Department of Electronic & Telecommunication Engineering
University of Moratuwa, Sri Lanka.

Simulation Assignment

Eye Diagrams and Equalization

210594J Senavirathne I.U.B
210609M Silva M.K.Y.U.N.

Submitted in partial fulfillment of the requirements for the module
EN 2074 Communication Systems Engineering

28/04/2024

Contents

1 Introduction

This report discusses and analyzes the performance of digital communication systems in various pulse shaping methodologies and environments. Further it explores the Impact of Additive White Gaussian Noise (AWGN) and Zero Forcing (ZF) Equalizer for multipath channels.

1.1 Pulse Amplitude Modulation (PAM)

Pulse Amplitude Modulation is a scheme in which a set of discrete amplitudes of a pulse train is chosen to represent a set of symbols made of bits in a bit stream.

In this assignment, we are simulating a baseband 2-PAM system, where the system contains two symbols +1 and -1, with each symbol representing a single bit ($\log_2(2) = 1$). This is similar to Binary Phase Shift Keying (BPSK), since both will result in a similar modulation.

$$0 \rightarrow -1 \quad (180^\circ \text{ phase})$$
$$1 \rightarrow +1 \quad (0^\circ \text{ phase})$$

1.2 Pulse Shaping

Digital signals are generally transmitted in a sequence of pulses. The shape of these pulses can change due to various reasons such as the bandwidth limitations, frequency and noise. These pulses should be in accordance with the Nyquist's First Criterion.

Rectangular Pulses have zero inter-symbol interference. However in the frequency domain this requires an infinite bandwidth. **Sinc pulses** which satisfies the Nyquist's Criterion are time unlimited.

As a solution for these problems, a scheme of modulation with **Raised Cosine Pulses** can be used. These pulses are band limited, and highly time limited. They compromise between the bandwidth and the inter-symbol interference (ISI). By controlling the roll off factor of the pulse, we can control the excess bandwidth.

1.3 Eye Diagrams

Eye diagrams are a tool that is used to analyse communication system, by superimposing multiple successive symbols. The resulting plot would resemble an human eye, giving it its name. An eye diagram can reveal following information about a system.[1]

- **Eye height** - Noise immunity
- **Eye width** - Error free region
- **Time variation at zero crossing** - A measure of jitter/ time offset
- **Thickness at the peaks** - Peak deviation

1.4 Channel Equalisation

The purpose of using equalisation schemes is to compensate for the frequency dependent amplitude and phase deviations, introduced to the original signal by the channel. These imparements include Inter-symbol Interference (ISI), multipath fading, and noise. These impairments can result in symbol errors. This can be managed by the introduction of equalizers into the communication system.

Equalizers change the frequency response of the channel, adjusting the amplitude and frequency of the signal. One of such equalizers that are being used in communication systems is known as the **Zero Forcing (ZF) Equaliser**, an equaliser which tries to apply the inverse of the frequency response of the channel on to the signal, which we will be using in this assignment.

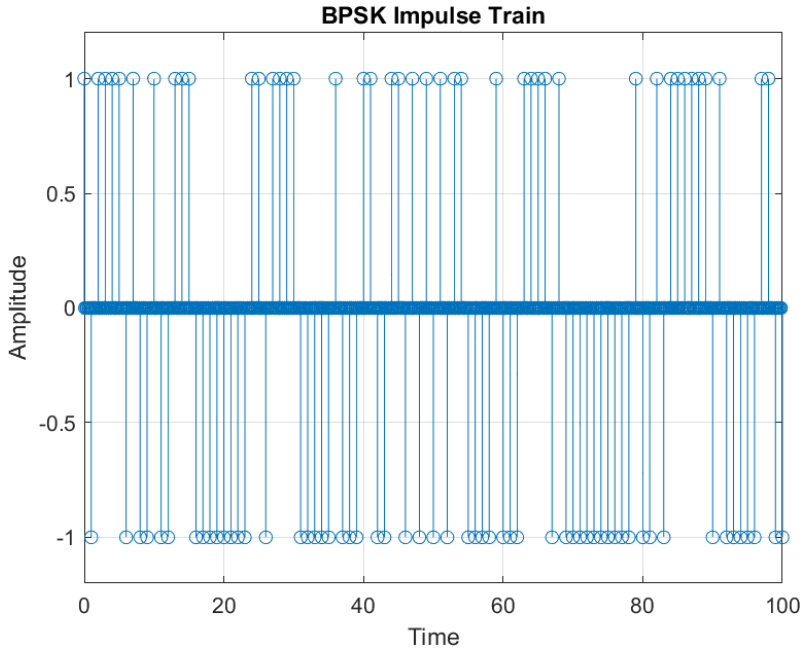
2 Assignment

2.1 Task I

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

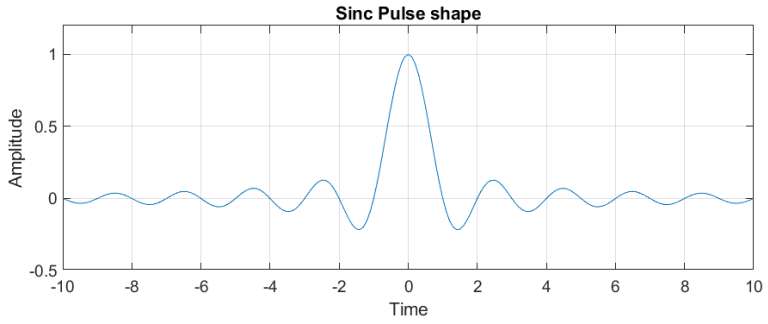
1. *Generate an impulse train representing BPSK symbols.*
2. *Obtain transmit signal by convolving the impulse train with a pulse shaping filter where the impulse response is a Sinc function.*
3. *Generate the eye diagram of the transmit signal*
4. *Repeat 1-3 for raised cosine pulse shaping filters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

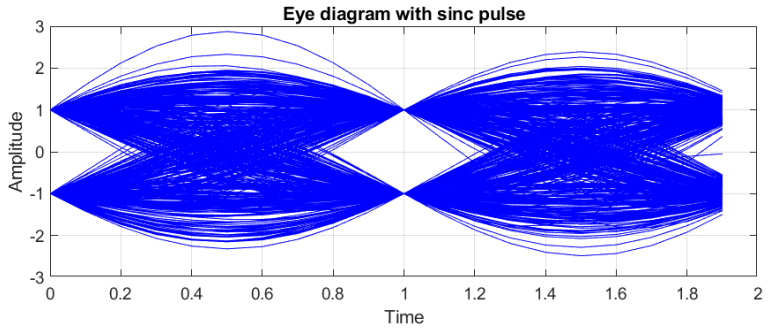
In this assignment, we have generated 1000 bit symbols, which have been mapped to -1 and +1 in a 2-PAM scheme, or to 0° and 180°. Each symbol has been sample 10 times, creating 10000 samples. Then this is convoluted with a Sinc pulse having 2000 samples.



Afterwards, these two were convoluted, and the following was obtained as the eye diagram of the resulting waveform.

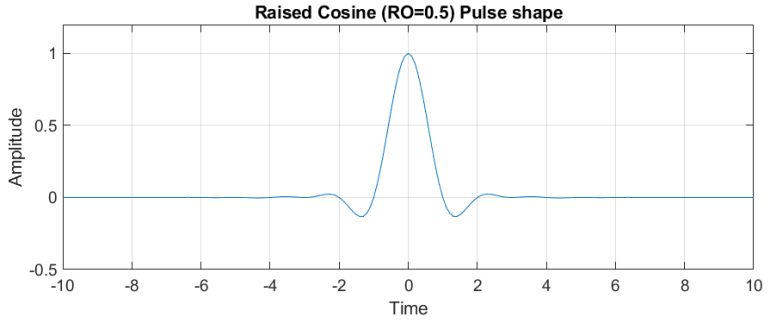
Now the impulse train was convoluted with two **Raised Cosine** pulse shaping filters, each having a *roll-off factor* of 0.5 and 1. The equation of a raised cosine pulse is as follows:

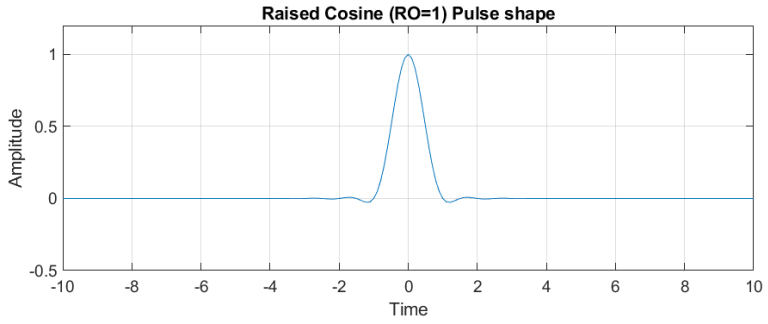




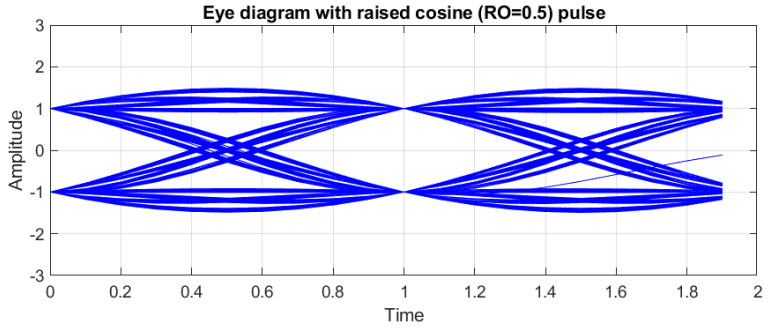
$$h(t) = \begin{cases} \frac{\pi}{4T}\,\mathrm{sinc}\left(\frac{1}{2\beta}\right), & t = \pm\frac{T}{2\beta} \\ \frac{1}{T}\,\mathrm{sinc}\left(\frac{t}{T}\right)\frac{\cos\left(\frac{\pi\beta t}{T}\right)}{1-\left(\frac{2\beta t}{T}\right)^2}, & \text{otherwise} \end{cases}$$

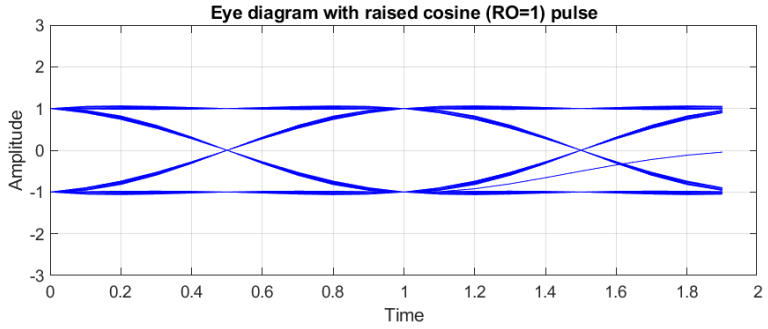
in which β is the Roll off factor, and T is the symbol period. In this assignment $T = 1$ has been used, and $\beta = 0.5$ and $\beta = 1$ which leads the following waveforms have been used.





When the impulse train is convoluted with the above waveforms, it lead with the following eye diagrams.





By the above eye diagrams, we can arrive at a number of conclusions:

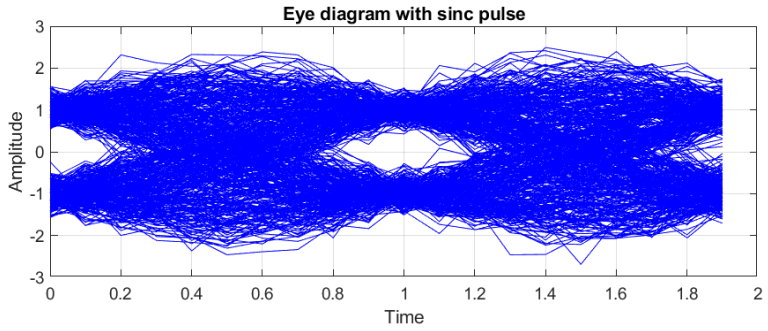
- Eye width in the increasing order can be seen in the sinc, raised cosine of rollover 0.5 and of 1. This implies that in the sinc pulse modulation, there is the least range for error free sampling, with raised cosines of rollover 0.5 and 1 having more range in the increasing order
- Time variation at the zero crossing decrease from sinc to raised cosine of rollover 0.5 to 1, signifying decreasing time jitter. 0.5 rollover scheme has almost no jitter at all.

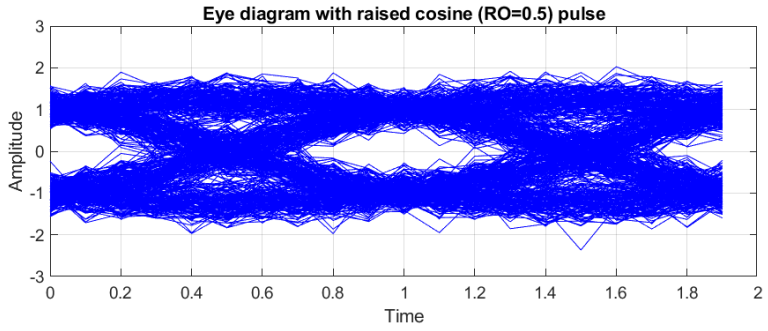
By the above conclusion, we can predict that the sinc pulse, raised cosine with 0.5 rollover, and 1 rollover will predict better in a noisy channel, in the increasing order.

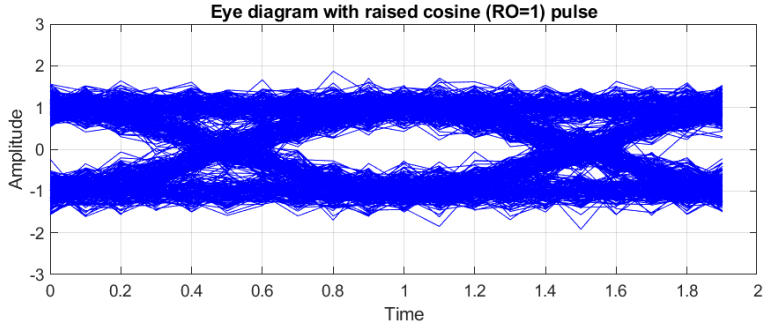
2.2 Task II

Requirement:
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* $\frac{E_b}{N_0} = 10dB$*, where* E_b *is the average bit energy and* N_0 *is the noise power spectral density.*

Here Additive White Gaussian Noise (AWGN) having a mean of 0 and a variance such that $\frac{E_b}{N_0} = 10dB$, where E_b is the average bit energy and N_0 is the noise power spectral density is added to the output of the filter. This leads following eye diagrams:







From the above eye diagrams, we can arrive at a number of conclusions:

- We can see the eye heights of all three diagrams have reduced due to the addition of AWGN. However, from sinc pulse to raised cosines with rollover of 0.5 and to that of 1, the eye height increases. This signifies the increasing noise immunity.
- Due to the new AWGN, the eye widths has reduced, but still are in the increasing order can be seen in the sinc, raised cosine of rollover 0.5 and of 1. This implies that in the sinc pulse

modulation, there is the least range for error free sampling, with raised cosines of rollover 0.5 and 1 having more range in the increasing order

- Time variation at the zero crossing has increased, but still decrease from sinc to raised cosine of rollover 0.5 to 1, signifying decreasing time jitter. In this scenario however, scheme with rollover of 1 is not devoid of jitter.
- Here in the presence of noise, there is a thickness at the peak of the eye diagram. This is due to the noise, and may cause some errors at sampling.

By the above conclusion, we can predict that the sinc pulse, raised cosine with 0.5 rollover, and 1 rollover will predict better in a noisy channel, in the increasing order.

2.3 Task III

Requirement:
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 0.7 0.4]`.
4. *Add White Gaussian noise such that* $\frac{E_b}{N_0} = 0dB$.
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* $\frac{E_b}{N_0}$ *values* 0-10dB.
9. *Plot the BER for all tap settings and* $\frac{E_b}{N_0}$ *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.*

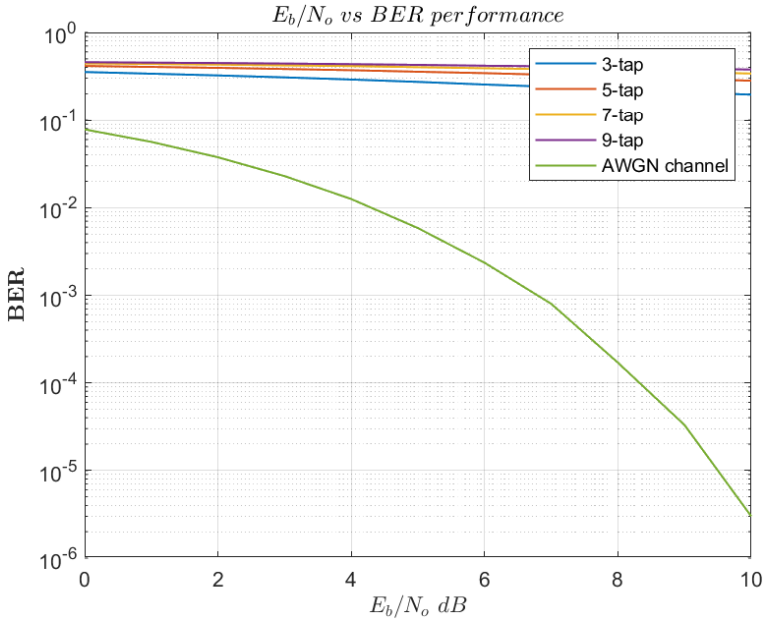
In this task we aim to revert the effects added onto a signal by a multipath channel, using a ZF equalizer. As an input for this, first a binary sequence has been generated. Then this sequence has been modulated using 2-PAM scheme, or BPSK scheme, mapping 0 to -1 and 1 to +1. Here we have not introduced a pulse shaping scheme as instructed.

We have then simulated a multipath channel by convolving the transmitted signals with a *3-tap impulse response*, represented by `h = [0.3, 0.9, 0.4]`, which would introduce inter-symbol interference. Multipath channel creates a signal using shifted and weighed original signal.

Afterwards AWGN has been added to simulate a realistic environment. At first $\frac{E_b}{N_0}$ is set to $0dB$. At the reciever, to combat the ISI, ZF filters at different tap lengths were made. Then this equalized signal is demodulated, and then converted back into a bit stream. Calculating the Bit Error Rate (BER), we got the following for each tap:

- BER at 3 = 0.236783
- BER at 5 = 0.263130
- BER at 7 = 0.270090
- BER at 9 = 0.273533

Now these steps are repeated, with varying $\frac{E_b}{N_0}$ values 0-10dB. This lead to the following graph of bit error rate with the increasing $\frac{E_b}{N_0}$:



In the AWGN channel, BER is very small compared to the ISI multipath channel with the ZF equalizer. Reason for this is the amplification of the noise at some frequency ranges by the ZF equalizer.[2]

Further, any imperfect channel estimations may lead to higher BER. Also, rather than forcing the pulses to be zero at all the instances, we select a finite number of instances to tap. The residual interference of this will also contribute to performance issues.[3]

Now let's consider the usage of binary orthogonal signalling, rather than BPSK. Here the distance between signals are smaller, as in BPSK it would have $2\sqrt{E}$ and binary orthogonal would have $\sqrt{2E}$ of a distance between the two symbols. Due to this, it will have a higher bit error rate than BPSK, in a given environment.[4]

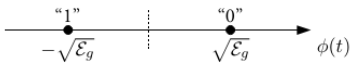


Figure 1: Signal Constellation for BPSK

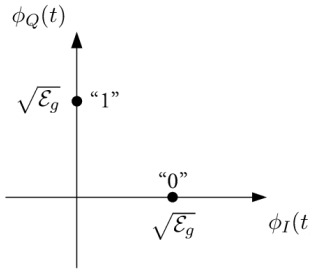


Figure 2: Signal Constellation for Binary Orthogonal Signalling

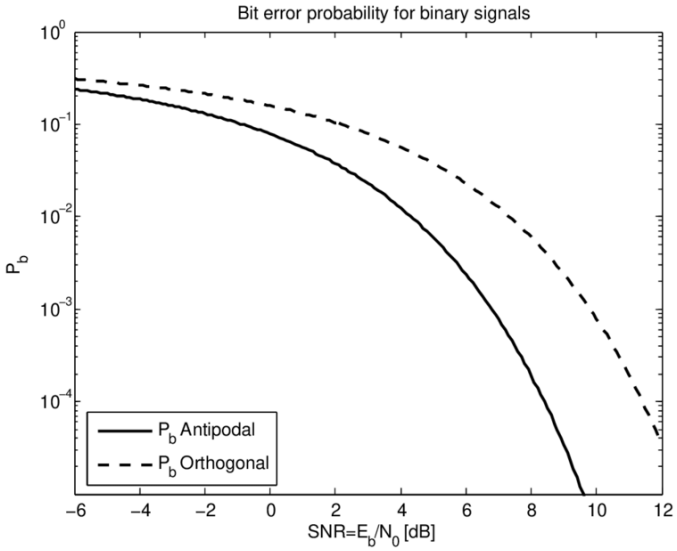


Figure 3: BPSK vs Binary orthogonal BER performance

References

[1] ON Semiconductor. *Understanding Data Eye Diagram Methodology for Analyzing High Speed Digital Signals*. Application Note. 2015. URL: https://www.onsemi.cn/PowerSolutions/document/AND9075-D.PDF (visited on 04/28/2024).

[2] K. Sankar. *BER for BPSK in ISI channel with Zero Forcing equalization*. Blog. 2009. URL: https://dsplog.com/2009/11/29/ber-bpsk-isi-channel-zero-forcing-equalization/ (visited on 04/28/2024).

[3] P. Sharma. *Performance Analysis of Zero-Forcing Equalizer for ISI Reduction In Wireless Channels*. Blog. 2012. URL: https://www.ijert.org/research/performance-analysis-of-zero-forcing-equalizer-for-isi-reduction-in-wireless-channels-IJERTV1IS8284.pdf (visited on 04/28/2024).

[4] S. Höst. "A Short Introduction to Digital Communications". In: (). URL: https://www.researchgate.net/publication/265490002_A_Short_Introduction_to_Digital_Communications.

3 Appendices

3.1 Code for Task I

```
% Parameters
fs = 10;                                    % Sample freq
nSymbols = 1000;                            % No of symbols
t = -10*fs:1/fs:9.99*fs;                    % time
ts = 0:1/fs:9999/fs;


% Generating BPSK symbols
symbols = 2*(rand(1,nSymbols)>0.5)-1;
symbols_upsampled = [symbols;zeros(fs-1,length(symbols))];
symbols_upsampled = symbols_upsampled(:).';

stem(ts, symbols_upsampled); xlabel('Time'); ylabel('Amplitude');
title('BPSK Impulse Train');
axis([0 100 -1.2 1.2]); grid on;


% Sinc filter convolution
sinc_num = sin(pi*t);
sinc_den = (pi*t);
sinc_zero = find(abs(sinc_den) < 10^-10);   % Finding the t=0 position
sinc_filter = sinc_num./sinc_den;
sinc_filter(sinc_zero) = 1;

tx_signal_sinc = conv(symbols_upsampled, sinc_filter, 'same');
tx_signal_sinc_trimmed = tx_signal_sinc(1:10000);


% Raised cosine filter (RO = 0.5)
roll_off = 0.5;
cos_num= cos(roll_off*pi*t);
cos_den = (1 - (2 * roll_off * t).^2);
cos_zero = abs(cos_den)<10^-10;
Raised_cosine = cos_num./cos_den;
Raised_cosine(cos_zero) = pi/4;
rc_roll05 = sinc_filter.*Raised_cosine;

tx_signal_rcroll05 = conv(symbols_upsampled, rc_roll05, 'same');
tx_signal_rcroll05_trimmed = tx_signal_rcroll05(1:10000);

% Raised cosine filter (RO = 1)
roll_off = 1;
cos_num= cos(roll_off*pi*t);
cos_den = (1 - (2 * roll_off * t).^2);
cos_zero = abs(cos_den)<10^-10;
Raised_cosine = cos_num./cos_den;
Raised_cosine(cos_zero) = pi/4;
rc_roll1 = sinc_filter.*Raised_cosine;

tx_signal_rcroll1 = conv(symbols_upsampled, rc_roll1, 'same');
tx_signal_rcroll1_trimmed = tx_signal_rcroll1(1:10000);


% Eye diagrams
conv_tx_sinc_reshape = reshape(tx_signal_sinc_trimmed, fs*2, nSymbols*fs/20).';
conv_tx_rcroll05_reshape = reshape(tx_signal_rcroll05_trimmed, fs*2, nSymbols*fs/20)
    .';
conv_tx_rcroll1_reshape = reshape(tx_signal_rcroll1_trimmed, fs*2, nSymbols*fs/20).';

% Sinc
figure;
subplot(6,2,[1,3]);
plot(t, sinc_filter);
```

```
title('Sinc Pulse shape');
xlabel('Time'); ylabel('Amplitude');
axis([-fs fs -0.5 1.2]);
grid on;

subplot(6,2,[2,4]);
plot(0:1/fs:1.99, real(conv_tx_sinc_reshape).', 'b');
title('Eye diagram with sinc pulse');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -3 3]);
grid on;

% Raised Cosine RO=0.5
subplot(6,2,[5,7]);
plot(t, rc_roll05);
title('Raised Cosine (RO=0.5) Pulse shape');
xlabel('Time'); ylabel('Amplitude');
axis([-fs fs -0.5 1.2]);
grid on;

subplot(6,2,[6,8]);
plot(0:1/fs:1.99, real(conv_tx_rcroll05_reshape).', 'b');
title('Eye diagram with raised cosine (RO=0.5) pulse');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -3 3]);
grid on;

% Raised Cosine RO=1
subplot(6,2,[9,11]); % Adjusted the index for the first subplot of RO=1
plot(t, rc_roll1);
title('Raised Cosine (RO=1) Pulse shape');
xlabel('Time'); ylabel('Amplitude');
axis([-fs fs -0.5 1.2]);
grid on;

subplot(6,2,[10,12]); % Adjusted the index for the second subplot of RO=1
plot(0:1/fs:1.99, real(conv_tx_rcroll1_reshape).', 'b');
title('Eye diagram with raised cosine (RO=1) pulse');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -3 3]);
grid on;

% Adjust the vertical position of the subplots to add spacing
h = gcf;
h.Position(2) = h.Position(2)  - 10;
```

3.2 Code for Task II

```
% Parameters
fs = 10;                                            % Sample freq
nSymbols = 1000;                                    % No of symbols
time = -fs:1/fs:fs;
ts = 0:1/fs:99/fs;


% Generating BPSK symbols
symbols = 2*(rand(1,nSymbols)>0.5)-1;
symbols_upsampled = [symbols;zeros(fs-1,length(symbols))];
symbols_upsampled = symbols_upsampled(:).';

stem(ts, symbols(1:100)); xlabel('Time'); ylabel('Amplitude');
title('BPSK Impulse Train');
axis([0 10 -1.2 1.2]); grid on;


% Generation of noise
SNR = 10;
PN  = 1./(10.^(0.1*SNR));
noise =((PN/2)^0.5)*randn(1,10000);


% Sinc filter convolution
sinc_num = sin(pi*t);
sinc_den = (pi*t);
sinc_zero = find(abs(sinc_den) < 10^-10);       % Finding the t=0 position
sinc_filter = sinc_num./sinc_den;
sinc_filter(sinc_zero) = 1;

tx_signal_sinc = conv(symbols_upsampled, sinc_filter, 'same');
tx_signal_sinc = tx_signal_sinc(1:10000);

% Sinc with noise
tx_sinc_noise = tx_signal_sinc+noise;


% Raised cosine filter (RO = 0.5)
roll_off = 0.5;
cos_num= cos(roll_off*pi*t);
cos_den = (1 - (2 * roll_off * t).^2);
cos_zero = abs(cos_den)<10^-10;
Raised_cosine = cos_num./cos_den;
Raised_cosine(cos_zero) = pi/4;
rc_roll05 = sinc_filter.*Raised_cosine;

tx_signal_rcroll05 = conv(symbols_upsampled, rc_roll05, 'same');
tx_signal_rcroll05 = tx_signal_rcroll05(1:10000);

% Raised Cosine (RO=0.5) with noise
tx_rcroll05_noise = tx_signal_rcroll05+noise;


% Raised cosine filter (RO = 1)
roll_off = 1;
cos_num= cos(roll_off*pi*t);
cos_den = (1 - (2 * roll_off * t).^2);
cos_zero = abs(cos_den)<10^-10;
Raised_cosine = cos_num./cos_den;
Raised_cosine(cos_zero) = pi/4;
rc_roll1 = sinc_filter.*Raised_cosine;

tx_signal_rcroll1 = conv(symbols_upsampled, rc_roll1, 'same');
tx_signal_rcroll1 = tx_signal_rcroll1(1:10000);

% Raised Cosine (RO=1) with noise
```

```
tx_rcroll1_noise = tx_signal_rcroll1+noise;


% Eye diagrams
conv_tx_sinc_reshape = reshape(tx_sinc_noise, fs*2, nSymbols*fs/20).';
conv_tx_rcroll05_reshape = reshape(tx_rcroll05_noise, fs*2, nSymbols*fs/20).';
conv_tx_rcroll1_reshape = reshape(tx_rcroll1_noise, fs*2, nSymbols*fs/20).';

% Sinc
figure;
subplot(6,2,[1,3]);
plot(t, sinc_filter);
title('Sinc Pulse shape');
xlabel('Time'); ylabel('Amplitude');
axis([-fs fs -0.5 1.2]);
grid on;

subplot(6,2,[2,4]);
plot(0:1/fs:1.99, real(conv_tx_sinc_reshape).', 'b');
title('Eye diagram with sinc pulse');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -3 3]);
grid on;

% Raised Cosine RO=0.5
subplot(6,2,[5,7]);
plot(t, rc_roll05);
title('Raised Cosine (RO=0.5) Pulse shape');
xlabel('Time'); ylabel('Amplitude');
axis([-fs fs -0.5 1.2]);
grid on;

subplot(6,2,[6,8]);
plot(0:1/fs:1.99, real(conv_tx_rcroll05_reshape).', 'b');
title('Eye diagram with raised cosine (RO=0.5) pulse');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -3 3]);
grid on;

% Raised Cosine RO=1
subplot(6,2,[9,11]); % Adjusted the index for the first subplot of RO=1
plot(t, rc_roll1);
title('Raised Cosine (RO=1) Pulse shape');
xlabel('Time'); ylabel('Amplitude');
axis([-fs fs -0.5 1.2]);
grid on;

subplot(6,2,[10,12]); % Adjusted the index for the second subplot of RO=1
plot(0:1/fs:1.99, real(conv_tx_rcroll1_reshape).', 'b');
title('Eye diagram with raised cosine (RO=1) pulse');
xlabel('Time'); ylabel('Amplitude');
axis([0 2 -3 3]);
grid on;

% Adjust the vertical position of the subplots to add spacing
h = gcf;
h.Position(2) = h.Position(2)  - 10;
```

3.3 Code for Task III

```
Rs =10 ;                                % Symbol rate for the signal
Ts = 1/Rs ;                             % Symbol frequency
time = 0:Ts:100000;


data_bits = randi([0 1],1,numel(time));
PAM2_data = real(pskmod(data_bits,2));  % Representing bit 1 -> +1 and bit 0 ->-1 (
    PAM)


% Model a 3-tap multipath channel
channel_taps = [0.3 0.7 0.4] ;

% Apply channel distortion (convolution)
received_signal = conv(PAM2_data,channel_taps,"same"); %convolving


%Calculate Bit energy
bit_energy = sum(PAM2_data.^2)/length(PAM2_data) ;


% creating Zero forcing equalizer
% M - tap equalizer
Eb_No_dB = 0:10 ; % dB

for M=3:2:9 % Number of equalizer taps (odd values)
    N = (M-1)/2 ;
    Po = zeros(1,M);
    Po(N+1) = 1;
    Pr = toeplitz([channel_taps(2) channel_taps(1) zeros(1,M-2)],[channel_taps(2)
    channel_taps(3) zeros(1,M-2)]) ; % Toeplitz matrix with filter coeffients
    C = Pr\Po' ;
    BER_ISI = zeros(1,11);

    for n=1:numel(Eb_No_dB)
        Eb_No = 10^(Eb_No_dB(n)/10);
        No = bit_energy/Eb_No ;
        sigma = sqrt(No/2);
        rs_an = received_signal + sigma*randn(1,numel(received_signal));%signal after
     adding noise
        rs_ae = conv(rs_an,C,"same"); % Signal after the equalizer
        received_data_bits_ISI = real(pskdemod(rs_ae,2));
        bit_errors_ISI=numel(find(received_data_bits_ISI-data_bits)); % calculation
    of number of bit errors
        BER_ISI(n)=bit_errors_ISI/numel(data_bits);                   % calculation
    of bit error rate
    end

    semilogy(Eb_No_dB,BER_ISI,'linewidth',1);
    grid on;
    hold on;
end


% plotting BER for a AWGN channel in the same figure
BER_awgn= zeros(1,11);

for n=1:numel(Eb_No_dB)
    Eb_No = 10^(Eb_No_dB(n)/10);
    No = bit_energy/Eb_No ;
    sigma = sqrt(No/2);

    %signal after adding noise
    rs_awgn =PAM2_data + sigma*randn(1,numel(PAM2_data));
    received_data_bits_awgn = real(pskdemod(rs_awgn,2));
    bit_errors_awgn=numel(find(received_data_bits_awgn-data_bits)); % calculating
```

```
    number of bit errors
    BER_awgn(n)=bit_errors_awgn/numel(data_bits);                    % calculating bit
     error rate
end


semilogy(Eb_No_dB,BER_awgn,'linewidth',1);
legend("3-tap","5-tap","7-tap","9-tap","AWGN channel");
title("$E_{b}/N_{o} \ vs \ BER \ performance$",'interpreter','latex');
xlabel("$E_{b}/N_{o} \ dB $",'interpreter','latex');
ylabel("$\textbf{BER}$",'interpreter','latex');
publish('script.m','pdf')
```