Cairo University
Faculty of Engineering

Department of Computer Engineering

ELC 3070 – Spring 2024

Communications 2

Project #2

Matched Filter

Submitted to

Dr. Mohamed Khairy

Dr. Mohamed Nafea

Eng. Mohamed Khaled

Submitted by

Team: 31

Name	ID	Sec.	B.N.
عبدالله ابراهيم السيد عبدالهادي	9210602	2	48
عمر احمد عبد العليم مرزوق	9210704	3	7
فتحى مصطفى فتحى عبدالحميد	9210808	3	19
محمود حمدى جابر عبد التواب	9211083	4	10

Contents

Figures

Role of each member

Members	Role
Abdallah Ibrahim	Requirement 2 (Noise Analysis)
Omar Marzouk	Requirement 3 (ISI and Raised Cosine)
Fathy Mostafa	Requirment 1 (Matched Filters and Correlators)
Mahmoud Hamdy	Report Organization
All of the members	checking code reliability

Requirement 1: Matched filters and correlators in noise-free environment

To simulate a binary PAM signaling system, we start by generating 10 random binary bits. These bits are then converted into PAM symbols, where logic 1 becomes +1 and logic 0 becomes -1. The symbol duration is 1 second.

To represent the pulse shaping function discreetly, we use 5 samples equally spaced, with a difference of 200 ms between each sample. This pulse shaping function is normalized to ensure that its energy is unity.

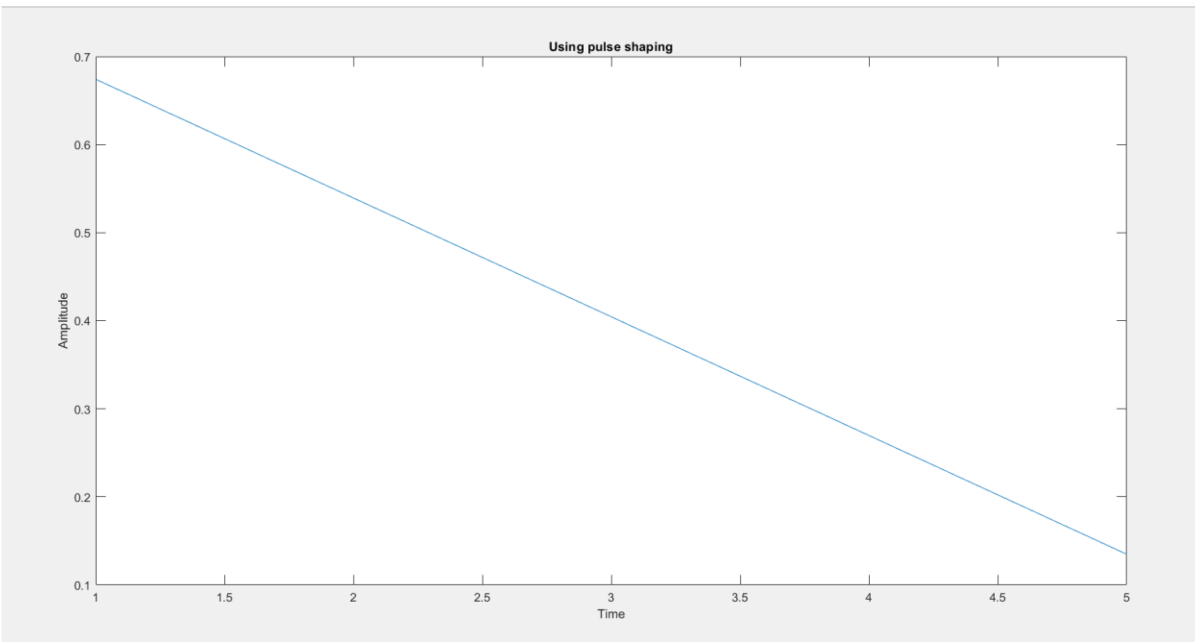

Figure 1 Pulse Shaping.

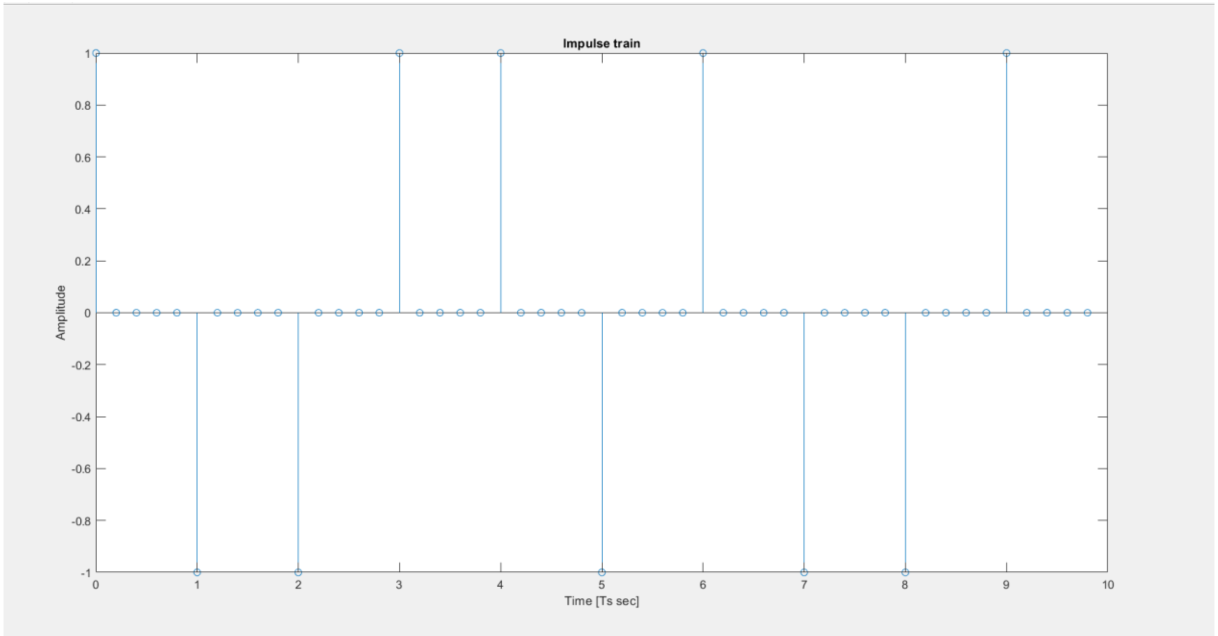

Figure 2 : UP Sampled Data.

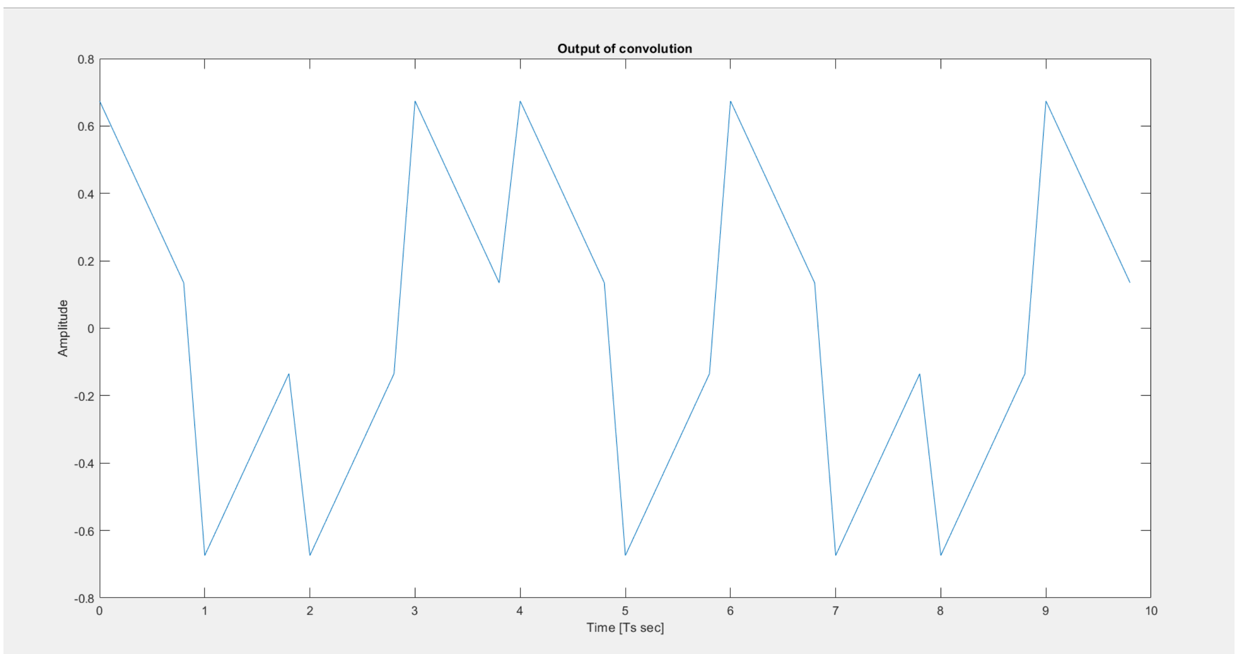

Figure 3 Output of Convolution

Part A: Matched and Unmatched filters Ouputs

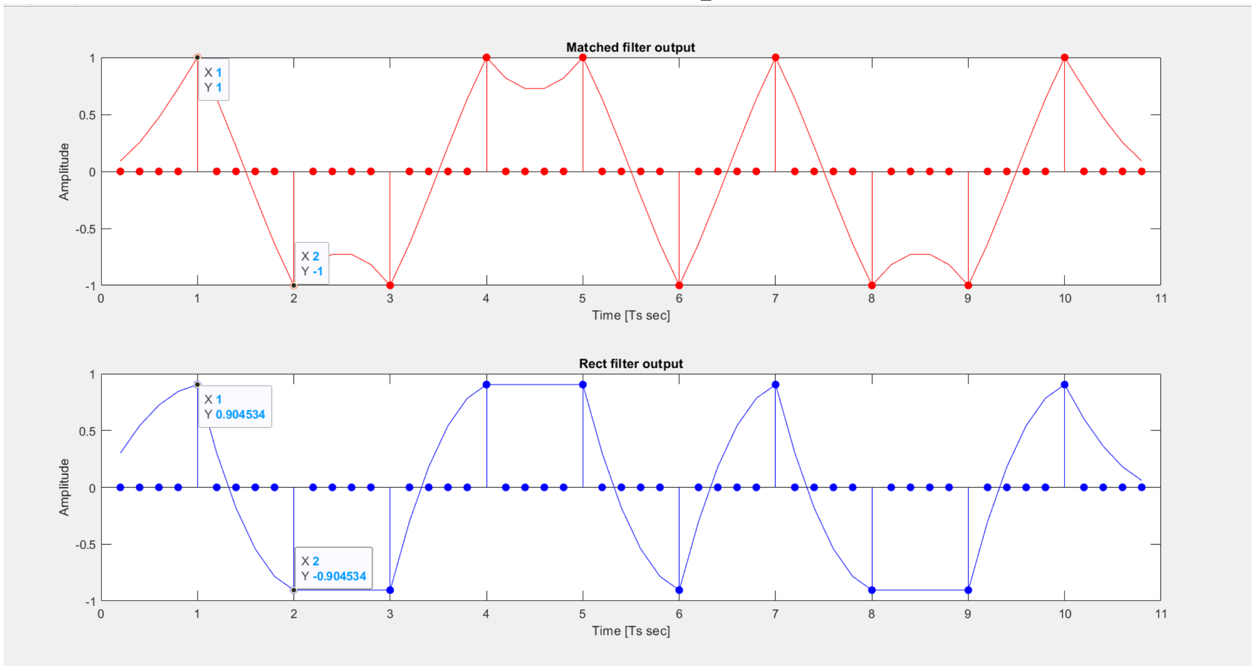

Figure 4 : Matched and Unmatched Filter Output

Comment:

At the sampling instance, the amplitude of the signal precisely reaches ±1 after passing through the matched filter. Conversely, when passing through the rectangular filter, the signal's level is slightly lower at 0.904, which is less than 1. As the amplitudes of the samples increase, using matched filter minimizing errors and maximizes the signal-to-noise ratio (SNR).

Part B: Correlator

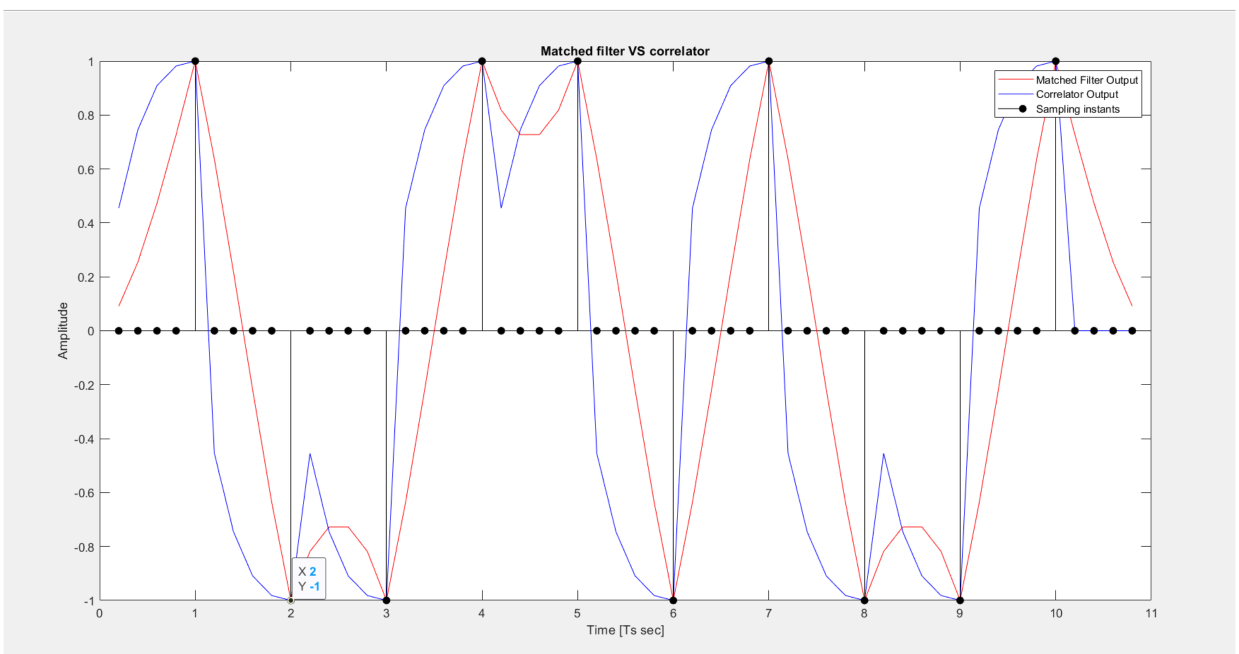

Figure 5 : Output of the matched filter and correlator

Comment:

The output of the correlator equals convolution between the received signal and the pulse shaping function, which is identical to the output of the matched filter at those instances. Since we are sampling at time equal TS, this equivalence maximizes the signal-to-noise ratio (SNR) at sampling instances. $Correlator_{OUT} = \int_0^{TS} X(t) * P(t)dt$

Requirement 2: Noise Analysis

The process involves expanding the initial bit sequence of 10,000 bits, generating Gaussian noise with zero mean and unity variance matching the size of the sequence. This noise is then scaled to achieve a variance of N0/2. It's added to the transmitted sequence, and the resulting signal is filtered using a matched filter or a rect filter and sampled. The probability of error is calculated based on a 10,000-sample array. Additionally, N0 is adjusted iteratively to vary Eb/N0 from -2 dB to 5 dB in 1 dB steps, with the BER calculated at each step

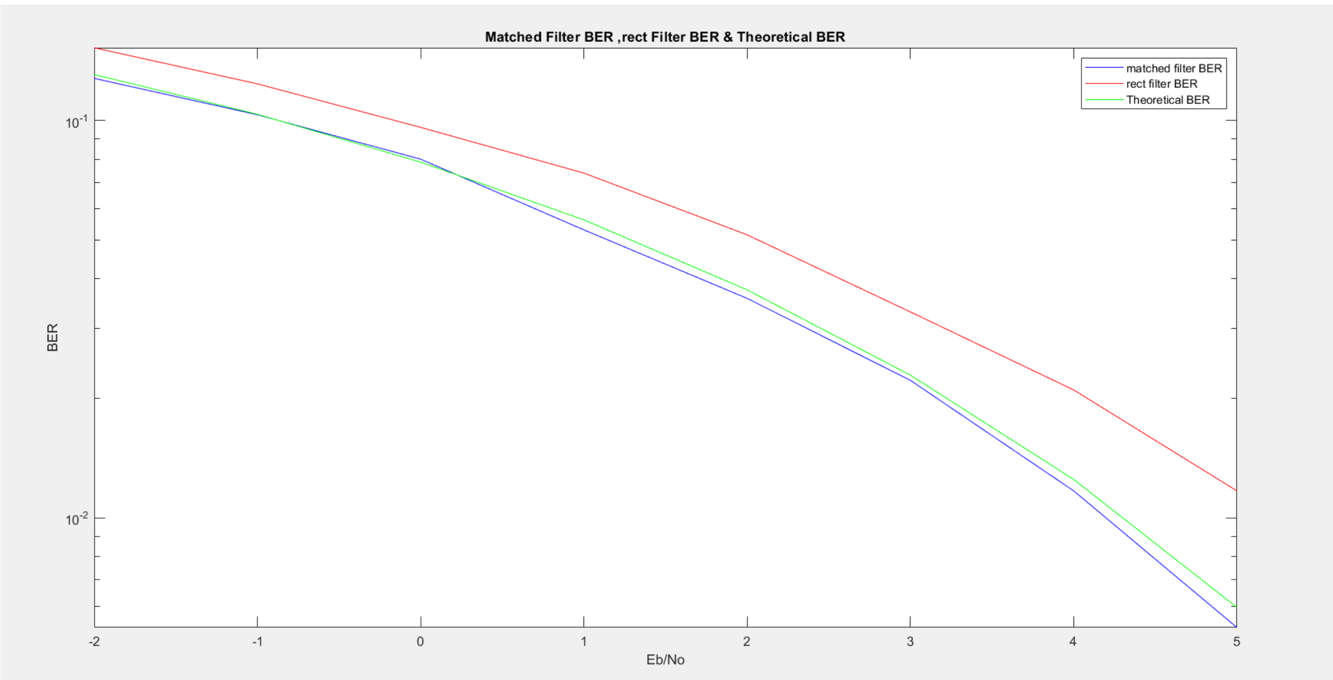

Figure 6 : BER of matched filter and unmatched filter with theoretical BER using 10000 bits

Comment:

Matched filters are designed to maximize the peak Signal-to-Noise Ratio (SNR), making them more efficient. So even with equivalent Energy to Noise Spectral Ratio (Eb/No), the Bit Error Rate (BER) tends to be lower with a matched filter than with an unmatched filter. Also, the matched filter BER is nearly equal to the theoretical BER $BER = \frac{1}{2}\, erfc(\sqrt{\frac{Eb}{No}})$

So to produce more accurate results, we used a larger number of bits (1,000,000)

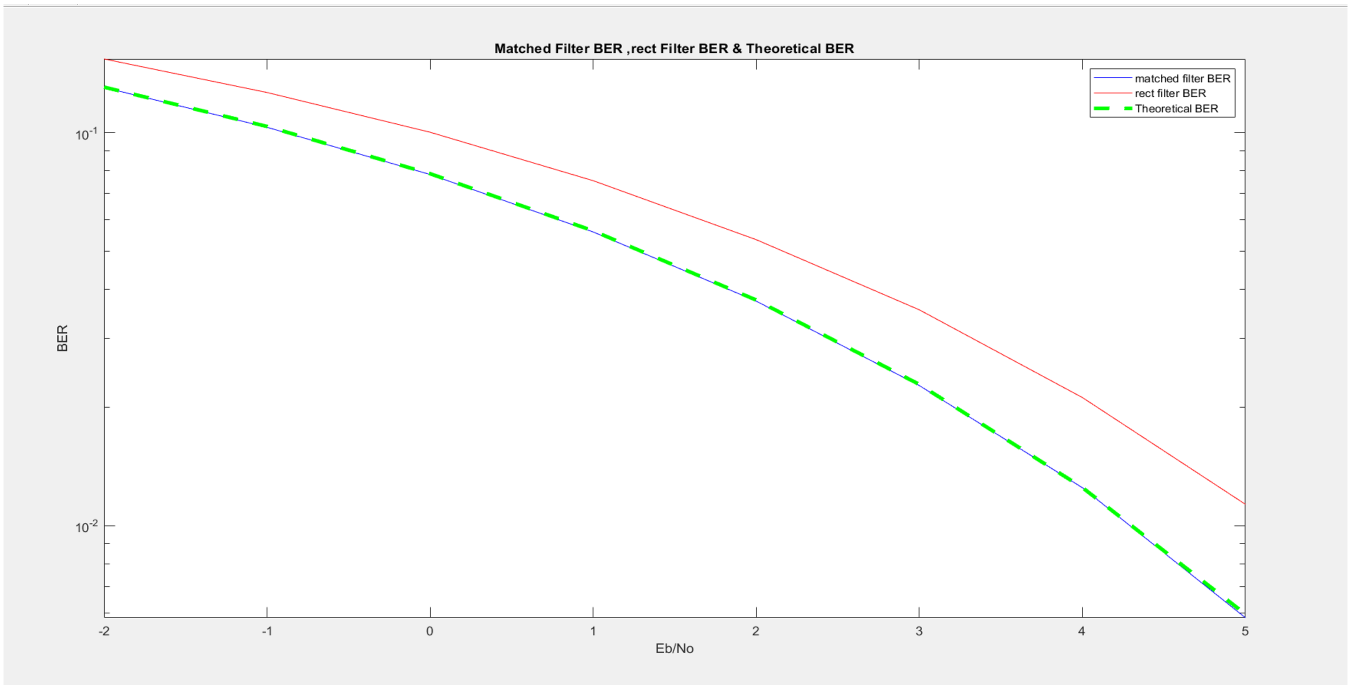

Figure 7: BER of matched filter and unmatched filter with theoretical BER using 1,000,000 bits

As the number of bits approaches infinity, the BER of the matched filter approaches the theoretical result.

Requirement 3: ISI and raised cosine

To illustrate the impact of Inter-Symbol Interference (ISI), we employ a noise-free system characterized by square root raised cosine filters for both transmission and reception. The raised cosine filter is chosen due to its ability to meet the Nyquist criterion, ensuring efficient data transmission by minimizing ISI. Although the ideal square root raised cosine filter is theoretically optimal, its infinite impulse response renders it impractical for real-world applications. Thus, we utilize a parameterized approach to define the practical length of the filter, represented by the delay parameter fig [9], and control the filter's bandwidth using the Rolloff factor, denoted as R. By employing square root raised cosine (SRRC) filtering at both ends, we aim to approximate the response of a raised cosine filter. This ensures that the resultant system exhibits the desired characteristics, facilitating a clearer understanding of ISI through the visualization of eye patterns, which graphically represent the received signal's characteristics over time.

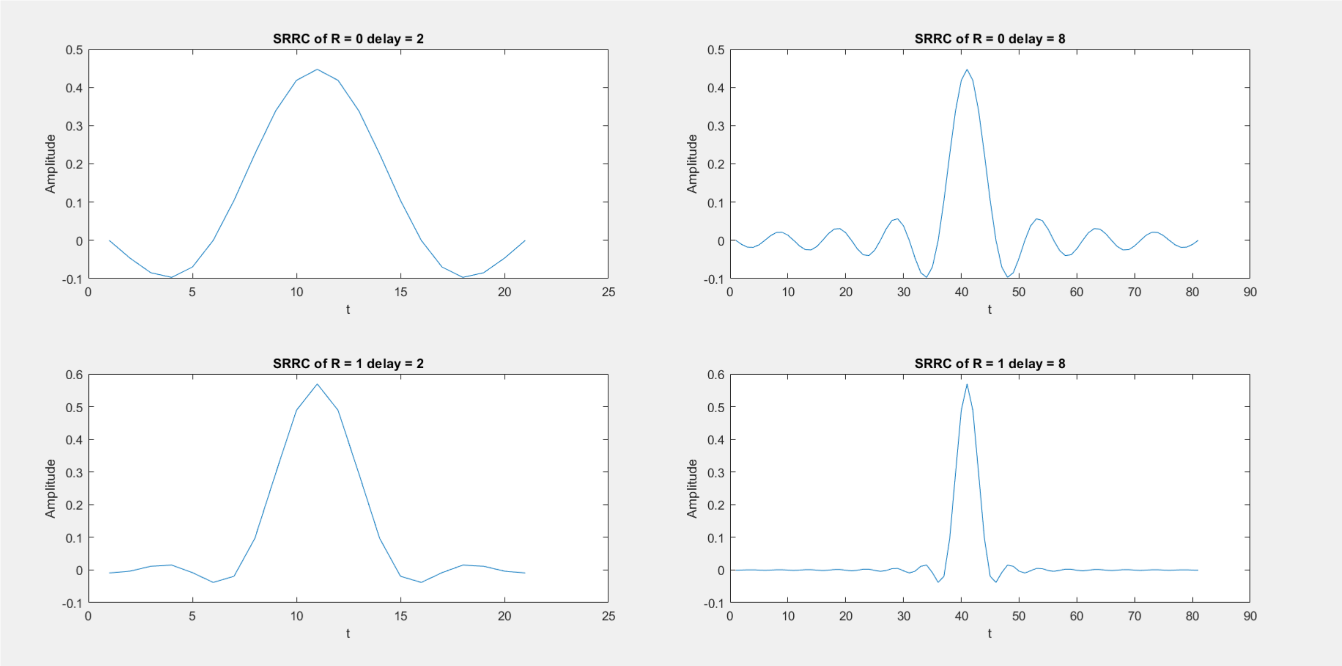

Figure 8 : square root raised cosine

At Point A: Transmitted signal

A. R = 0, delay = 2.

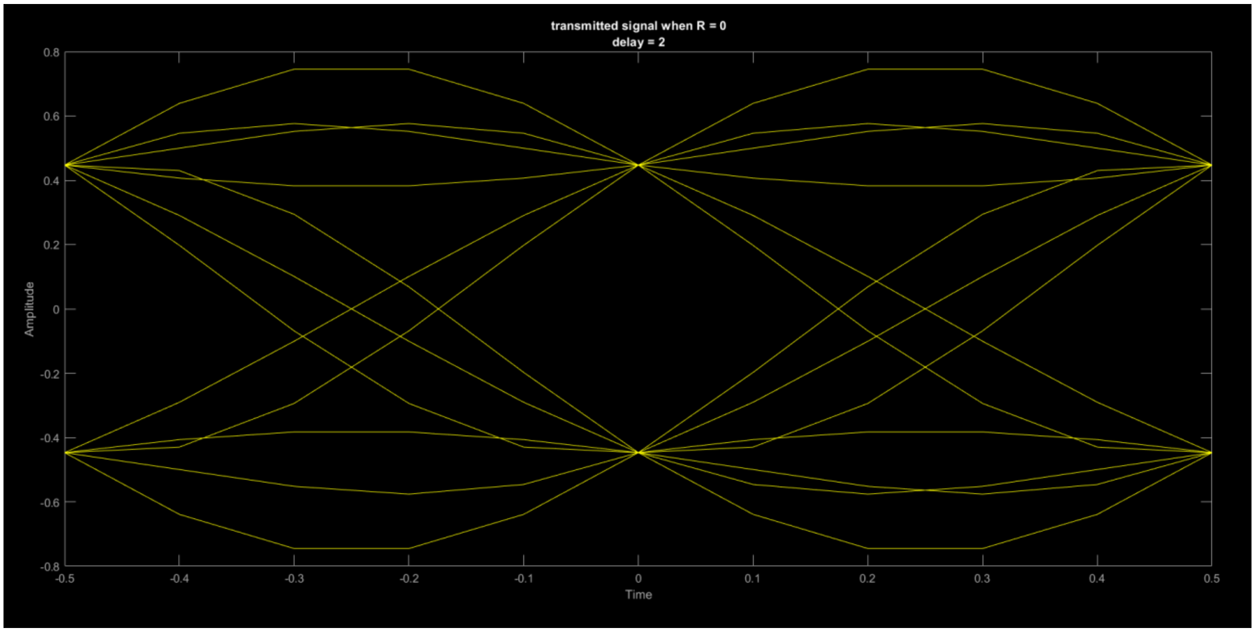

Figure 9: Transmitted signal Eye diagram at R=0 & Delay=2

B. R = 0, delay =8.

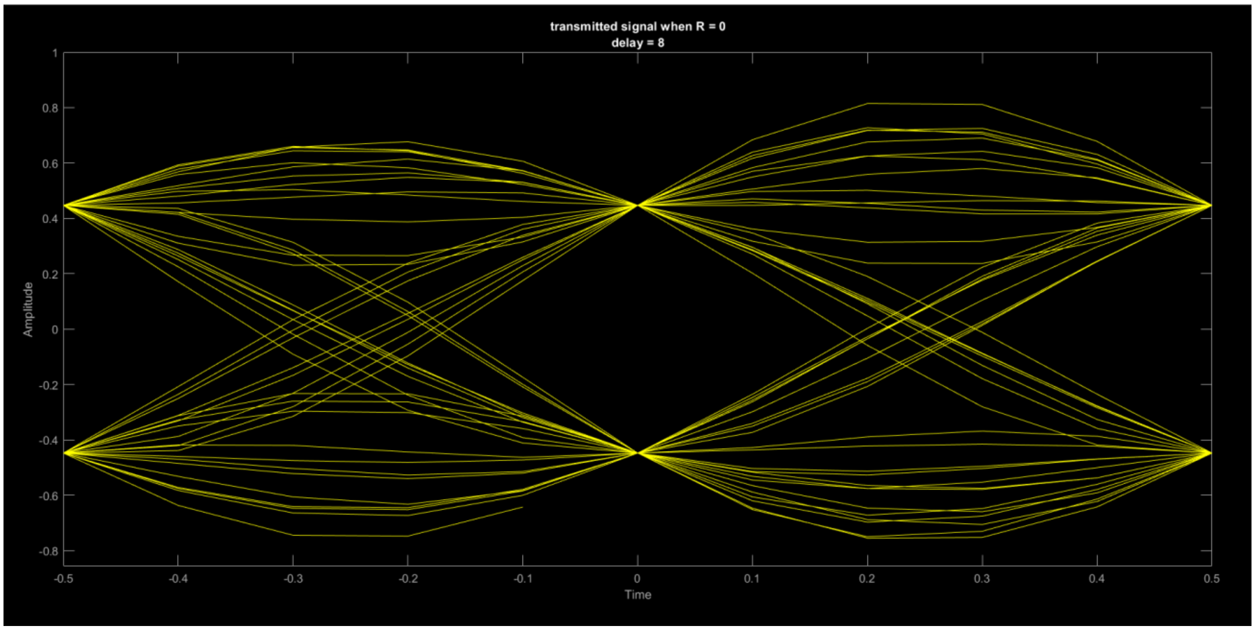

Figure 10: Transmitted signal Eye diagram at R=0 & Delay=8

C. R = 1, delay =2.

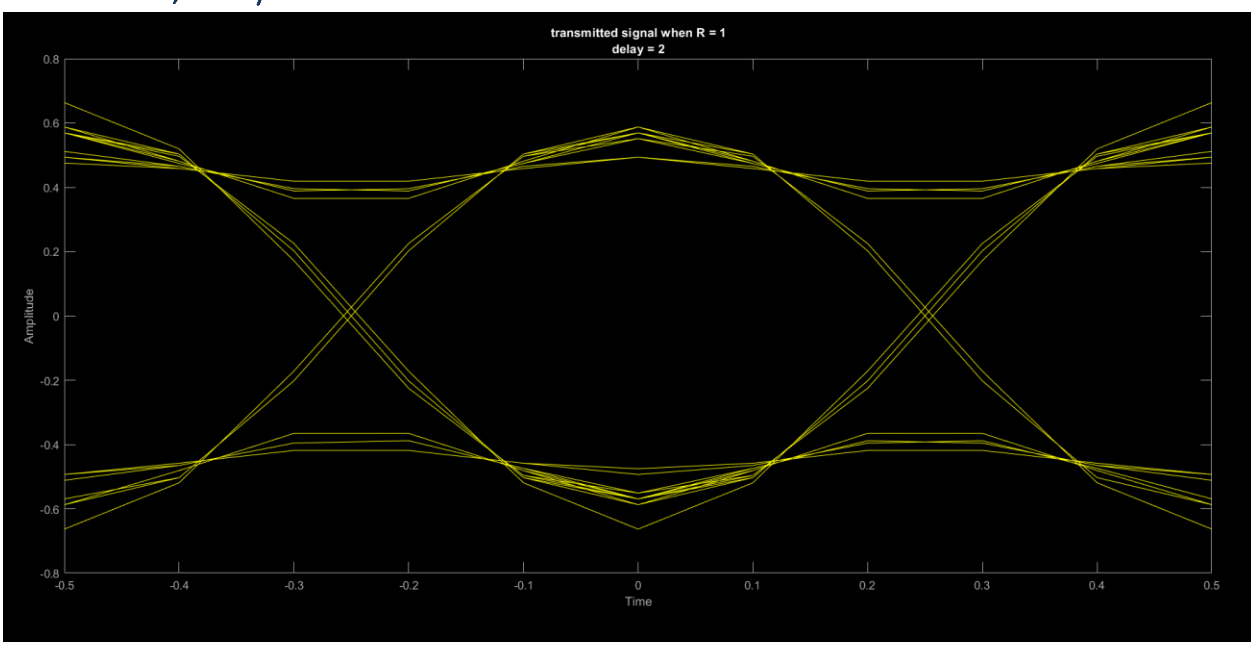

Figure 11: Transmitted signal Eye diagram at R=1 & Delay=2

D. R = 1, delay =8.

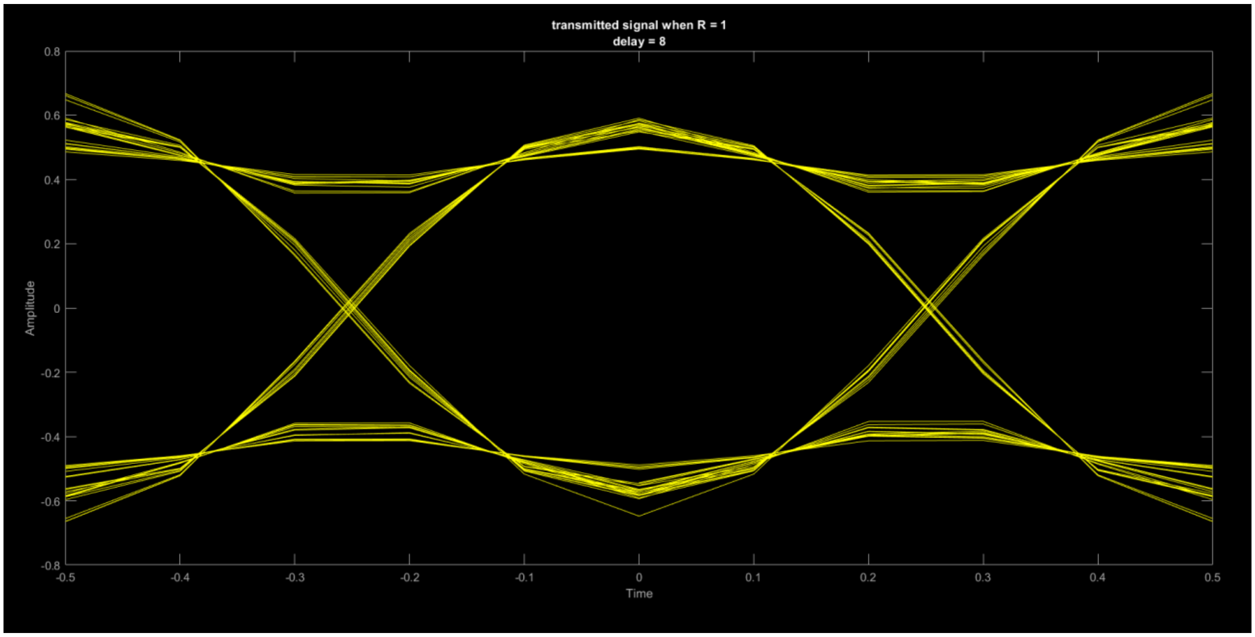

Figure 12: Transmitted signal Eye diagram at R=1 & Delay=8

Comment:

The Optimal sampling instant corresponds to the center of the eye-opening in an eye diagram. A wider eye opening indicates a greater tolerance for timing errors, whereas a narrower opening suggests a higher risk of sampling errors. Optimal sampling instant placement maximizes signal integrity and minimizes inter-symbol interference. A higher roll-off factor typically results in a wider eye-opening but increasing bandwidth.

- For R = 0:

square root raised cosine filter is an infinite impulse response (Sinc function for R=0). Thus, we utilize a parameterized approach to define the practical length of the filter by the delay parameter which decides the number of peaks taken as in Fig [8]. In Fig [9][10] we can observe that There is no noticeable change in the eye pattern as after the transmitter the signal is exposed to only one SRRC filter. The effect of delay appears clearly after the receiver (raised cosine effect) where a clear eye pattern appears with delay =8 as in Fig [13][14]. This happens because Sinc's pulse doesn't die quickly and needs more time to ignore the other pecks.

- For R = 1:

the bandwidth is increased to allow the pulse to die quickly which improves ISI and isn't affected by the delay as side peaks in impulse response have ignored amplitudes compared to the main peak. These results appear clearly in figures [11][12] after the transmitter and figures [16][17] after the receiver.

At Point B: Received signal

A. R = 0, delay =2.

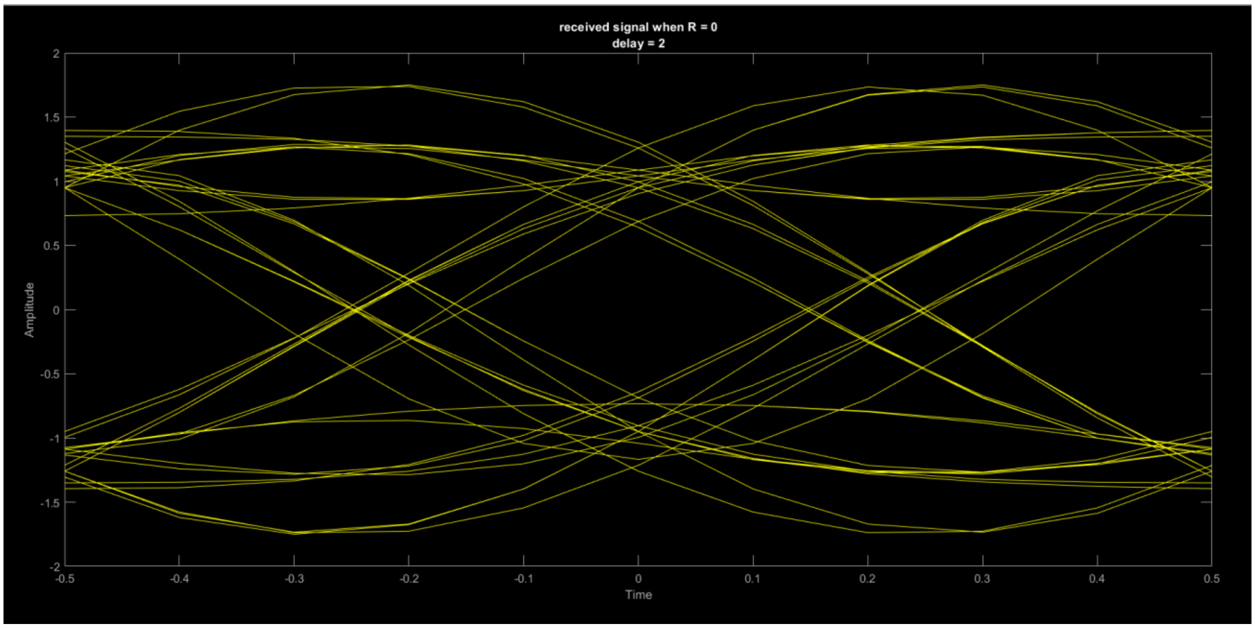

Figure 13: Received signal Eye diagram at R=0 & Delay=2

B. R = 0, delay =8.

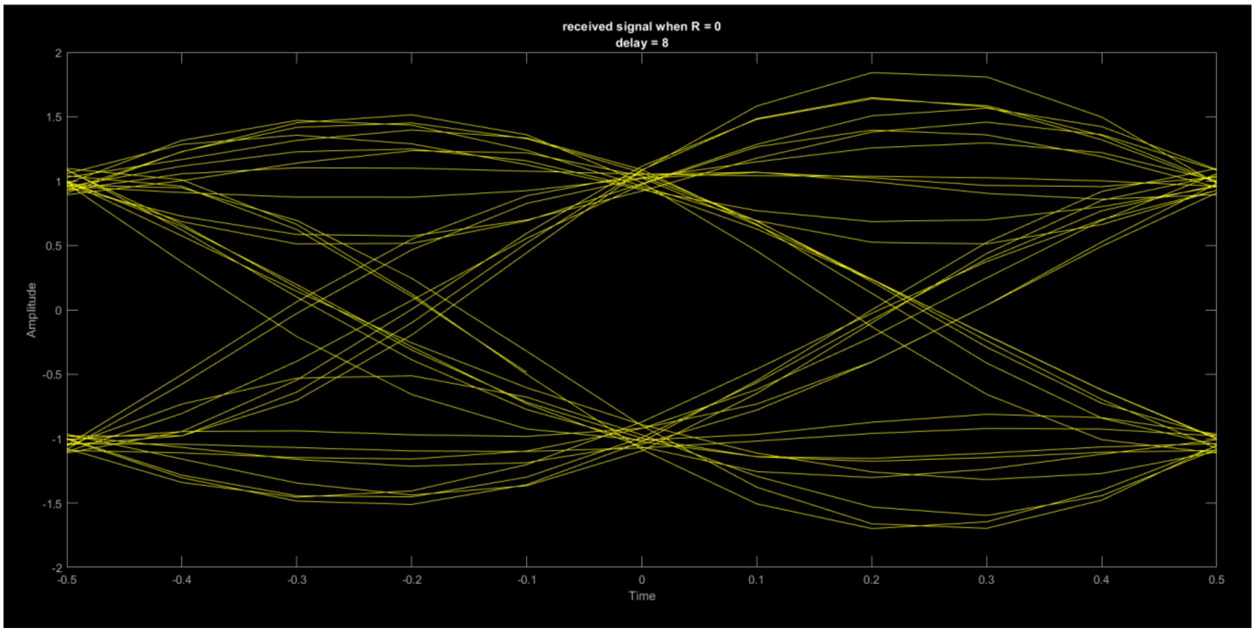

Figure 14: Received signal Eye diagram at R=0 & Delay=8

C. R = 1, delay =2.

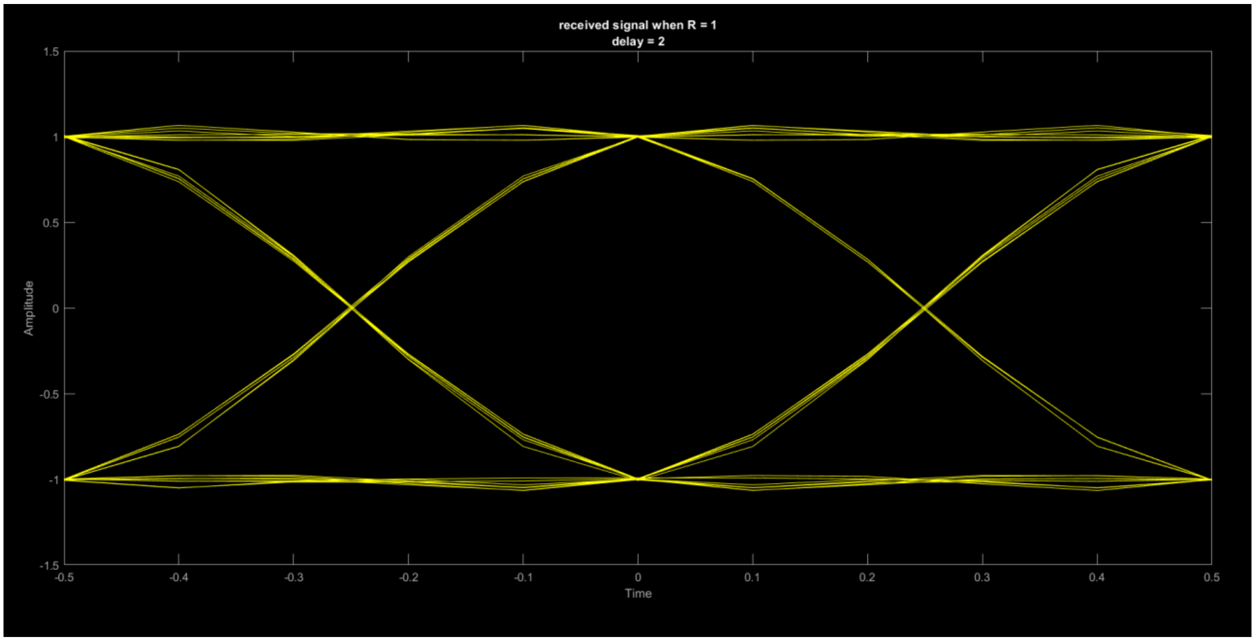

Figure 15: Received signal Eye diagram at R=1 & Delay=2

D. R = 1, delay =8.

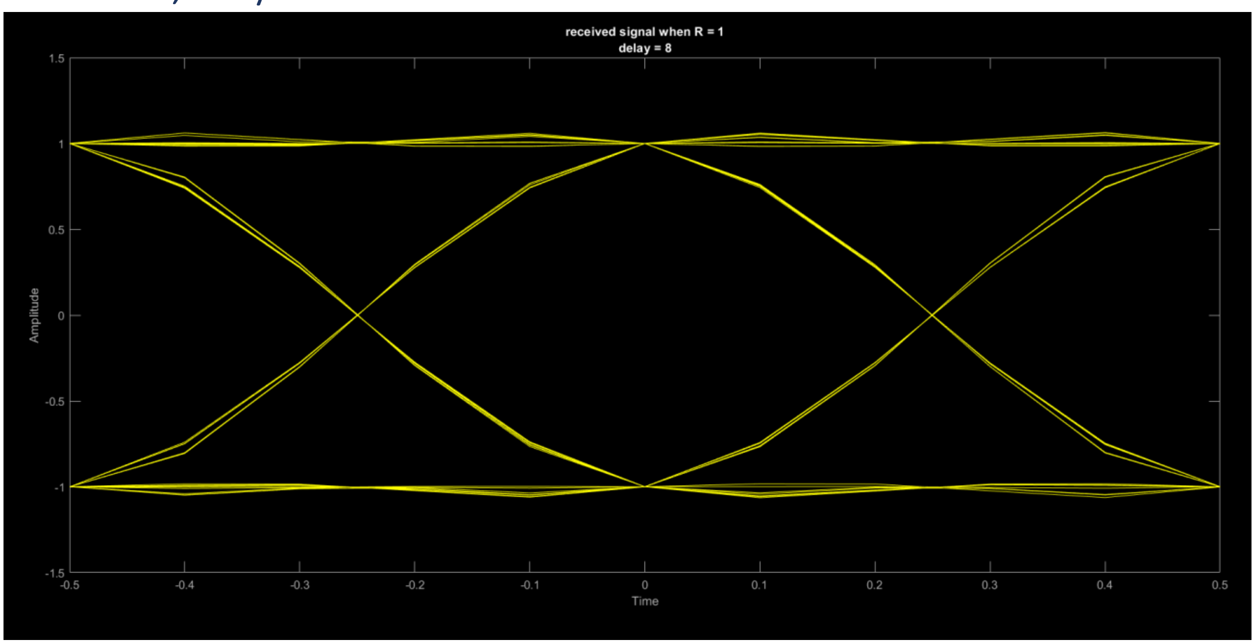

Figure 16: Received signal Eye diagram at R=1 & Delay=8

Comment:

The received signal might experience distortion or noise, which can further affect the eye-opening. As it is considered the system is noise-free, the eye diagram won't affected by these factors. Note that we utilize an SRRC pulse for both transmission and reception to ensure that the final result is a raised cosine pulse. The comment on the figures for this section has been clarified in the previous comment section.

Index:

```
%%%%%%%%%%Matched Filters, Correlators, ISI, and raised cosine%%%%%%%%%%%%
close all;
clear;
clc;
%%part 1 Matched filters and correlators in noise free environment%%
A=1;   %voltage level
TS=1; %bit duartion
sampling_time=0.2; % we will take 5 samples for each bit
num_bits=10;
num_sample=num_bits*(TS/sampling_time);
data =randi([0,1],1,num_bits);  %generate bits
data_mapping=2*data-1;   %mapping data to-1,1
data_upsampling=upsample(data_mapping,(TS/sampling_time)); %represnting every bit using 5 samples
%and the rest be zeros
p=[5 4 3 2 1]/sqrt(55);    %pulse shaping with unit energy
y=conv(data_upsampling,p);  % y equal convolution of data_upsampling and the pulse shaping signal
y=y(1:num_sample); %convolution change size of data streamed
t = 0 : sampling_time : (num_sample - 1)*sampling_time; %start from zero until reach all samples
%plot the using pulse shaping
figure ;
plot(p);
title("Using pulse shaping");
xlabel("Time");
ylabel("Amplitude");
figure;
stem(t,data_upsampling);
title("Impulse train");
xlabel("Time [Ts sec]");
ylabel("Amplitude");
figure ;
plot(t,y);
title("Output of convolution");
xlabel("Time [Ts sec]");
ylabel("Amplitude");
%%design of matched filter
matched_filter=fliplr(p);   % Matched filter (flipped pulse)
filter_2=ones(1,(TS/sampling_time))/sqrt(TS/sampling_time);     %rect filter normalized
matched_filter_output=conv(matched_filter,y);
filter_2_output=conv(filter_2,y);
t_conv_adjust=sampling_time : sampling_time :
(size(matched_filter_output,2))*sampling_time;
matched_filter_output_sampled=zeros(size(matched_filter_output));
filter_2_output_sampled=zeros(size(matched_filter_output));
for i=1:5:num_sample
    matched_filter_output_sampled(i+4)=matched_filter_output(i+4);
    filter_2_output_sampled(i+4)= filter_2_output(i+4);
end
figure;
subplot(2,1,1);
plot(t_conv_adjust,matched_filter_output,'r');
hold on
stem(t_conv_adjust,matched_filter_output_sampled,'r','filled')
title("Matched filter output");
xlabel("Time [Ts sec]");
```

```
ylabel("Amplitude");
axis ([0 11 -1 1]);
subplot(2,1,2);
plot(t_conv_adjust,filter_2_output,'b');
hold on
stem(t_conv_adjust,filter_2_output_sampled,'b','filled')
title("Rect filter output");
xlabel("Time [Ts sec]");
ylabel("Amplitude");
axis ([0 11 -1 1]);

%%correlator%%
correlator_output = zeros(size(matched_filter_output));
%repeat the pulse shape function by number of bits
pulse_shaping_correlator = repmat(p, 1, num_bits);   %repeat pulse shaping the
number of bits
sum_corr = 0; %initialization of integrator result

for i = 1 : num_sample
    if mod((i-1), 5) == 0    %(i-1)%5
        sum_corr = 0;
    end
    correlator_output(i) = sum_corr + (pulse_shaping_correlator(i) * y(i));
    sum_corr = correlator_output(i);
end
correlator_output_sampled=zeros(size(matched_filter_output));
for i=1:5:num_sample
    correlator_output_sampled(i+4)=correlator_output(i+4);
end
figure
subplot(2,1,1)
plot(t_conv_adjust,matched_filter_output,'r');
hold on ;
stem(t_conv_adjust,matched_filter_output_sampled,'r','filled');
title("Matched filter Output");
xlabel("Time [Ts sec]");
ylabel("Amplitude");
axis ([0 11 -1 1]);
subplot(2,1,2)
plot(t_conv_adjust,correlator_output,'b');
hold on ;
stem(t_conv_adjust,matched_filter_output_sampled,'b','filled');
title("Correlator Output");
xlabel("Time [Ts sec]");
ylabel("Amplitude");
axis ([0 11 -1 1]);
figure
plot(t_conv_adjust,matched_filter_output,'r');
hold on ;
plot(t_conv_adjust,correlator_output,'b');
hold on;
stem(t_conv_adjust,matched_filter_output_sampled,'k','filled');
title("Matched filter VS correlator");
xlabel("Time [Ts sec]");
ylabel("Amplitude");
legend('Matched Filter Output', 'Correlator Output','Sampling instants');
axis ([0 11 -1 1]);
%%%
noise_num_bits=10000;
noise_data = randi([0,1],1,noise_num_bits); %generating 10000 random bits
```

```
noise_data_mapping = (2*noise_data - 1); %converting 0 to -1 and 1 to +1
noise_data_upsampling = upsample(noise_data_mapping,5); %sapmle every 200ms
y_noise= conv(noise_data_upsampling,p); %output of the transmitter
y_noise(:,50001:50004) = []; %taking only nonzero values
noise = randn(1,size(y_noise,2));
No=1;
Eb=sum(p .^2);
x=1;
Rect_Filter_BER=zeros(1,8);
Matched_Filter_BER=zeros(1,8);
Theoretical_BER=zeros(1,8);
for j=-2:5
    No=Eb/(10^(j/10));
    noise_scaled = noise.*sqrt(No/2);
    v_noise = y_noise+noise_scaled; %adding noise to signal
%Matched and normalized filters--------------------------------
matched_filter_noise = fliplr(p);
matched_filter_noise_out = conv(v_noise , matched_filter_noise);
rect_filter_noise = [5 5 5 5 5]/sqrt(125);
rect_filter_noise_out = conv(v_noise,rect_filter_noise);
for i=1:noise_num_bits
    matched_filter_noise_out_Sampled(i) = matched_filter_noise_out(i*5); %sampling
every Ts=5*200ms
    rect_filter_noise_out_Sampled(i) = rect_filter_noise_out(i*5);
end
%Comparing with threshold for matched filter
for i = 1:length(matched_filter_noise_out_Sampled)
    if matched_filter_noise_out_Sampled(i)<0
        matched_filter_noise_out_Sampled(i) = -1;
    elseif matched_filter_noise_out_Sampled(i)>0
        matched_filter_noise_out_Sampled(i) = 1;
    end
end
%Comparing with threshold for matched filter
for i = 1:length(rect_filter_noise_out_Sampled)
    if rect_filter_noise_out_Sampled(i)<0
        rect_filter_noise_out_Sampled(i) = -1;
    elseif rect_filter_noise_out_Sampled(i)>0
        rect_filter_noise_out_Sampled(i) = 1;
    end
end
%Getting BER-------------------------------------------------------
%ber
ber_matched_fitler_sum=0;
ber_RECT_sum=0;
for i=1:length(matched_filter_noise_out_Sampled)
    if(matched_filter_noise_out_Sampled(i)~= noise_data_mapping(i))
        ber_matched_fitler_sum=ber_matched_fitler_sum+1;
    end
end
for i=1:length(rect_filter_noise_out_Sampled)
    if(rect_filter_noise_out_Sampled(i)~= noise_data_mapping(i))
        ber_RECT_sum=ber_RECT_sum+1;
    end
end
Theoretical_BER(x)=0.5*erfc(sqrt(Eb/No));%Getting theoritical BER
Matched_Filter_BER(x)=ber_matched_fitler_sum/noise_num_bits;
Rect_Filter_BER(x)=ber_RECT_sum/noise_num_bits;
x=x+1;
end
```

```matlab
%Plots-------------------------------------------------------------
figure;
semilogy(-2:5,Matched_Filter_BER,'b');
hold on
semilogy(-2:5,Rect_Filter_BER,'r');
hold on
semilogy(-2:5, Theoretical_BER,'g');
xlabel('Eb/No');
ylabel('BER');
legend('matched filter BER','rect filter BER','Theoretical BER');
title('Matched Filter BER ,rect Filter BER & Theoretical BER');
%%ISI %%%%%%
ISI_num_bits=100;
ISI_data=randi([0,1],1,ISI_num_bits);
ISI_data_mapping=(2*ISI_data - 1);
ISI_data_upsampling=upsample(ISI_data_mapping,5);
R=[0 0 1 1];  %different roll of factor
delay=[2 8 2 8];  %different delays
for i=1:4
    square_root_raised_cosine=rcosine(TS,TS/sampling_time,'sqrt', R(i), delay(i));
    figure ('Name', 'SRRC');
    plot( square_root_raised_cosine);
    title(sprintf('SRRC of R = %d delay = %d', R(i), delay(i)));

    TX=conv(ISI_data_upsampling,square_root_raised_cosine,'valid');
    RX=conv(TX,square_root_raised_cosine,'valid');
    eyediagram(TX,2*TS/sampling_time);
    title(sprintf('transmitted signal when R = %s\ndelay = %s', string(R(i)),
string(delay(i))));
    eyediagram(RX,2*TS/sampling_time);
    title(sprintf('received signal when R = %s\ndelay = %s', string(R(i)),
string(delay(i))));
end
```