Department of Electronic & Telecommunication Engineering

University of Moratuwa

EN2074 - Communication Systems Engineering

Eye diagrams and Equalization

Name	Index Number
Jayasuriya C.L.	200262G
Kariyawasam K.K.D.	200289U

Contents

1 Introduction

This assignment focuses on digital communication systems using baseband 2-PAM signaling. We explore the effects of pulse shaping filters, additive white Gaussian noise (AWGN), and zero-forcing equalization for a multipath channel.

In Task 1, we generate BPSK symbols and obtain the transmit signal by convolving them with different pulse shaping filters. The eye diagram of the transmit signal is then generated to assess signal quality and robustness of the communication scheme.

Task 2 extends Task 1 by introducing additive white Gaussian noide (AWGN) to evaluate system robustness against noise, sampling time, and synchronization errors.

Task 3 centers on designing a zero-forcing equalizer for a 3-tap multi-path channel. We compute the equalization filters, calculate the bit error rate (BER), and compare it with an AWGN channel.

Throughout the assignment, eye diagrams serve as valuable graphical tools for analyzing the quality of a signal in digital communication systems. They provide visual representations that enable a comprehensive assessment of various factors affecting the signal's performance, including inter-symbol interference (ISI), additive noise immunity, and jitter effects.

2 Simulation Results

For the simulation purpose in this assignment, we used **Matlab** software. **Matlab** is a powerful programming and computing tool which is widely used in various fields. Its rich set of functions and tools make it suitable for implementing and evaluating different communication techniques.

2.1 Task 1

2.1.1 Impulse train representing BPSK symbols

As the first step a random bit sequence was generated using the *randi* function. Length of the bit sequence was set to 1000 bits. Then that bit sequence mapped into BPSK symbols.

Mapping was done according to the following rule.

$$\text{Amplitude} = \begin{cases} 1, & \text{if bit value = 1} \\ \text{-1}, & \text{if bit value = 0} \end{cases}$$

Then the impulse train was generated. Figures of plots of original bit stream and BPSK impulse train are shown below.

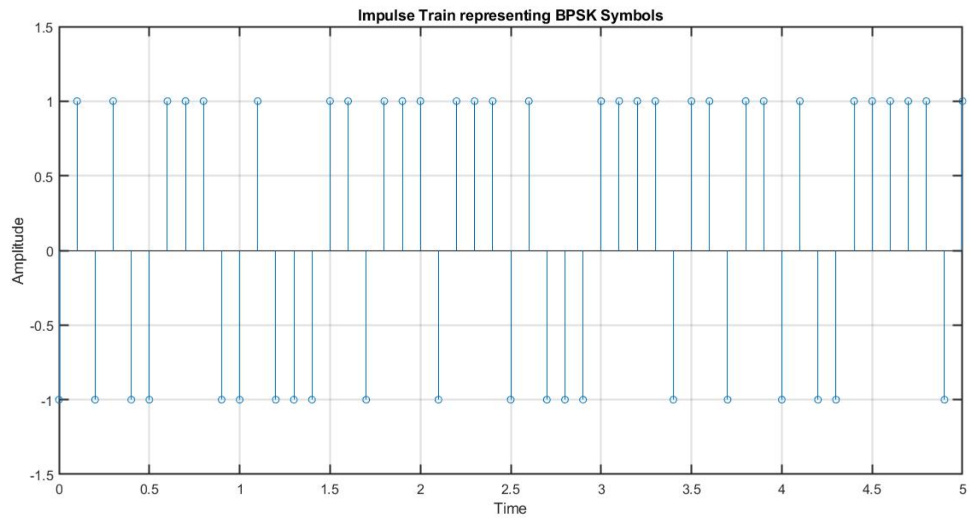


Figure 1: *BPSK Impulse Train*

2.1.2 Sinc pulse shaping filter

In this step, we obtain the transmit signal by convolving the BPSK impulse train, which represents the BPSK symbols, with a pulse shaping filter where the impulse response is a sinc function. To accomplish this, we first generate the sinc pulse in **MATLAB** using the **sinc** function. The **sinc** function creates a sinc waveform, which is a basic building block for many pulse shaping filters. The sinc waveform has a central peak surrounded by smaller lobes. Next, we perform pulse shaping by convolving the impulse train with the sinc pulse. By convolving the impulse train with the sinc pulse, we obtain the transmit signal.

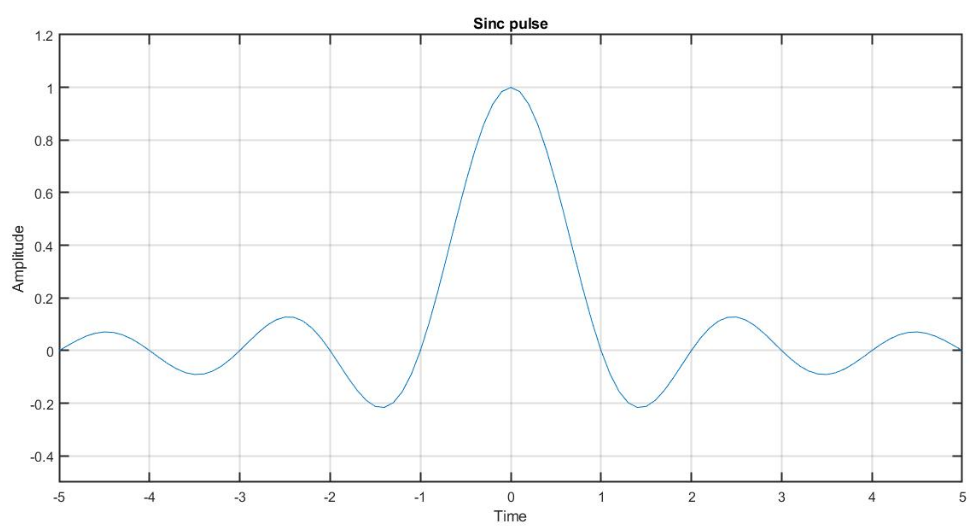


Figure 2: *Sinc pulse*

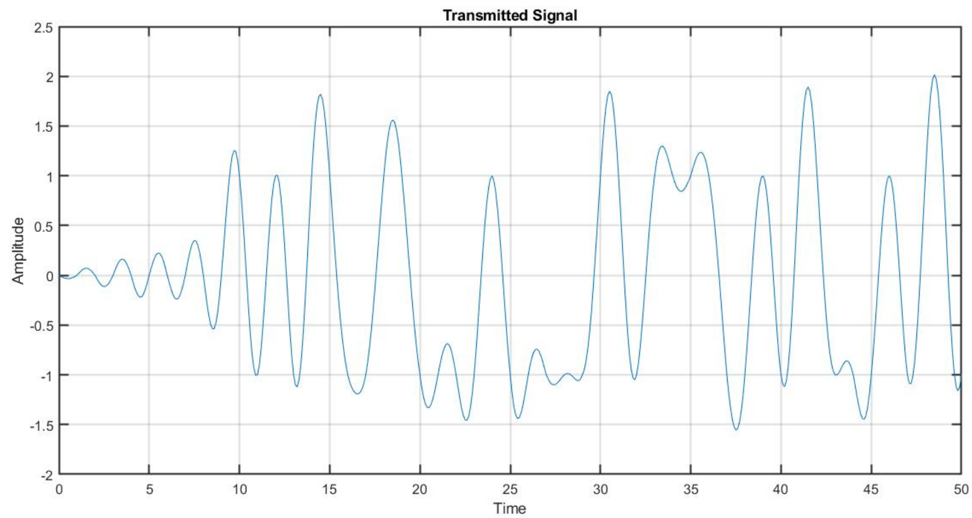


Figure 3: *Transmit signal*

2.1.3 Eye diagram of the transmit signal (sinc pulse shaping)

The eye diagram is a graphical representation of the signal's behavior over multiple symbol intervals. It provides insights into the signal's timing and amplitude variations, allowing us to assess the quality of the transmitted signal. To generate the eye diagram, we utilize the **eyediagram** function in **MATLAB**. This function takes the transmit signal as input and plots eye diagram.

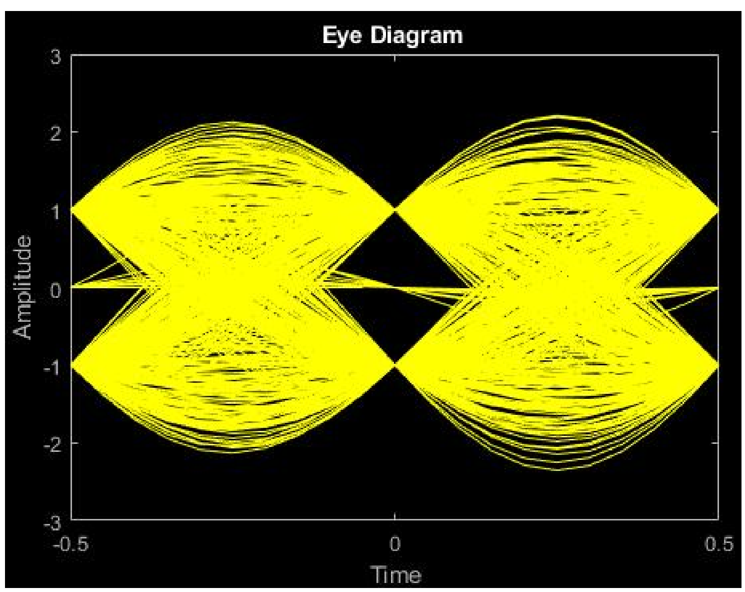


Figure 4: *Eye diagram for transmit signal*

2.1.4 Raised cosine pulse shaping filters

In this task, we developed a dedicated **MATLAB** function called **raised_cosine_filter** (See A.2) to design raised cosine pulse shaping filters. This function is designed to generate the raised cosine pulse based on the provided roll-off factor and sampling frequency. Then transmit signal was generated by convolving above raised cosine and the BPSK representing impulse train.

For Roll-off factor = 0.5

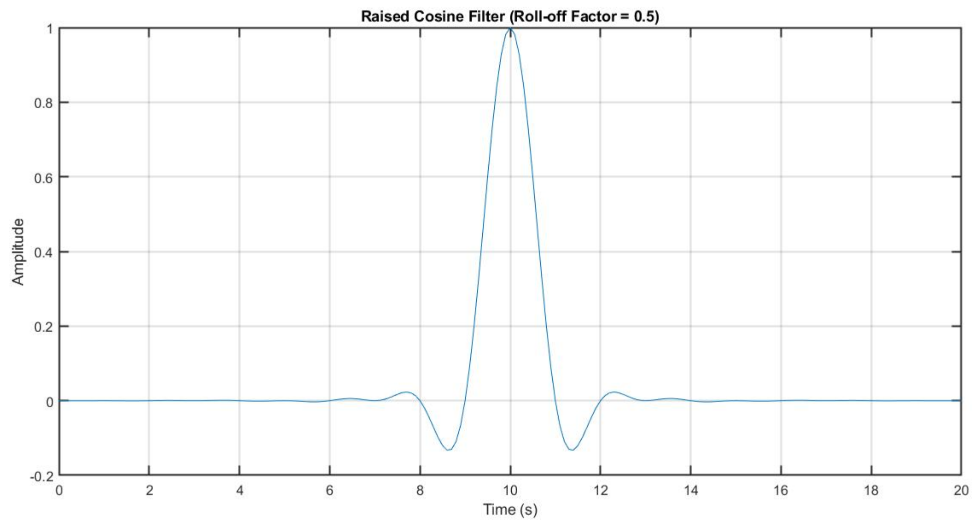


Figure 5: *Raised cosine pulse shaping filter with roll-off = 0.5*

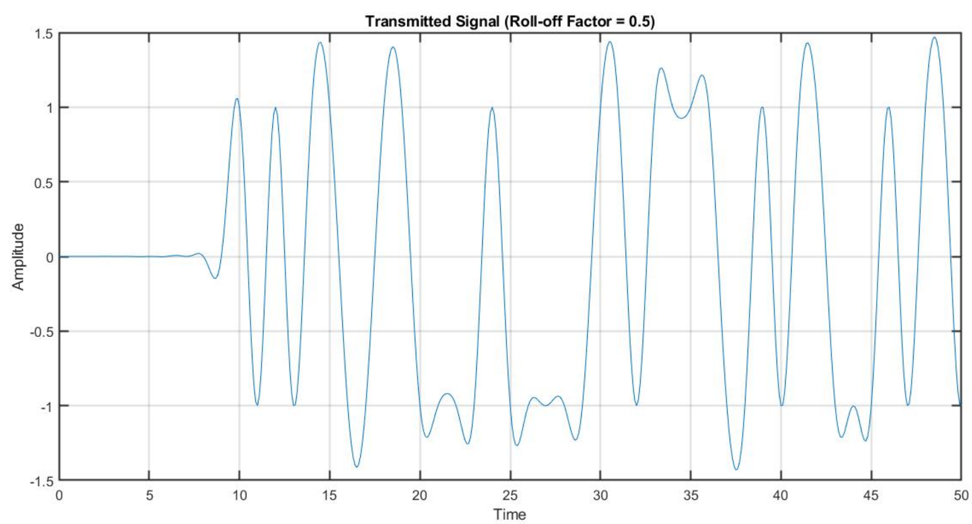


Figure 6: *Transmit signal*

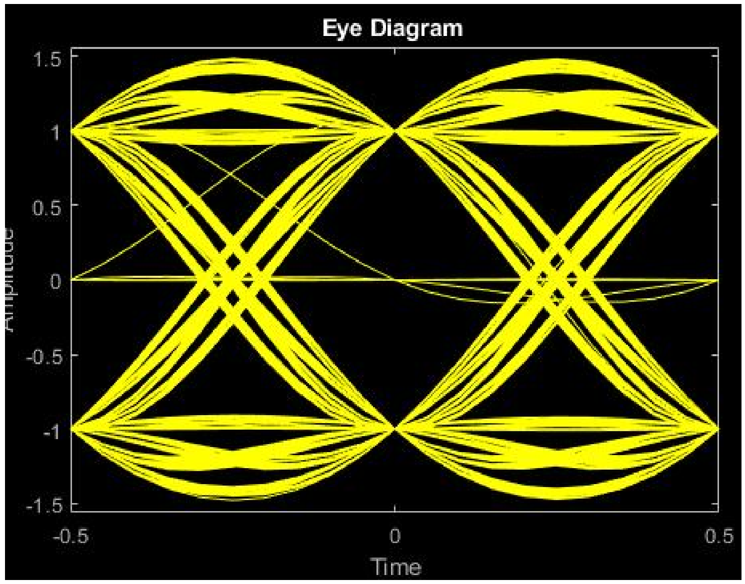


Figure 7: *Eye diagram for transmit signal*

For Roll-off factor = 1

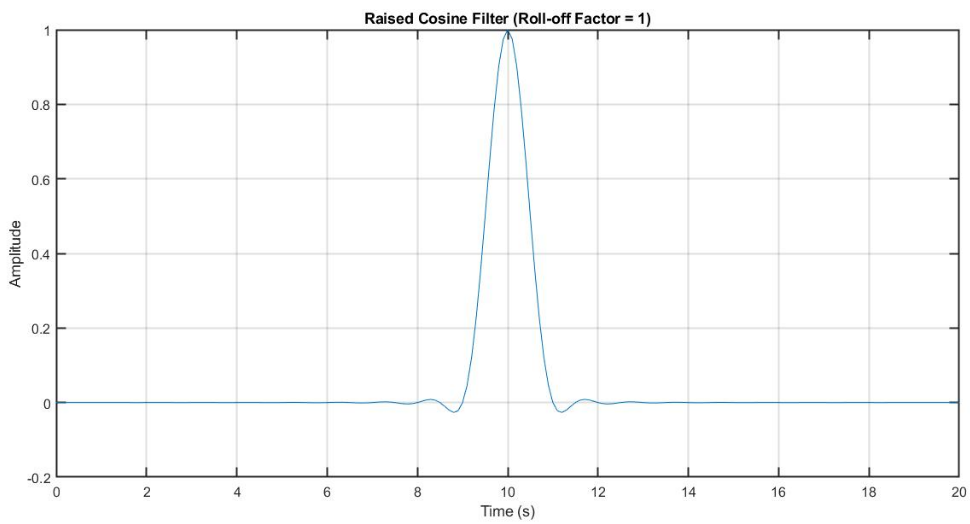


Figure 8: *Raised cosine pulse shaping filter with roll-off = 0.5*

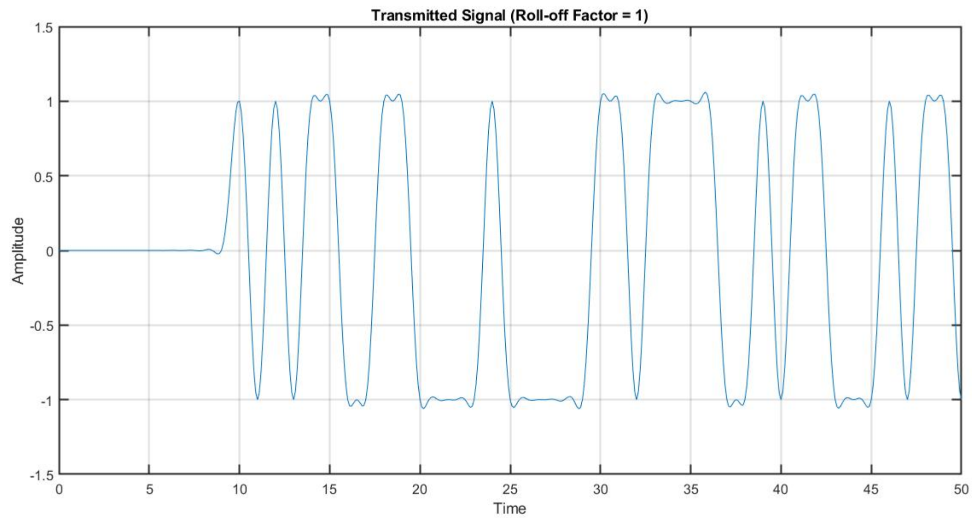


Figure 9: *Transmit signal*

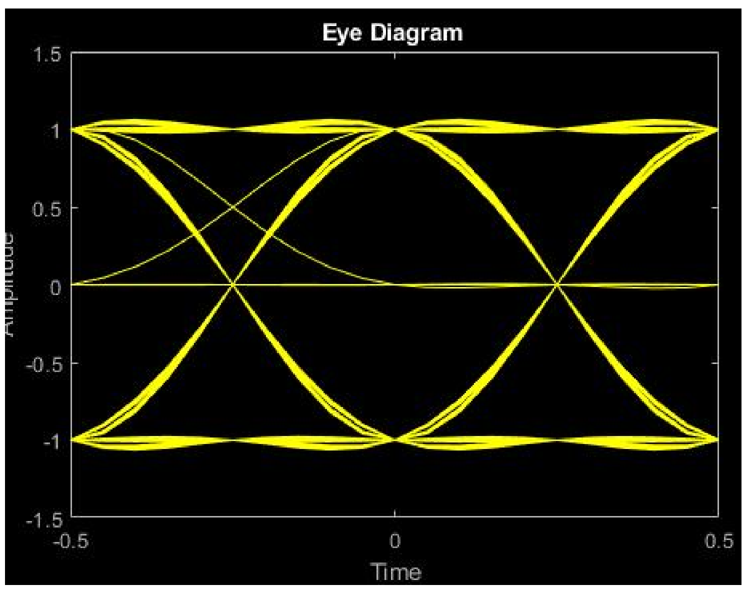


Figure 10: *Eye diagram for transmit signal*

2.1.5 Comparison of robustness of the system

We can get several information by looking at the eye diagrams.

Eye-diagram feature	Information that can be obtained
Eye height	Noise added to the signal
Eye overshoot/undershoot	Malformations in the signal path due to inteferences
Eye width	Timing synchronization & jitter effects
Eye closure	Intersymbol interference, additive noise

Table 1: *Eye diagram features and their measurements.*

When we examine the 3 eye diagrams (4, 7 and 10), we notice that they all have the same noise margin at the best sampling point, which is the highest vertical part of the eye opening. However, the error-free sampling region, which is the longest horizontal part of the eye opening, varies between the systems. This causes differences in the slope of the eye pattern and the width of the eye corners.

The system with a Raised Cosine filter and $\alpha = 1$ has the largest error-free sampling region, the smallest eye corner width, and the gentlest slope. This means that it is unlikely to have inter-symbol interference (ISI) and is less sensitive to timing errors and jitter. It can be considered the most robust system out of the three when it comes to dealing with noise, sampling time, and synchronization errors.

On the other hand, the system with a Raised Cosine filter and $\alpha = 0.5$ has the second largest error-free sampling region, the second smallest eye corner width, and the second gentlest slope. While it is not as robust as the first system, it still offers better resilience compared to the system using a sinc function.

2.2 Task 2

AWGN stands for Additive White Gaussian Noise. It is a commonly used model for representing the noise present in communication systems and signal processing.

Task 2 extends Task 1 by introducing additive white Gaussian noide (AWGN) to evaluate system robustness against noise, sampling time, and synchronization errors.

After adding the additive white Gaussian noise (AWGN) to the transmitted signal following results are obtained.

2.2.1 Transmit signal with noise (Sinc pulse shaping)

When an AWGN (Additive White Gaussian Noise) is added to the transmit signal, the quality and reliability of the received signal can be affected. The AWGN introduces random variations and disturbances to the transmitted signal.

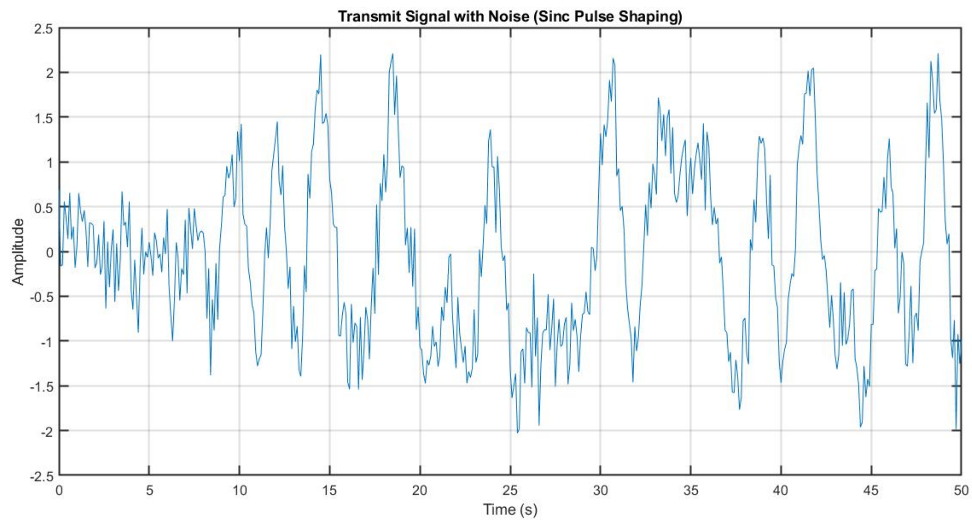


Figure 11: *Transmit signal with noise*

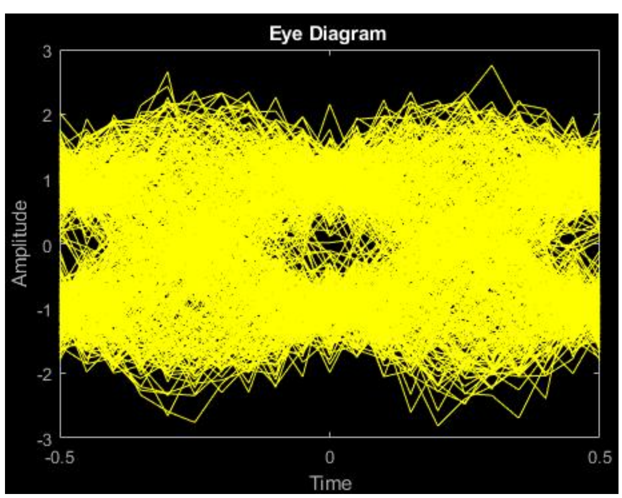


Figure 12: *Eye diagram when noise added*

2.2.2 Transmit signal with noise (Raised cosine with roll-off = 0.5)

When AWGN (Additive White Gaussian Noise) is added to the transmit signal with a raised cosine filter pulse shaping using a roll-off factor of 0.5, the following results can be observed.

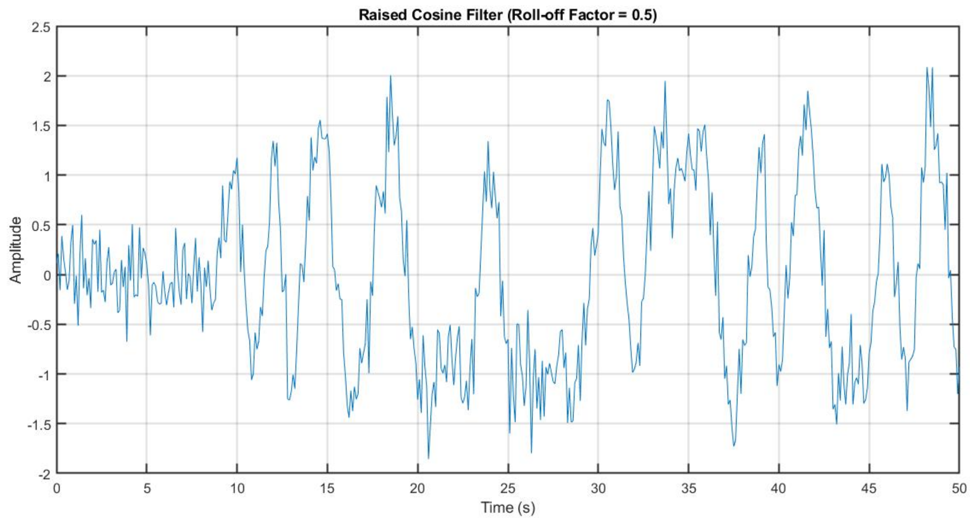


Figure 13: *Transmit signal with noise*

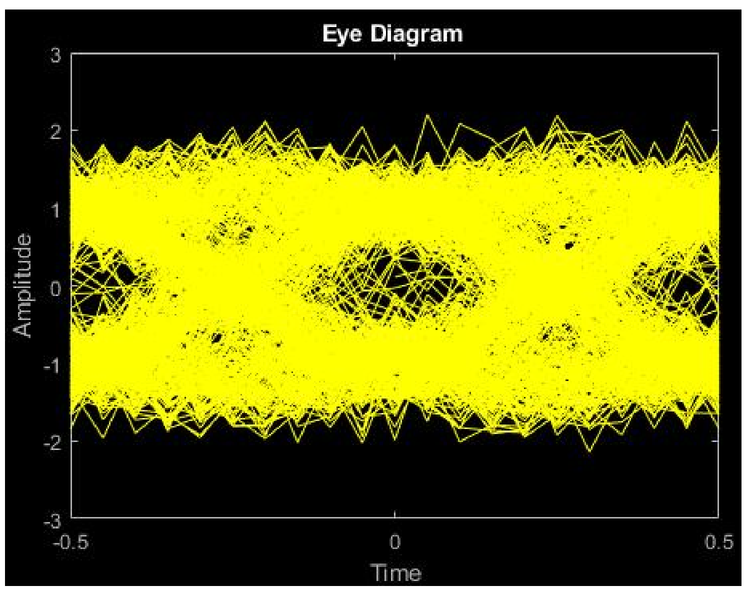


Figure 14: *Eye diagram when noise added*

2.2.3 Transmit signal with noise (Raised cosine with roll-off = 1)

The addition of AWGN (Additive White Gaussian Noise) to the transmit signal, when employing a raised cosine filter pulse shaping with a roll-off factor of 1, yields the following results.

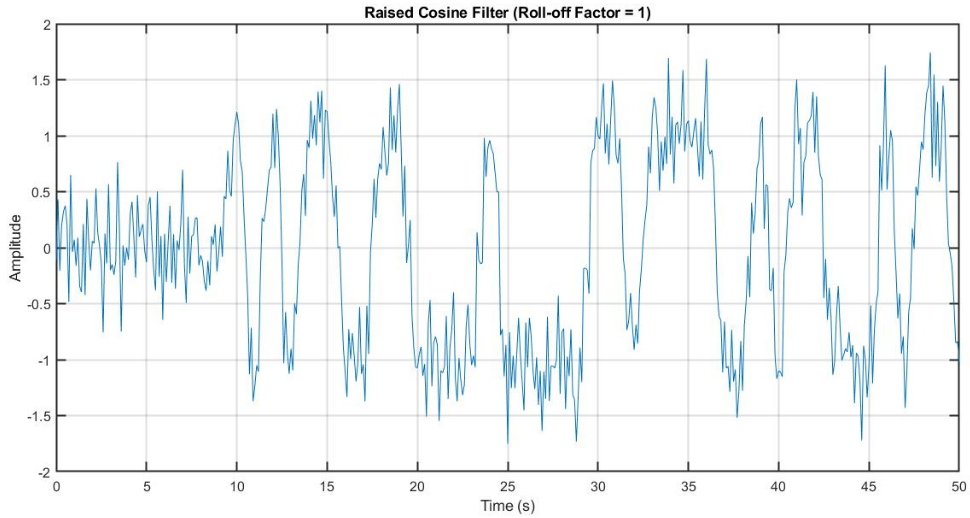


Figure 15: *Transmit signal with noise*

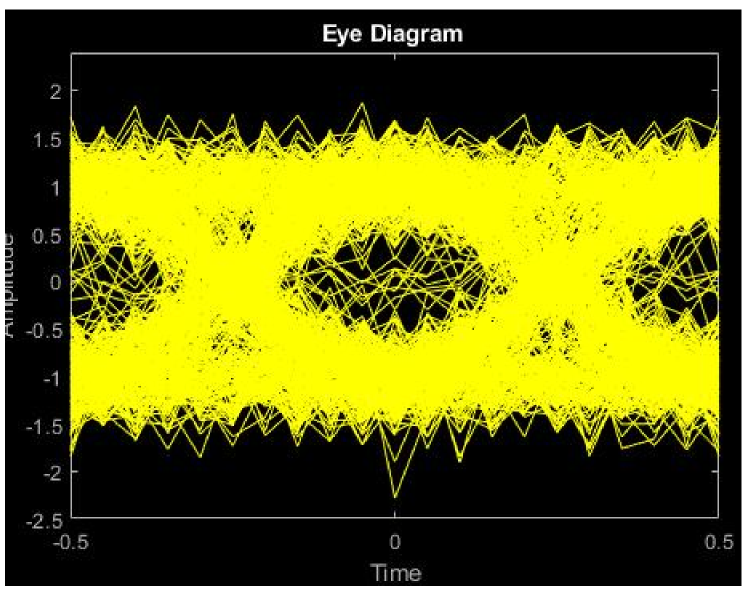


Figure 16: *Eye diagram when noise added*

2.2.4 Comparison of robustness of the system

Due to the presence of Additive White Gaussian Noise, it can be observed that all three systems experience a reduction in both vertical and horizontal eye openings compared to Task 1. Additionally, the width of the eye corners has significantly increased.

Upon examining the eye diagrams, it is still evident that the system employing the Raised Cosine function with an alpha value of 1 exhibits the highest level of noise margin. It also possesses the narrowest width of the eye corner and a steep slope, indicating a robust performance in the presence of noise, sampling time variations, and synchronization errors. Therefore, this system remains the most resilient.

The second most robust system is the one utilizing the Raised Cosine function with an alpha value of 0.5. It offers a high noise margin and a substantial error-free sampling region, accompanied by a narrower width of the eye corner and a lower slope compared to the system employing the Sinc function.

2.3 Task 3

Task 3 centers on designing a zero-forcing equalizer for a 3-tap multi-path channel. We compute the equalization filters, calculate the bit error rate (BER), and compare it with an AWGN channel.

2.3.1 Bit error rate calculation

$\frac{E_b}{N_0}$	3-Tap	5-Tap	7-Tap	9-Tap
0	0.2240	0.2500	0.2580	0.2602
1	0.1988	0.2246	0.2326	0.2352
2	0.1736	0.1988	0.2066	0.2091
3	0.1492	0.1720	0.1800	0.1825
4	0.1251	0.1452	0.1520	0.1543
5	0.1033	0.1193	0.1251	0.1271
6	0.0816	0.0936	0.0987	0.1005
7	0.0638	0.0702	0.0739	0.0752
8	0.0480	0.0508	0.0527	0.0535
9	0.0358	0.0350	0.0355	0.0360
10	0.0251	0.0219	0.0216	0.0215

Table 2: *Bit Error Rate (BER)*

2.3.2 BER for all tap settings and BER for an additive white Gaussian noise (AWGN) channel

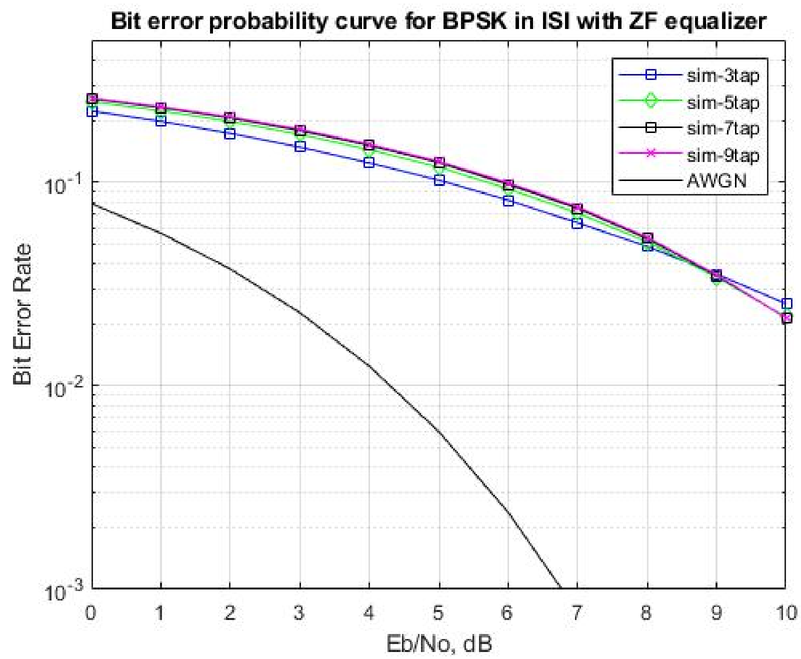


Figure 17: *Bit error probability curve*

2.3.3 Discrepancy between the AWGN channel BER and the ZF equalized multipath channel

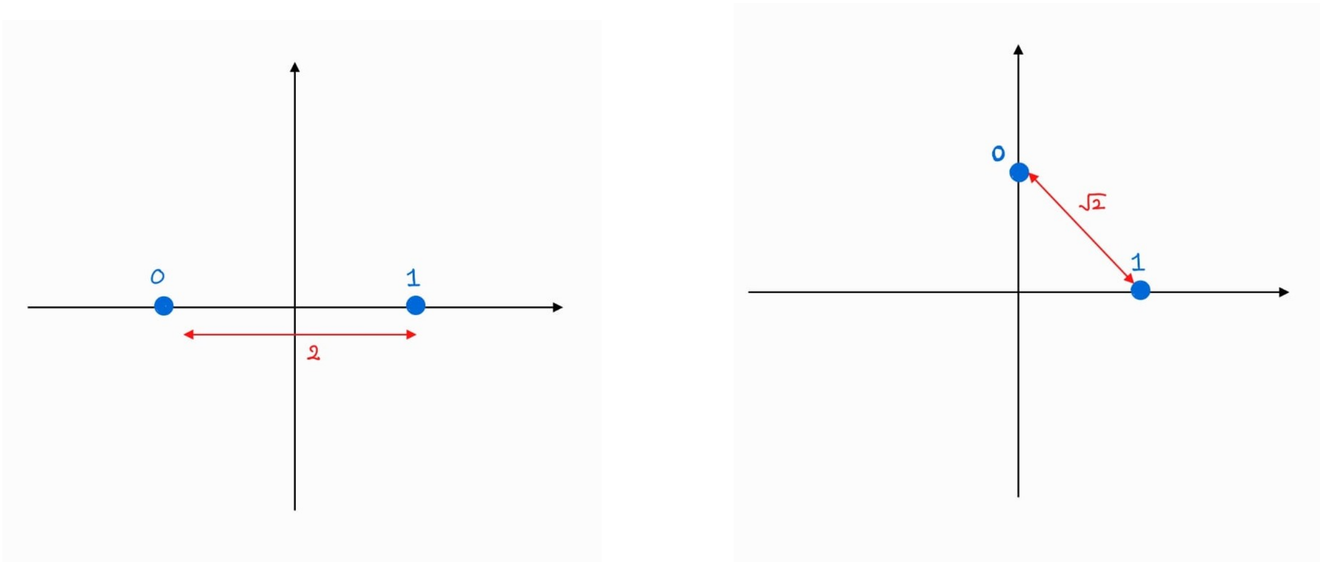
In an AWGN channel, the main source of impairment is the additive white Gaussian noise, which is assumed to be uncorrelated and has a constant power spectral density.

On the other hand, a ZF equalizer is designed to mitigate the effects of multipath propagation. It aims to remove the interference caused by multiple paths with different delays and attenuations. The ZF equalizer tries to "invert" the channel response, effectively canceling out the multipath distortion.

However, the ZF equalizer's performance is affected by **noise amplification**. The noise amplification occurs because the equalizer aims to cancel out the interference, but in doing so, it may amplify the noise as well. This noise amplification can increase the BER compared to an ideal AWGN channel.

Therefore, the BER in a ZF equalized multi-path channel is typically higher than that in an AWGN channel.

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



(a) *BPSK* **(b)** *Binary orthogonal BPSK*

Figure 18: *BPSK diagrams*

The Bit Error Rate (BER) performance of BPSK is superior to that of binary orthogonal signaling due to the larger distance between the BPSK symbols. In BPSK, the symbols are represented by +1 and -1, resulting in a distance of 2 between them. On the other hand, binary orthogonal signaling involves a phase difference of $\frac{\pi}{2}$ between the symbols, leading to a minimum distance of $\sqrt{2}$. Consequently, BPSK exhibits better performance since the larger symbol distance reduces the likelihood of bit errors.

In summary, the advantage of BPSK over binary orthogonal signaling can be attributed to the increased distance between symbols, resulting in fewer errors.

References

[1] *Eye pattern Wikipedia*. URL: https://en.wikipedia.org/wiki/Eye_pattern.

[2] *Eye diagram with raised cosine filters*. URL: https://dsplog.com/2008/05/01/eye-diagram-plot-matlab-raised-cosine-filter/.

[3] *AWGN - Wikipedia*. URL: https://en.wikipedia.org/wiki/Additive_white_Gaussian_noise#:~:text=Additive%20white%20Gaussian%20noise%20(AWGN,intrinsic%20to%20the%20information%20system..

A Appendix

A.1 Matlab code for Task 1 and Task 2

```
close all; clc; clear;

%% Task I

N = 1000;                  % Number of bits
fs = 10;                   % Sampling frequency
Tb = 1/fs;                 % Bit duration
dt = 0.001;
t = 0:1/fs:999/fs;            %time vector
SNR_dB = 10;
NoisePower = 1./(10.^(0.1*SNR_dB));

%% Step 1
% Generating BPSK symbols
Bit_Stream = rand(1,N)>0.5;
BPSK_Symbols = Bit_Stream* 2 - 1;   % Mapping to BPSK symbols  0 -> -1 and 1 -> 1

%Plotting the BPSK impulse train
figure;
stem(t, BPSK_Symbols);
xlabel('Time');
ylabel('Amplitude');
title('Impulse Train representing BPSK Symbols');
ylim([-1.5 1.5])
xlim([0 5])
grid on;
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;
%% Step 2
t = -fs:1/fs:fs;             %Updating the time vector
Sinc_Filter = sinc(t);       %Defining sinc pulse shaping filter

%Plotting sinc pulse
figure
plot(t, Sinc_Filter);
xlabel('Time');
ylabel('Amplitude');
title('Sinc pulse');
ylim([-0.5 1.2])
xlim([-5 5])
grid on;
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

% Upsampling the BPSK sequence
t = 0:Tb:99*Tb;                        %Updating the time vector
BPSK_Symbols = upsample(BPSK_Symbols, fs);

figure;
stem(t, BPSK_Symbols(1:100));
```

```
xlabel('Time');
ylabel('Amplitude');
title('Upsampled BPSK Impulse Train');
axis([0 10 -1.2 1.2]);
grid on;
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

% Generating transmit signal by convolving with sinc pulse
Tx_Signal = conv(BPSK_Symbols, Sinc_Filter);
t = 0:Tb:(length(Tx_Signal)-1)*Tb;

%Plotting the transmit signal
figure
plot(t, Tx_Signal);
xlabel('Time');
ylabel('Amplitude');
title('Transmitted Signal');
xlim([0 50])
grid on;
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

%% Step 3
%Generating the eye diagram
eyediagram(Tx_Signal, 2*fs);

%% Step 4
%Defining raised cosine filter with roll-off = 0.5
rcos_alpha5 = raised_cosine_filter(0.5, fs);
rcos_alpha1 = raised_cosine_filter(1, fs);

% Generating transmit signal by convolving with raised cosine
Tranmit_Signal1 = conv(BPSK_Symbols, rcos_alpha5);
Tranmit_Signal2 = conv(BPSK_Symbols, rcos_alpha1);

% Plotting
figure;
t = 0:Tb:(length(rcos_alpha5)-1)*Tb;
plot(t, rcos_alpha5);
grid on;
title('Raised Cosine Filter (Roll-off Factor = 0.5)');
xlabel('Time (s)');
ylabel('Amplitude');
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

% Plotting
figure;
t = 0:Tb:(length(rcos_alpha1)-1)*Tb;
plot(t, rcos_alpha1);
grid on;
title('Raised Cosine Filter (Roll-off Factor = 1)');
```

```
xlabel('Time (s)');
ylabel('Amplitude');
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

%Plotting the transmit signals
figure;                                       %with roll-off = 0.5
t = 0:Tb:(length(Tranmit_Signal1)-1)*Tb;
plot(t, Tranmit_Signal1);
xlabel('Time');
ylabel('Amplitude');
title('Transmitted Signal (Roll-off Factor = 0.5)');
xlim([0 50])
grid on;
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

figure;                                      %with roll-off = 1
t = 0:Tb:(length(Tranmit_Signal2)-1)*Tb;
plot(t, Tranmit_Signal2);
xlabel('Time');
ylabel('Amplitude');
title('Transmitted Signal (Roll-off Factor = 1)');
xlim([0 50])
grid on;
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

%Generating eye diagrams
eyediagram(Tranmit_Signal1, 2*fs);          %For roll-off = 0.5
eyediagram(Tranmit_Signal2, 2*fs);          %For roll-off = 1

%% Task 2

%% Adding Noise
AWGN_TX_Sinc = awgn(Tx_Signal,SNR_dB,'measured');
AWGN_TX_roll5 = awgn(Tranmit_Signal1,SNR_dB,'measured');
AWGN_TX_roll1 = awgn(Tranmit_Signal2,SNR_dB,'measured');

%Plotting the figures
figure;
t = 0:Tb:(length(AWGN_TX_Sinc)-1)*Tb;
plot(t, AWGN_TX_Sinc);
grid on;
title('Transmit Signal with Noise (Sinc Pulse Shaping)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([0 50]);
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

figure;
```

```
t = 0:Tb:(length(AWGN_TX_roll5)-1)*Tb;
plot(t, AWGN_TX_roll5);
grid on;
title('Raised Cosine Filter (Roll-off Factor = 0.5)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([0 50]);
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

figure;
t = 0:Tb:(length(AWGN_TX_roll1)-1)*Tb;
plot(t, AWGN_TX_roll1);
grid on;
title('Raised Cosine Filter (Roll-off Factor = 1)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([0 50]);
ax = gca;
ax.LineWidth = 1.5;
ax.FontSize = 12;

% eye diagrams with noise
eyediagram(AWGN_TX_Sinc, 2*fs);            %For sinc pulse
eyediagram(AWGN_TX_roll5, 2*fs);           %For roll-off = 0.5
eyediagram(AWGN_TX_roll1, 2*fs);           %For roll-off = 1
```

A.2 Raised cosine filter function

```
function h = raised_cosine_filter(alpha, Fs)
    % Define time vector
    t = -Fs:1/Fs:Fs;

    % Generate sinc filter
    sinc_filter = sinc(t);

    % Compute cosine component
    cos_num = cos(alpha*pi*t);
    cos_den = (1 - (2*alpha*t).^2);
    cos_den_zero = find(abs(cos_den) < 10^-10);
    cos_op = cos_num./cos_den;
    cos_op(cos_den_zero) = pi/4;

    % Compute raised cosine filter
    h = sinc_filter .* cos_op;
end
```

A.3 Matlab code for Task 3

```
clear; clc;
N = 10^6;                 % Number of bits
SNR = [0:10];             % Eb/N0 values
K = 4;                    % Number of tap settings

for i = 1:length(SNR)
   Bit_Stream = rand(1,N)>0.5;            % Generating random binary sequence
   BPSK_Symbols = 2*Bit_Stream-1;         % Mapping to BPSK symbols  0 -> -1 and 1 -> 1

   % Defining multipath channel impulse response
   h = [0.3 0.9 0.4];

   %Generating received signal by convolving
   Received_Signal = conv(BPSK_Symbols,h);

   % Adding White Gaussian noise
   Noise = 1/sqrt(2)*[randn(1,N+length(h)-1) + 1i*randn(1,N+length(h)-1)];
   Signal_With_Noise = Received_Signal + 10^(-SNR(i)/20)*Noise;                % Noise
       addition

   for k = 1:K
     L  = length(h);
     Toeplitz_Matrix = toeplitz([h([2:end]) zeros(1,2*k+1-L+1)], [ h([2:-1:1]) zeros
         (1,2*k+1-L+1) ]);
     d  = zeros(1,2*k+1);
     d(k+1) = 1;
     Coeff  = [inv(Toeplitz_Matrix)*d.'].';

     % Matched filter
     yFilt = conv(Signal_With_Noise,Coeff);
     yFilt = yFilt(k+2:end);
     yFilt = conv(yFilt,ones(1,1));          % Performing convolution
     ySamp = yFilt(1:1:N);                   % Sampling

     % receiver - hard decision decoding
     E_Bits = real(ySamp)>0;

     % Counting number of bit errors
     Num_Errors(k,i) = size(find([Bit_Stream- E_Bits]),2);

   end
end
length(Bit_Stream)
disp(Num_Errors);

simBer = Num_Errors/N; % simulated ber
theoryBer = 0.5*erfc(sqrt(10.^(SNR/10))); % theoretical ber

% plotting
close all
figure
semilogy(SNR,simBer(1,:),'bs-')
hold on
semilogy(SNR,simBer(2,:),'gd-')
```

```
semilogy(SNR,simBer(3,:),'ks-')
semilogy(SNR,simBer(4,:),'mx-')
semilogy(SNR, theoryBer, 'k-')
axis([0 10 10^-3 0.5])
grid on
legend('sim-3tap', 'sim-5tap','sim-7tap','sim-9tap','AWGN');
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');
title('Bit error probability curve for BPSK in ISI with ZF equalizer');
```