University of Moratuwa

Department of Electronic & Telecommunication Engineering

EN2074 Communication Systems Engineering

Lab Assignment – Eye Diagrams and Equalization

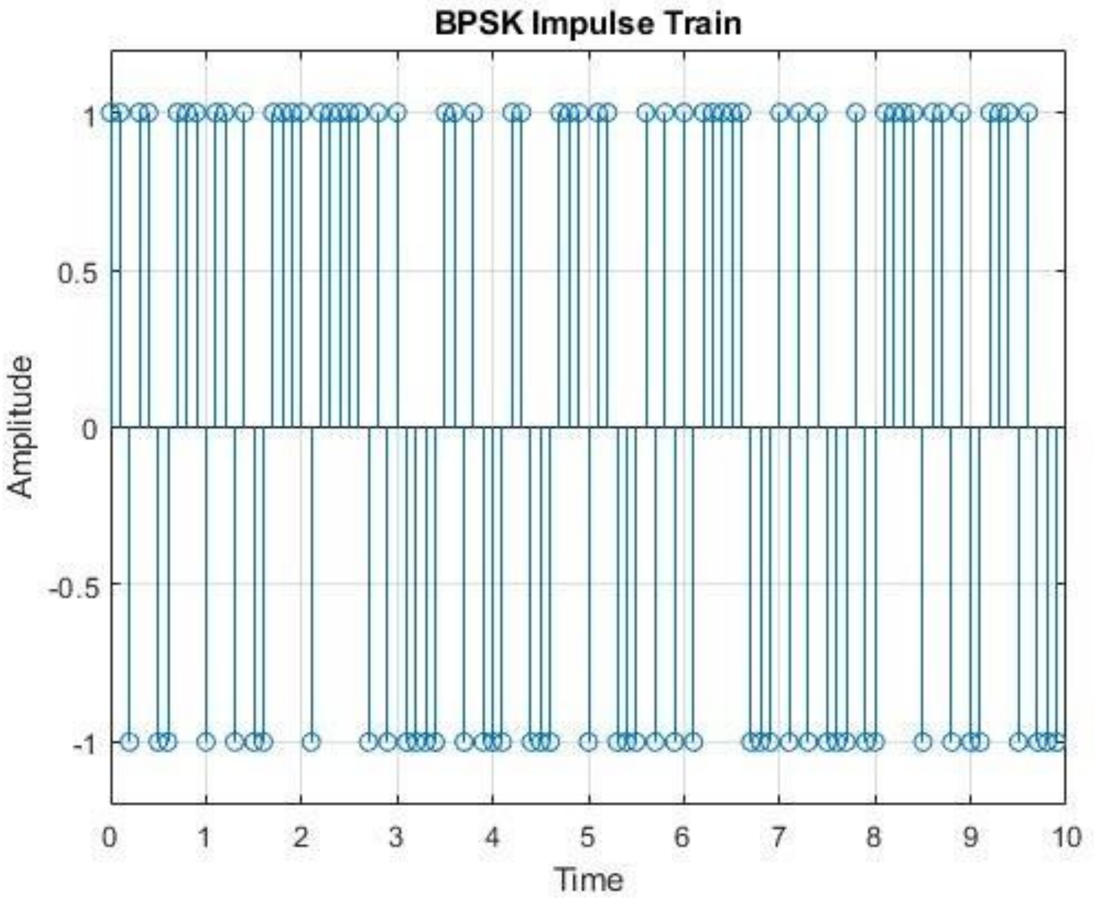
Name	Index No
Dodangoda D.K.S.J.	210150V
Wathudura T.R.	210682D

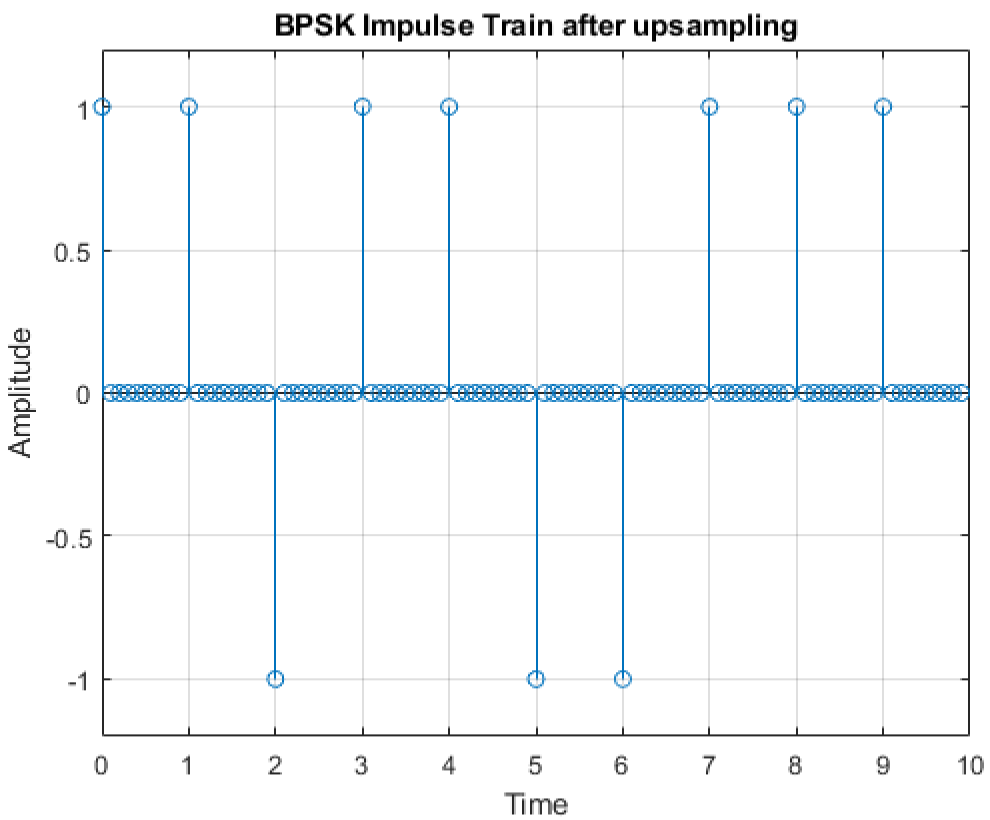
Content

Used Software - MATLAB R2022b

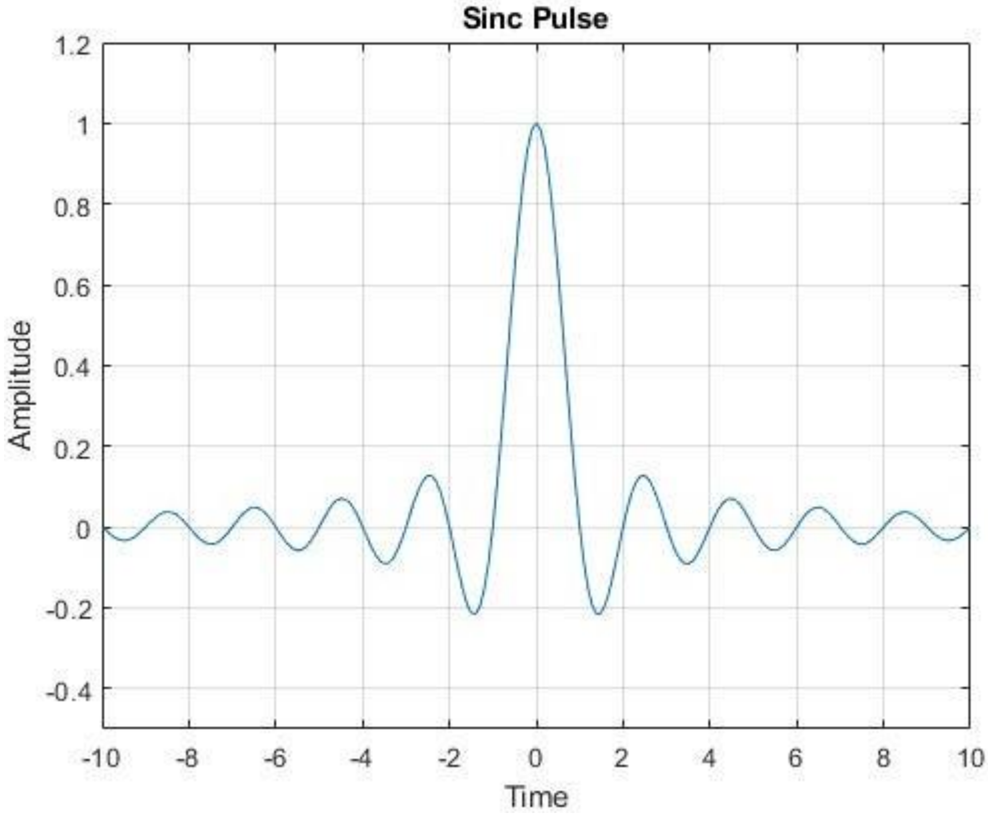
1) Task 1

1.1)



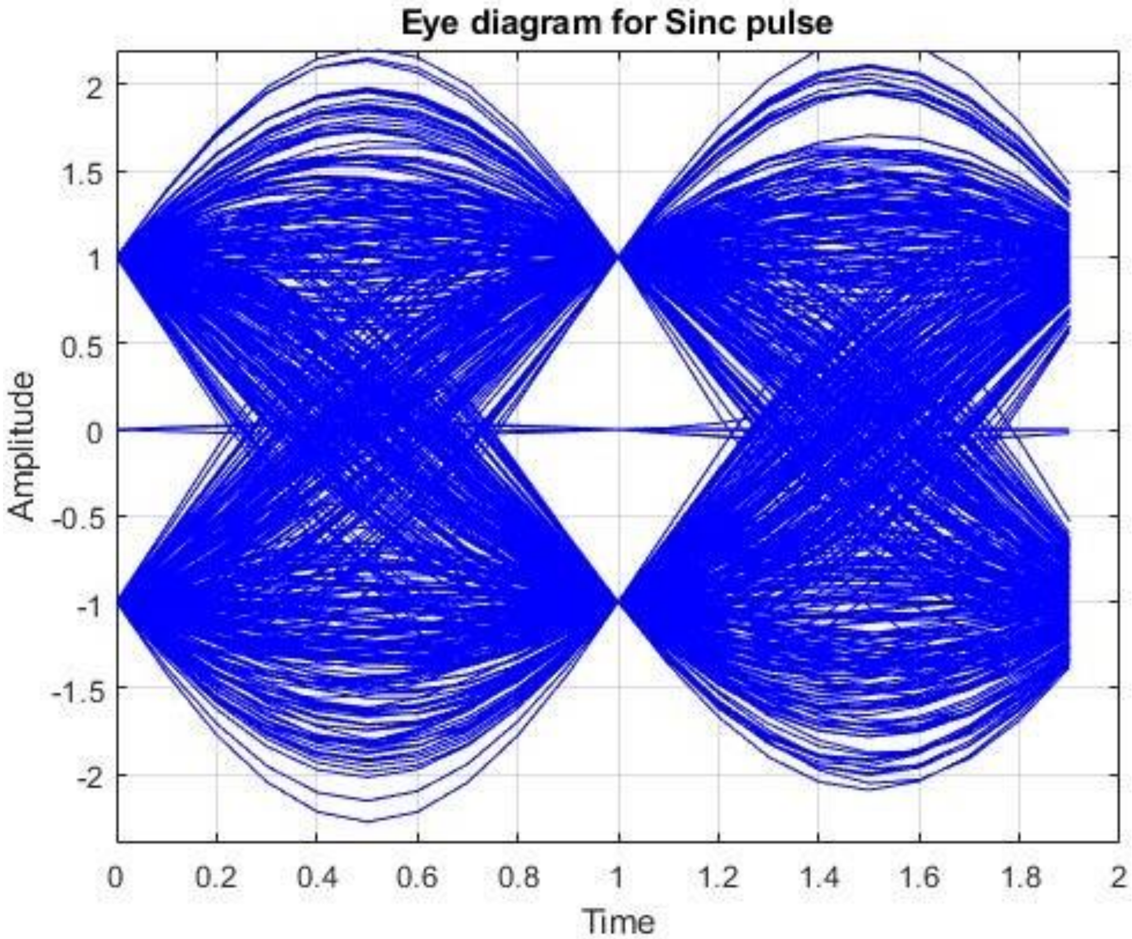


1.2)



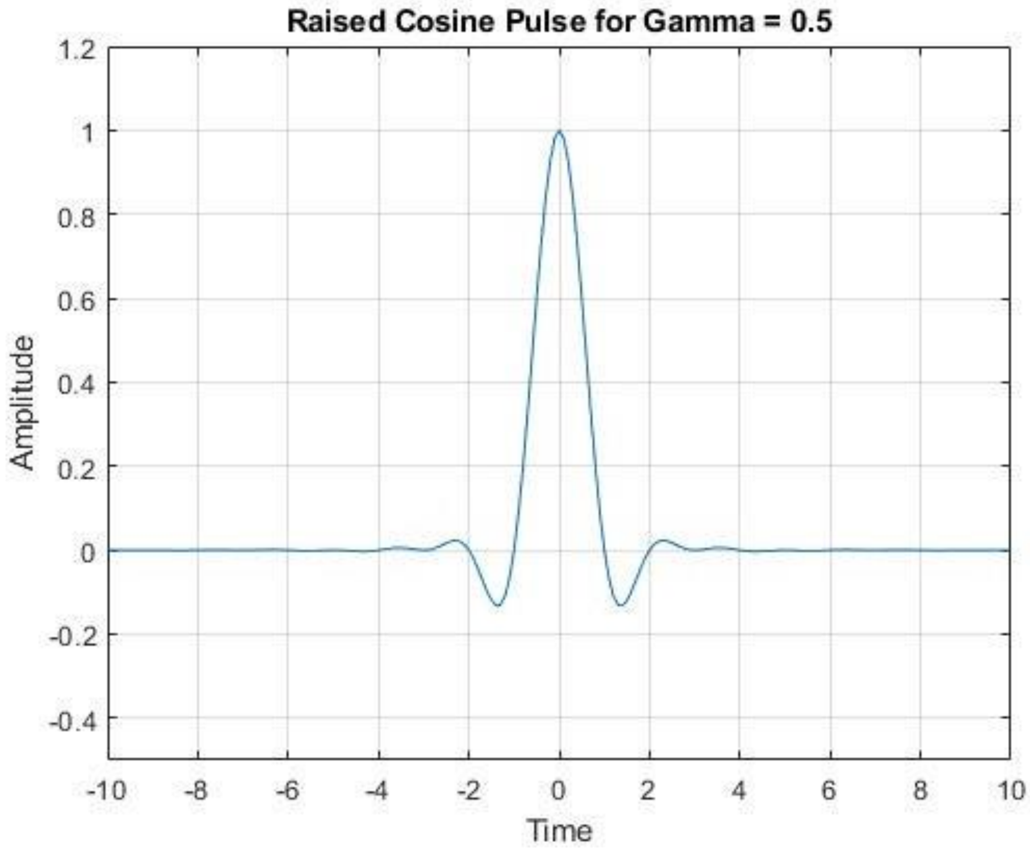
Sinc Pulse
Amplitude
Time

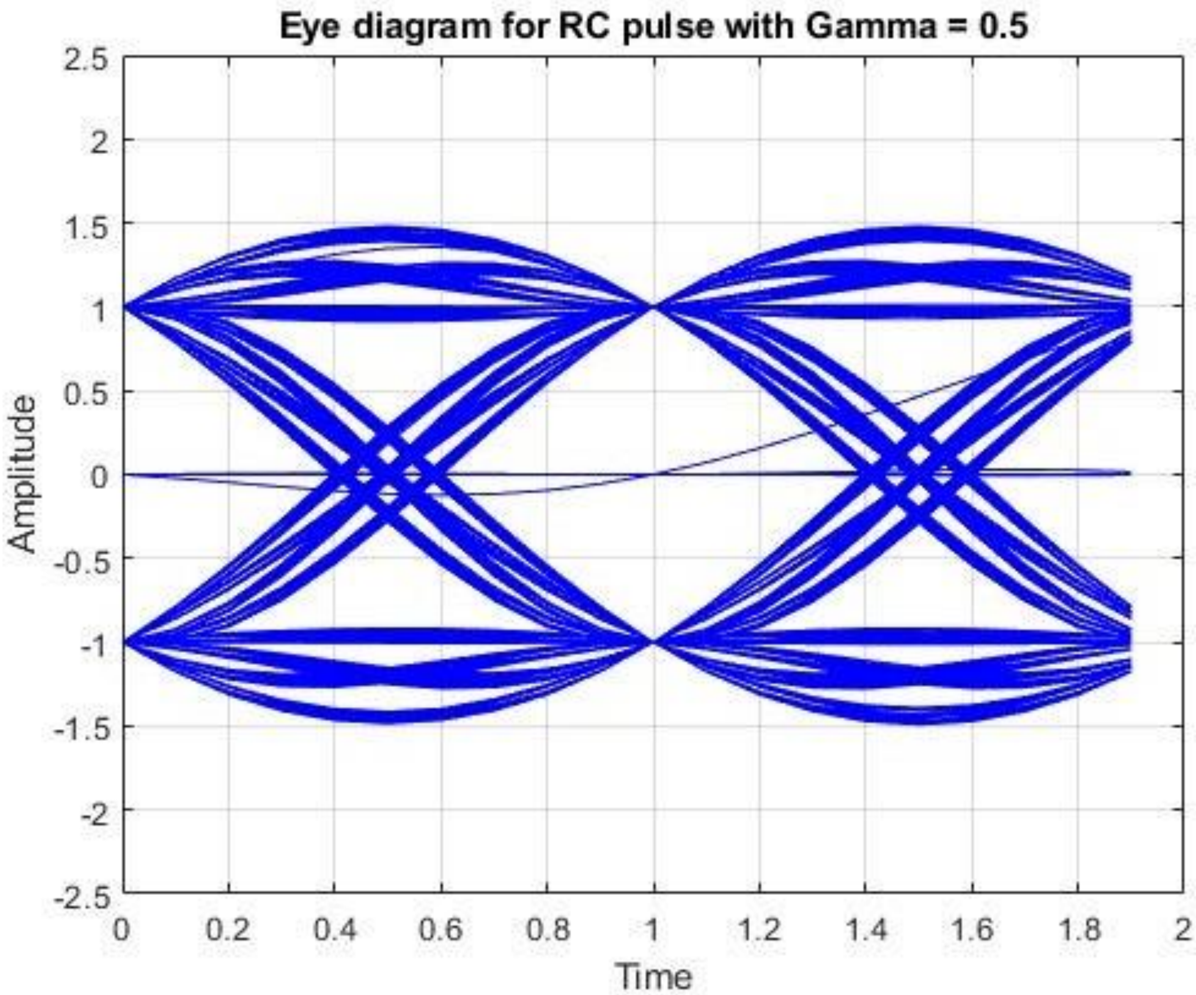
1.3)



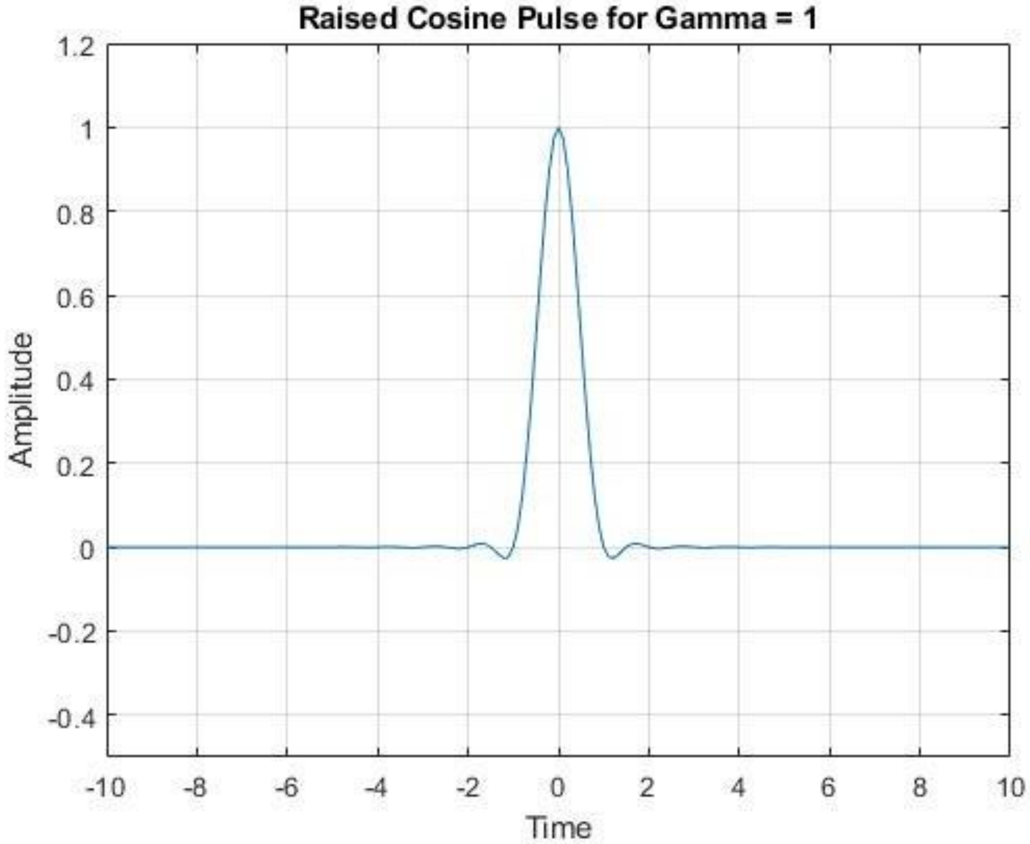
Eye diagram for Sinc pulse
Amplitude
Time

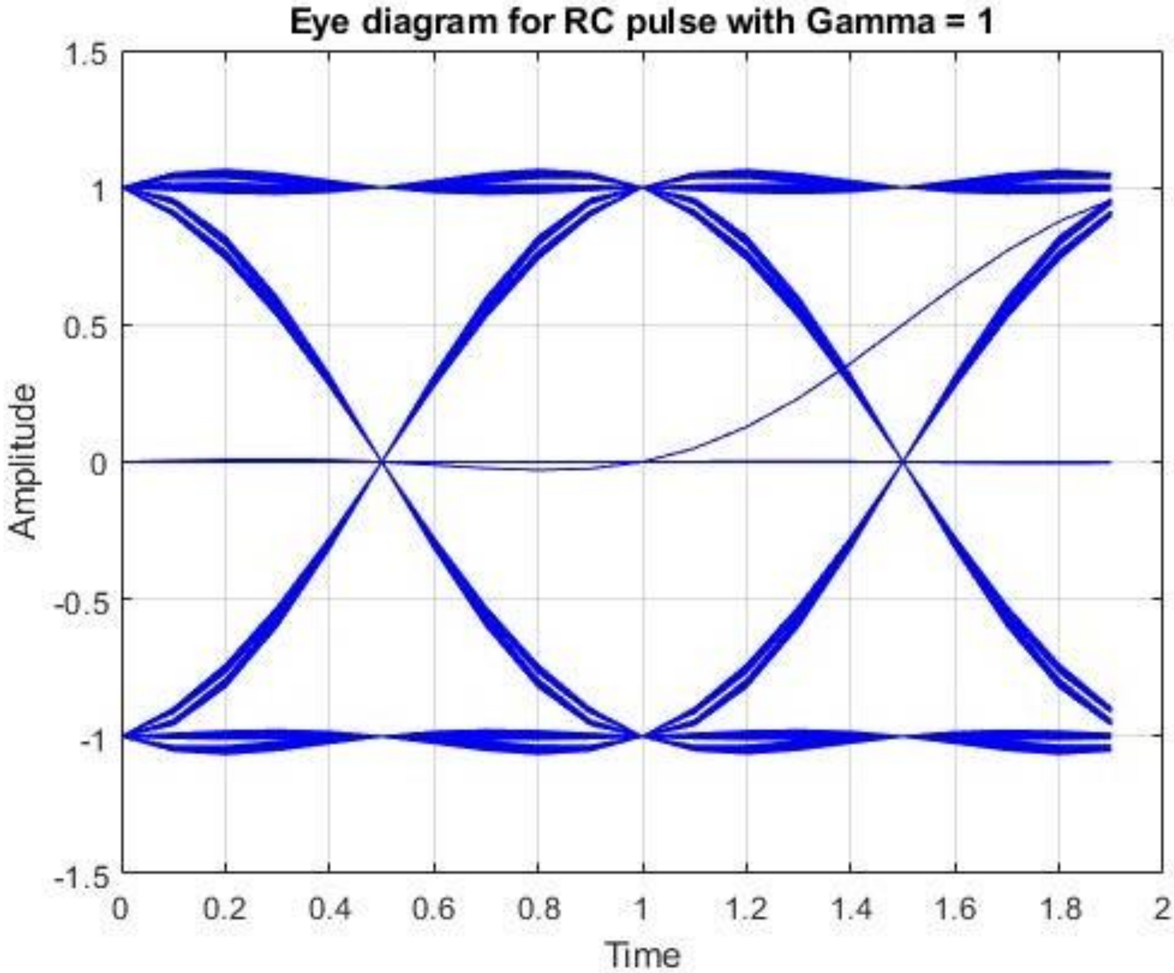
1.4. a) Raised cosine pulse with roll-off factor 0.5



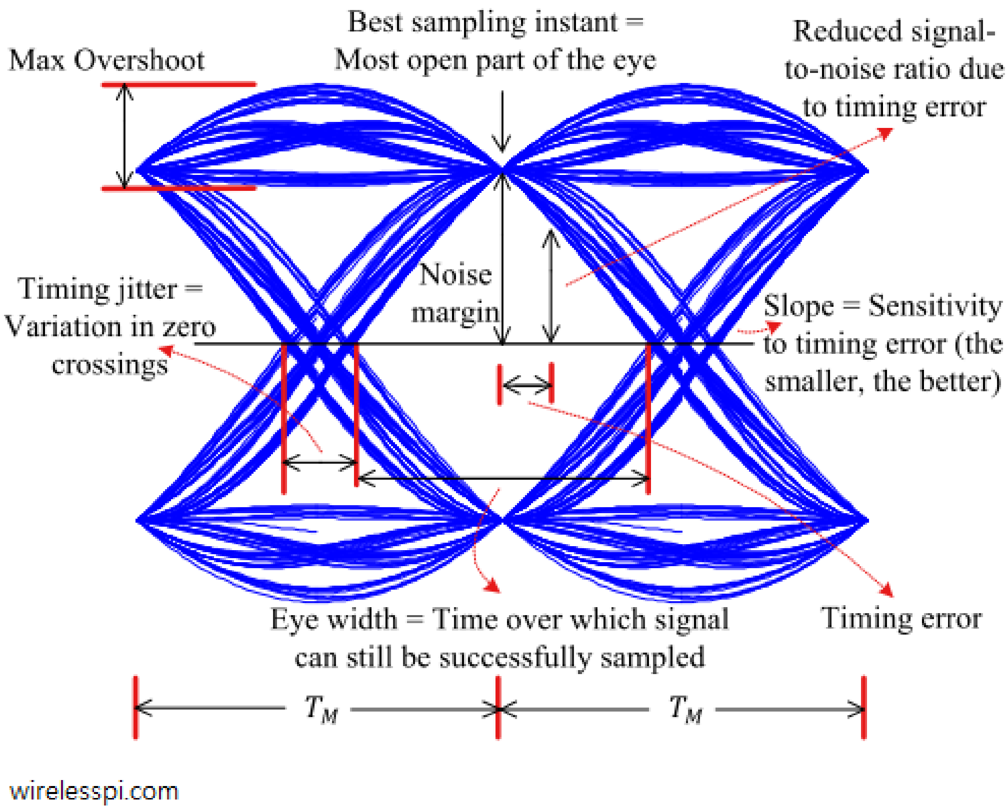


1.4. b) Raised cosine pulse with roll-off factor 1





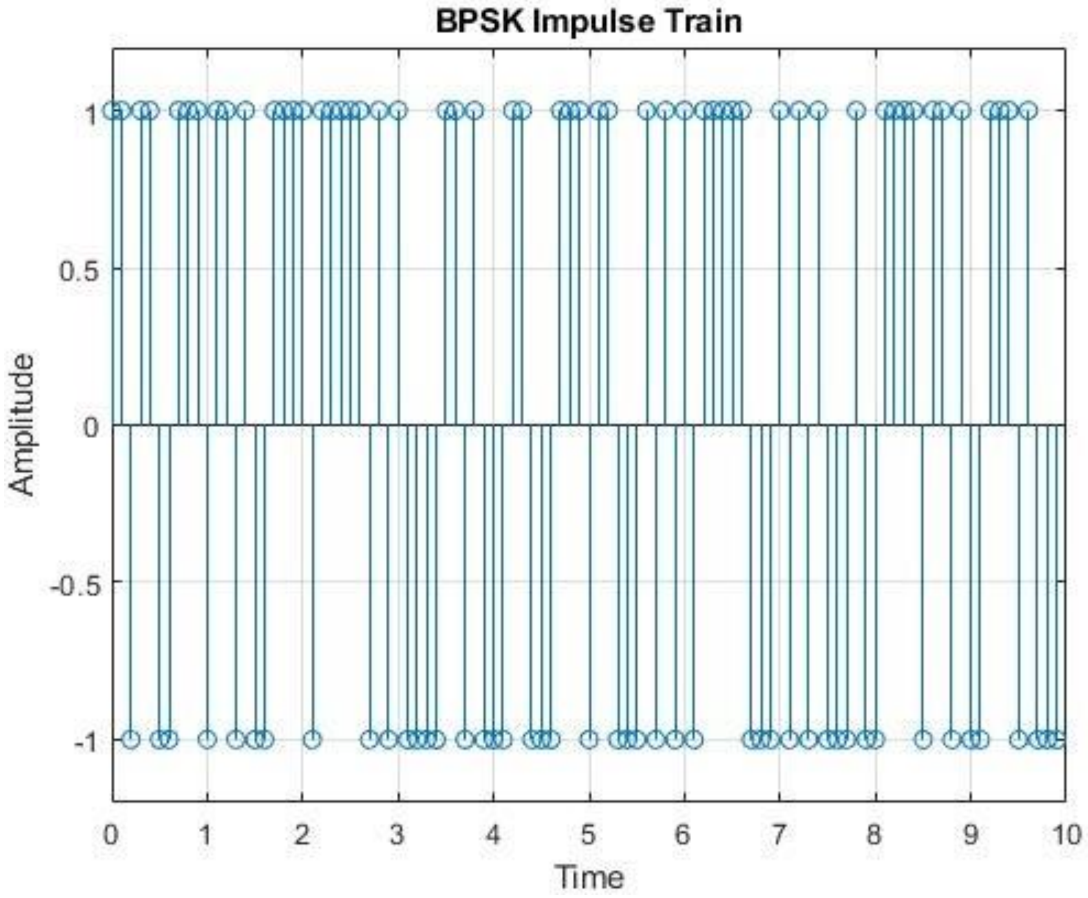
1.5) Comparison of the robustness of the system with respect to noise, sampling time and synchronization errors



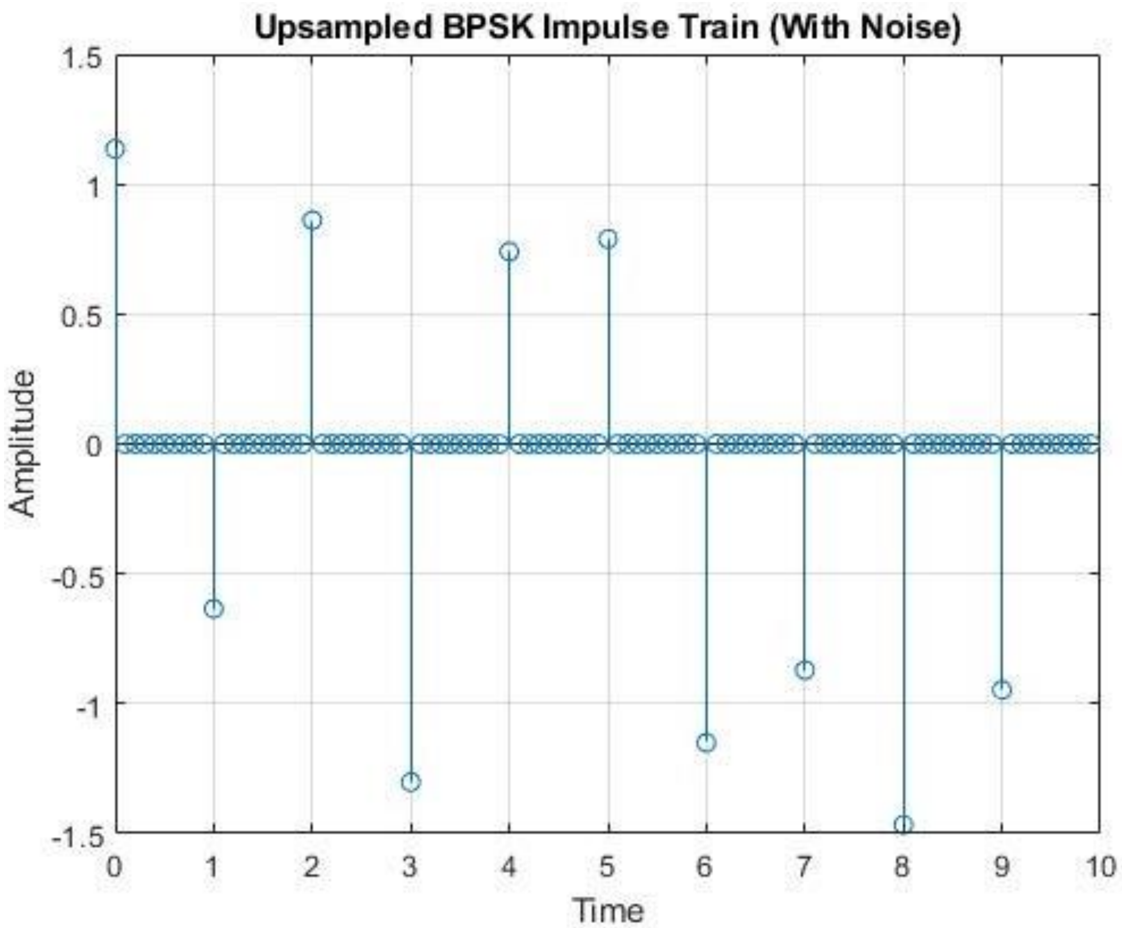
Characteristic	**Eye Parameter**	**Sinc Pulse**	**Raised Cosine Pulse (Roll-off = 0.5)**	**Raised Cosine Pulse (Roll-off =1)**
Noise (Noise Immunity)	Maximum height of the center of the eye. (Height of Eye at optimum sampling point) When this gap is wider, the noise immunity is higher.	No noise is added. Good noise margin is Present.	No noise is added. Good noise margin is Present.	No noise is added. Good noise margin is Present.
Sampling Time	Horizontal eye width. When this width is wider, the sampling instance can vary more without leading to errors.	Lowest of the three. Has the least range for error-free sampling. Least robustness to sampling errors (ISI).	Wider than the Sinc Pulse but narrower than the RC pulse with roll-off = 1. Medium range for error-free sampling. Medium robustness to sampling errors (ISI).	Widest of the three. Has the largest range for error-free sampling. Highest robustness to sampling errors (ISI).
Synchronization Errors	Slope of the eye. When the slope is larger, the probability of synchronization errors (sensitivity to sampling time) is higher.	Highest slope among the three. Highest probability for timing errors.	Lowest slope among the three. Best robustness to timing errors.	Medium slope among the three. More robust to timing errors than the Sinc pulse but less robust than the RC pulse with roll-off = 0.5.

2) Task 2

2.1)

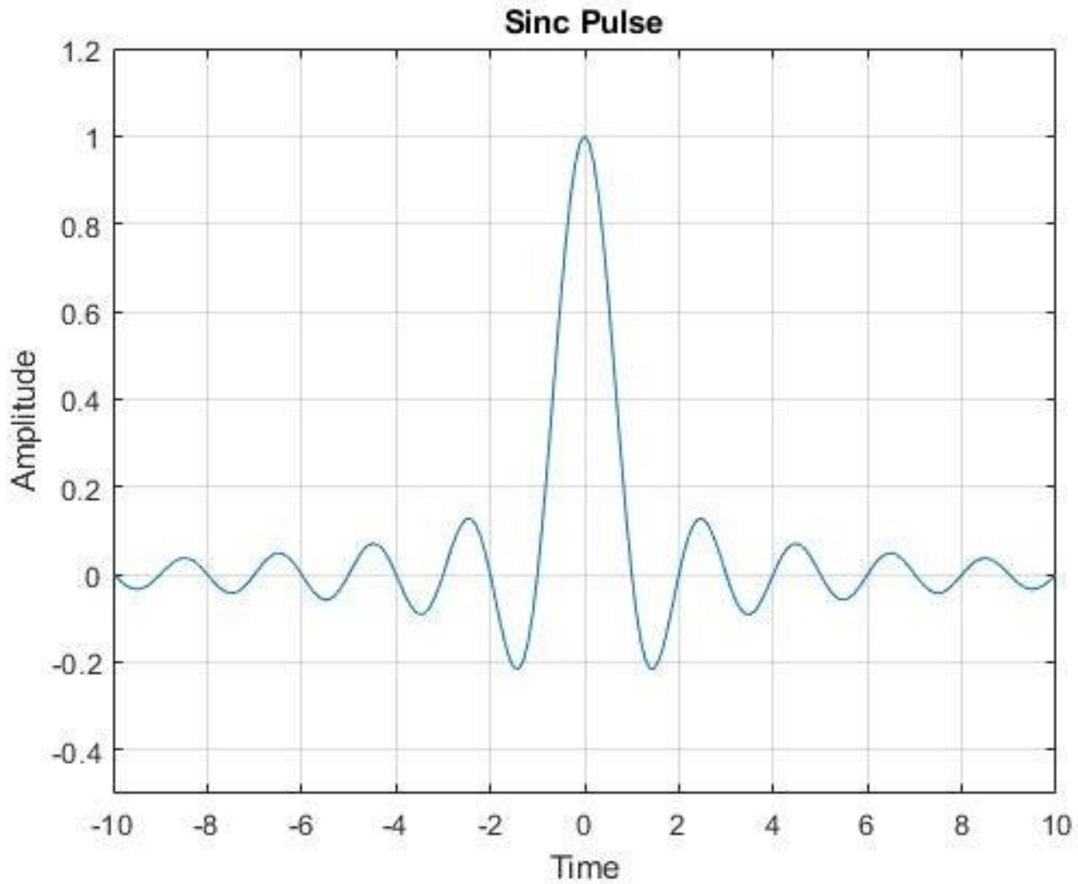


BPSK Impulse Train
Amplitude
Time
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10



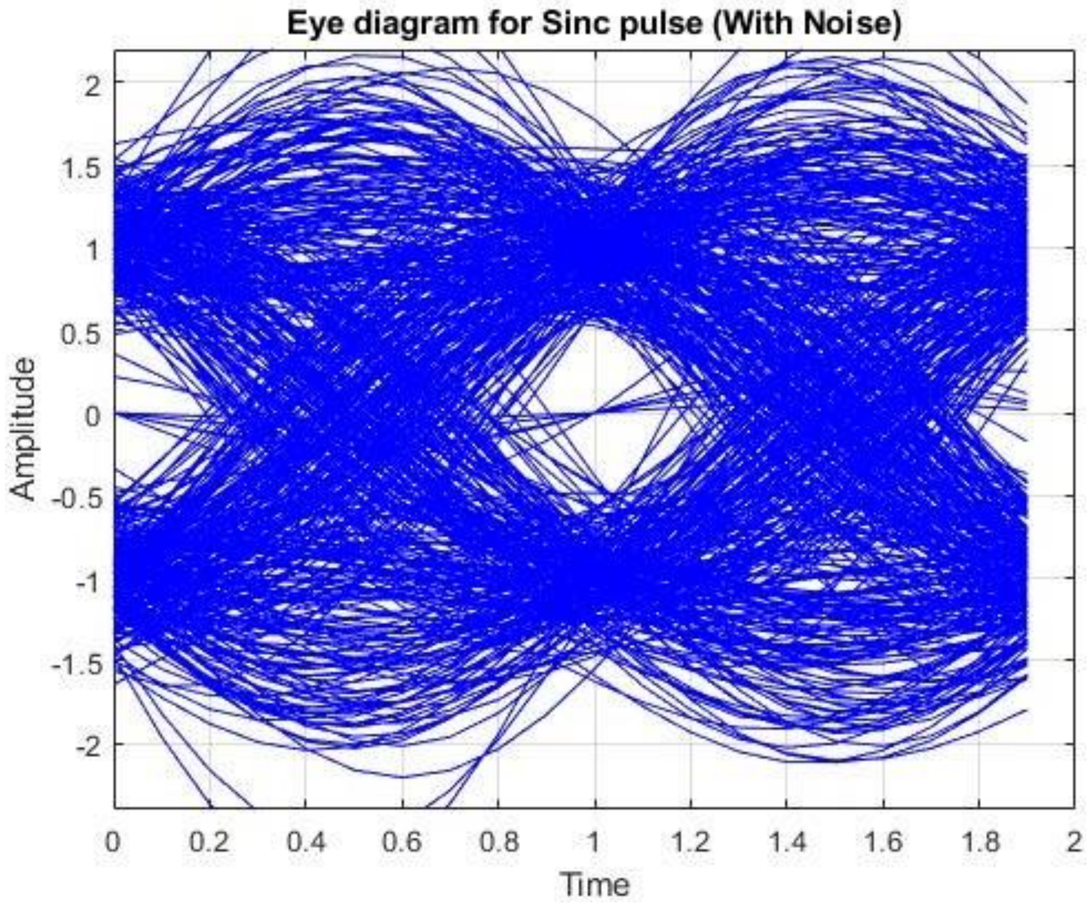
Upsampled BPSK Impulse Train (With Noise)
Amplitude
Time
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
6
7
8
9
10

2.2)



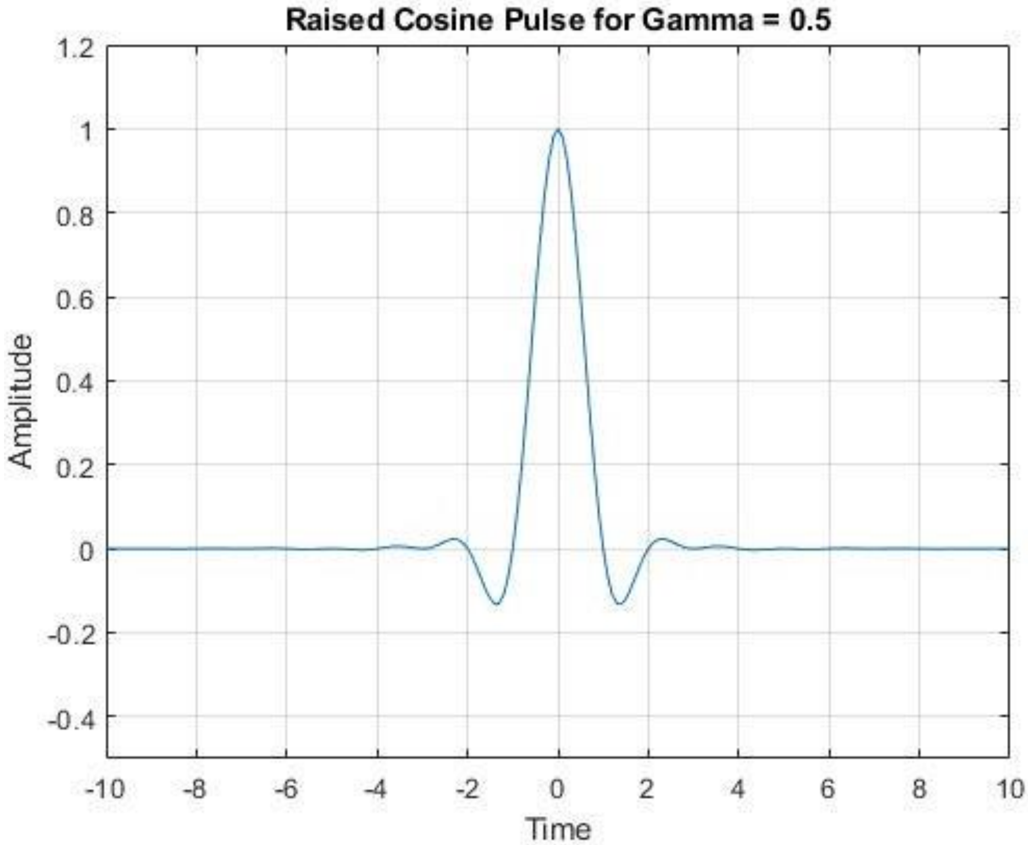
Sinc Pulse
Amplitude
Time
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-8
-6
-4
-2
0
2
4
6
8
10

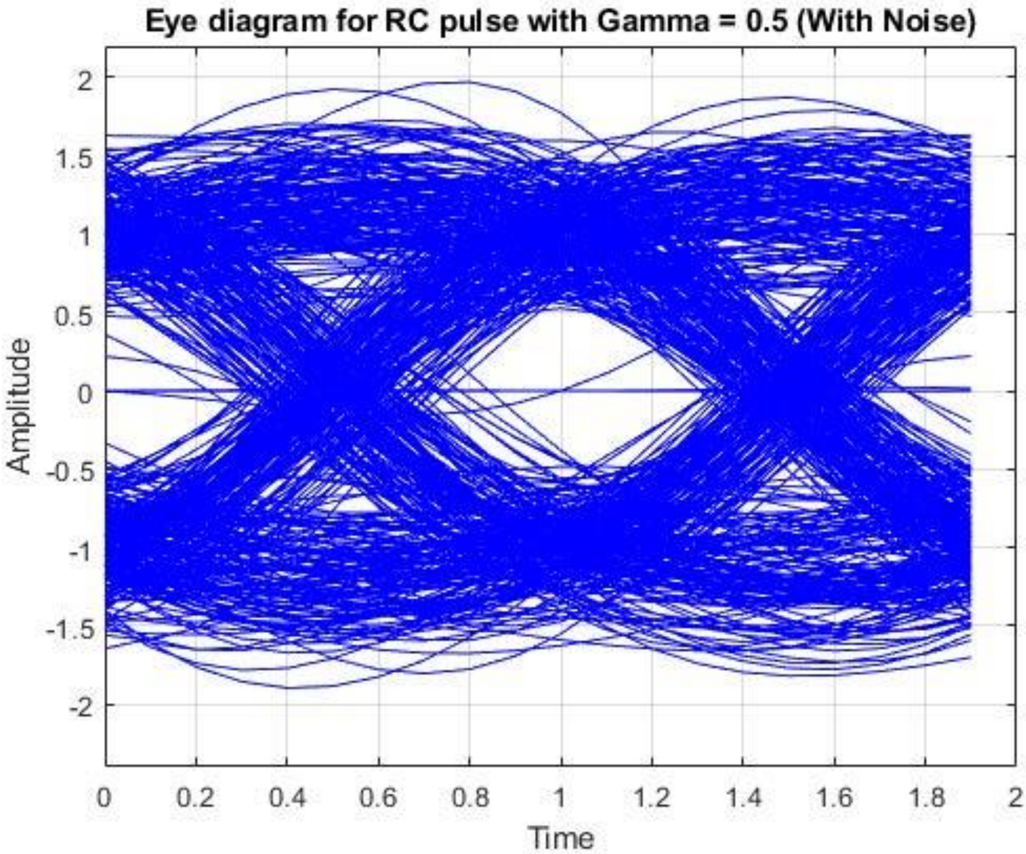
2.3)



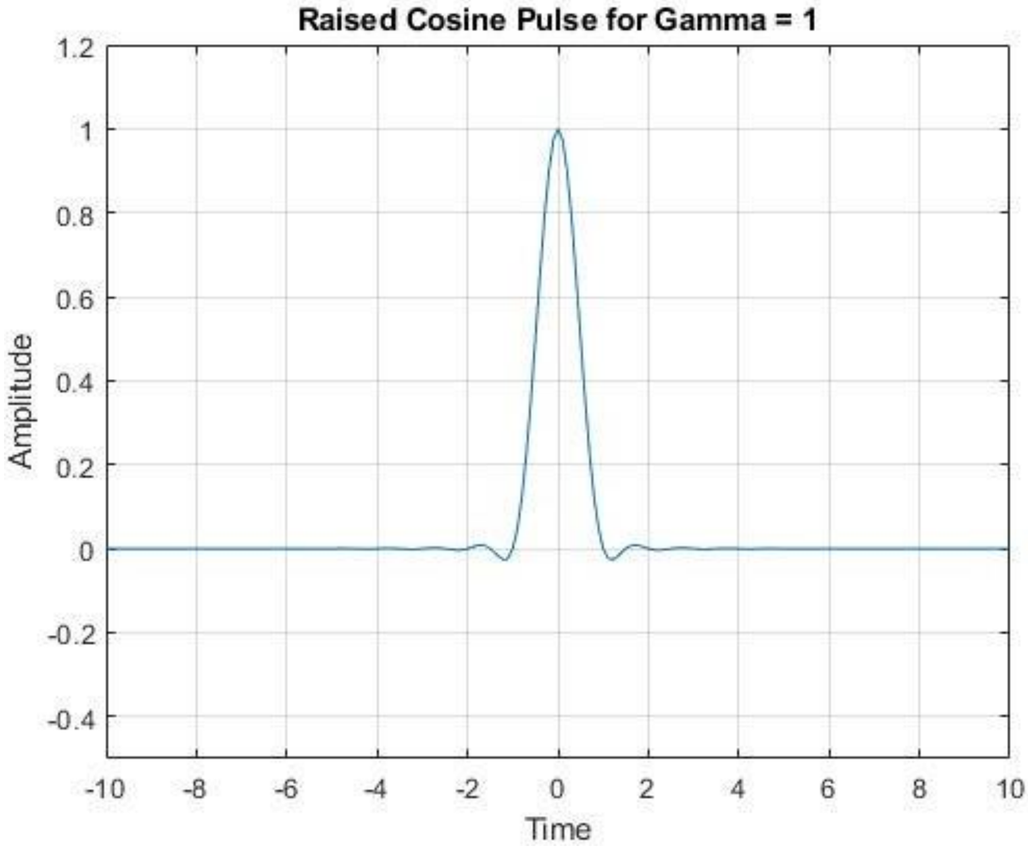
Eye diagram for Sinc pulse (With Noise)
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

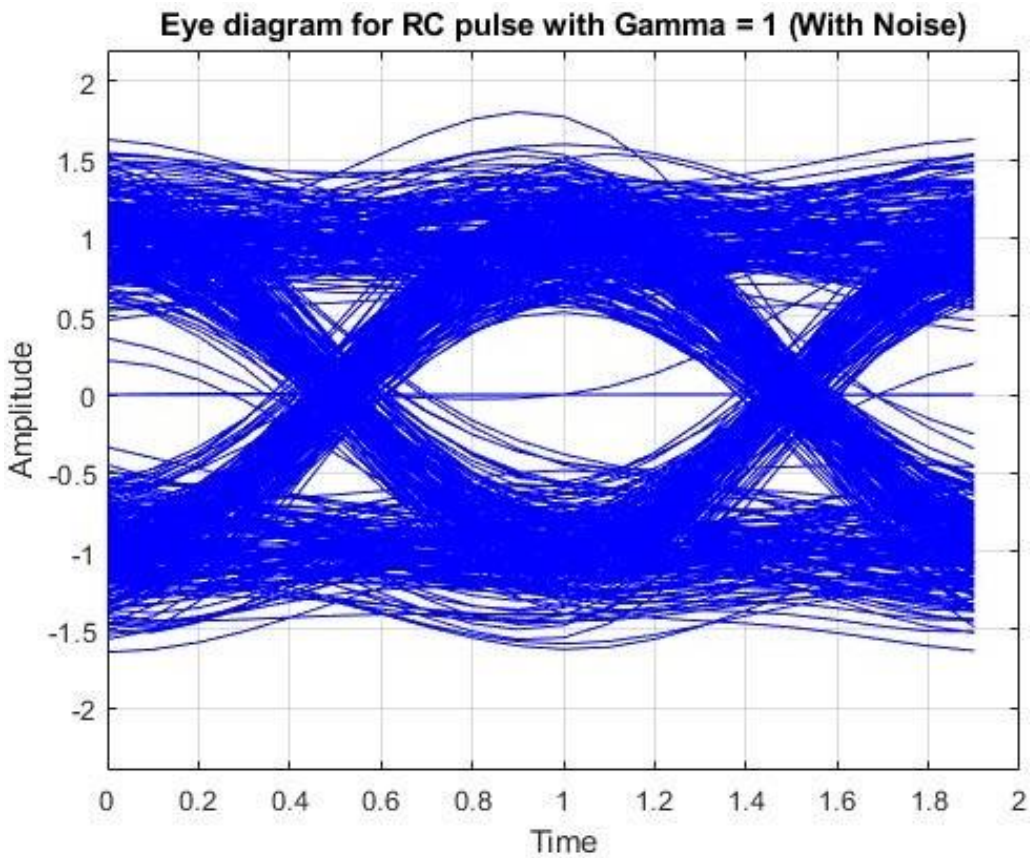
2.4.a) Raised cosine pulse with roll-off factor 0.5





2.4.b) Raised cosine pulse with roll-off factor 1





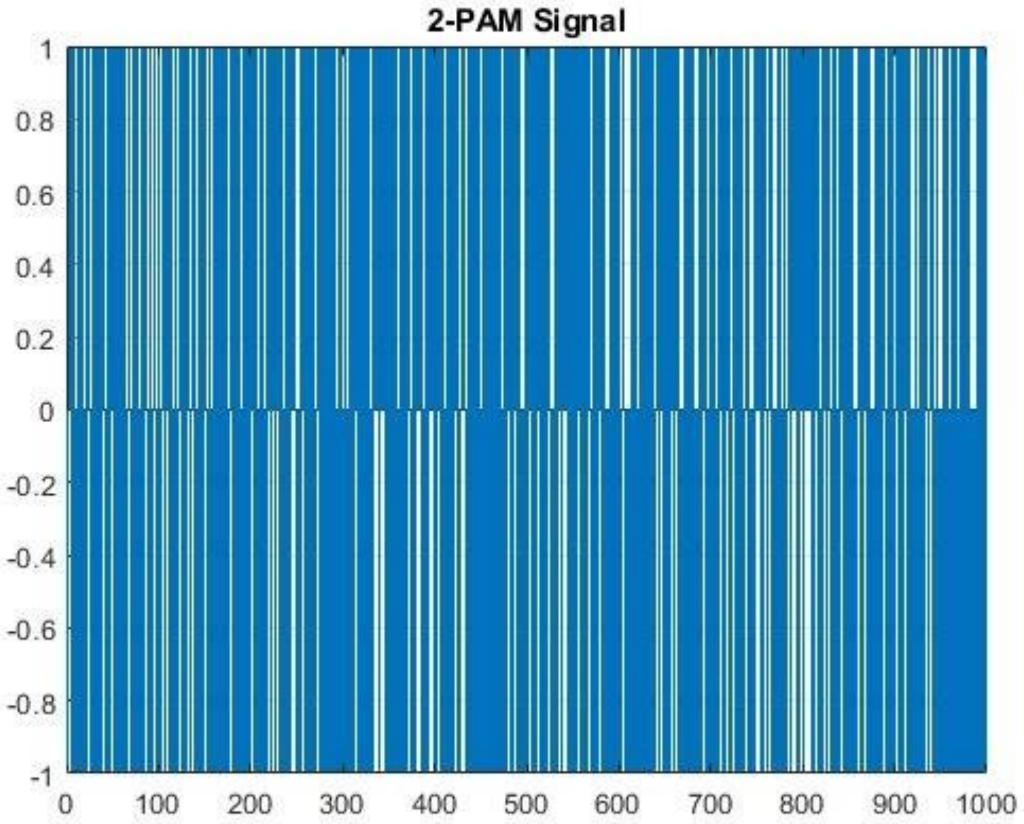
2.5 Comparison of the robustness of the system with respect to noise, sampling time and synchronization errors.

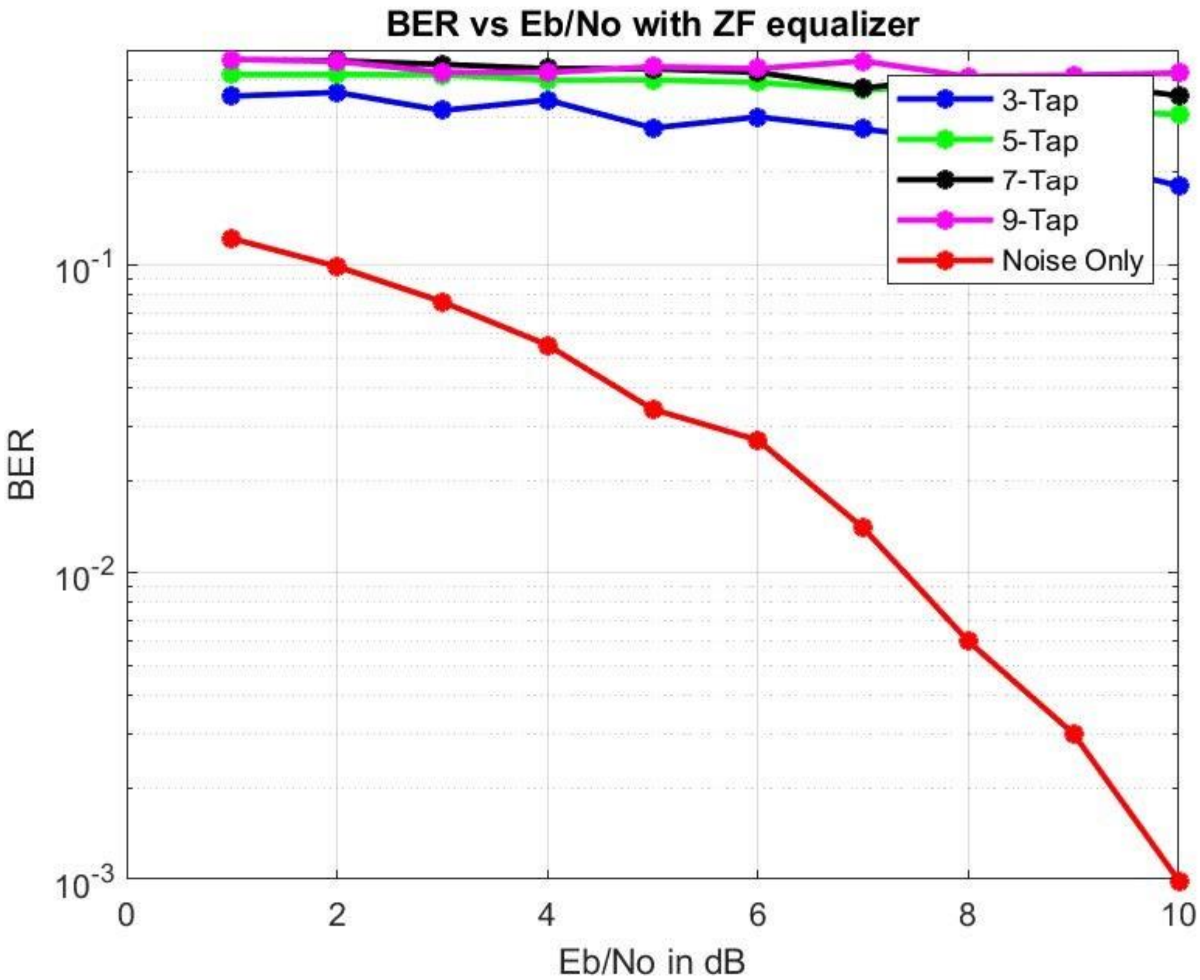
Characteristic	Eye Parameter	Sinc Pulse	Raised Cosine Pulse (Roll-off = 0.5)	Raised Cosine Pulse (Roll-off =1)
Noise (Noise Immunity)	Maximum height of the center of the eye. (Height of Eye at optimum sampling point) When this gap is wider, the noise immunity is higher.	Least height among the three. Lowest noise margin. (Least noise immunity.)	Medium height among the three. Medium noise margin. (Medium noise immunity.)	Highest height among the three. Highest noise margin. (Highest noise immunity.)

The other characteristics remain same as in Task 1.

3) Task 3

3.1 - 3.10)





3.11 Discrepancy between the AWGN channel BER and the ZF equalized multipath Channel

When comparing the performance of a channel with an impulse response that spreads the signal (such as a multipath channel) to that of a pure AWGN (Additive White Gaussian Noise) channel, several key differences exist.

In a pure AWGN channel, the bit error rate tends to be lower because the main source of degradation is Gaussian noise, which is easier to manage. In contrast, a channel with an impulse response that spreads the signal, such as a multipath channel, introduces Inter Symbol interference (ISI), causing overlapping of adjacent symbols and increasing the bit error rate.

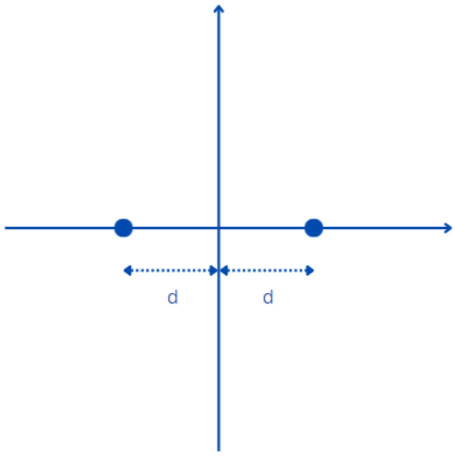
A zero-forcing equalizer is commonly used to counteract the effects of the channel's impulse response by inverting the channel's effect on the transmitted signal. However, this equalizer can amplify any noise present in the channel, leading to higher noise levels in the decoded signal. This amplified noise results in increased errors in the decoded signal, ultimately making the overall signal quality lower in a channel with an impulse response that spreads the signal compared to a pure AWGN channel.

3.12 BER performance if binary orthogonal signaling was used instead of BPSK

The distance between symbols in the signal space is a key factor influencing the bit error rate (BER) performance.

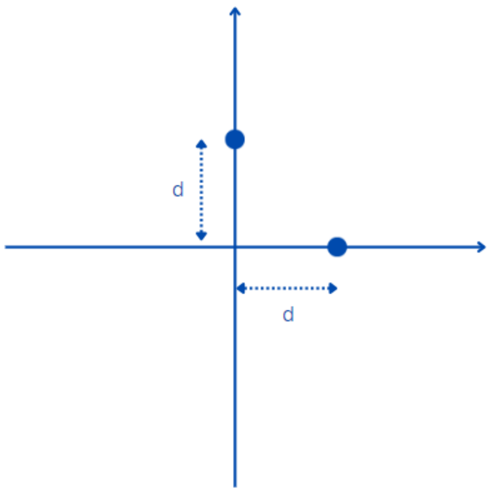
When comparing binary phase-shift keying (BPSK) with Binary orthogonal signaling, the distance between symbols is different.

In BPSK, the two symbols (representing binary 0 and 1) are encoded as phase shifts of 0 and 180 degrees, respectively. This results in the maximum possible distance between symbols in the signal space, which minimizes the likelihood of confusion between them and reduces the error rate.



(a) BPSK Constellation Diagram

On the other hand, Binary orthogonal signaling uses two orthogonal signals to represent binary 0 and 1. These signals occupy separate frequency bands and are chosen to be orthogonal to each other. However, the distance between the symbols in binary orthogonal signaling is smaller compared to BPSK. This smaller distance means the receiver is more likely to confuse the two signals, particularly in the presence of noise, resulting in a higher BER.



(b) Binary Orthogonal Singal Constellation Diagram

Therefore, while binary orthogonal signaling offers some advantages in terms of separating the signals and providing a measure of resistance to certain types of interference, the greater distance between symbols in BPSK generally leads to better BER performance. This makes BPSK more robust and reliable in terms of error rates, especially in channels where noise and interference are significant factors.

```matlab
% Simulation Assignment – Eye diagrams and Equalization
% Wathudura T.R. - 210682D
% Dodangoda D.K.S.J. - 210150V

% Task 1

clear all;
close all;
clc;

% System Parameters
BitLength = 10^3; % No of Bits Transmitted
SampleFreq = 10; % Sampling frequency (Hz)
time = -SampleFreq:1/SampleFreq:SampleFreq; % Time Array

% Generate the BPSK Signal
% Map 0 -> -1 and 1 -> 1 (0-phase and 180-phase)
BPSKSignal = 2*(rand(1,BitLength)>0.5)-1;
t = 0:1/SampleFreq:99/SampleFreq;
stem(t, BPSKSignal(1:100)); xlabel('Time'); ylabel('Amplitude');
title('BPSK Impulse Train');
axis([0 10 -1.2 1.2]); grid on;

% Sinc Pulse
Sinc_Num = sin(pi*time); % numerator sinc
Sinc_Den = (pi*time); % denominator sinc
Sinc_DenZero = find(abs(Sinc_Den) < 10^-10); % Find t=0 position
Sinc_Filt = Sinc_Num./Sinc_Den;
Sinc_Filt(Sinc_DenZero) = 1; % Define t=0 value
figure;
plot(time, Sinc_Filt);
title('Sinc Pulse');
xlabel('Time'); ylabel('Amplitude');
axis([-SampleFreq SampleFreq -0.5 1.2]); grid on

% Raised Cosine Pulse for 0.5 roll-off
roll_off = 0.5;
cos_Num = cos(roll_off*pi*time);
cos_Den = (1 - (2 * roll_off * time).^2);
cos_DenZero = abs(cos_Den)<10^-10;
RaisedCosine = cos_Num./cos_Den;
RaisedCosine(cos_DenZero) = pi/4;
RC_gamma5 = Sinc_Filt.*RaisedCosine; % Getting the complete raised cosine pulse
figure;
plot(time, RC_gamma5);
title('Raised Cosine Pulse for Gamma = 0.5');
xlabel('Time'); ylabel('Amplitude');
axis([-SampleFreq SampleFreq -0.5 1.2]); grid on

% Raised Cosine Pulse for 1 roll-off
roll_off = 1;
cos_Num = cos(roll_off * pi * time);
cos_Den = (1-(2 * roll_off * time).^2);
cos_DenZero = find(abs(cos_Den)<10^-20);
RaisedCosine = cos_Num./cos_Den;
RaisedCosine(cos_DenZero) = pi/4;
RC_gamma1 = Sinc_Filt.*RaisedCosine; % Get the complete raised cosine pulse
figure;
plot(time, RC_gamma1);
title('Raised Cosine Pulse for Gamma = 1');
xlabel('Time'); ylabel('Amplitude');
axis([-SampleFreq SampleFreq -0.5 1.2]); grid on
```

```
% Upsample the transmit sequence
BPSK_Upsample = [BPSKSignal;zeros(SampleFreq-1,length(BPSKSignal))]; % Upsample the BPSK to match the sampling frequency
BPSK_U = BPSK_Upsample(:).';
figure;
stem(t, BPSK_U(1:100)); xlabel('Time'); ylabel('Amplitude');
title('BPSK Impulse Train after upsampling');
axis([0 10 -1.2 1.2]); grid on;

% Pulse Shaped sequences
Conv_sincpulse = conv(BPSK_U, Sinc_Filt);
Conv_RCgamma5 = conv(BPSK_U,RC_gamma5);
Conv_RCgamma1 = conv(BPSK_U,RC_gamma1);


% Take only the first 10000 samples
Conv_sincpulse = Conv_sincpulse(1:10000);
Conv_RCgamma5 = Conv_RCgamma5(1:10000);
Conv_RCgamma1 = Conv_RCgamma1(1:10000);

% Reshape the sequences to build Eye Diagrams
Conv_sincpulse_reshape = reshape(Conv_sincpulse, SampleFreq*2, BitLength*SampleFreq/20).';
Conv_RCgamma5_reshape = reshape(Conv_RCgamma5,SampleFreq*2,BitLength*SampleFreq/20).';
Conv_RCgamma1_reshape = reshape(Conv_RCgamma1,SampleFreq*2,BitLength*SampleFreq/20).';

% Plot the Eye Diagrams
% Eye diagram for Sinc pulse
figure;
plot(0:1/SampleFreq:1.99, real(Conv_sincpulse_reshape).', 'b');
title('Eye diagram for Sinc pulse');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -2.4 2.2]);
grid on

% Eye diagram for RC pulse with Gamma = 0.5
figure;
plot(0:1/SampleFreq:1.99, Conv_RCgamma5_reshape.','b');
title('Eye diagram for RC pulse with Gamma = 0.5');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -2.5 2.5]);
grid on

% Eye diagram for RC pulse with Gamma = 1
figure;
plot(0:1/SampleFreq:1.99, Conv_RCgamma1_reshape.','b');
title('Eye diagram for RC pulse with Gamma = 1');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -1.5 1.5 ]);
grid on
```

```
% Simulation Assignment - Eye diagrams and Equalization
% Wathudura T.R. - 210682D
% Dodangoda D.K.S.J. - 210150V

% Task 2
% With Additive White Gaussian Noise (AWGN)

clear all;
close all;
clc;

% System Parameters
BitLength = 10^3; % No of Bits Transmitted
SampleFreq = 10; % Sampling frequency (Hz)
time = -SampleFreq:1/SampleFreq:SampleFreq; % Time Array
SNR_dB = 10;
NoisePower = 1./(10.^(0.1*SNR_dB)); % Noise Power (Eb = 1 in BPSK)

% Generate the BPSK Signal (With noise)
% Map 0 -> -1 and 1 -> 1 (0-phase and 180-phase)
BPSKSignal = 2*(rand(1,BitLength)>0.5)-1;
t = 0:1/SampleFreq:99/SampleFreq;

% Noise Array Generation using SNR = 10dB
Noise1D = normrnd (0 , sqrt(NoisePower/2), [1, BitLength]);
AWGN_TX = BPSKSignal + Noise1D;
figure;
stem(t, AWGN_TX(1:100)); xlabel('Time'); ylabel('Amplitude');
title('BPSK Impulse Train (With Noise)');
axis([0 10 -1.5 1.5]); grid on;

% Sinc Pulse
Sinc_Num = sin(pi*time); % numerator sinc
Sinc_Den = (pi*time); % denominator sinc
Sinc_DenZero = find(abs(Sinc_Den) < 10^-10); % Find t=0 position
Sinc_Filt = Sinc_Num./Sinc_Den;
Sinc_Filt(Sinc_DenZero) = 1; % Define t=0 value

% Raised Cosine Pulse for 0.5 roll-off
roll_off = 0.5;
cos_Num = cos(roll_off*pi*time);
cos_Den = (1 - (2 * roll_off * time).^2);
cos_DenZero = abs(cos_Den)<10^-10;
RaisedCosine = cos_Num./cos_Den;
RaisedCosine(cos_DenZero) = pi/4;
RC_gamma5 = Sinc_Filt.*RaisedCosine; % Getting the complete raised cosine pulse

% Raised Cosine Pulse for 1 roll-off
roll_off = 1;
cos_Num = cos(roll_off * pi * time);
cos_Den = (1-(2 * roll_off * time).^2);
cos_DenZero = find(abs(cos_Den)<10^-20);
RaisedCosine = cos_Num./cos_Den;
RaisedCosine(cos_DenZero) = pi/4;
RC_gamma1 = Sinc_Filt.*RaisedCosine; % Get the complete raised cosine pulse
```

```matlab
% Upsample the transmit sequence (With Noise)
AWGNTx_Upsample = [AWGN_TX;zeros(SampleFreq-1,length(BPSKSignal))];
AWGNTx_U = AWGNTx_Upsample(:);
figure;
stem(t, AWGNTx_U(1:100)); xlabel('Time'); ylabel('Amplitude');
title('Upsampled BPSK Impulse Train (With Noise)');
axis([0 10 -1.5 1.5]); grid on;

% Pulse Shaped sequences (With Noise)
Conv_sincnoise = conv(AWGNTx_U,Sinc_Filt);
Conv_RC5noise = conv(AWGNTx_U,RC_gamma5);
Conv_R1noise = conv(AWGNTx_U,RC_gamma1);

% Take only the first 10000 samples (With noise)
Conv_sincnoise = Conv_sincnoise(1:10000);
Conv_RC5noise = Conv_RC5noise(1:10000);
Conv_R1noise = Conv_R1noise(1:10000);

% Reshape the sequences to build Eye Diagrams (With Noise)
Conv_sincnoise_reshape = reshape(Conv_sincnoise, SampleFreq*2, BitLength*SampleFreq/20).';
Conv_RC5noise_reshape = reshape(Conv_RC5noise,SampleFreq*2,BitLength*SampleFreq/20).';
Conv_R1noise_reshape = reshape(Conv_R1noise,SampleFreq*2,BitLength*SampleFreq/20).';

% Plot the Eye Diagrams (With Noise)
% Eye diagram for Sinc pulse
figure;
plot(0:1/SampleFreq:1.99, Conv_sincnoise_reshape.', 'b');
title('Eye diagram for Sinc pulse (With Noise)');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -2.4 2.2]);
grid on

% Eye diagram for RC pulse with Gamma = 0.5
figure;
plot(0:1/SampleFreq:1.99, Conv_RC5noise_reshape.', 'b');
title('Eye diagram for RC pulse with Gamma = 0.5 (With Noise)');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -2.4 2.2]);
grid on

% Eye diagram for RC pulse with Gamma = 1
figure;
plot(0:1/SampleFreq:1.99, Conv_R1noise_reshape.', 'b');
title('Eye diagram for RC pulse with Gamma = 1 (With Noise)');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -2.4 2.2]);
grid on
```

```
% Simulation Assignment - Eye diagrams and Equalization
% Wathudura T.R. - 210682D
% Dodangoda D.K.S.J. - 210150V

% Task 3
% Zero-forcing (ZF) equalizer for a 3-tap Multipath Channel

clear all;
close all;
clc;

% System Parameters
BitLength = 10^3;

% Generate a random binary sequence
BinarySequence = randi([0,1],1,BitLength);

% 2-PAM Signal
PAMsignal = 2 * BinarySequence - 1;
figure
stem(PAMsignal,'Marker','none');
title('2-PAM Signal');
grid


% Generate the received signal samples by convolving with channel response
h = [0.3 0.7 0.4];
ReceivedSignal = conv(PAMsignal,h);

BitErrors = zeros(10,5);
value = zeros(1,BitLength);

for n = 1:4
    % The matrix to obtain the ZF equalizer
    SignalWithNoise = awgn(ReceivedSignal,0);

    ShiftedMatrix = toeplitz([h([2:end]) zeros(1, 2 * n -1)],[h([2:-1:1]) zeros(1,2 * n - 1)]);
    RequiredMatrix = zeros(1,2 * n + 1);
    RequiredMatrix(n + 1) = 1;
    ZF_equalizerCoefficients = [inv(ShiftedMatrix)*RequiredMatrix.'].';

    for k = 1:10
        % Add White Gaussian Noise
        SignalWithNoise = awgn(ReceivedSignal,k);

        % Convolve with filter to get response
        y_filtered = conv(SignalWithNoise,ZF_equalizerCoefficients);
        y_filtered = y_filtered(n+2:n+1+BitLength);% Consider the appropriate samples
            for i = 1:length(PAMsignal)
                if (y_filtered(i) > 0)
                    value(i) = 1;
                else
                    value(i) = 0;
                end
            end
        NoOfBitErrors = sum(value ~= BinarySequence);
```

```
        BitErrors(k,n) = NoOfBitErrors / BitLength;
    end
end

% AWGN Channel (Only noise)
for k = 1:10
    SignalWithNoise = awgn(PAMsignal,k);
     for i = 1:length(PAMsignal)
        if (SignalWithNoise(i) > 0)
            value(i) = 1;
        else
            value(i) = 0;
        end
     end
     NoOfBitErrors = sum(value ~= BinarySequence);
     BitErrors(k,5) = NoOfBitErrors / BitLength;
end

% BER vs SNR graphs
figure
semilogy([1:10],BitErrors(:,1),'b*-','Linewidth',1.8);
hold on
semilogy([1:10],BitErrors(:,2),'g*-','Linewidth',1.8);
semilogy([1:10],BitErrors(:,3),'k*-','Linewidth',1.8);
semilogy([1:10],BitErrors(:,4),'m*-','Linewidth',1.8);
semilogy([1:10],BitErrors(:,5),'r*-','Linewidth',1.8);
axis([0 10 10^-3 0.5])
grid on
legend('3-Tap', '5-Tap','7-Tap','9-Tap','Noise Only');
xlabel('Eb/No in dB');
ylabel('BER');
title('BER vs Eb/No with ZF equalizer');
```