplot(6,1)(make sure that all the types have the same periodic time Ts).
(4) Find the power spectrum density of each code, and plot them in the same figure using subplot as previous (you may try "`psd`" function on the matlab).
(5) Comment on each of the figures.

Report requirement:

(1) Well commented M-file.
(2) Softcopy report containing required figures and the comments on each figure.
(3) Which type of signal has the highest bandwidth? Comment.
(4) Mention the advantages and disadvantages of each line code.
(5) Mention in the report 2 other used line codes and explain them mentioning the main advantages or disadvantages.

Experiment (7)
Channel coding

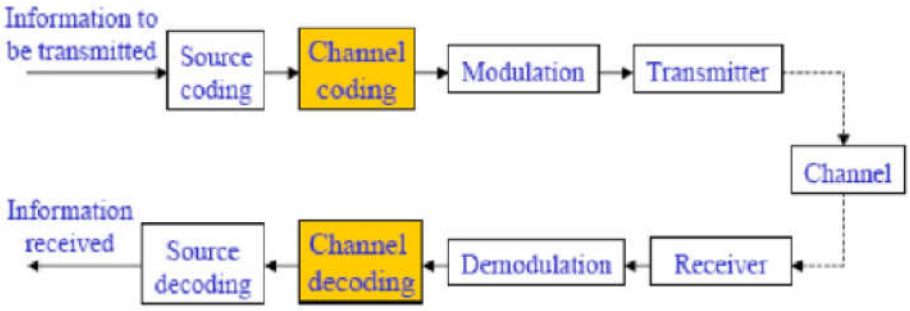
Objective:

(1) Investigate the importance of channel coding.
(2) Investigate the repetition codes.
(3) Investigate the linear block codes.
(4) Investigate the convolutional codes.
(5) Investigate the main parameters of the convolutional codes.
(6) Calculation of BER in case of coded transmission.

Theoritical Background:

(A) Channel coding:

Coding is achieved by adding properly designed redundant bits to each message, or make an operation on the message to get it encoded with some methods. These redundant bits (digits) are used for detecting and/or correcting transmission errors, in other words for protecting data against channel impairments (e.g., noise, fading, interference). There are many codes that are used in different applications.



(B) repetition codes:

In repetition codes, every bit is transmitted an odd number of times so that the receiver can detect and correct the errors generated due channel effect via majority decoding
Ex. If m=101 , transmitter will send it 3 times so transmitted code will be c=111 000 111 if 2 errors are introduced r=110 010 111 so the receiver will decide based on the majority of bits so the decision will be 1 0 1

(c) Linear block codes:

A code is said to be linear if any two code words in the code can be added in modulo-2 arithmetic to produce a third code word in the code. Consider then an (n, k) linear block code, in which k bits of the n code bits are always identical to the message sequence to be transmitted. The n-k bits in the remaining portion are computed from the message bits in accordance with a prescribed encoding rule that determines the mathematical structure of the code. Accordingly, these n - k bits are referred to as generalized parity check bits or simply parity bits. Block codes in which the message bits are transmitted in unaltered form are called systematic codes

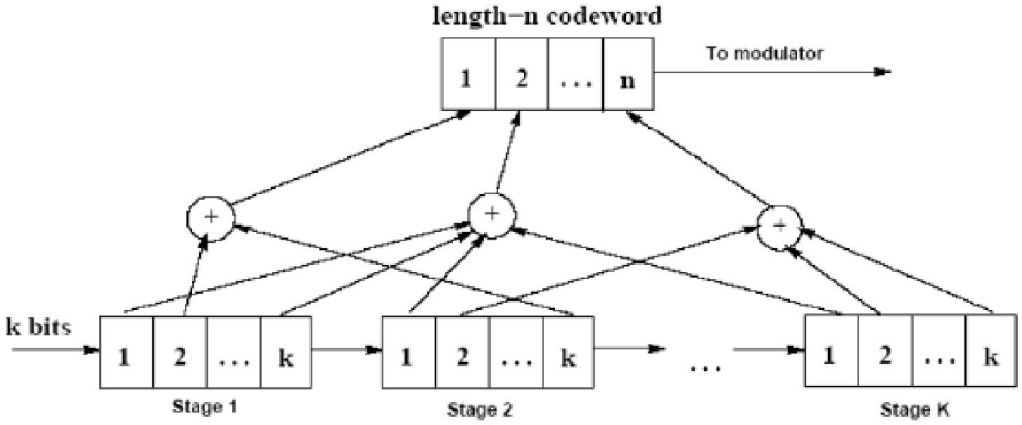
The encoding process is done as follows:

$C = mG$

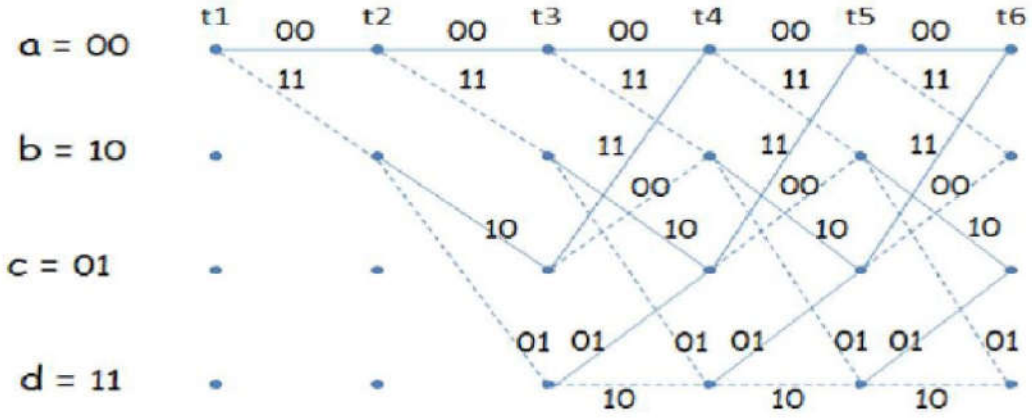
Where m is the message vector and G is the generator matrix.
The decoding procedure is done by Syndrome decoding.

(C) Convolutional codes:

A convolutional code is described by three integers, n, k. and *K,* where. The integer *K* is a parameter known as the constraint length*,* it represents the number of k stages in the encoding shift register.



The encoder is usually represented by the trellis structure
The decoding process is done by the Viterbi algorithm.



Procedure:

(1) Simulation parameters:

a. Number of bits/frame=1e6
b. Signal to noise ratio range=0 to 30 dB with 2 dB steps.
c. Modulation type: BPSK.
d. Fading model:AWGN

(2) Generate random data sequence.
(3) Choose a suitable systematic linear block code generating matrix.
(4) Encode the sequence using your generating matrix (you can use `encode, decode, gen2par, syndtable`)
(5) Modulate the encoded sequence using BPSK modulation.
(6) Refer to experiment (1) to completely evaluate the BER for systematic linear block code case.

For repetition code

Modify the Simple detector system in experiment 1 (part2) to have a repetition code based system (you may repeat the transmitted vector "n" times and then reshape). Apply simple detection rule and majority decoding find P_e of the new system for n=3,5,11.

Hint: Note that the calculation of signal power in this case is analogous to MF experiment

For convolutional code:

(1) Use the following steps for encoding:

```
constlength=9;
traceback=5*constlength;
polynomial=[657 435];
trellis = poly2trellis(constlength,polynomial);
x=convenc(BITS,trellis);
```

(2) For Decoding process: use the following

```
vitdec(x,trellis,traceback,'trunc','hard')
```

(3) Compare the performance of the convolutional code,linear block code case, uncoded case.
(4) Change the parameters of the convolutional code (constraint length , puncture pattern ,....) .what do you notice.

Report requirement:

(1) Well-commented M-file.
(2) All required curves. compare and comment.