UNIVERSITY OF MORATUWA, SRI LANKA

Faculty of Engineering

Department of Electronic and Telecommunication Engineering

Semester 4 (Intake 2020)

EN2074 - Communication Systems Engineering

Lab Assignment – Eye diagrams and Equalization

Pramuditha A. A. H. – 200476P

Wijetunga W. L. N .K. – 200733D

This report is submitted as a partial fulfillment for the module EN2074 - Communication Systems Engineering, Department of Electronic and Telecommunication Engineering, University of Moratuwa.

ABSTRACT

Digital communication is the physical transfer of data over Point-To-Point or Point to Multipoint communication channel. Any communication system can be divided into three major parts named transmitter, medium (channel), and the receiver. When it comes to digital communication systems, after the modulation, each baseband signal should be passed through a “Pulse Shaping Filter” to generate waveforms for the transmission purpose. When designing these pulse shapes, it is mandatory to consider the errors and impairments that can be generated by the transmission medium (channel). So, we need to analyze the robustness of the channel to these erroneous occurrences and prepare the transmitter and receiver circuitries to compensate these errors. For this purpose, we can use the tool called “Eye diagrams”. In this report we have obtained eye diagrams for PAM (Pulse Amplitude Modulation) scheme using two pulse shaping filters and explained the information which is obtained from those eye diagrams.

ASSIGNMENT DESCRIPTION

Lab Assignment – Eye diagrams and Equalization

Please note that each task needs to be completed using MATLAB or Octave. (Please state in the report which software you have used.)

Task 1

In Task I, you are expected to generate eye diagrams for baseband 2-PAM signaling with different pulse shaping filters.

1. Generate an impulse train representing BPSK symbols.
2. Obtain transmit signal by convolving the impulse train with a pulse shaping filter where the impulse response is a sinc function.
3. Generate the eye diagram of the transmit signal.
4. Repeat 1-3 for raised cosine pulse shaping filters with roll-off factor 0.5 and 1.
5. Compare the robustness of the system with respect to noise, sampling time and synchronization errors.

Task 2

In Task 2, you are required to repeat Task 1, in the presence of additive white Gaussian noise (AWGN). To generate noise, use 'randn' function. Set the variance of noise such that $Eb\ N0$ = 10 dB, where Eb is the average bit energy and $N0$ is the noise power spectral density.

Task 3

For Task 3, you will design a zero-forcing (ZF) equalizer for a 3-tap multipath channel. Please follow the following steps for Task 3.

1. Generate a random binary sequence.
2. 2-PAM modulation - bit 0 represented as -1 and bit 1 represented as +1. You can ignore pulse shaping here. Assume you are transmitting impulses.
3. Generate the received signal samples by convolving the symbols with a 3-tap multipath channel with impulse response h = [0.3 0.9 0.4].
4. Add White Gaussian noise such that $Eb\ N0$ = 0 dB.
5. Computing the ZF equalization filters at the receiver for 3, 5, 7, and 9 taps in length.
6. Demodulation and conversion to bits
7. Calculate the bit error rate (BER) by counting the number of bit errors.
8. Repeat steps 1-7 for $Eb\ N0$ values 0 -10 dB.
9. Plot the BER for all tap settings and $Eb\ N0$ values in the same figure.

10. Plot the BER for an additive white Gaussian noise (AWGN) channel in the same figure.
11. Why there is a discrepancy between the AWGN channel BER and the ZF equalized multipath channel. Explain your results referring to the design of the ZF equalizer.
12. Comment on the BER performance if binary orthogonal signaling was used instead of BPSK.

TASK 01 – Discussion

In this assignment we are analyzing the robustness of a digital communication system with respect to noise, sampling time, and synchronization errors. For this purpose, we use the engineering tool called “Eye Diagrams”. Eye diagram is a common indicator of the quality of signals in high-speed digital transmissions. Oscilloscopes can generate these diagrams by overlapping sweeps of different segments of a long data stream driven by a master clock.

First, we created an equi-probable bit array of 0s and 1s which is 2000 bits long. The reason to take an array with a large bit count is that we can ensure there are enough samples to create a more accurate eye diagram.

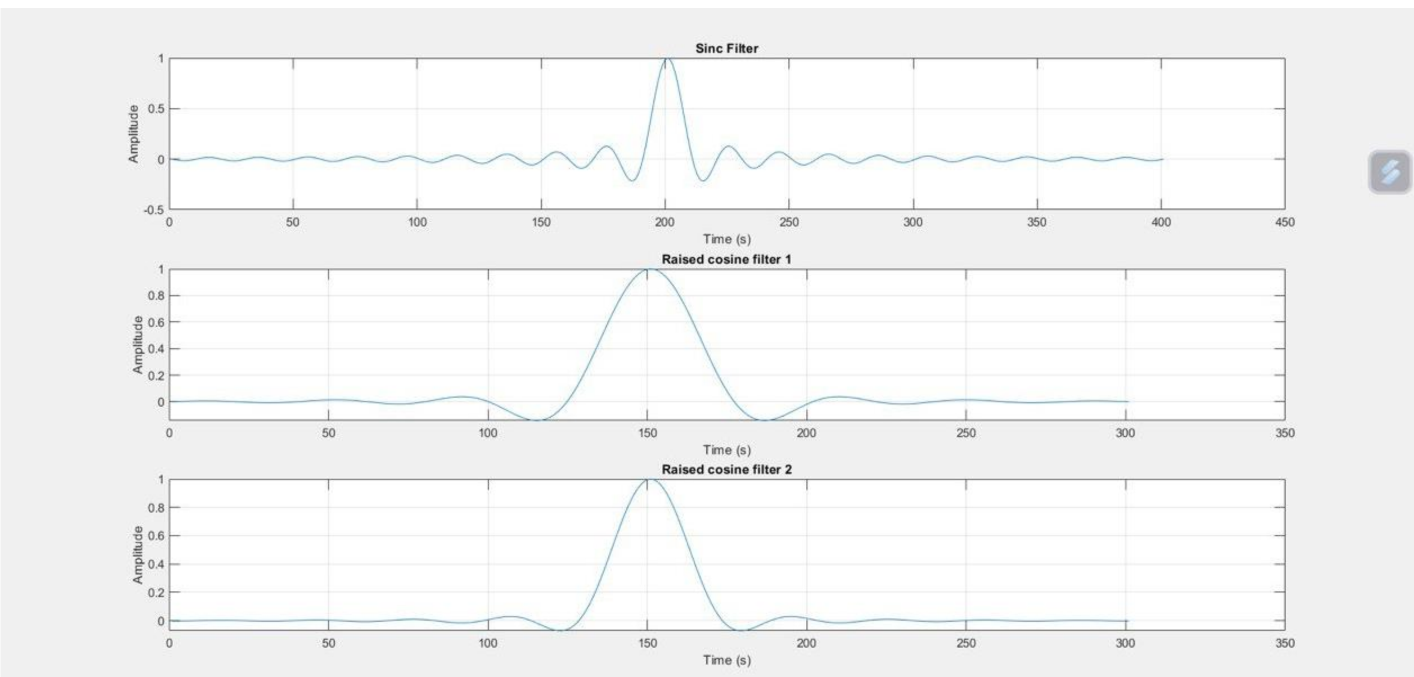
In task 01, we mapped these bits to a set of two symbols using BPSK modulation technique.

All 0s → -1 (180^0 phase)

All 1s → +1 (0^0 phase)

For the pulse shaping task we used three different pulses in our pulse shaping filter.

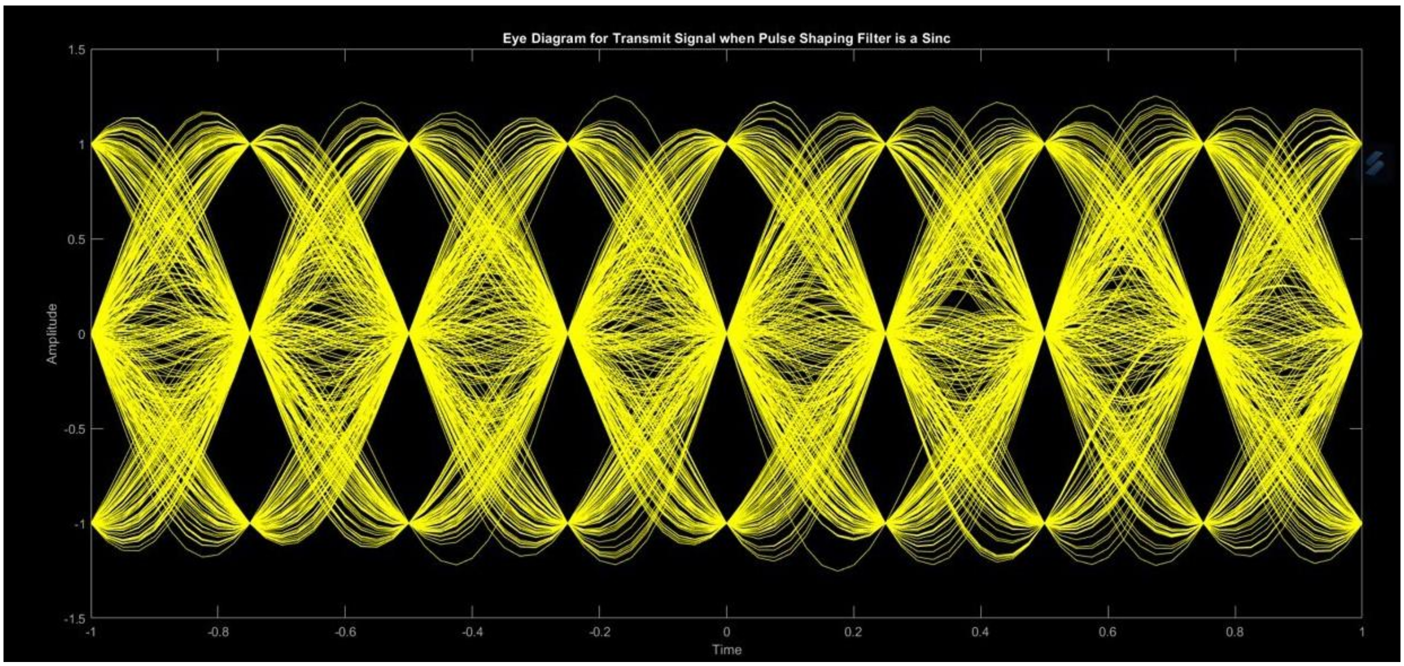
I. Sinc Pulse
II. Raised Cosine Pulse with roll-off = 0.5
III. Raised Cosine Pulse with roll-off = 1

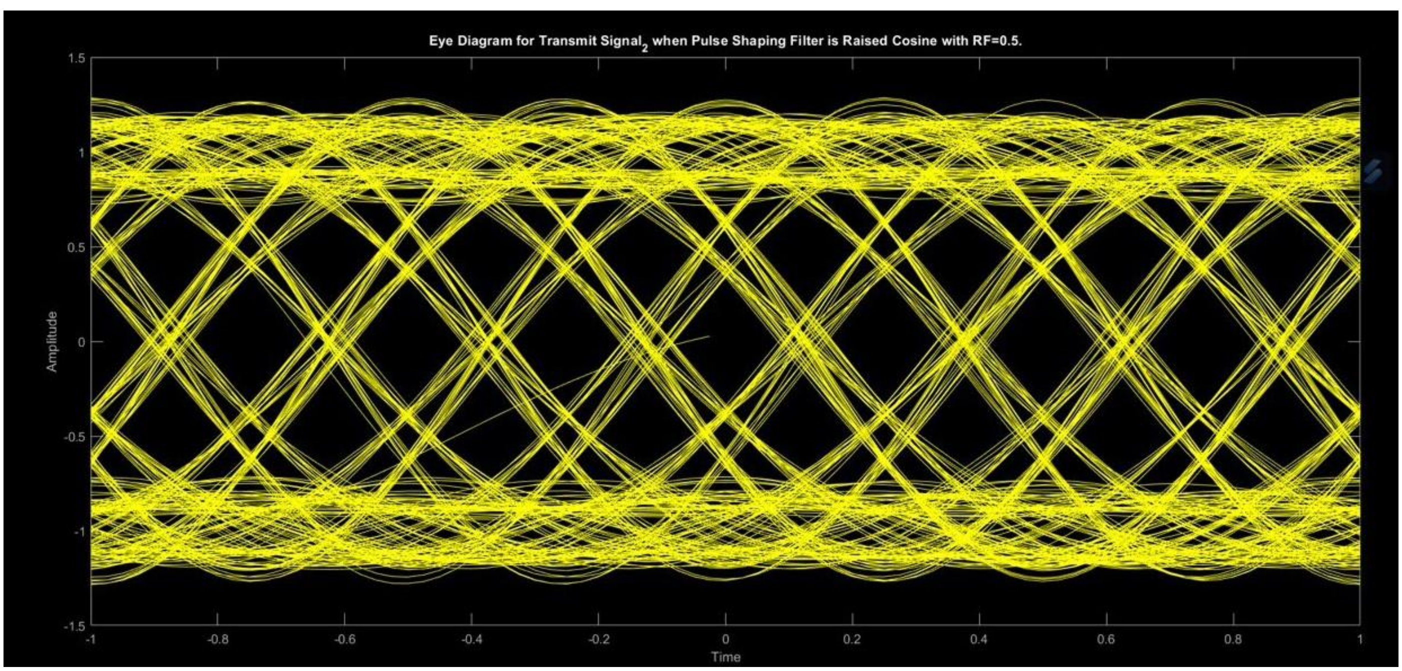


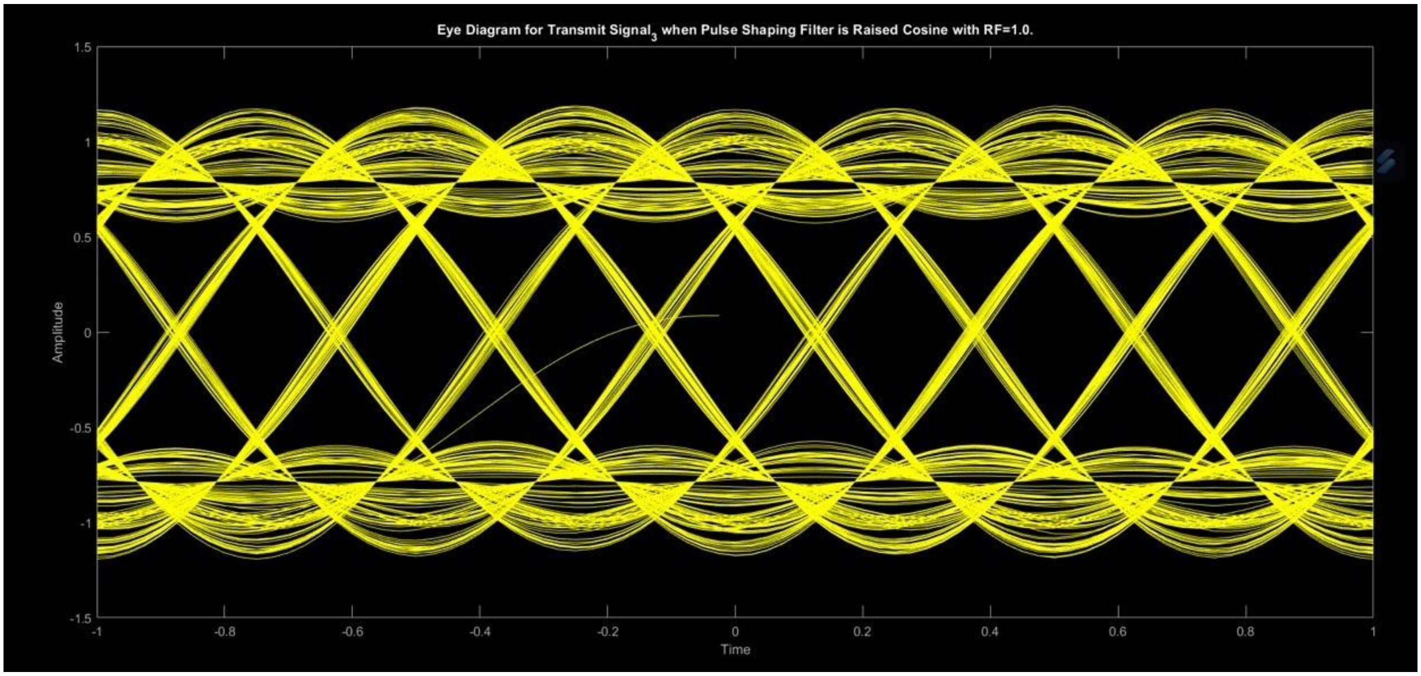
According to the above graphs representing the pulse shapes that we used for the analysis, It can be clearly seen that considerable fluctuations of amplitude can be seen in the sinc pulse over the whole time domain while the raised cosing pulses with roll-off factors 0.5 and 1 die down down after a particular time period. When it comes to the side lobes, raised cosine pulses have smaller side lobes compared to the sinc pulse. (Higher the roll-off, smaller the side lobes)

So, according to these basic features we can conclude that raised cosine pulses are more robust to the errors and impairements than the sinc pulse in digital communication systems. This can be further verified by obtaining eye diagrams for different pulse shaped signals.

In task 1, we check and verify the robustness of the system with respect to sampling time (ISI – Inter Symbol Interference) and synchronization errors.





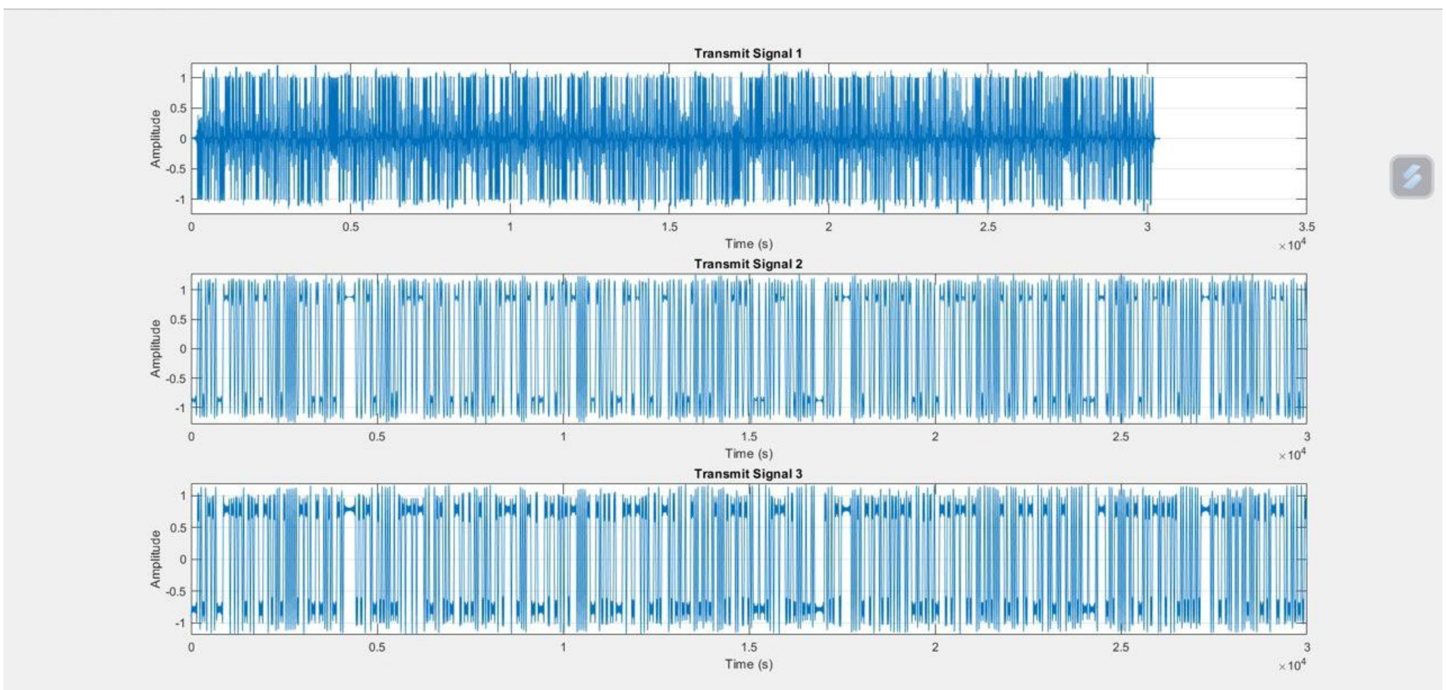


When conveying the information and conclusions about the differences between these generated waveforms and the effect they have on the transmission can be decided by observing some specific characteristics of an eye diagram. They can be listed as follows:

Characteristic of the Eye diagram	Information revealed by the characteristic	Sinc Pulse	Raised Cosine Pulse with Roll-off factor = 0.5	Raised Cosine Pulse with Roll-off factor = 1
1. Slope of the eye	Level of synchronization errors	Has the height slope among three diagrams. High probability for timing errors.	Has the lowest slope compared with the other two diagrams. So more robust to timing errors.	Has an average slope which is less than the sinc pulse and greater than the raised cosine pulse with 0.5 roll-off. Robust to timing errors than sinc pulse.
2. Height of the eye	Immunity to noise (half of height of the eye at optimum sampling point)	At this stage no noise is added. So, it has a better noise margin.	At this stage no noise is added. So, it has a better noise margin.	At this stage no noise is added. So, it has a better noise margin.
3. Width of the eye	Error free sampling region	Has the lowest eye width. Has a less range to error free sampling and less	Wider than the sinc pulse but narrower than the raised cosine pulse with 1.0 roll-off. Has a	Has the largest width of three occasions. So, the region for error free sampling is larger

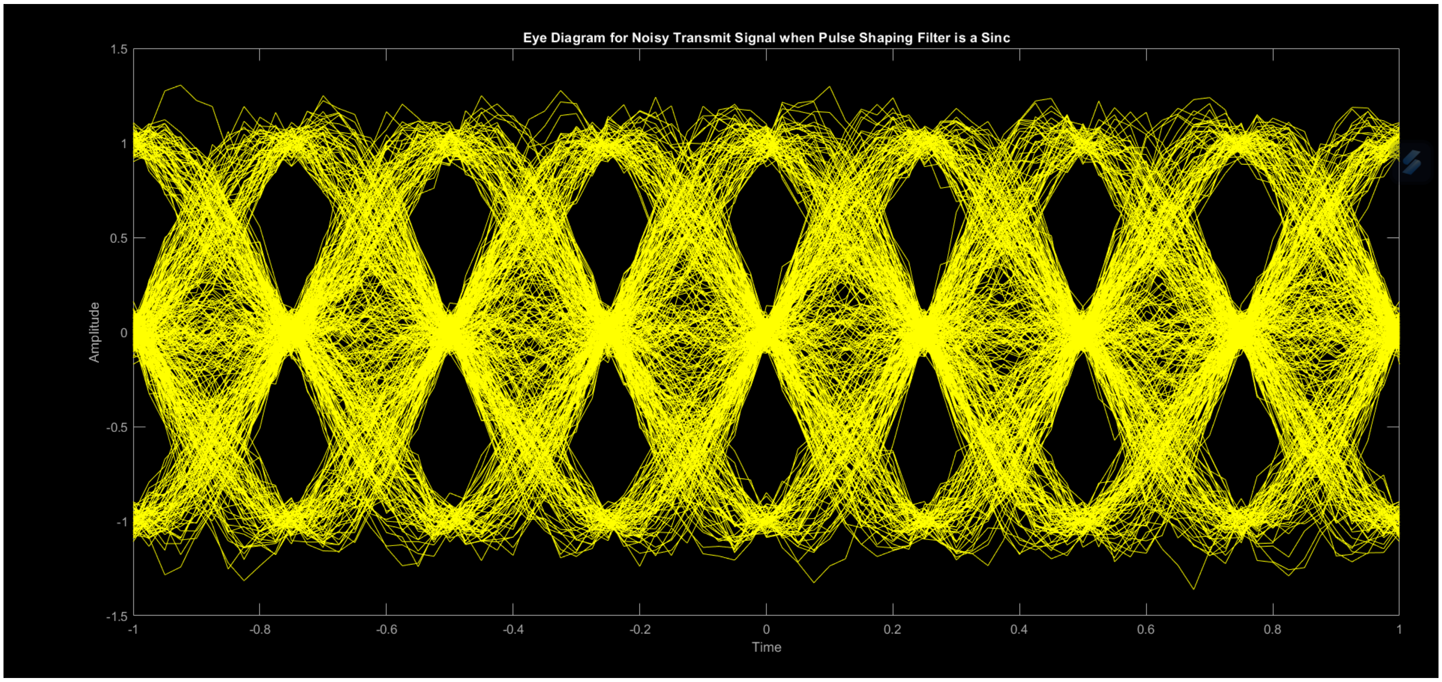
		robust to the sampling errors.	medium range for error free sampling.	than the other two occasions.
4. Thickness at the peak	Peak deviation	No peak deviation since we didn't apply AWGN at this stage.	No peak deviation since we didn't apply AWGN at this stage.	No peak deviation since we didn't apply AWGN at this stage.
5. Time variation at zero-crossing	Timing off-set (Time jitter)	Time variation is highest at zero crossing points. The difference between rise and fall time is quite high.	Has some jitter but is lower than the sinc pulse. But not good as raised cosine pulse with 1.0 roll-off.	Has the smallest jitter between the three graphs. The rise and fall curves of the pulse almost converges at zero crossing points. High robustness for timing errors and ISI errors.

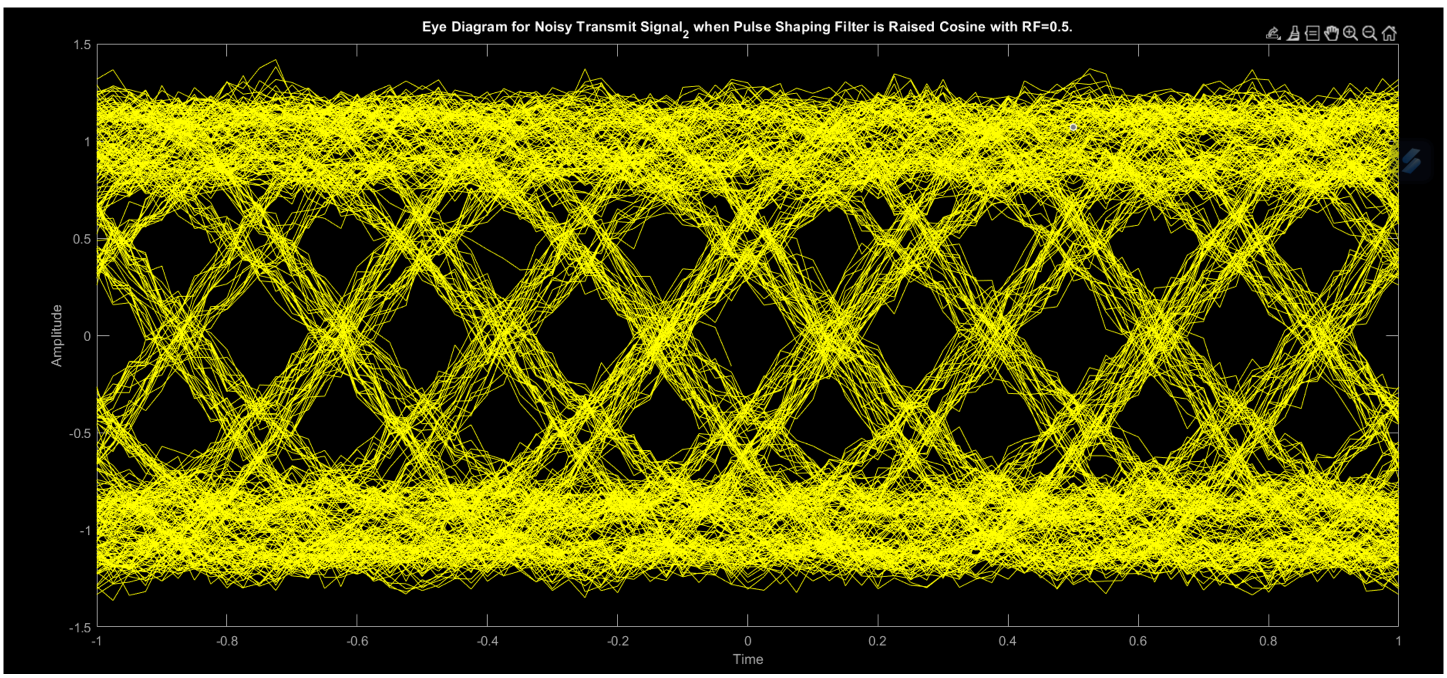
Under the above pulse shaping filters with specifications, the transmitted signals were obtained and can be viewed as follows:

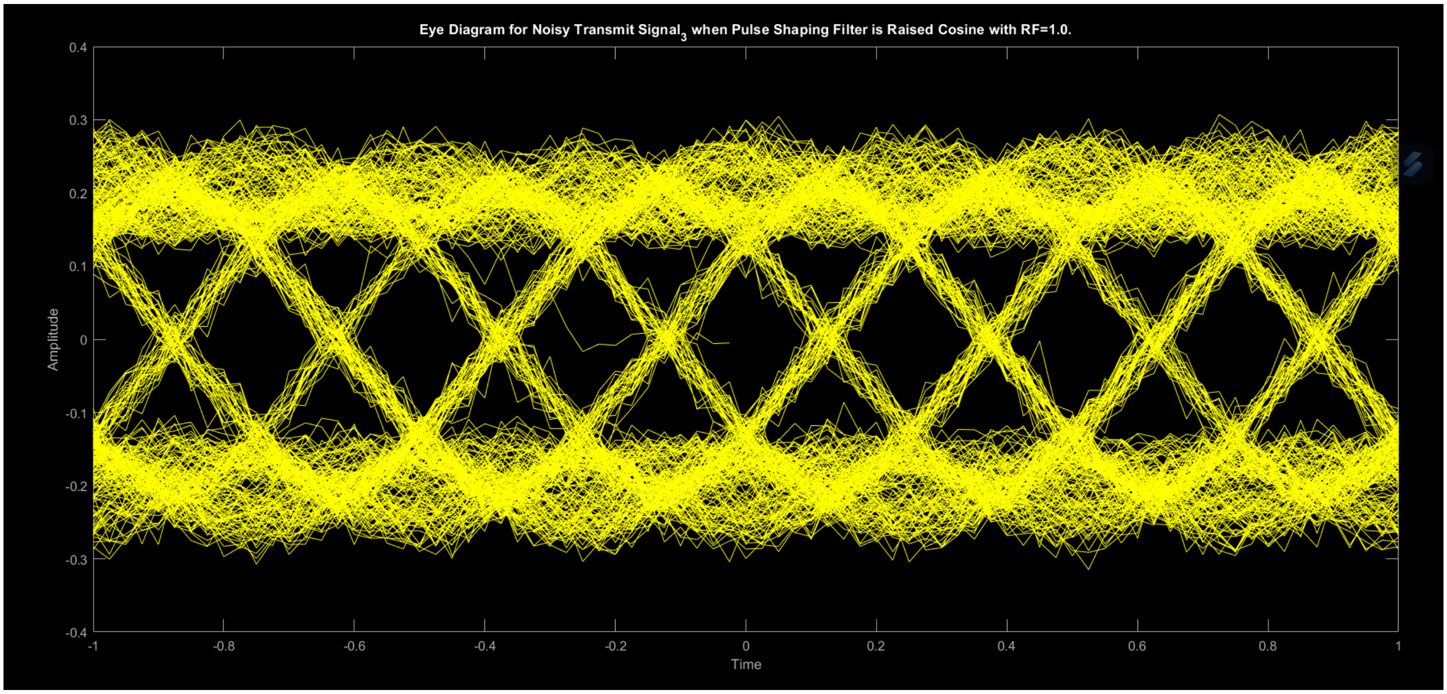


TASK 02 – Discussion

In this stage, an AWGN (Additive White Gaussian Noise) is added to the system to generate noise with variance of $\frac{E_b}{N_o} = 10dB$, Where E_b is the average energy per bit and N_o is the noise power spectral density.







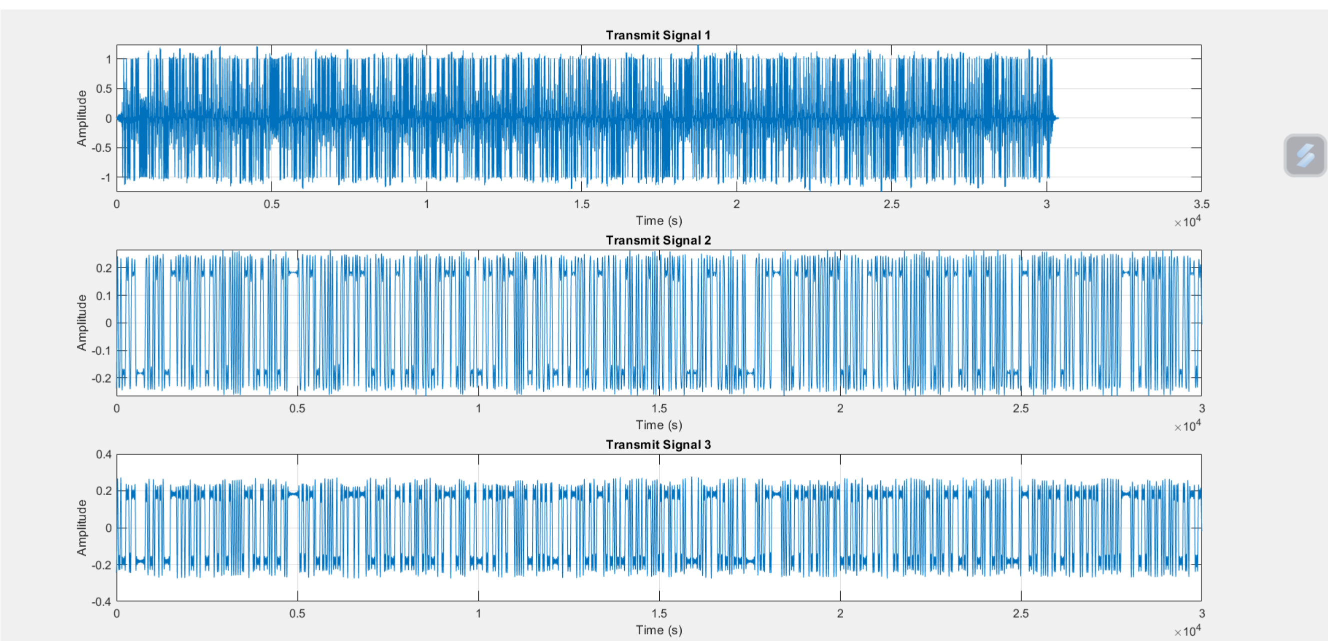
When AWGN is added, our BPSK impulse train will show different amplitudes apart from ±1 and will change the amplitudes of the pulses that we transmit which will cause errors which were not presented in the previous scenario.

After AWGN is added, the robustness of the system for different erroneous occasions can be explained by using the same characteristics of the eye diagrams as follows:

Characteristic of the Eye diagram	Information revealed by the characteristic	Sinc Pulse	Raised Cosine Pulse with Roll-off factor = 0.5	Raised Cosine Pulse with Roll-off factor = 1
1. Slope of the eye	Level of synchronization errors	Slope is very high which causes a high possibility for synchronization error occurrences.	Has the lowest slope among the three occasions. Low possibility for synchronization errors.	Has an average slope which is less than the sinc pulse and greater than the raised cosine pulse with 0.5 roll-off. Average possibility for synchronization errors.
2. Height of the eye	Immunity to noise (half of height of the eye at optimum sampling point)		The height was diminished on an average scale. Possibility of sampling errors has increased.	Height has reduced than the previous scenario but still it has the maximum out of three. Better robustness to sampling errors when comparing other two pulses.

3. Width of the eye	Error free sampling region	Has the lowest eye width. Almost all the region is not good enough for error-free sampling (Low robustness).	Width has reduced due to the AWGN. But still a small region is suitable for error free sampling.	Width has reduced due to the AWGN but still maintain enough error free sampling region. Compared to other two pulses, robustness for sampling errors is high.
4. Thickness at the peak	Peak deviation	Peak deviation can be seen due to the presence of SNR value which cause some distortions in pulse amplitudes.	Peak deviation can be seen due to the presence of SNR value in the sampling region.	Peak deviation can be seen due to the presence of SNR value in the sampling region. Deviations are almost same as the raised cosine pulse with 0.5 roll-off.
5. Time variation at zero-crossing	Timing off-set (Time jitter)	Jitter has increased compared with the previous case. Lowest robustness to timing offsets.	Jitter has increased and has average robustness to timing offsets.	Some jitter is available but still it is the lowest of all the three pulses. Better robustness for timing offsets than others.

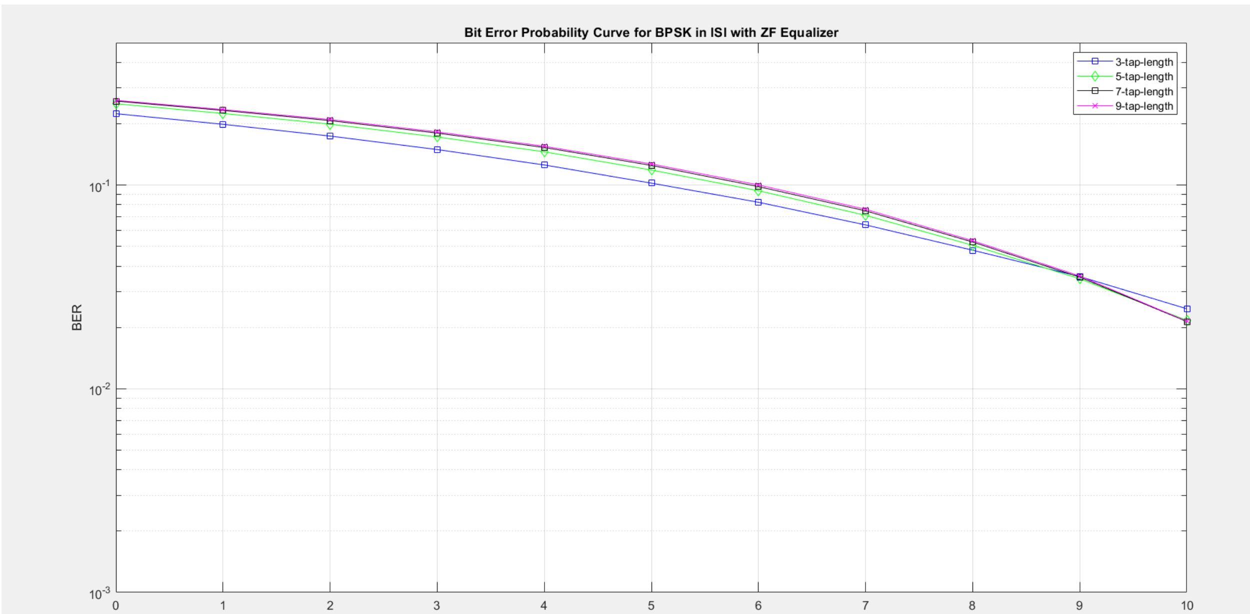
Under the above pulse shaping filters with specifications and AWGN added, the transmitted signals were obtained and can be viewed as follows:



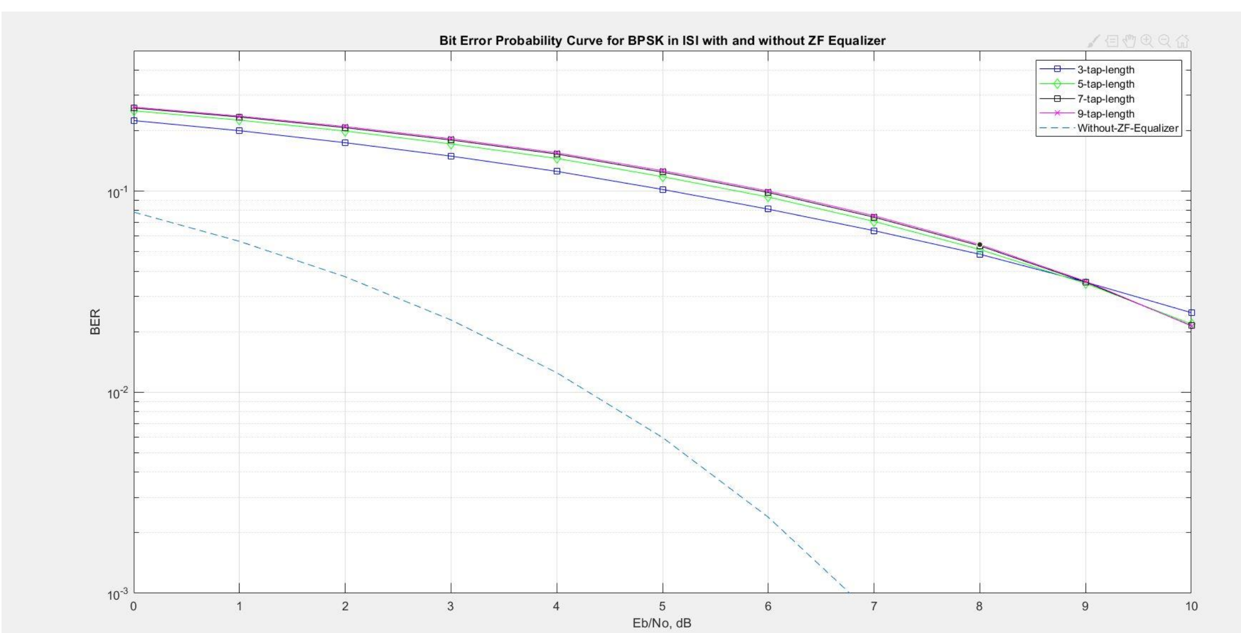
TASK 03 – Discussion

In this stage, we designed a zero-forcing (ZF) equalizer for a 3-tap multipath channel.

After generating the signals and demodulations and converting them to bits, under the presence of ZF equalization filters with different tap settings, the BER (Bit Error Rate) were calculated by counting the number of bit errors for different AWGN levels and plotted $\frac{E_b}{N_o}$ values for all tap settings in the same figure which can be viewed as follows:



Then the BER for AWGN (without ZF equalization) were calculated and plotted in the same figure which can be viewed as follows:



It can be clearly seen that there is a considerable discrepancy between the AWGN channel BER and the ZF equalized multipath channel BER. The BER values for AWGN channel are quite low than the ZF equalized channel and it decays faster when $\frac{E_b}{N_o}$ ratio is increased. We think that applying a 3-tap channel equalizer solves the multipath propagation problem. But there is also a noise associated with our signal and it gets amplified. Due to this unavoidable noise amplification part, we get this discrepancy between two types of channels.

Comment on the BER performance if binary orthogonal signaling was used instead of BPSK.

If we use binary orthogonal signaling instead of BPSK, even at a small level of noise, our existing BER performance becomes much worse because in orthogonal signal space representation, the distance from the center of gravity of the signal space to each signal point is lower than the distance value in BPSK which cause a lower decision region for those signal points. So, there is a much higher probability to change the transmitted symbols (Shrinking the distance between two signal points in the constellation diagram will cause higher error probability).

APPENDICES

1. **Task 1:**

```
clear;
clc;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%% Question 1

% Parameters
numSymbols = 1000; % Number of symbols to generate(Since BPSK, this is
equal to no.of bits)

bitRate = 1; % Bit rate (symbols per second)
samplingRate = 10 * bitRate; % Sampling rate (samples per second)

symbolDuration = 1 / bitRate; % Duration of each symbol in seconds

% Generate random binary data
binaryData = randi([0, 1], 1, numSymbols);

% Generate BPSK symbols as impulse train
bpskSymbols = 2 * binaryData - 1;
%disp(bpskSymbols)


%It is important that the convolution between the filters should occur at
%symbol periods. So, it is necessary to adjust the Sinc filter such that
%the data points on such periods are present. Here 30 zeros have been put
%between the impulses in the bpsk vector to achieve it.
padded_bpsk_vector = reshape([bpskSymbols;
zeros(29,numel(bpskSymbols))],1,[]);
%disp(padded_bpsk_vector)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

%% Question 2

t = -20:0.1:20;

% Put time vector into sinc filter and plotting it
sinc_Filter = sinc(t);

% Convolve BPSK symbols with pulse shaping filter
transmitSignal_1 = conv(padded_bpsk_vector, sinc_Filter);

%n = length(transmitSignal_1);
```

```
%disp(n)
%disp(transmitSignal)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

%% Question 3

figure;
eyediagram(transmitSignal_1,80,2*symbolDuration);     % Plot eye-diagram
for Transmit Signal_1.
title('Eye Diagram for Transmit Signal when Pulse Shaping Filter is a
Sinc')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%

%% Question 4

% Roll-off factors for the raised cosine filters.
rolloff_factor1 = 0.5;
rolloff_factor2 = 1;

r_c_f_1 = rcosdesign(rolloff_factor1,10,30,'sqrt');    % Make raised cosine
filter 1
r_c_f_1 = r_c_f_1 / max(r_c_f_1);% Normalize the filter to ensure that the
center amplitude is 1


r_c_f_2 = rcosdesign(rolloff_factor2,10,30,'sqrt');    % Make raised cosine
filter 2
r_c_f_2 = r_c_f_2 / max(r_c_f_2); % Normalize the filter to ensure that
the center amplitude is 1

% Make transitSignal_2 by convolving raised cosine filter and padded bpsk
vector
transmitSignal_2 = conv(padded_bpsk_vector, r_c_f_1,'same');


% Make transitSignal_3 by convolving raised cosine filter and padded bpsk
vector
transmitSignal_3 = conv(padded_bpsk_vector, r_c_f_2,'same');

%% Plotting the eye diagrams

% Plot eye-diagram for Transmit Signal_2
figure;
eyediagram(transmitSignal_2,80,2*symbolDuration);
title('Eye Diagram for Transmit Signal_2 when Pulse Shaping Filter is
Raised Cosine with RF=0.5.');
```

```
% Plot eye-diagram for Transmit Signal_2.
figure;
eyediagram(transmitSignal_3,80,2*symbolDuration);
title('Eye Diagram for Transmit Signal_3 when Pulse Shaping Filter is
Raised Cosine with RF=1.0.');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting all the filters
figure;

subplot(3,1,1);
plot(sinc_Filter)
title('Sinc Filter');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3,1,2);
plot(r_c_f_1)
title('Raised cosine filter 1');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3,1,3);
plot(r_c_f_2)
title('Raised cosine filter 2');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting all the transmitted signals
figure;

subplot(3,1,1);
plot(transmitSignal_1)
title('Transmit Signal 1');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3,1,2);
plot(transmitSignal_2)
title('Transmit Signal 2');
xlabel('Time (s)');
ylabel('Amplitude');
```

```
grid on;

subplot(3,1,3);
plot(transmitSignal_3)
title('Transmit Signal 3');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%% Close all the empty figures
close(1);
close(3);
close(5);
```

2. **Task 2:**

```
clear;
clc;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

% Parameters
numSymbols = 1000; % Number of symbols to generate(Since BPSK, this is
equal to no.of bits)

bitRate = 1; % Bit rate (symbols per second)
samplingRate = 10 * bitRate; % Sampling rate (samples per second)

symbolDuration = 1 / bitRate; % Duration of each symbol in seconds

% Generate random binary data
binaryData = randi([0, 1], 1, numSymbols);

% Generate BPSK symbols as impulse train
bpskSymbols = 2 * binaryData - 1;
%disp(bpskSymbols)


%It is important that the convolution between the filters should occur at
%symbol periods. So, it is necessary to adjust the Sinc filter such that
%the data points on such periods are present. Here 30 zeros have been put
%between the impulses in the bpsk vector to achieve it.
padded_bpsk_vector = reshape([bpskSymbols;
zeros(29,numel(bpskSymbols))],1,[]);
%disp(padded_bpsk_vector)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

t = -20:0.1:20;

% Put time vector into sinc filter and plotting it
sinc_Filter = sinc(t);

% Convolve BPSK symbols with pulse shaping filter
transmitSignal_1 = conv(padded_bpsk_vector, sinc_Filter);

%n = length(transmitSignal_1);
%disp(n)
%disp(transmitSignal)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

rolloff_factor1 = 0.5;       % Roll-off factor for first raised cosine
filter 1.
rolloff_factor2 = 1;         % Roll-off factor for second raised cosine
filter 2.

r_c_f_1 = rcosdesign(rolloff_factor1,10,30,'sqrt');    % Make raised cosine
filter 1
r_c_f_1 = r_c_f_1 / max(r_c_f_1);

r_c_f_2 = rcosdesign(rolloff_factor2,10,30,'sqrt');    % Make raised cosine
filter 2
r_c_f_2 = r_c_f_2 / max(r_c_f_2);

% Make transitSignal_2 by convolving raised cosine filter and padded bpsk
vector
transmitSignal_2 = conv(padded_bpsk_vector, r_c_f_1,'same');


% Make transitSignal_3 by convolving raised cosine filter and padded bpsk
vector
transmitSignal_3 = conv(padded_bpsk_vector, r_c_f_2,'same');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%

%% Additive White Gaussian Noise is in action from here onwards.

% The average bit energy (Eb)
Eb = sum(abs(padded_bpsk_vector).^2)/numSymbols;

% No where Eb/No = 10dB
```

```
EbNo_dB = 10;
No = Eb/(10^(EbNo_dB/10));

% Noise variance
noise_variance = No/2;


%It is necessary to generate the noise with zero mean and the calculated
variance
%seperately for all 3 transmitted signals since they have different
%lengths.

n_1 = noise_variance*normrnd(0,1,1,length(transmitSignal_1));
n_2 = noise_variance*normrnd(0,1,1,length(transmitSignal_2));
n_3 = noise_variance*normrnd(0,1,1,length(transmitSignal_3));

% Add the noise to each transmitted signals
noisy_transmitSignal_1 = transmitSignal_1 + n_1;
noisy_transmitSignal_2 = transmitSignal_2 + n_2;
noisy_transmitSignal_3 = transmitSignal_3 + n_3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%

%% Plot eye-diagrams for each transmit signal
figure;
eyediagram(noisy_transmitSignal_1,80,2*symbolDuration);
title('Eye Diagram for Noisy Transmit Signal when Pulse Shaping Filter is
a Sinc')

figure;
eyediagram(noisy_transmitSignal_2,80,2*symbolDuration);
title('Eye Diagram for Noisy Transmit Signal_2 when Pulse Shaping Filter
is Raised Cosine with RF=0.5.');

figure;
eyediagram(noisy_transmitSignal_3,80,2*symbolDuration);
title('Eye Diagram for Noisy Transmit Signal_3 when Pulse Shaping Filter
is Raised Cosine with RF=1.0.');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Plotting all the filters
figure;

subplot(3,1,1);
plot(sinc_Filter)
title('Sinc Filter');
xlabel('Time (s)');
```

```
ylabel('Amplitude');
grid on;

subplot(3,1,2);
plot(r_c_f_1)
title('Raised cosine filter 1');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3,1,3);
plot(r_c_f_2)
title('Raised cosine filter 2');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Plotting all the transmitted signals
figure;

subplot(3,1,1);
plot(transmitSignal_1)
title('Transmit Signal 1');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3,1,2);
plot(transmitSignal_2)
title('Transmit Signal 2');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3,1,3);
plot(transmitSignal_3)
title('Transmit Signal 3');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

%% Close all the empty figures
close(1);
close(3);
```

```
close(5);
```

3. **Task 3:**

```
clear;
clc;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Number of symbols to generate(Since BPSK, this is equal to no.of bits)

numSymbols = 10^6;

% Multiple Eb/N0 values
EbN0_dB = [0:10];

%Number of taps used in the equalizer.
no_of_taps = 4;

for ii = 1:length(EbN0_dB)

   %% Transmitter

   % Generate random binary data
   binaryData = randi([0 1],1,numSymbols);

   % Generate BPSK symbols as impulse train
   bpskSymbols = 2*binaryData-1;

   %% Characteristics of the channel

   %No. of taps
   nTap = 3;

   %The impulse reponse of the channel
   impulse_response = [0.3 0.9 0.4];

   %% Simulating the effect of a multipath channel.
   channel_output_signal = conv(bpskSymbols,impulse_response);

   %% Noise calculations

   %Generating random numbers from a standard normal distribution
   %with a mean of 0 and variance of 1
   x = randn(1,numSymbols+length(impulse_response)-1);

   %Gaussian noise with mean 0 and variance 1
   gaussian_noise = 1/sqrt(2)*[x + 1j*x]; % white gaussian noise, 0dB
variance
```

```
   % Noise addition
   noise_power = 10^(-EbN0_dB(ii)/20)*gaussian_noise;

   noisy_signal = channel_output_signal + noise_power;

   for kk = 1:no_of_taps

     L  = length(impulse_response);

     %Create a Toeplitz matrix that represents
     %the channel impulse response for the equalizer
     column = [impulse_response([2:end]) zeros(1,2*kk+1-L+1)];
     row = [ impulse_response([2:-1:1]) zeros(1,2*kk+1-L+1) ];
     channel_impulse_response = toeplitz(column, row);

     %Create a unit impulse response at the desired tap position
     d  = zeros(1,2*kk+1);
     d(kk+1) = 1;

     %Calculate the equalizer coefficients
     equilizer_coefficients  = [inv(channel_impulse_response)*d.'].';

     %% mathched filter
     yFilt = conv(noisy_signal,equilizer_coefficients);
     yFilt = yFilt(kk+2:end);
     yFilt = conv(yFilt,ones(1,1)); % convolution
     ySamp = yFilt(1:1:numSymbols);  % sampling at time T


     %% At the receiver, hard decision decoding is performed
     recieved_binary_data = real(ySamp)>0;

     % Counting the errors
     no_of_bit_errors(kk,ii) = size(find([binaryData-
recieved_binary_data]),2);

   end


end

% Simulated BER
simulated_BER = no_of_bit_errors/numSymbols;

% Theoretical BER
theoryBer = 0.5*erfc(sqrt(10.^(EbN0_dB/10)));

%% Plotting BER for different no.of taps
close all
```

```
figure
semilogy(EbN0_dB,simulated_BER(1,:),'bs-'),'Linewidth';2;
hold on
semilogy(EbN0_dB,simulated_BER(2,:),'gd-'),'Linewidth';2;
semilogy(EbN0_dB,simulated_BER(3,:),'ks-'),'Linewidth';2;
semilogy(EbN0_dB,simulated_BER(4,:),'mx-'),'Linewidth';2;
semilogy(EbN0_dB,theoryBer,'--'),'Linewidth';2;
axis([0 10 10^-3 0.5])
grid on
legend('3-tap-length', '5-tap-length','7-tap-length','9-tap-
length','Without-ZF-Equalizer');
xlabel('Eb/No, dB');
ylabel('BER');
title('Bit Error Probability Curve for BPSK in ISI with and without ZF
Equalizer');
```