Faculty of Engineering

Cairo University

# Digital Communication Course

# Project (2): Matched Filter

## Presented to:

## Dr. Mohamed Nafie

## TA: Eng/ Mohamed Khaled

| ID | B.N. | Sec. | الإسم |
|---|---|---|---|
| 9213073 | 16 | 1 | أحمد عادل يونس سيد |
| 9210688 | 3 | 3 | علي مختار علي الدهشوري |
| 9210824 | 22 | 3 | فيلوباتير عوني عبدالله صليب |
| 9211295 | 34 | 4 | نور الدين تميم محمد محمد |
| 9170663 | - | 3 | عمرو مصطفي كمال محمد |

- ## Table of Contents


**Requirement 1:**......3

**Part A:**......4

**Comment:**......4

**Part B:**......5

**Comment:**......5

**Requirement 2:**......6

**Comment:**......6

**Requirement 3:**......7

**At Point A:**......7

**A.** ......7

**B.** ......7

**C.** ......8

**D.** ......8

**Comment:**......8

**At Point B:**......9

**A.** ......9

**B.** ......9

**C.** ......10

**D.** ......10

**Comment:**......11

**Index:**......12


- ## Table of Figures


**Figure 1 Fig**......3
**Figure 2 Fig**......4
**Figure 3 fig**......5
**Figure 4 Fig**......6
**Figure 5 Fig**......7
**Figure 6 Fig**......7
**Figure 7 Fig**......8
**Figure 8 Fig**......8
**Figure 9 Fig**......9
**Figure 10 Fig**......9
**Figure 11 Fig**......10
**Figure 12 Fig**......10

## ➢ Requirement 1:

- In this requirement we will build the Matched filter and the Correlator in Free Noise environment (before adding the Noise).

- First, we create the pulse signal p(t) with symbol duration Ts = 1 sec, then we sample it 5 times ($T_{sampling} = 200\ ms$).

- In addition to that we normalize the pulse signal to obtain unity energy ($E_p = 1$).

- We generate 10 random bits and from them we create the impulse train and to apply the convolution between the pulse signal p(t) and the impulse train we want to make each symbol in the impulse train take 5 samples not only one. Therefore, we apply up sampling to insert 4 zeros after each symbol in the impulse train.

- after applying the convolution between the pulse signal and the impulse train → The resultant signal will be the output signal from the transmitter ($Tx_{out}$) and this output signal will pass through the channel and reach to the Receiver (Assuming free noise).

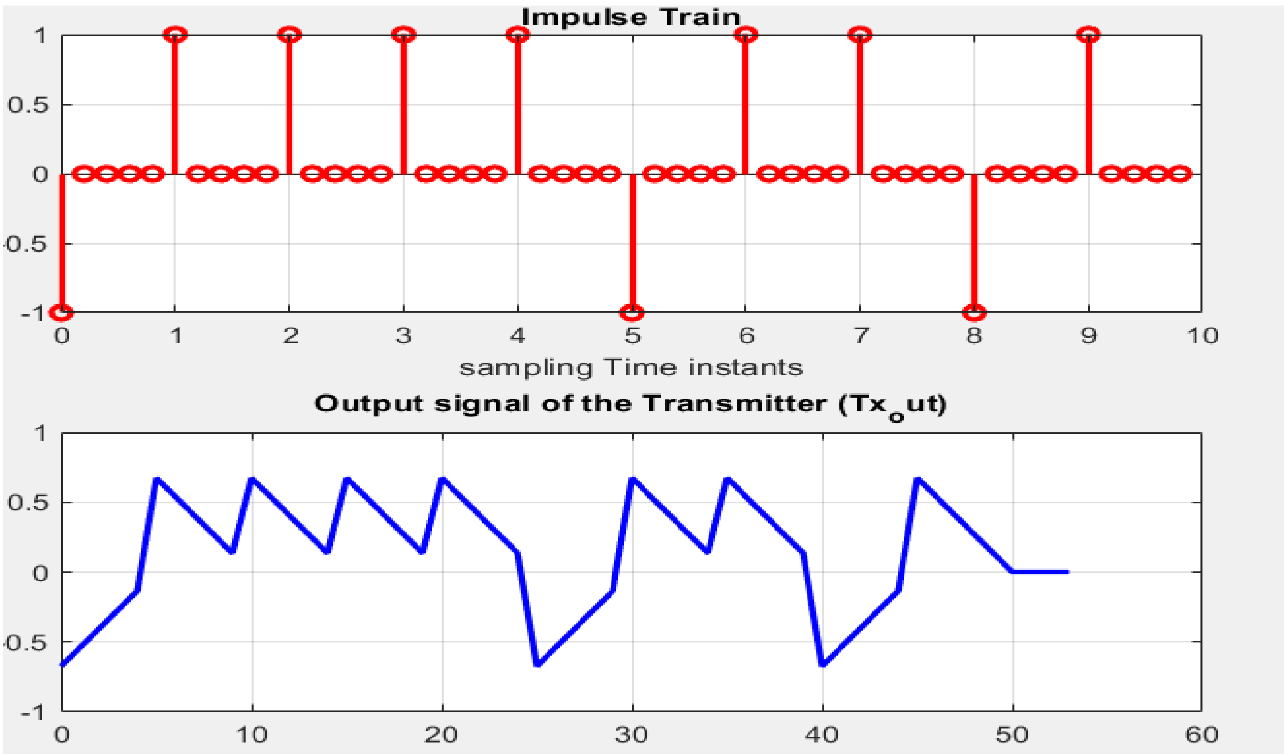


**Figure 1 Fig**

## ➢ Part A:

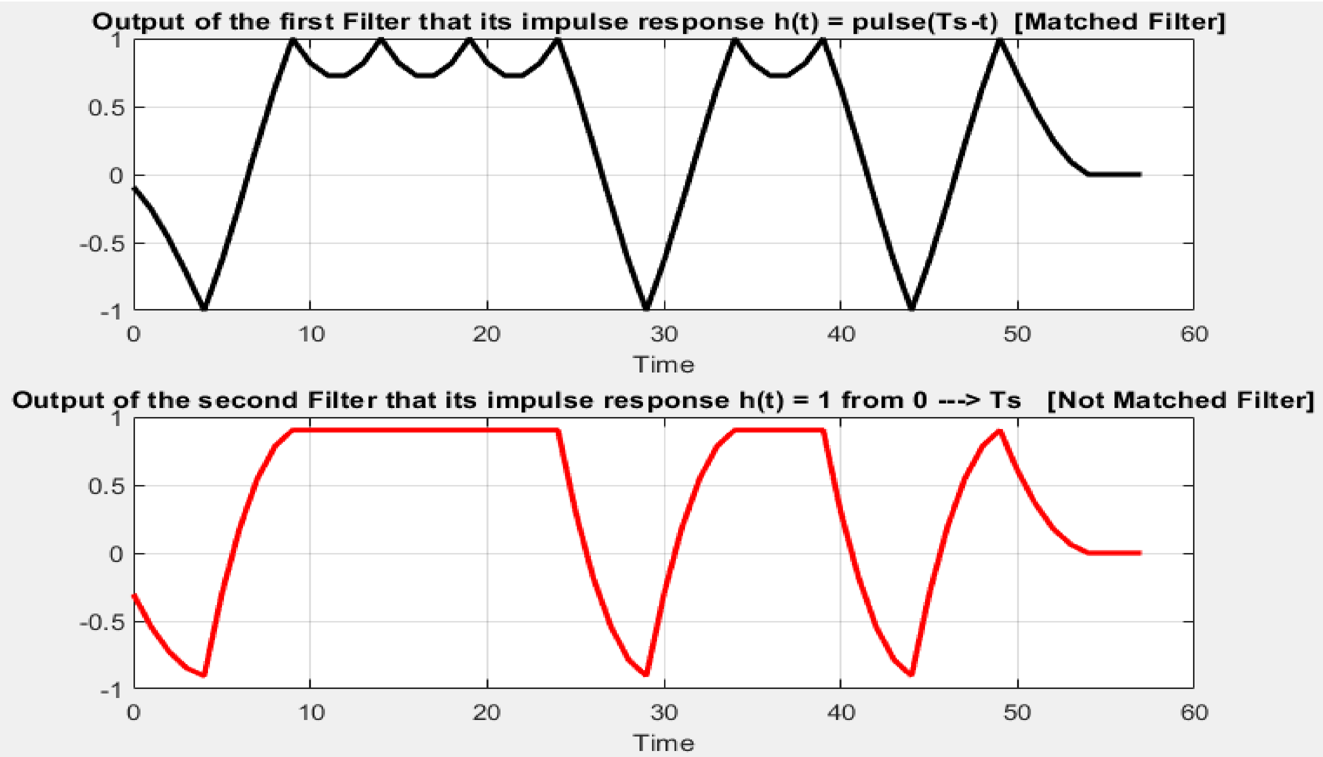


**Figure 2 Fig**

## ➢ Comments:

- In the above figure (figure 1), we compare the output that result from the Matched filter with other filter (Not Matched Filter) that it's impulse response h(t) = 1 from (0 → Ts) but we must normalize the impulse response first to make fair comparison between both filters.

- To normalize any discrete signal, we bring its energy first from this equation
  → $E_s = \sum_{n=-\infty}^{n=\infty}(x[n])^2$ , After that we bring $X[n]_{normalized} = \frac{X[n]}{\sqrt{E_s}}$ .

- In this simulation the Random bits generated are **[0 1 1 1 1 0 1 1 0 1]** as shown in figure (1). Therefore, the Matched filter output curve verifies that, as there are **peaks at 1** that represent **ones** and **minimum values at -1** that represent **zeros**.

## Part B:

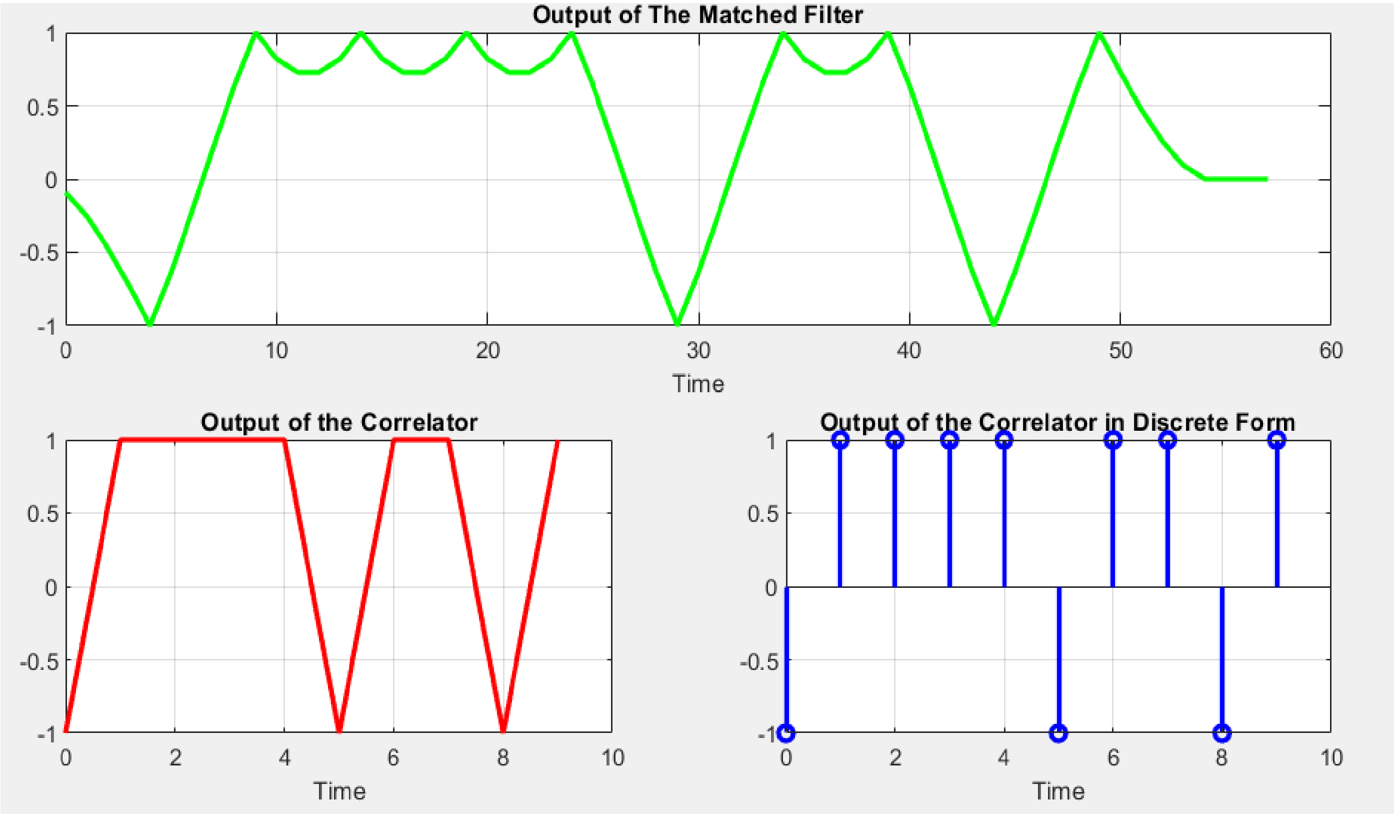


**Figure 3 Fig**

## Comments:

- In the above figure (figure 3), we compare the output that result from the Matched filter with the output of the correlator where the concept of the correlator is that we take every 5 samples of the output signal produced from the transmitter ($Tx_{out}$) and multiply them by the pulse signal p(t) which have the same length =5 then we sum the product and store it in the $\mathrm{correlator}_{\mathrm{out}}$ vector which has length equal to the number of bits (symbols) that are generated.

- As shown in figure (3) we draw the output of the correlator two times one continuous and the other is discrete where in discrete domain it is obvious that the output of the correlator represents the values of the random bits.
  (1 represent **one** & -1 represent **zero**)

## ➢ Requirement 2:

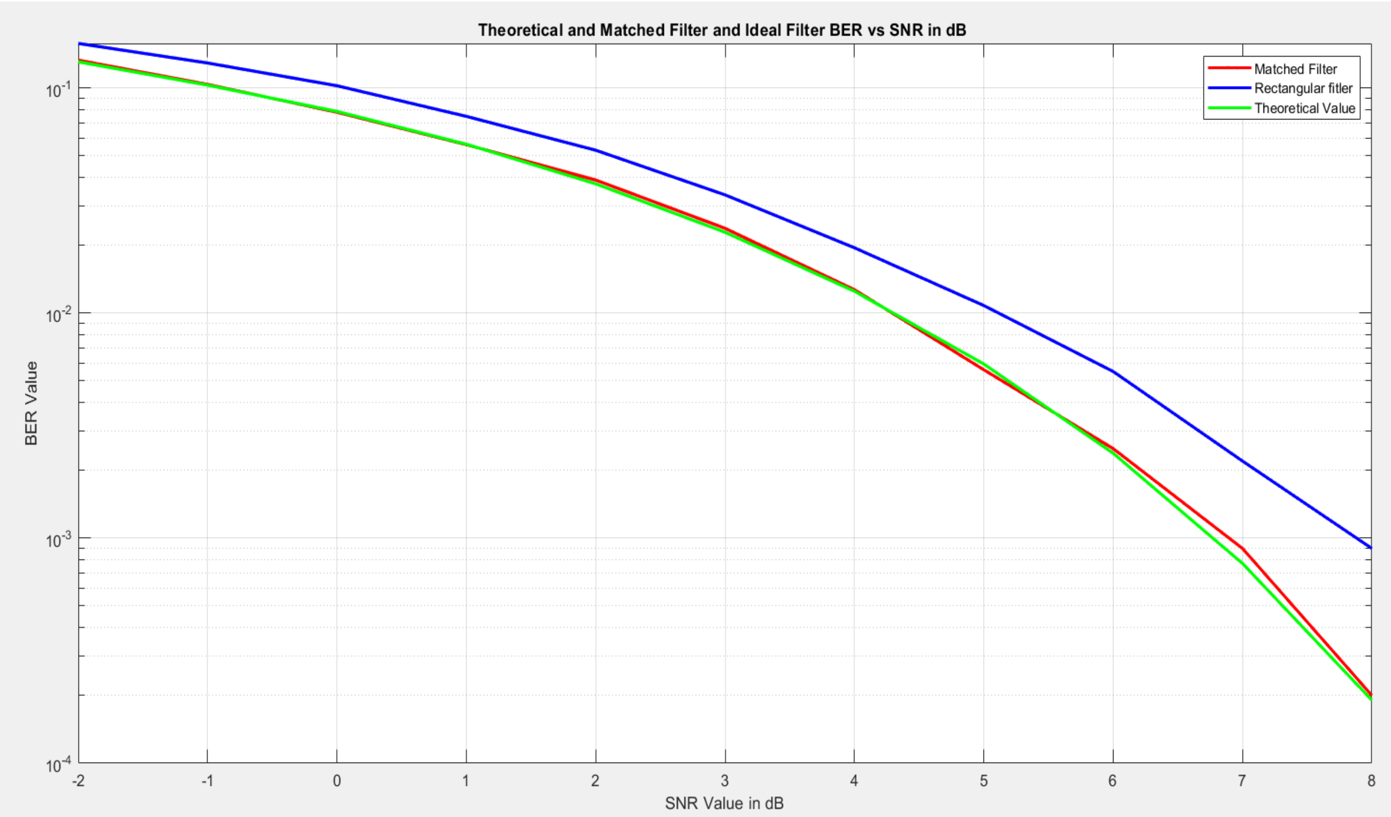


**Figure 4 fig**

## ➢ Comments:

- **For this requirement White gaussian noise is added to the system with different variances to obtain the required SNR.**
- **The Noisy output is sampled every five Samples (Ts), the sampled output is compared with the threshold, if it's below the threshold (zero) then the received bit is decoded as zero if its higher it's decoded as one.**
- **The decoded bits are then compared with the original bits and the bit error rate for each output is calculated.**
- **For the same energy of sent pulse the matched filter provides a lower bit error rate and better performance than using a rectangular filter as matched filter provides highest SNR value.**
- **Increasing Number of bits increases the simulation resolution as the generated gaussian noise mean approaches zero more and the variance approaches unity more, so the simulated results become even closer to the theoretical values.**

# ➢ Requirement 3

## - At Point A:

### A. Roll-off = 0 & Delay = 2

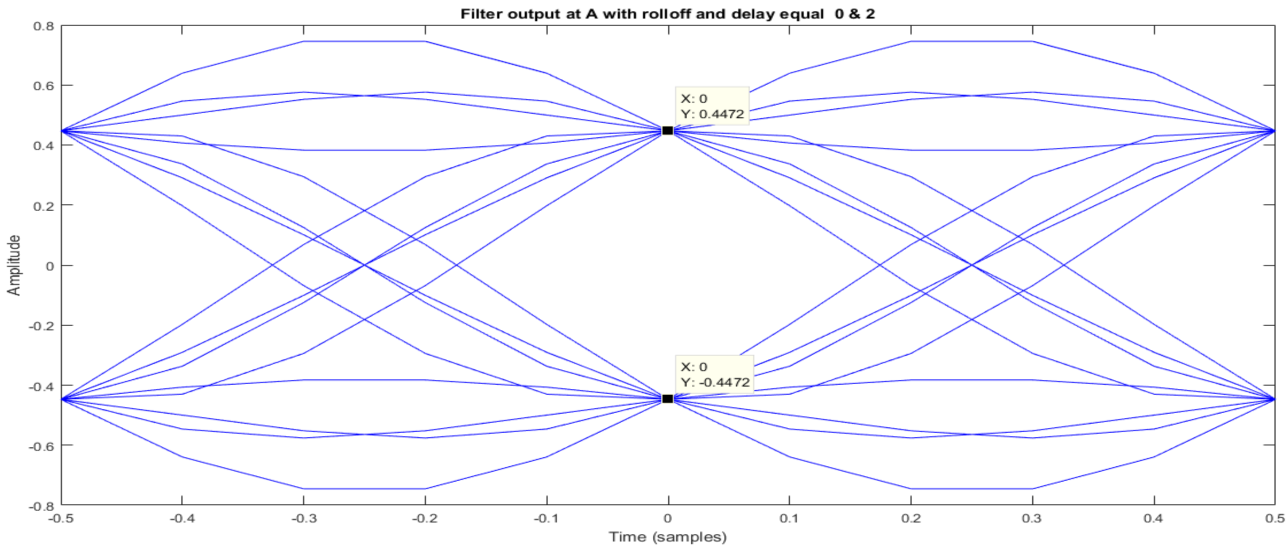


**Figure 5: Eye diagram at point A with (r=0, d=2)**

### B. Roll-off = 0 & Delay = 8

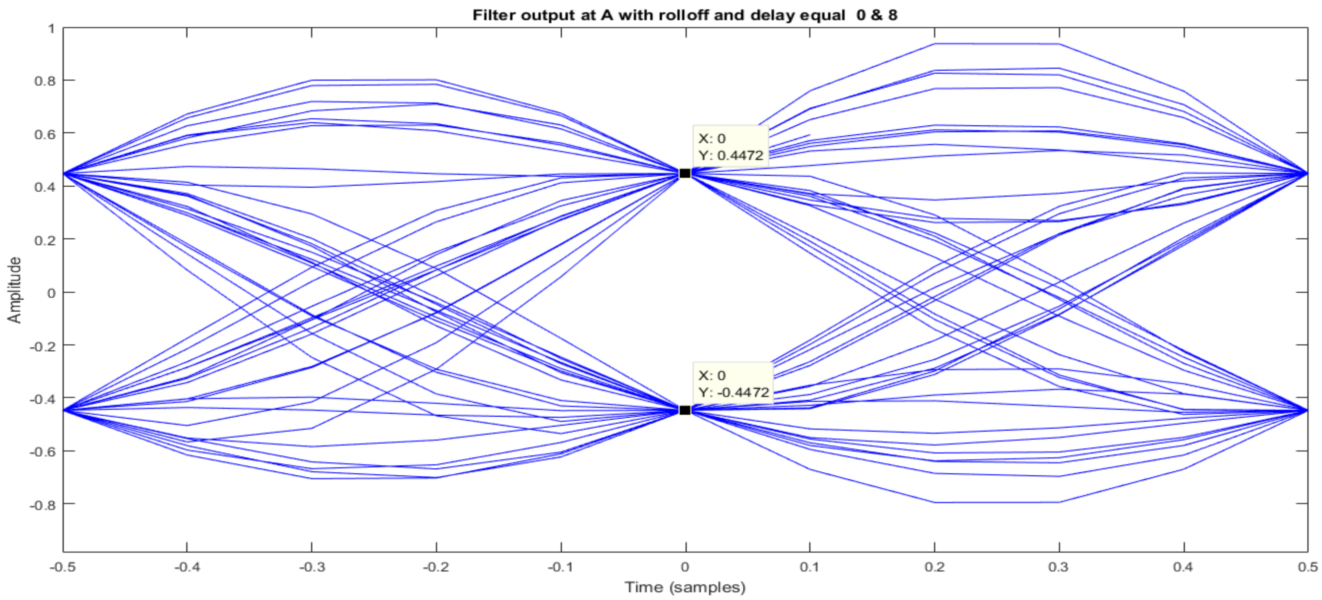


**Figure 6: Eye diagram at point A with (r=0, d=8)**

## C. Roll-off = 1 & Delay = 2

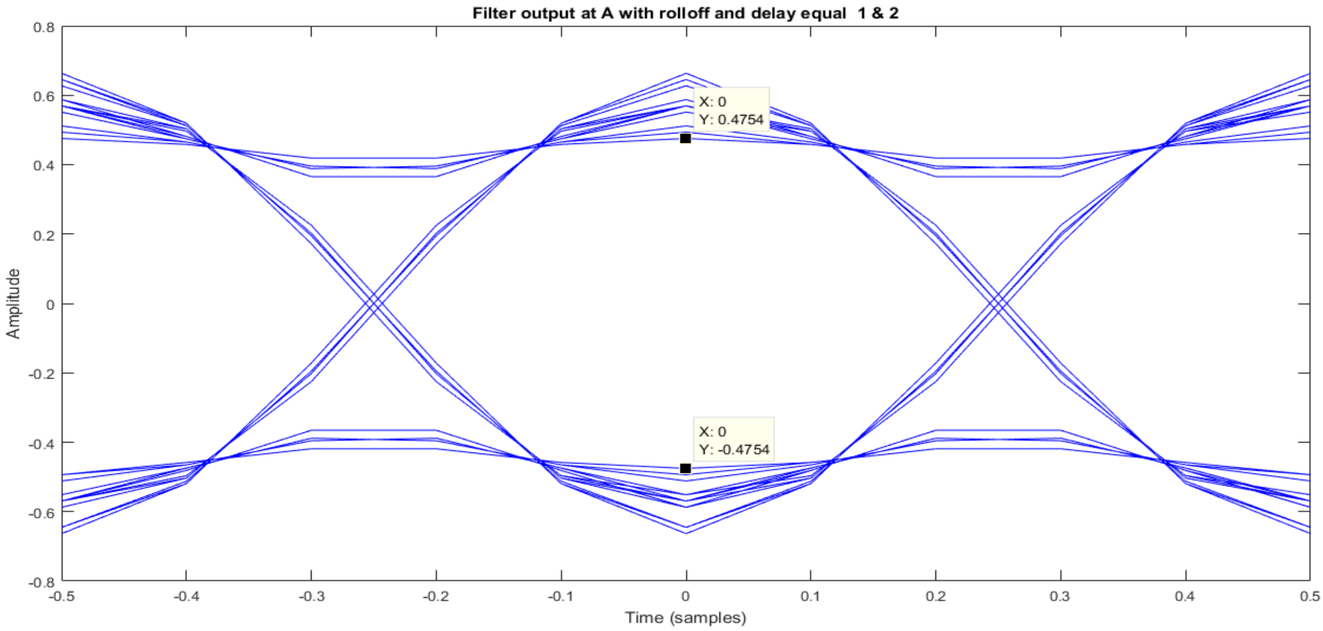


**Figure 7: Eye diagram at point A with (r=1, d=2)**

## D. Roll-off = 1 & Delay = 8

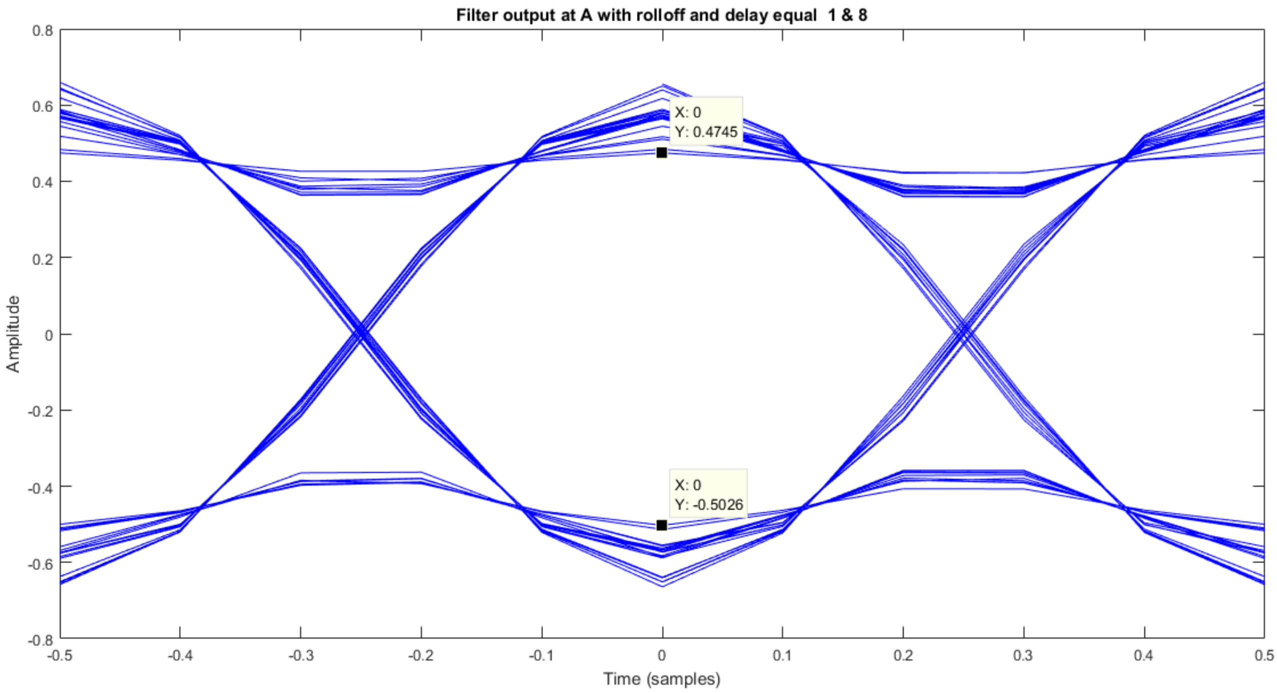


**Figure 8: Eye diagram at point A with (r=1, d=8)**

## - <u>At Point B:</u>

### A. Roll-off = 0 & Delay = 2

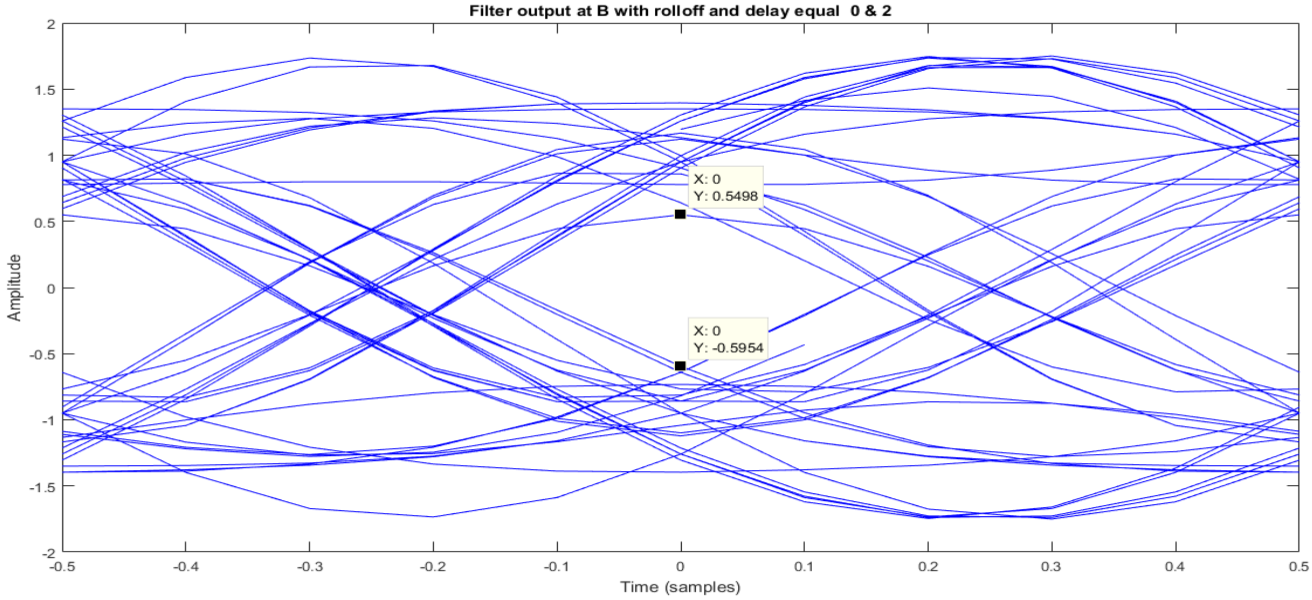


**Figure 9: Eye diagram at point B with (r=0, d=2)**

### B. Roll-off = 0 & Delay = 8

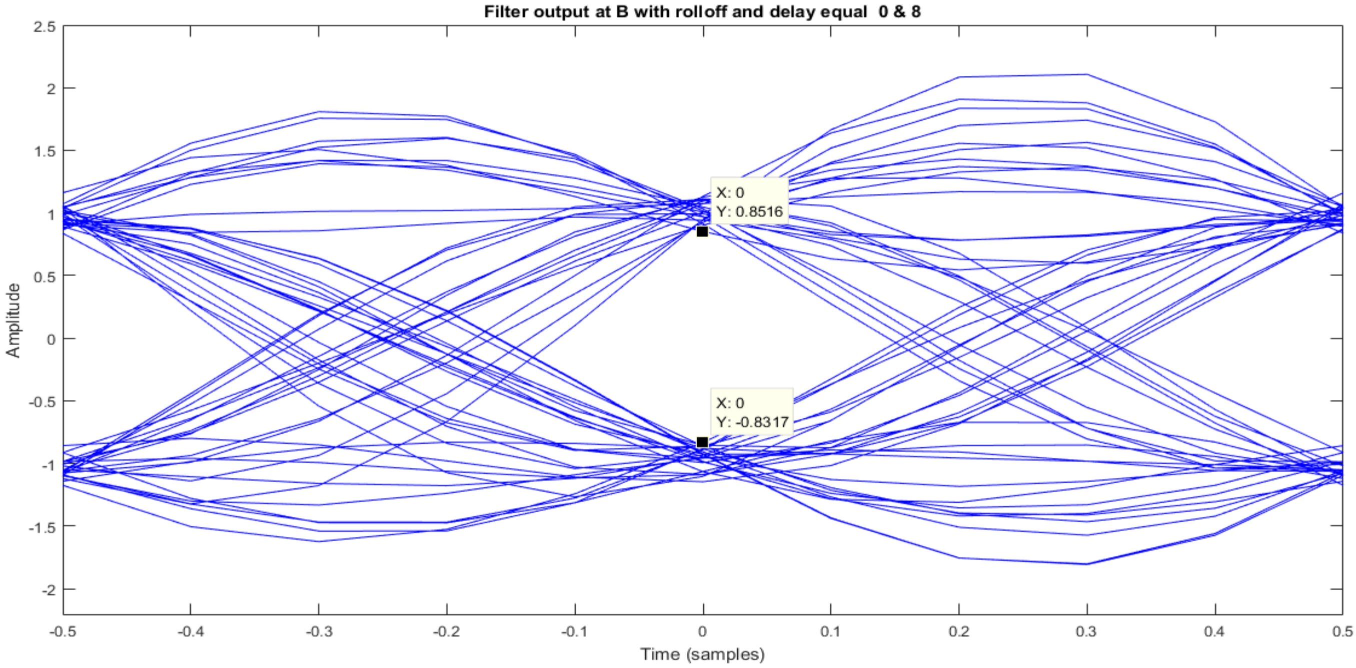


**Figure 10: Eye diagram at point B with (r=0, d=8)**

## C. Roll-off = 1 & Delay = 2

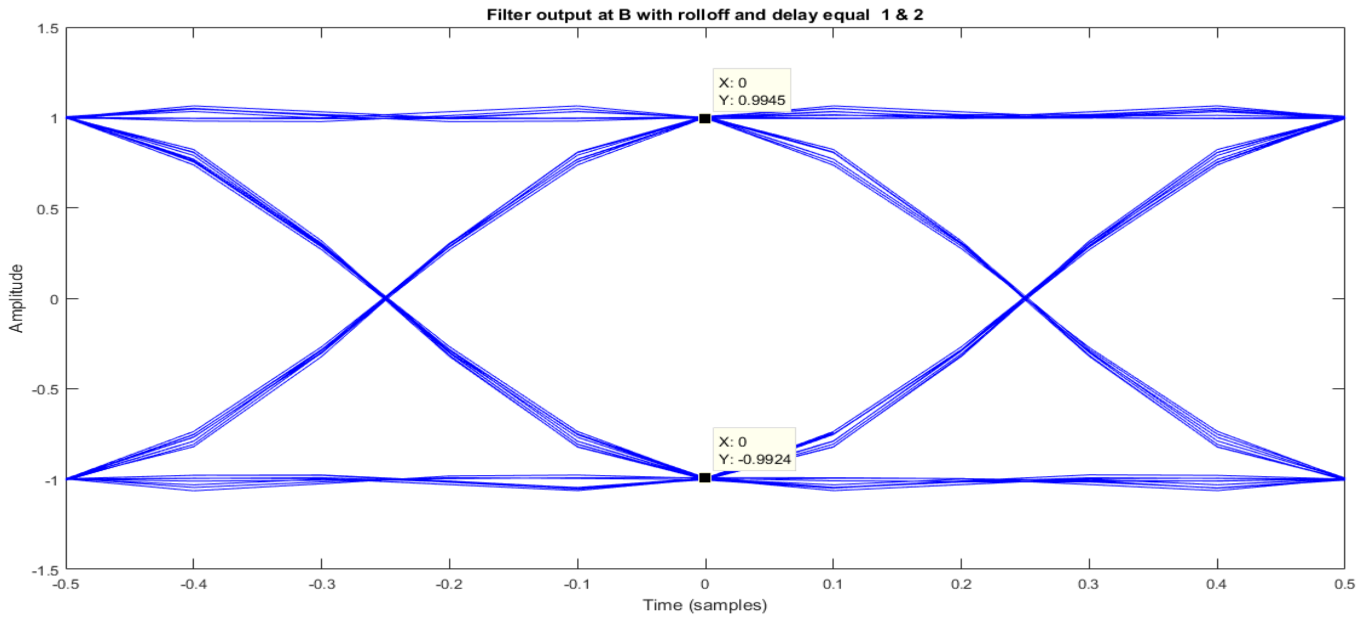


**Figure 11: Eye diagram at point B with (r=1, d=2)**

## D. Roll-off = 1 & Delay = 8

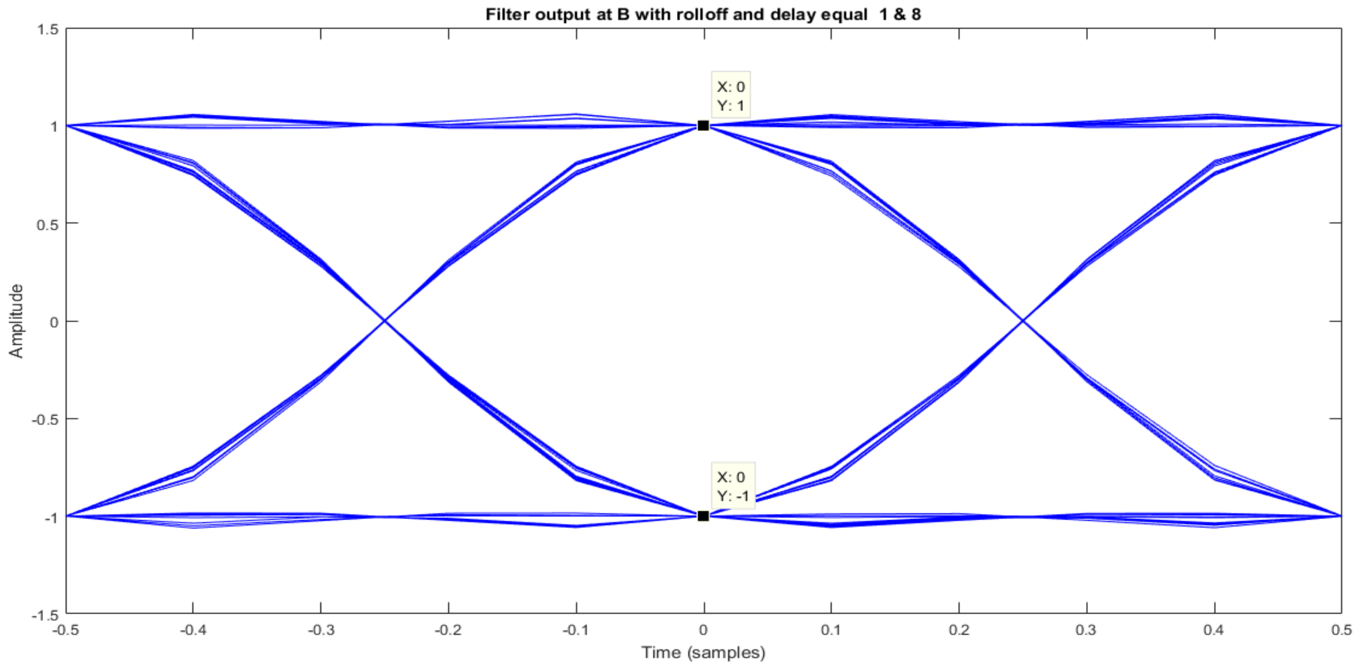


**Figure 12: Eye diagram at point B with (r=1, d=8)**

## ➢ Comments:

- **This part consists mainly of three parts after up sampling the random data ranging between [-1,1] we generated the SRRC filter coefficients using Rcosine filter giving it the following parameters:**
    1) **Sampling frequency (Fs)**
    2) **Fd = Fs *Sps where represents the no. of samples taken per symbol preferably taken a large number to increase the filter resolution.**
    3) **Shape of desired raised cosine filter: 'sqrt'**
    4) **Roll-off factor**
    5) **delay: represents the number of zeros seen in a single side of the filter describing the length of the finite window taken from the IIR of the filter.**
- **We generated the SRRC filter coefficients then we convoluted it with the up-sampled data to generate the transmitted data from A and plot its eye diagram then we repeated the convolution between the Tx signal and the same SRRC coefficients to get the received signal at B and plot the eye diagram.**
- **In the code: when plotting the eye diagram at either point A or B we didn't consider the entire length of the convolution output. Instead, we specifically extracted the primary intended part of the signal, which corresponds to the main lobe. Therefore, we began with the very first main lobe resulting from the convolution with the first bit and ended with the last one, extracting some side lobes from the start and the end of the convolution output.**
- **About the eye opening and the sampling time relation as the eye remains quite open for a larger period around the optimum sampling time (center of the eye opening). The system becomes more tolerant of sampling timing errors making the system more immune to synchronization errors.**
- **Observations:**
    1) **When the roll-off factor increases this means the bandwidth used increases so the signal shrinks in time domain decreasing the ISI and leading to larger eye opening (better sampling accuracy).**
    2) **Also, when the delay increases, the filter response expands in time making it more similar to the ideal response so, the eye opening slightly widens leading to a larger noise margin.**

3) **The effects of the roll-off factor and the delay parameters are much noticeable at the receiver (point B) than at the transmitter (point A) as at Tx data is convolved with the SRRC filter once before transmission while at Rx the Tx signal is received by a matching SRRC filter to maximize SNR resulting to a nearly raised cosine overall effect this could contribute to the difference of the observed results.**

## ➢ Index:

```
clc
clear
close all;

%% -----------------------Matched filter & correlator part------------------------%%

Ts = 1;                        % it refer to the symbol duration Ts = 1 sec
% generation of pulse p[n] but divided over sqrt(55) to normalize it's energy
pulse = [5 4 3 2 1]/sqrt(55);
Number_of_bits = 10;           % this variable represent the number of the random bits that are generated
Data = randi([0 1] , 1 , Number_of_bits);    % array contain the random data bits (0 & 1)

Amplitude = 1;         % it refer to the amplitude of the impulse train
PAM_signal = (2 * Data - 1) * Amplitude;     % This is the PAM sampling which represent (+Amplitude or -Amplitude) every Ts = 1 sec
impulse_train = upsample(PAM_signal,5);      % we upsample the input signal (PAM_signal) by inserting 4 zeros (5-1) in each symbol duration Ts

Tx_out = conv(impulse_train,pulse);     % where Tx_out refer to the signal at the output of the transmitter
% where the length(Tx_out) = length(impulse_train) + length(pulse) - 1     [According to the discrete convolution]

% For plotting, we will use the stem() function instead of the plot() function as stem() is specified for the discrete plotting
figure;
subplot(2,1,1);
title('Output signal of the Transmitter (Tx_out)');
xlabel('Time');
grid on;

%------------ Methods of filtering the Tx_out signal (2 Methods)------%

% First Method is by take Tx_out & apply convolution with impulse response of the Matched filter h[n] = Pulse[Ts - n]     (Matched Filter)

impulse_response_MF1 = fliplr(pulse);            % Matched filter impulse response
Rx_out1 = conv(Tx_out, impulse_response_MF1);      % Rx_out is the output signal of the matched filter

% second Method is by take Tx_out & apply convolution with impulse response of another filter h(t) = 1 from 0 --> Ts    (NOT Matched Filter)

impulse_response_MF2 = [1 1 1 1 1]/sqrt(5);       % First step we must normalize the impulse response to obtain unity energy
Rx_out2 = conv(Tx_out, impulse_response_MF2);

% we make subplotting to compare between the two outputs of two different filters (Rx_out1,Rx_out2)

figure;
subplot(2,1,1);
plot(0:length(Rx_out1)-1 , Rx_out1 , 'k', 'LineWidth', 2);
title('Output of the first Filter that its impulse response h[n] = pulse[Ts-n]');
xlabel('Time');
grid on;

subplot(2,1,2);
plot(0:length(Rx_out2)-1, Rx_out2, 'r', 'LineWidth', 2);
title('Output of the second Filter that its impulse response h[n] = 1 from 0 ---> Ts');
xlabel('Time');
```

```
grid on;

figure;
subplot(2,1,1);
stem(0:length(Rx_out1)-1 , Rx_out1 , 'k', 'LineWidth', 2);
title('Output of The First Matched Filter in Discrete Form');
xlabel('Time');
grid on;

subplot(2,1,2);
stem(0:length(Rx_out2)-1, Rx_out2, 'r', 'LineWidth', 2);
title('Output of the second Filter in Discrete Form');
xlabel('Time');
grid on;

% Build the correlator & compare it's output with the output of the Matched Filter

correlator_out = zeros(1,length(Number_of_bits));  %the output signal of the correlator is the same length as the number of random bits generated

j = 1;            % j variable is used to increment the index of correlator_out
for i = 1 : 5 : length(Tx_out)-5
        % Calculate the correlation for each segment (5 samples) of Tx_out and update correlator_out
        correlator_out(1,j) = sum( pulse .* Tx_out(1,i:i+4) );
        j = j + 1;
end

% plot the output of the correlator and compare it with the Matched Filter
figure;
subplot(2,2,[1 2]);
plot(0:length(Rx_out1)-1 , Rx_out1 , 'g', 'LineWidth', 2);
title('Output of The Matched Filter');
xlabel('Time');
grid on;

subplot(2,2,3);
plot(0:length(correlator_out)-1, correlator_out , 'r', 'LineWidth', 2);
title('Output of the Correlator');
xlabel('Time');
grid on;

subplot(2,2,4);
stem(0:length(correlator_out)-1, correlator_out , 'b', 'LineWidth', 2);
title('Output of the Correlator in Discrete Form');
xlabel('Time');
grid on;


%% --------------------------------Noise Analysis part------------------------- %%
%  a):
Number_of_bits = 10000;          % this variable represent the number of the random bits that are generated
pulse = [5 4 3 2 1]/sqrt(55);
Data = randi([0 1] , 1 , Number_of_bits);    % array contain the random data bits (0 & 1)
Amplitude = 1;         % it refer to the amplitude of the impulse train
PAM_signal = (2 * Data - 1) * Amplitude;     % This is the PAM sampling which represent (+Amplitude or -Amplitude) every Ts = 1 sec
impulse_train = upsample(PAM_signal,5);      % we upsample the input signal (PAM_signal) by inserting 4 zeros (5-1) in each symbol duration Ts
Tx_out = conv(impulse_train,pulse);     % where Tx_out refer to the signal at the output of the transmitter
impulse_response_MF1 = fliplr(pulse);            % Matched filter impulse response
impulse_response_MF2 = [5 5 5 5 5]/sqrt(125);       % First step we must normalize the impulse response to obtain unity energy
```

```
% b):
Gaussian_Noise = randn( 1,length(Tx_out) );  % generate gaussian white noise of zero mean and unity variance of length equal to that of transmitted signal

% c):
Noise_Variance = [1.585 1.25 1 0.794 0.631 0.501 0.3981 0.316 0.2511 0.1995 0.15845]  ;  % Variable to control gaussian noise variance value

% d):
BER_Simulated = zeros(2,length(Noise_Variance));
BER_Theoretical = zeros(1,length(Noise_Variance));
for j= 1 :length(Noise_Variance)
    Noisy_Tx_out = Tx_out + ( Gaussian_Noise * sqrt( Noise_Variance(j)/2 )) ;
    Rx_out1 = conv(Tx_out, impulse_response_MF1);      % Noisy_Rx_out1 is the output signal of the matched filter with noise added
    Noisy_Rx_out1 = conv(Noisy_Tx_out, impulse_response_MF1);      % Noisy_Rx_out1 is the output signal of the matched filter with noise added
    Noisy_Rx_out2 = conv(Noisy_Tx_out, impulse_response_MF2);
    Sampled_Output1 = zeros(1,Number_of_bits); %Samples Noisy_Rx_out1
    Sampled_Output2 = zeros(1,Number_of_bits); %Samples Noisy_Rx_out2

    Sampled_Output_Index = 1;
    for i= 5:5: (Number_of_bits*5)
        Sampled_Output1(Sampled_Output_Index ) = Noisy_Rx_out1(i);
        Sampled_Output2(Sampled_Output_Index ) = Noisy_Rx_out2(i);
        Sampled_Output_Index = Sampled_Output_Index + 1;
    end
    Number_Of_Errors = zeros(1,2);
    %Calculation Of BER
    for i = 1:Number_of_bits
        if(PAM_signal(i) == 1 && Sampled_Output1(i) <=0 )
            Number_Of_Errors(1) = Number_Of_Errors(1) + 1;
        elseif(PAM_signal(i) == -1 && Sampled_Output1(i) >0 )
            Number_Of_Errors(1) = Number_Of_Errors(1) + 1;
        end
        if(PAM_signal(i) == 1 && Sampled_Output2(i) <=0 )
            Number_Of_Errors(2) = Number_Of_Errors(2) + 1;
        elseif(PAM_signal(i) == -1 && Sampled_Output2(i) >0 )
            Number_Of_Errors(2) = Number_Of_Errors(2) + 1;
        end
    end
    BER_Simulated(1,j) = Number_Of_Errors(1) / Number_of_bits  ;
    BER_Simulated(2,j) = Number_Of_Errors(2) / Number_of_bits  ;
    BER_Theoretical(1,j) = 0.5 * erfc(sqrt(1/Noise_Variance(j) ) );
end

%plotting BER vs SNR
figure;
SNR_DB=-2:length(Noise_Variance)-3;
semilogy(SNR_DB, BER_Simulated(1,:) ,'r', 'linewidth', 2)
hold on;
semilogy(SNR_DB, BER_Simulated(2,:),'b', 'linewidth', 2 )
hold on;
semilogy(SNR_DB, BER_Theoretical , 'g', 'linewidth', 2)
legend('Matched Filter','Ideal fitler','Theoretical Value')
grid on
xlabel("SNR Value in dB")
ylabel("BER Value")
title("Theoretical and Matched Filter and Ideal Filter BER vs SNR in dB ")
```

```
%% %%****************** ISI and Raised cosine filter part*****************
% Parameters
numBits = 100;          % Number of bits
rolloff = [0,0,1,1];          % Roll-off factor
delay=[2,8,2,8];          %Delay introduced to filter coefficient
sps =5;       %Samples taken per symbol preferred large to increase resolution
Fs = 1;     %Sampling freq  where each bit is represented by a symbol

% Generate random bits
dataTx = randi([0,1],1,numBits)*2-1;  % Randomize bits to -1 or 1
dataTx_upsampled = upsample(dataTx,sps);

for i=1:4

    % Transmitter
        %Generating the filter coefficients
            srrc = rcosine(Fs ,sps*Fs, 'sqrt' , rolloff(i), delay(i) );

       % Apply SRRC filter to the upsampled data by convolution
            txSig = conv(dataTx_upsampled, srrc);

        %Calculating the start of the main intended output
        start = length(srrc)+1;

    % Eyediagram at A(transmitter)
        eyediagram(txSig(start-1:end-start-sps-1),2*sps);
        xlabel('Time (samples)');
        title(['Filter output at A with rolloff and delay equal  '
            num2str(rolloff(i)) ' & ' num2str(delay(i))]);

    % Receiver
        %Using the same filter srrc as the transmitter
            RxSig = conv(txSig,srrc);

        % Eyediagram at B(reciever)
            eyediagram(RxSig(start-1:end-start-sps-1),2*sps);
            xlabel('Time (samples)');
            title(['Filter output at B with rolloff and delay equal  '
                num2str(rolloff(i)) ' & ' num2str(delay(i))]);

end
```