Digital Communications (CT303)
Lab 7 - 8 Report

Dhrumil Mevada – 201901128

Lab 7

1. Understanding the concept of the Eye Diagram of a Qua- ternary PAM baseband transmission.(4.38)

➢ Here Input Output relationship is $x(t) = s(t) + as^2(t)$

1.1 **Code :-**

```
clear;        %Generate eye diagrams
clf;
a = 0.1;
data = sign(randn(1,400)) + 2*sign(randn(1,400));
%PAM symbols stem(data);
output = data + a.*(data.^2); tau = 64;
%Symbol period
dataup = upsample(output,tau);
%Impulse train
yrz = conv(dataup,prz(tau))    ;
%Return to zero polar signal yrz = yrz(1:end-tau+1);
ynrz = conv(dataup,prect(tau)) ;
%Non return to zero polar ynrz = ynrz(1:end-tau+1);
ysine=conv(dataup,psine(tau));
% half sinusoid polar ysine = ysine(1:end-tau+1);
Td = 4 ;    % truncating raised cosine to 4 periods
yrcos = conv(dataup,prcos(0.5,Td,tau));
% roll off factor = 0.5
yrcos = yrcos(2*Td*tau:end-2*Td*tau+1);
% generating RC pulse train

eye1 = eyediagram(yrz,2*tau,tau,tau/2);
title('RZ Eye - Diagram for a = 0.1');
eye2 = eyediagram(ynrz,2*tau,tau,tau/2);
title('NRZ Eye - Diagram for a = 0.1');
```

```
eye3 = eyediagram(ysine,2*tau,tau,tau/2);
title('Half - Sine Eye - Diagram for a = 0.1');
eye4 = eyediagram(yrcos,2*tau,tau);
title('Raised - Cosine Eye - Diagram for a = 0.1');
```

- Pnrz code:-

```
%generating a rectangular pulse of width T
% Usage function pout = pnrz(T);
function pout = prect(T)
    pout = ones(1,T) ;
end
```

- Prz code:-

```
function pout=prz(T)
    pout= [zeros(1,T/4) ones(1,T/2) zeros(1,T/4)];
end
```

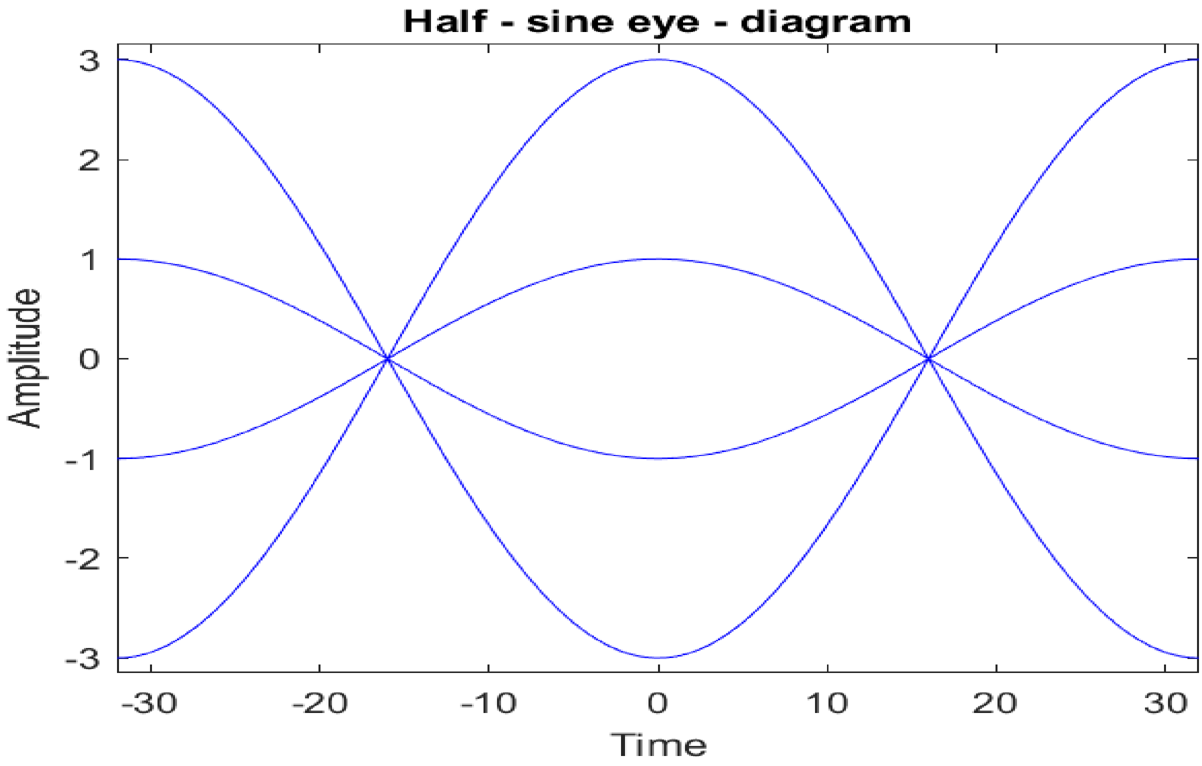
- Psin code:-

```
% (ps ine . rn)
% generating a s inusoid pul se of width T
%
function pout=psine (T)
    pout=sin (pi * [ 0 : T- 1 ] / T ) ;
end
```

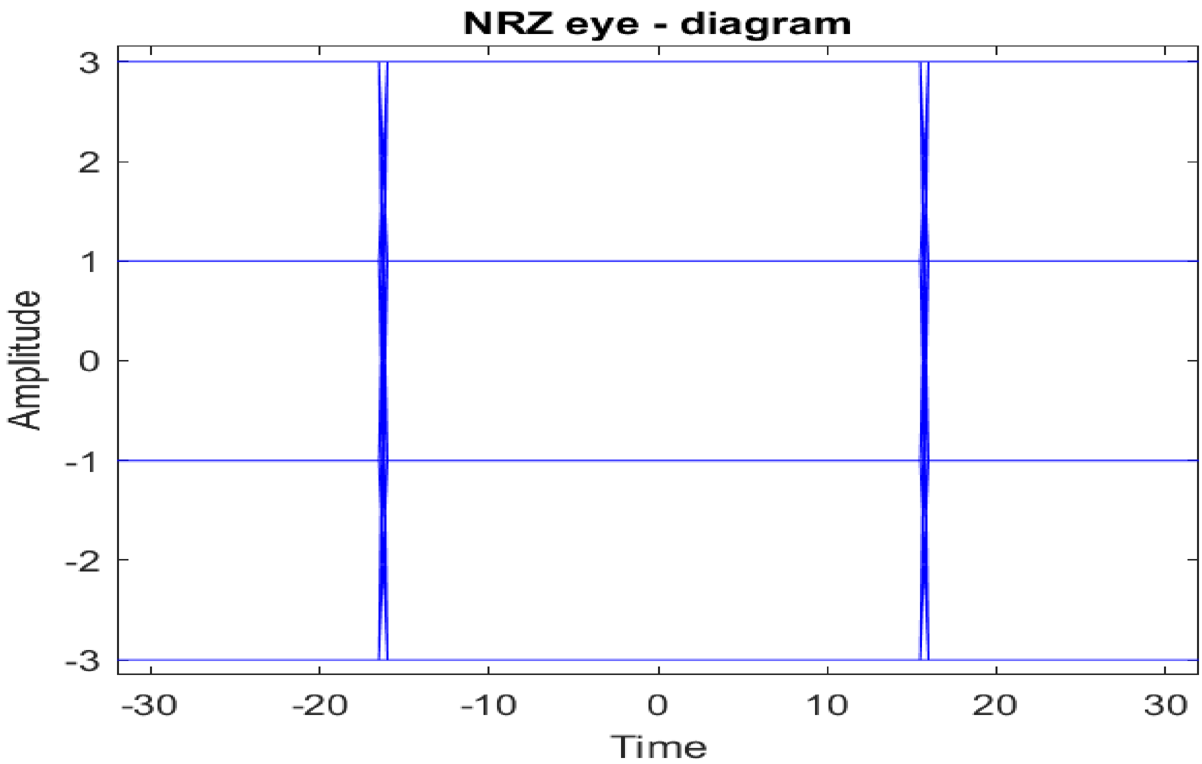
- Prcos:-

```
% ( prcos . m)
% Usage y=prcos ( rollfac , length , T)
function y=prcos ( rollfac , length, T)
    % rol l fac = 0 to 1 is the rol loff factor
    % l ength is the onesided pul se l ength in the
    number of T
    % l ength = 2 T+ l ;
    % T i s the oversampling rate
    y= rcosfir(rollfac,length,T,1,'normal') ;
end
```

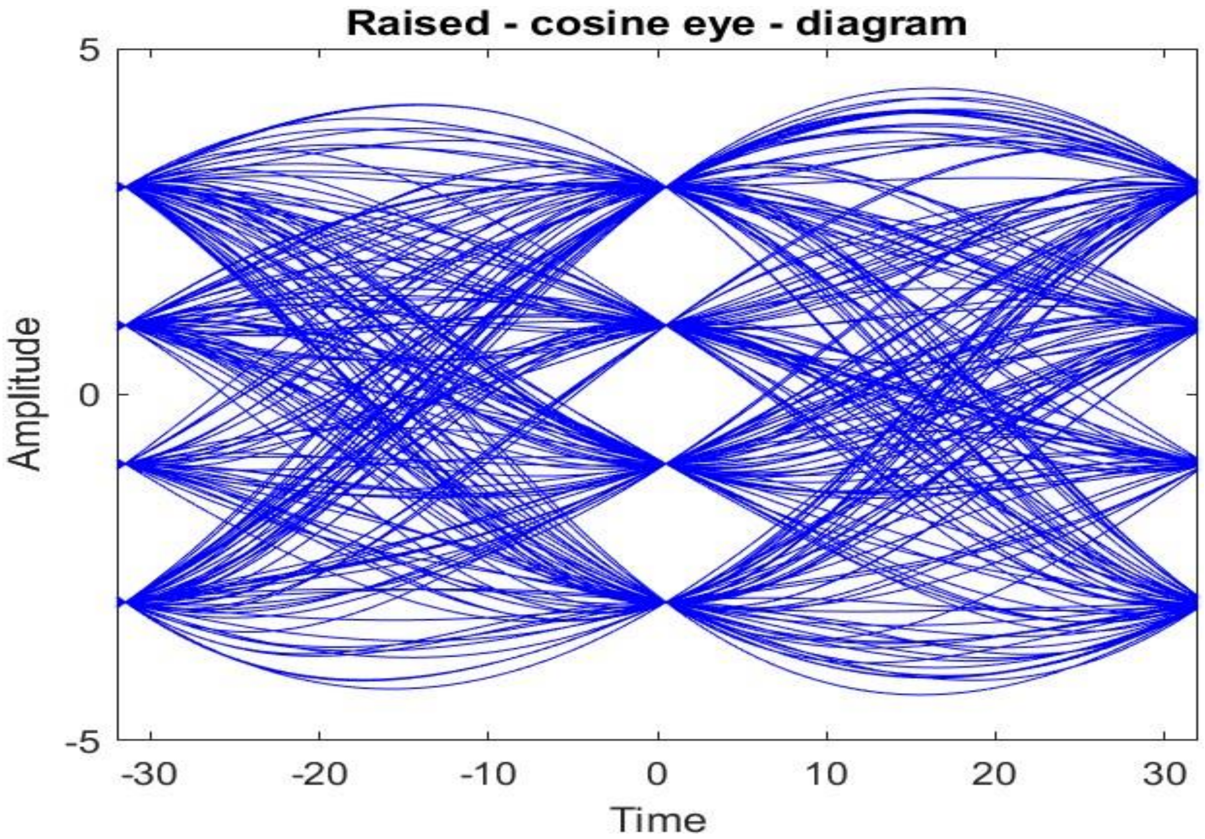
1.2 For a = 0 ,



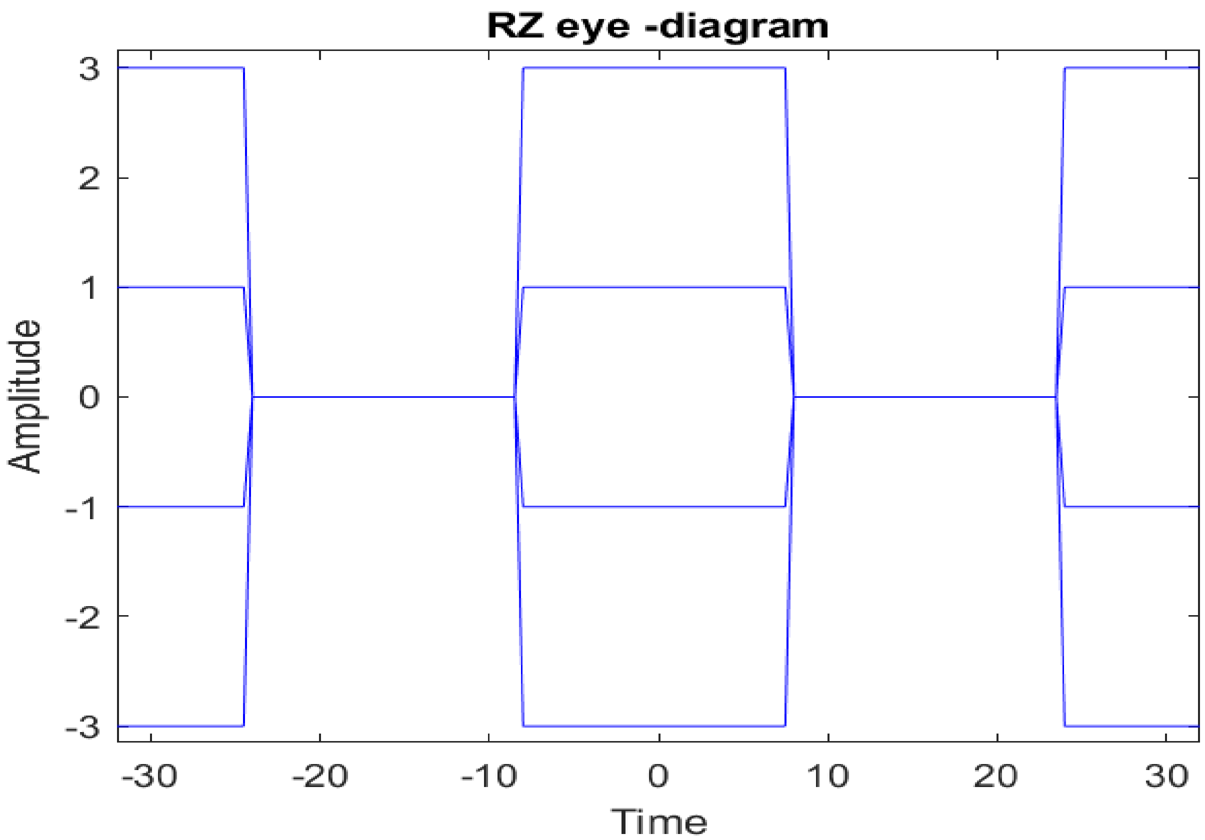
Half - Sine Eye - Diagram for a = 0



NRZ Eye - Diagram for a = 0

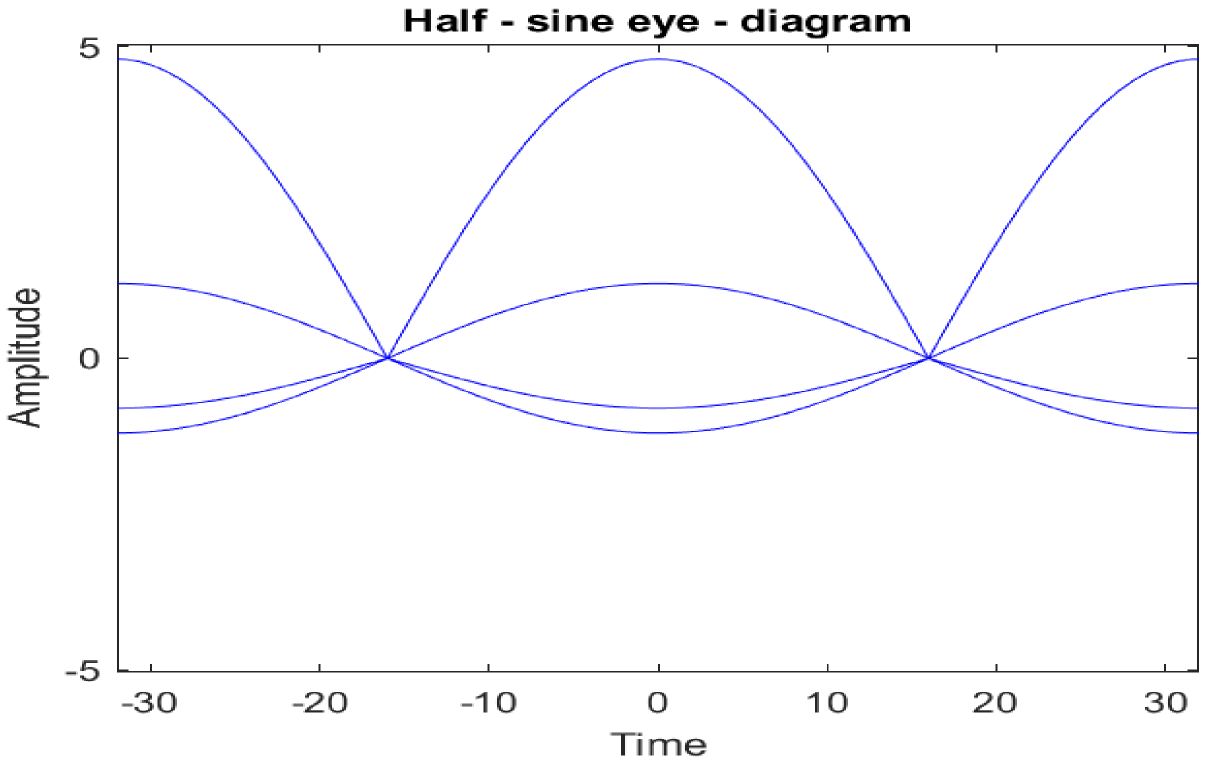


Raised - Cosine Eye - Diagram for a = 0

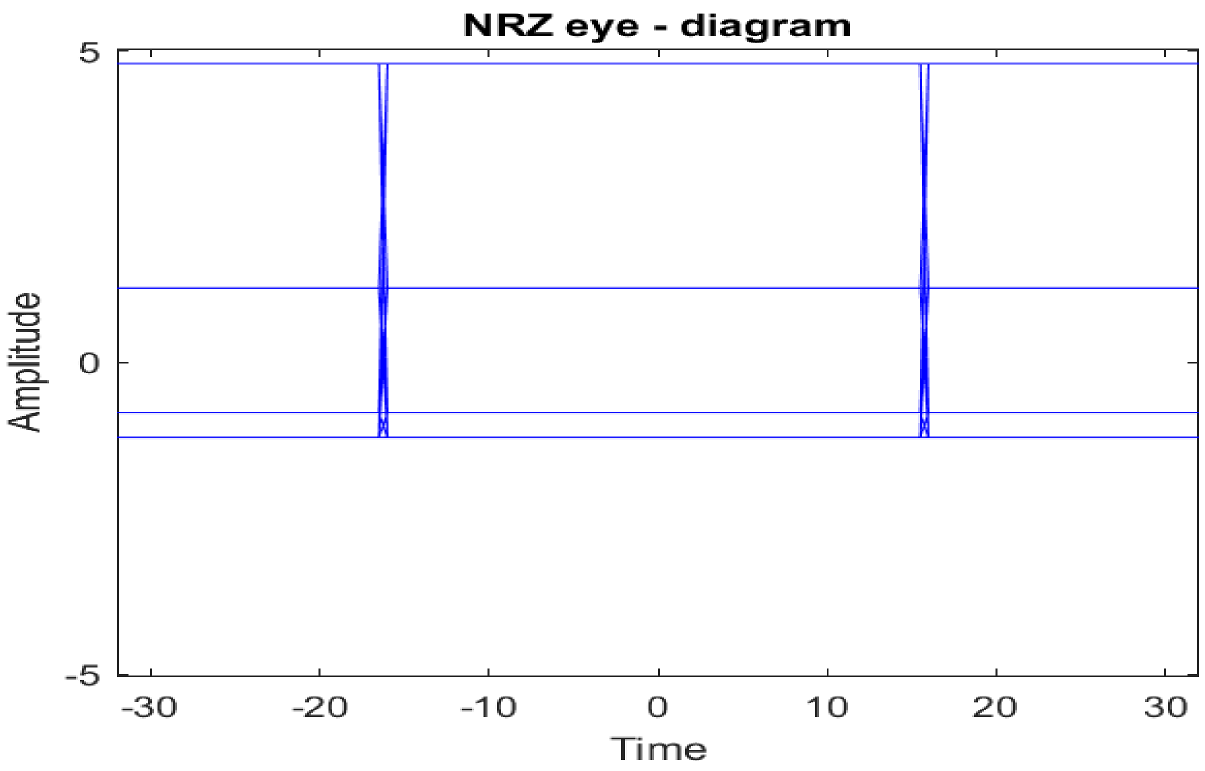


RZ Eye - Diagram for a = 0

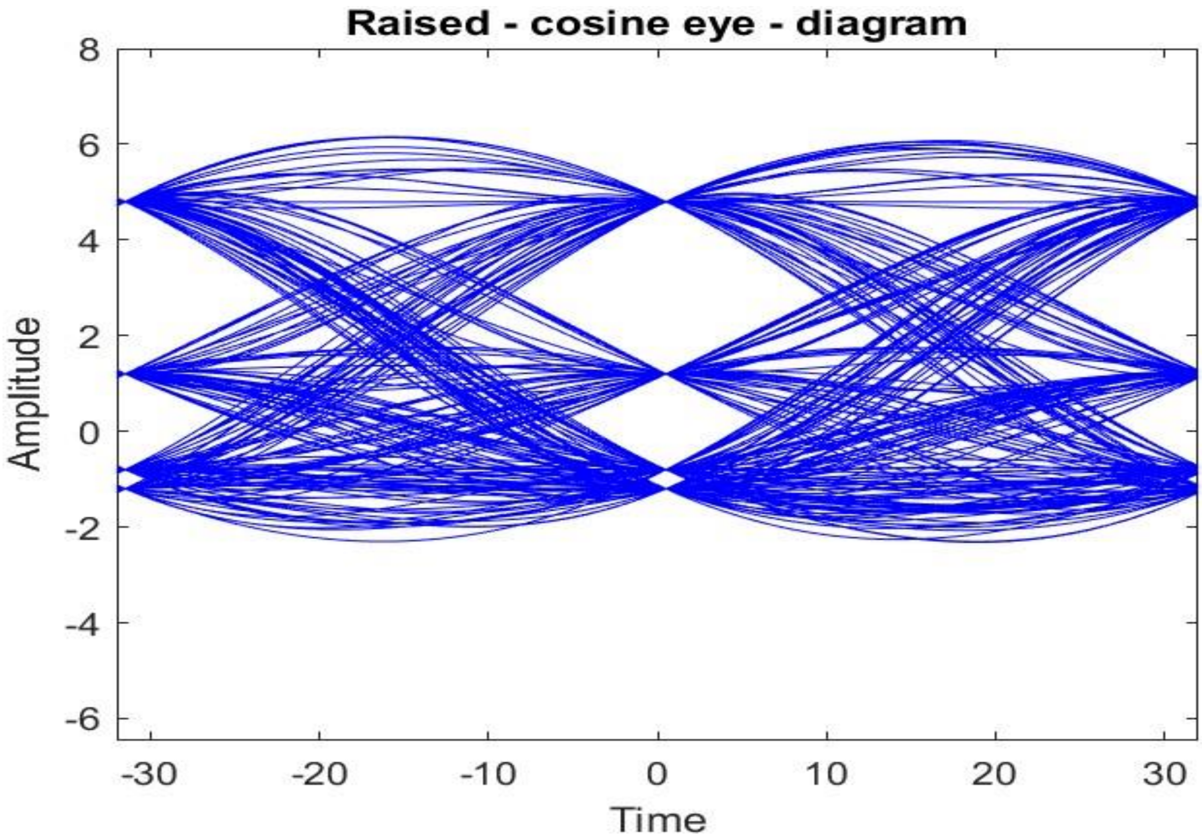
1.3 For a = 0.05 ,



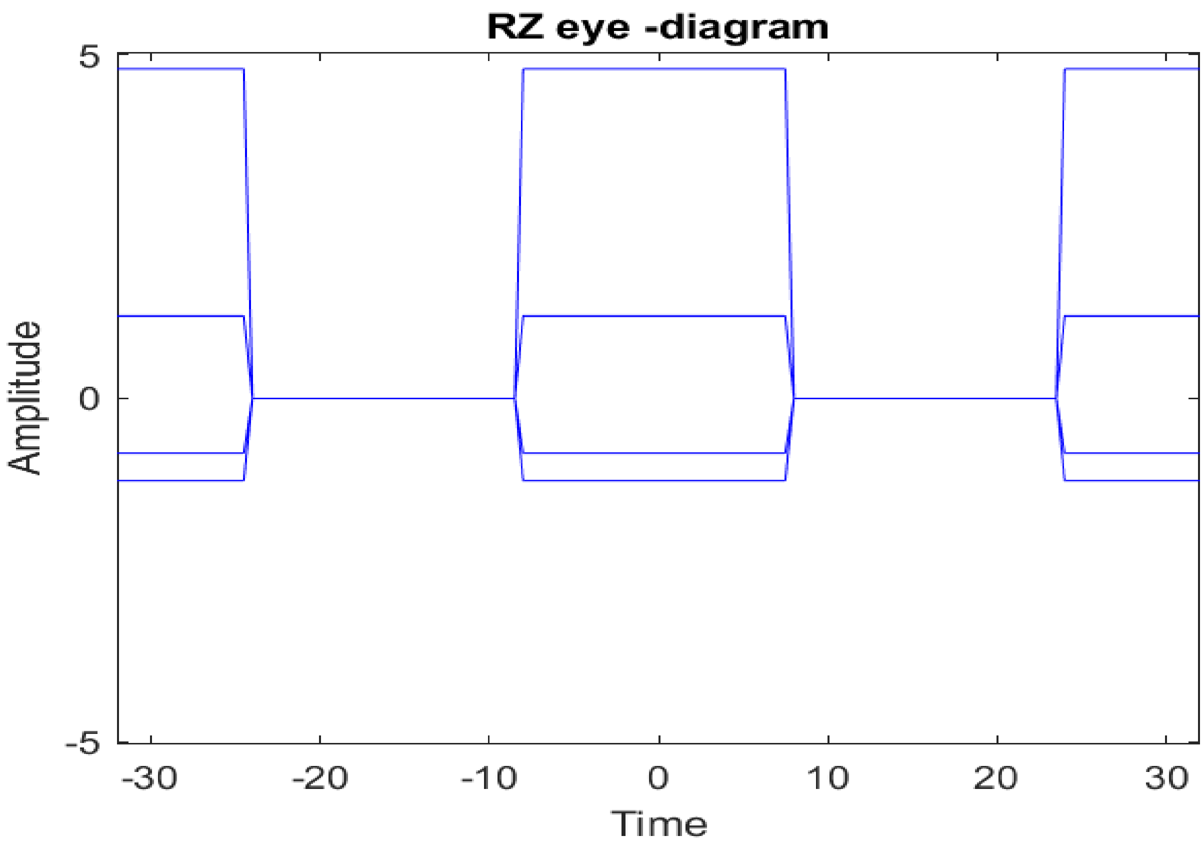
Half - Sine Eye - Diagram for a = 0.05



NRZ Eye - Diagram for a = 0.05

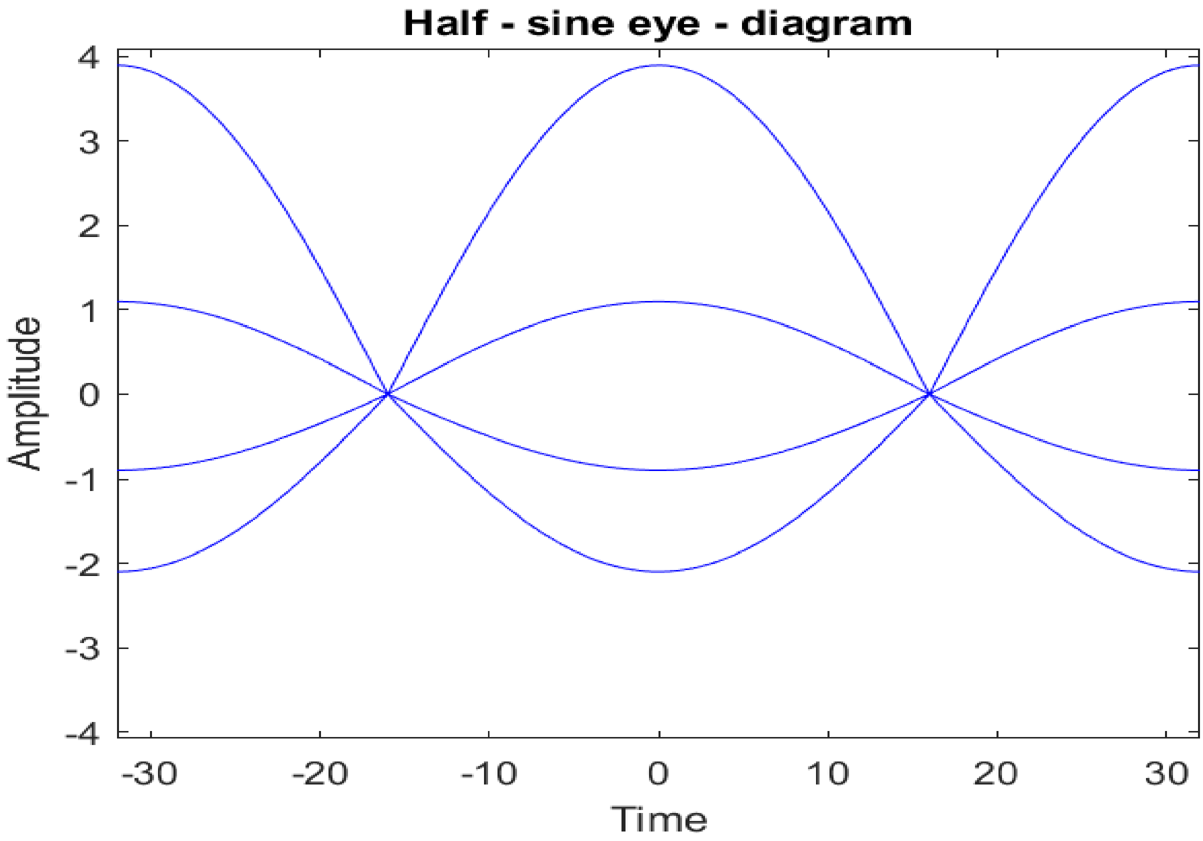


Raised - Cosine Eye - Diagram for a = 0.05

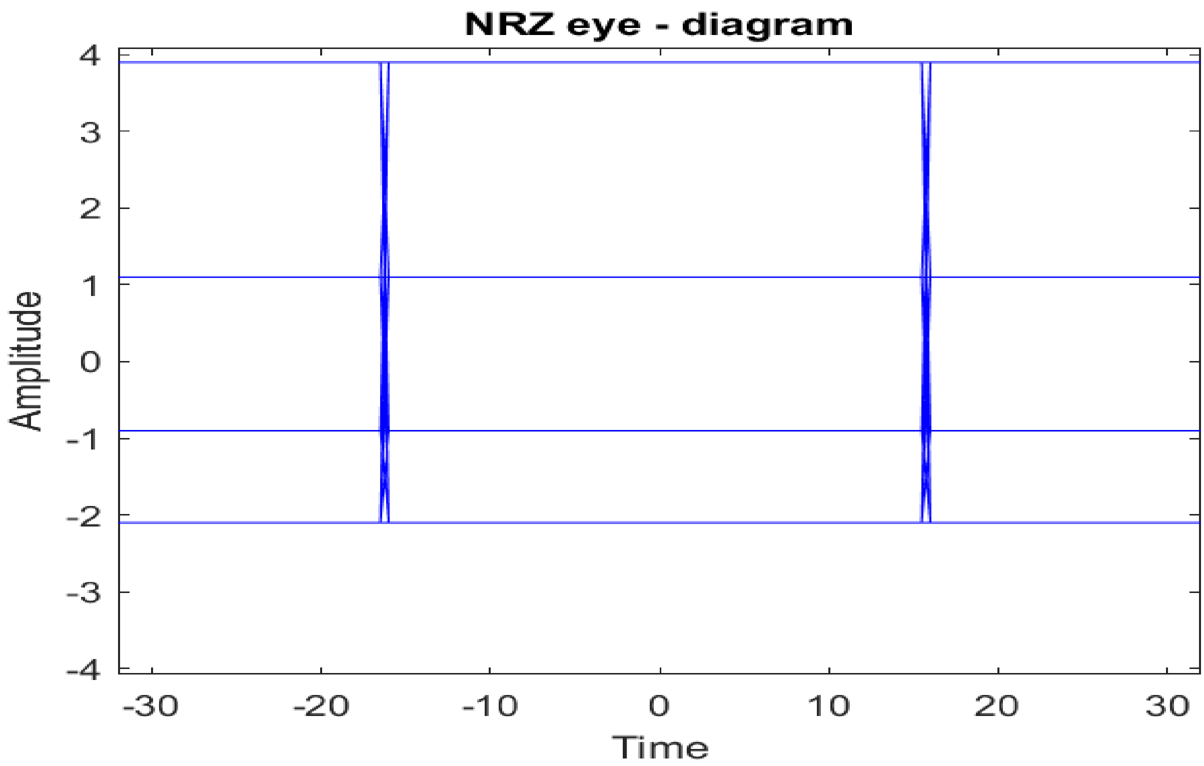


RZ Eye - Diagram for a = 0.05

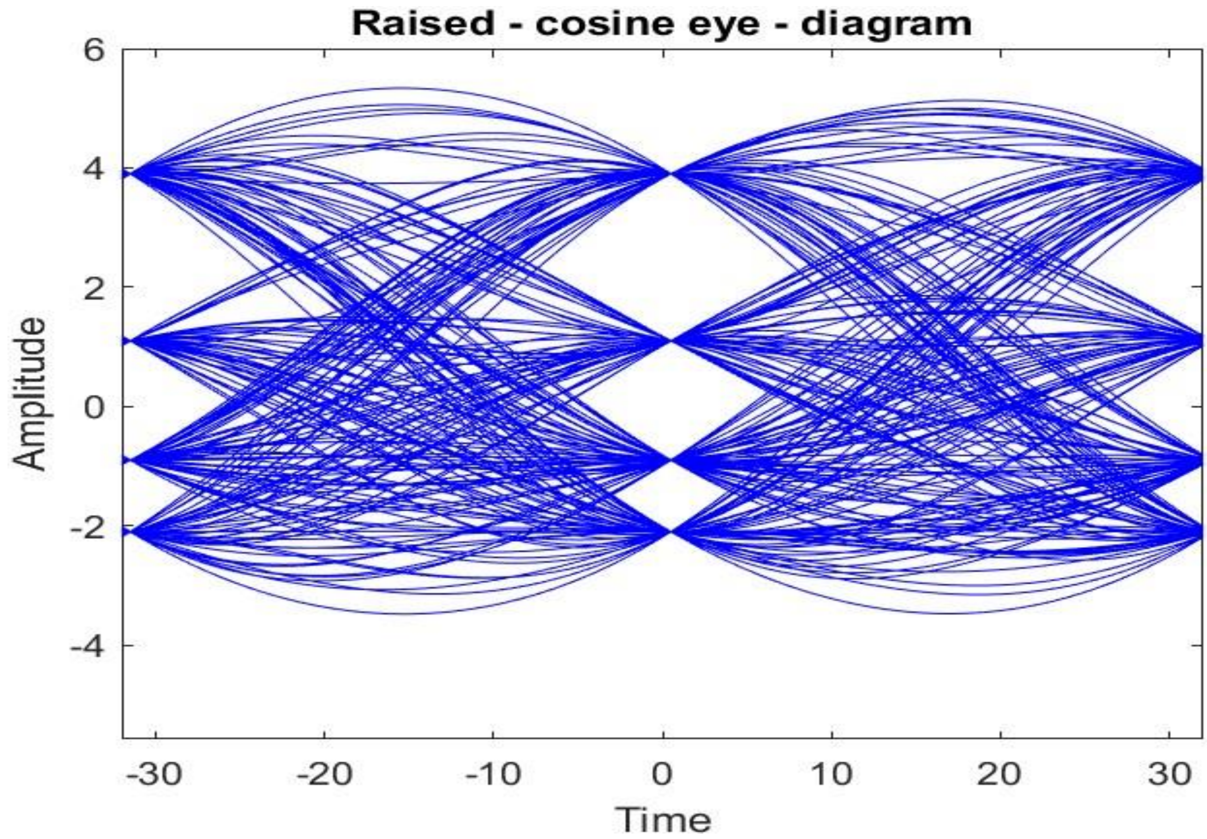
1.4 For a = 0.1 ,



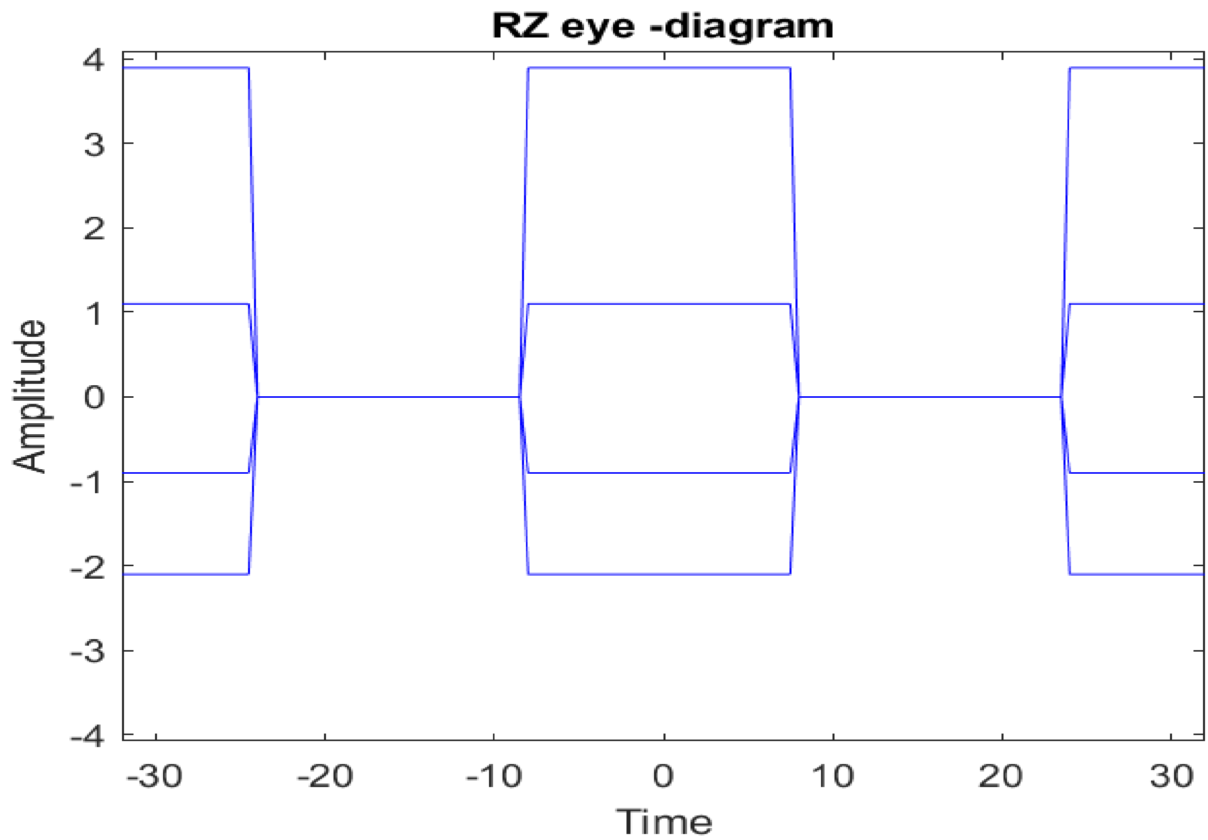
Half - Sine Eye - Diagram for a = 0.1



NRZ Eye - Diagram for a = 0.1

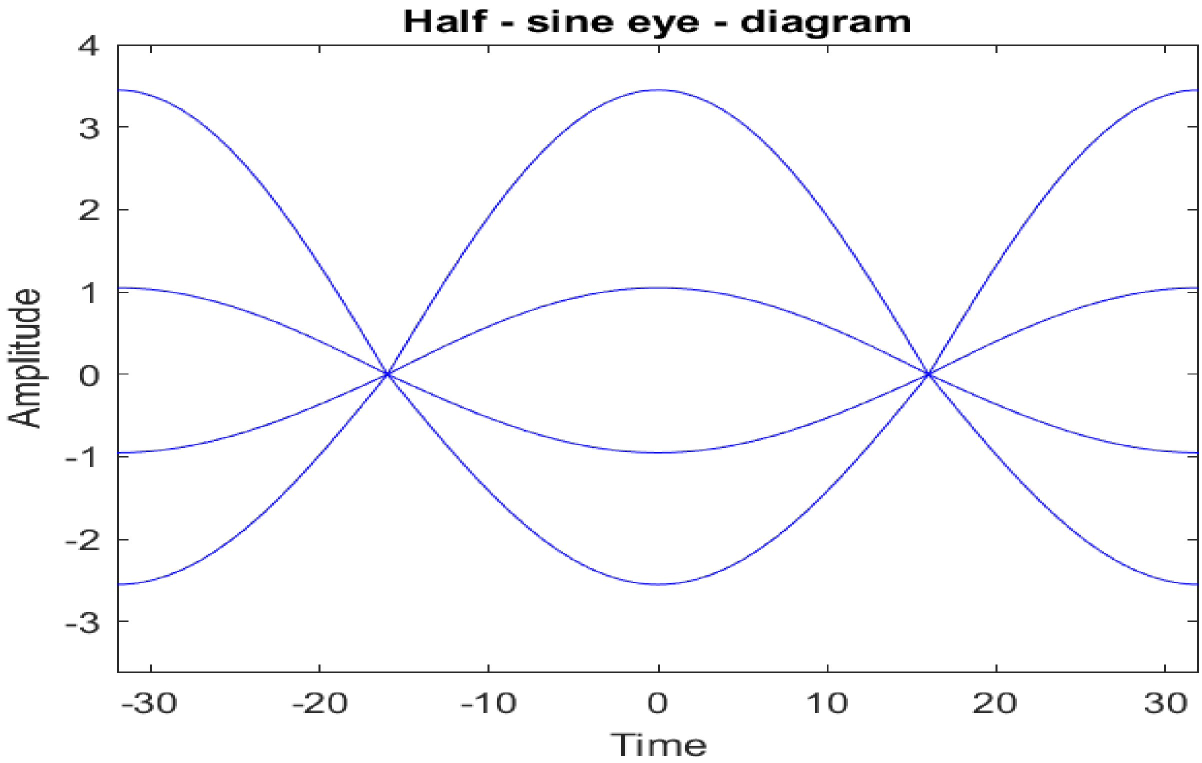


Raised - Cosine Eye - Diagram for a = 0.1

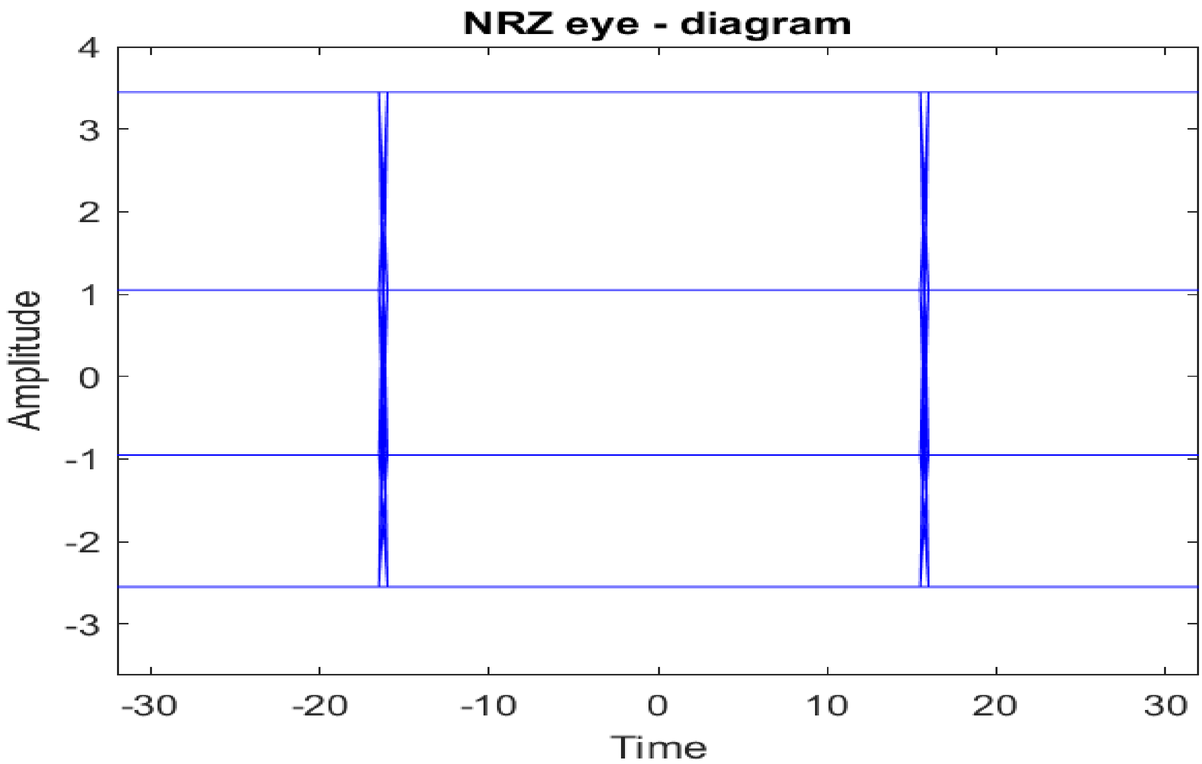


RZ Eye - Diagram for a = 0.1

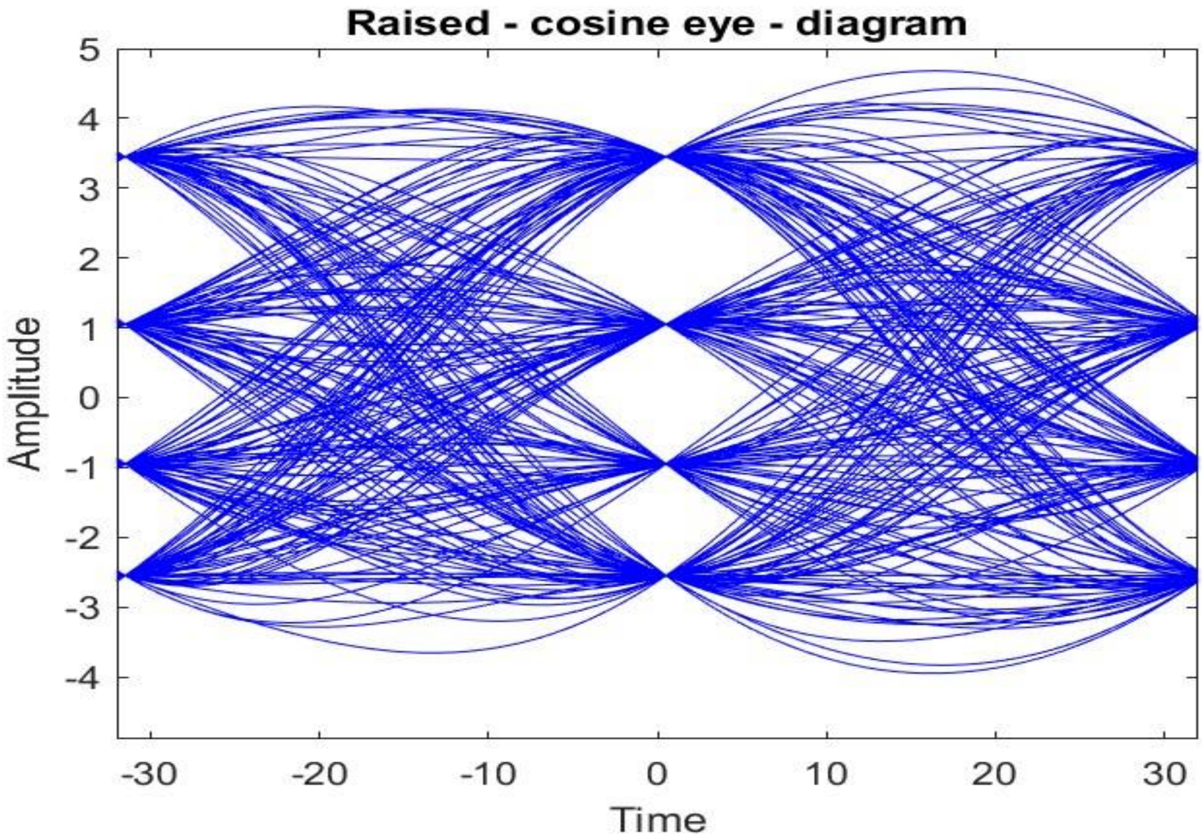
1.5 For a = 0.2 ,



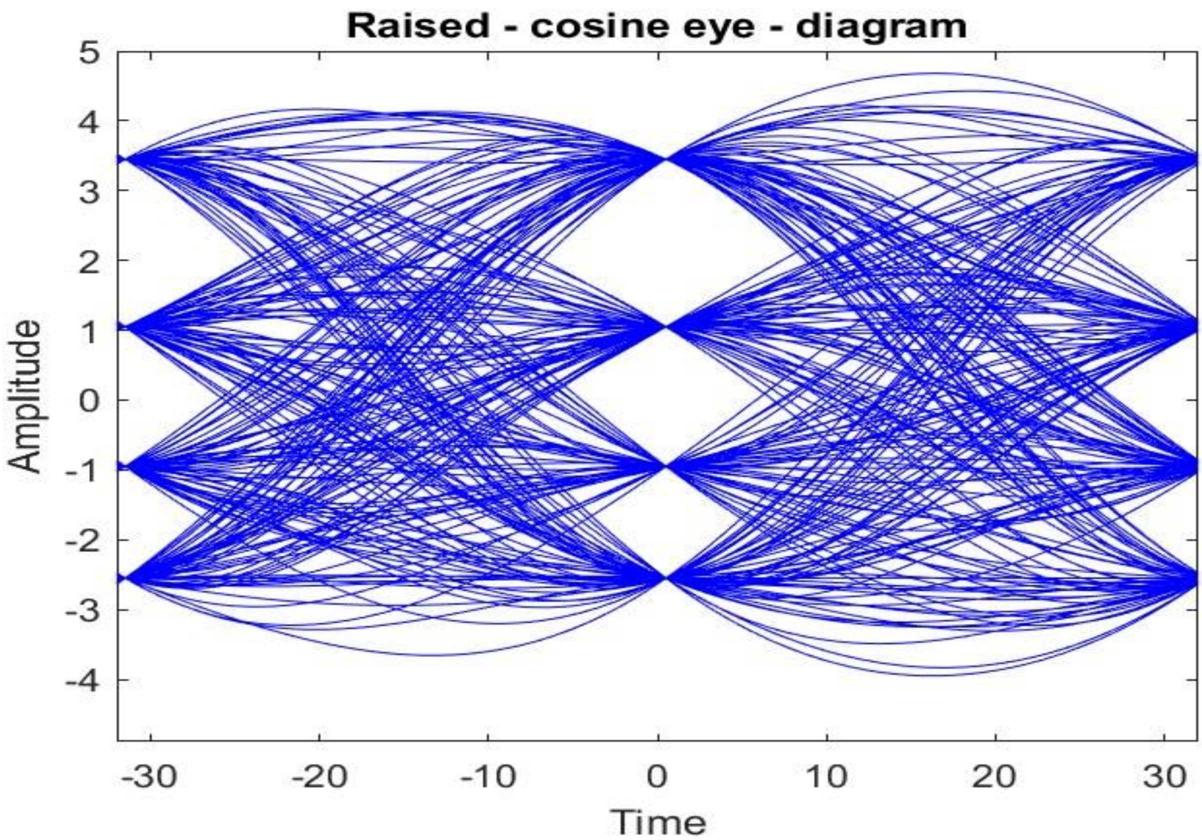
Half - Sine Eye - Diagram for a = 0.2



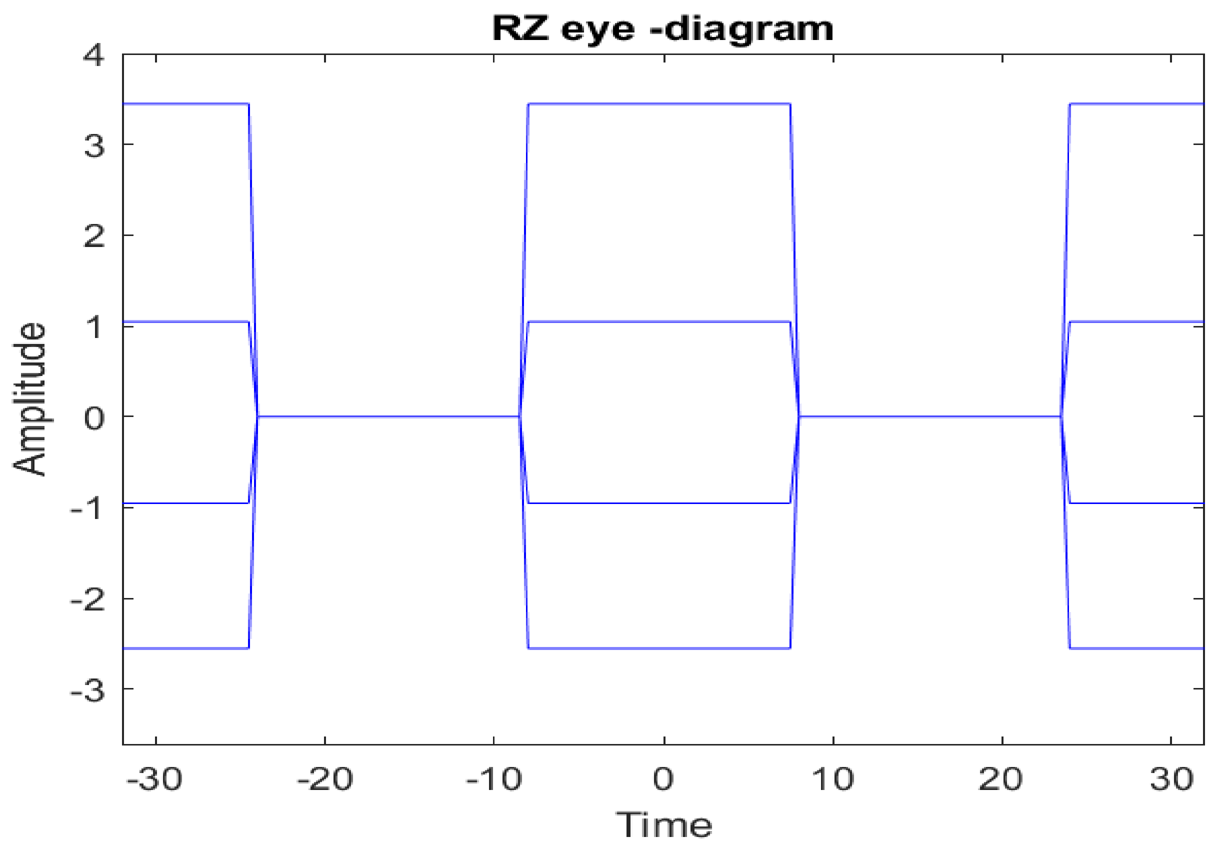
NRZ Eye - Diagram for a = 0.2



Raised - Cosine Eye - Diagram for a = 0.2



RZ Eye - Diagram for a = 0.2



Eye - Diagram for a = 0.2

1.6 Observation : -

- It becomes difficult to separate symbols with values -1 and -3.
- Here as the value of a is increasing the openings of the eye is decreasing.

2. DPCM Scheme

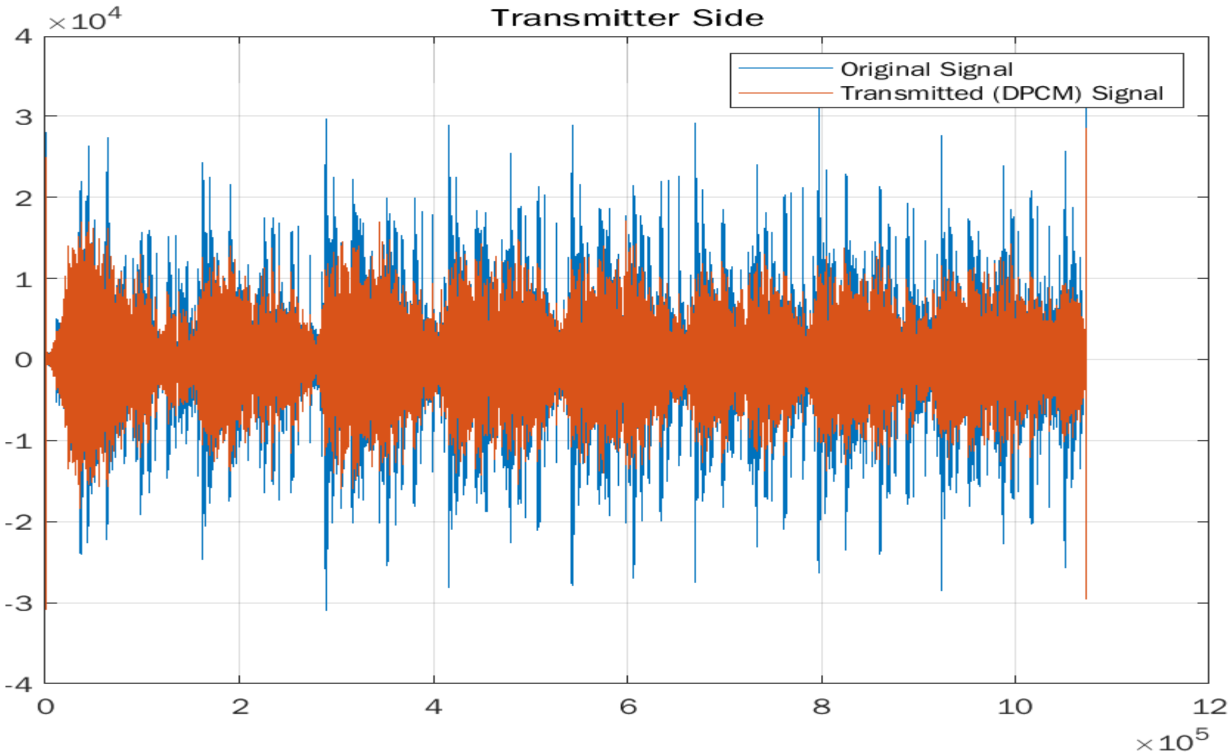
2.1 Code :-

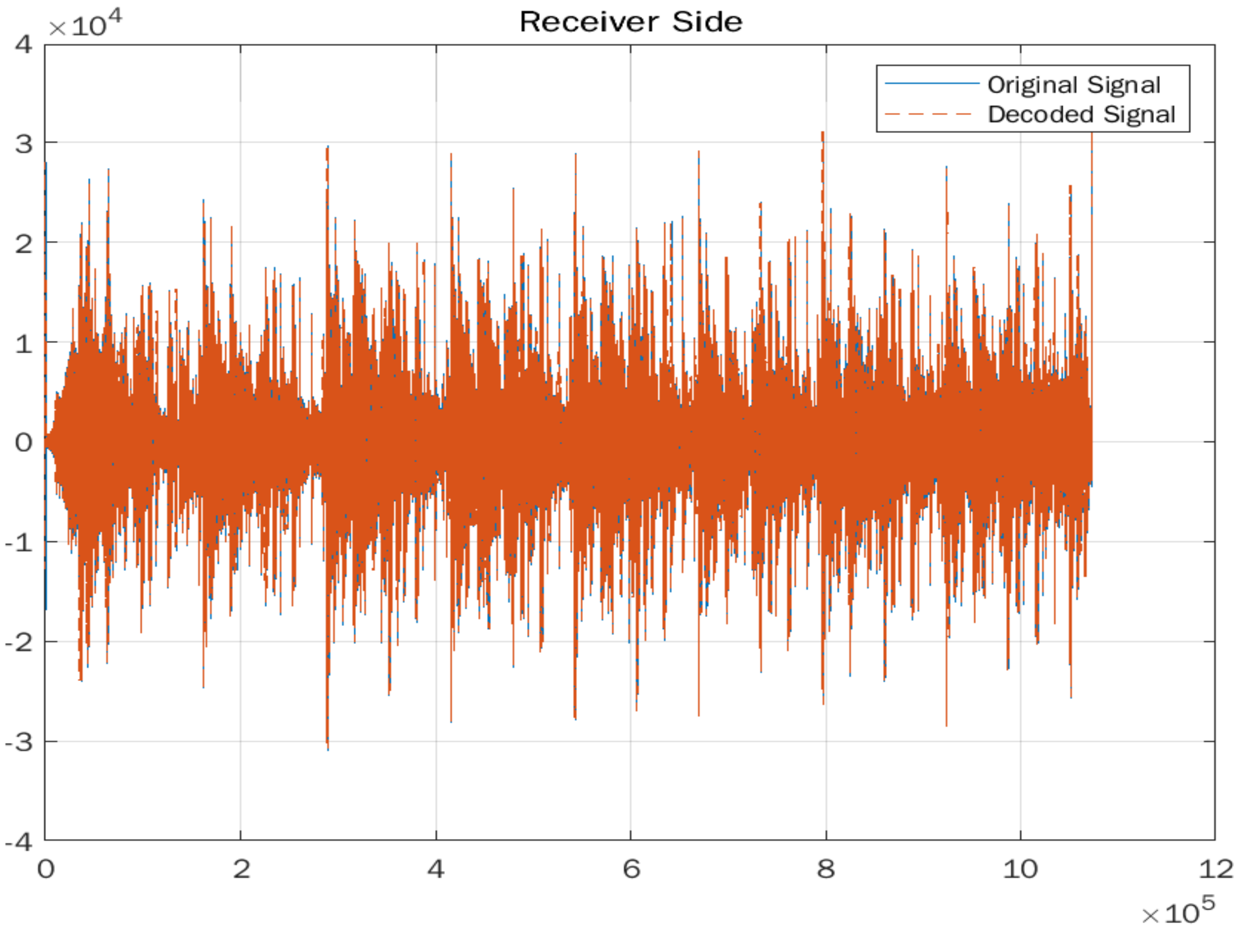
```
clc;
clearvars;
close all;
s = fopen('test.wav','r');
s = fread(s,'int16');
s = s';
mp = max(s);
mn = min(s);
%  Transmitter (DPCM)
N=8;
d = zeros(1,length(s));
d_q = zeros(1,length(s));
```

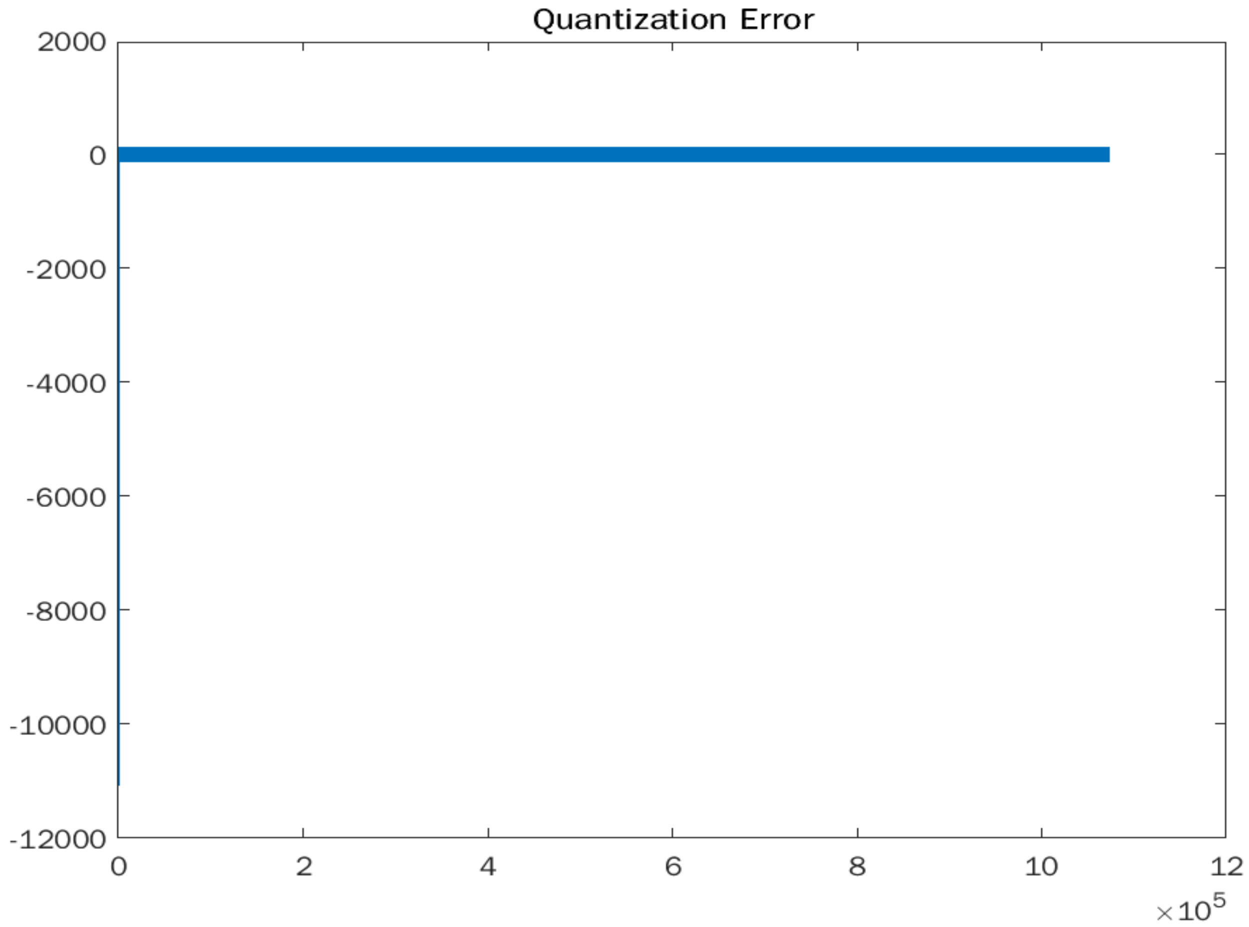
```
s_q = zeros(1,length(s));
for n=1:length(s)
   if n==1
       d(n)=s(n);
       d_q(n)=quantizer(d(n),N,mn,mp);
       s_q(n)=d_q(n);
   else
       d(n)=s(n)-s_q(n-1);
       d_q(n)=quantizer(d(n),N,mn,mp); %Transmitted
signal
       s_q(n)=d_q(n)+s_q(n-1);
   end
end
%Original Signal and Transmitted DPCM signal
figure(1);
plot(s);
hold on;
grid on;
plot(d_q);
legend('Original Signal','Transmitted (DPCM) Signal');
title('Transmitter Side');
% Reciever (DPCM)
s_qr = zeros(1,length(s));
for n=1:length(s)
   if n==1
       s_qr(n)=d_q(n);
   else
       s_qr(n)=d_q(n)+s_qr(n-1);
   end
end
%Plot Original Signal and Decoded signal at receiver
figure(2);
plot(s);
hold on;
grid on;
plot(s_qr,'--');
legend('Original Signal','Decoded Signal');
title('Receiver Side');
% Quantization error
sqnr=20*log10(norm(s)/norm(s-s_qr));
figure(3);
plot(s-s_qr);
title('Quantization Error');
function y_final = quantizer(y,N,min_value,max_value)
% Need to restrict y inside [min_val,max_val]
   if y < min_value
       y = min_value;
```

```
    end
    if y > max_value
        y = max_value;
    end
    % Find area width step D
    D = max_value/(2^(N-1));
    % Find the centers
    centers = zeros(2^N,1);
    centers(1) = max_value - D/2;
    centers(2^N) = min_value + D/2;
    for i=2:2^N-1
        centers(i) = centers(i-1) - D;
    end
    y_final = 1;

    % Find the range of y
    for i=1:2^N
        if ((y <= centers(i)+D/2) && (y>= centers(i)-D/2))
            y_final = i;
        end
    end
    % Find quantified sample
    y_final = centers(y_final);
end
```

2.2 Output :-


Transmitter Side
Original Signal
Transmitted (DPCM) Signal
×10^4
×10^5


Receiver Side
Original Signal
Decoded Signal
×10^4
×10^5


Quantization Error
2000
0
-2000
-4000
-6000
-8000
-10000
-12000
0
2
4
6
8
10
12
$\times 10^5$

Lab 8

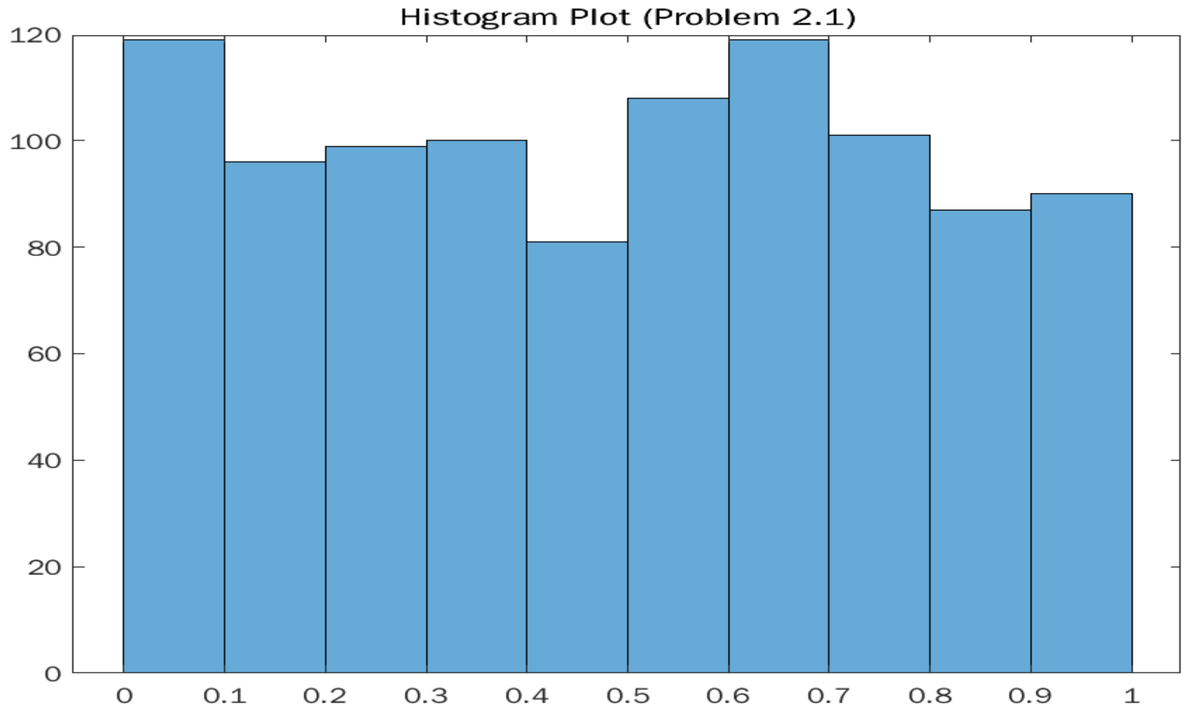
3. Generation of Random Variables

3.1 Question 1

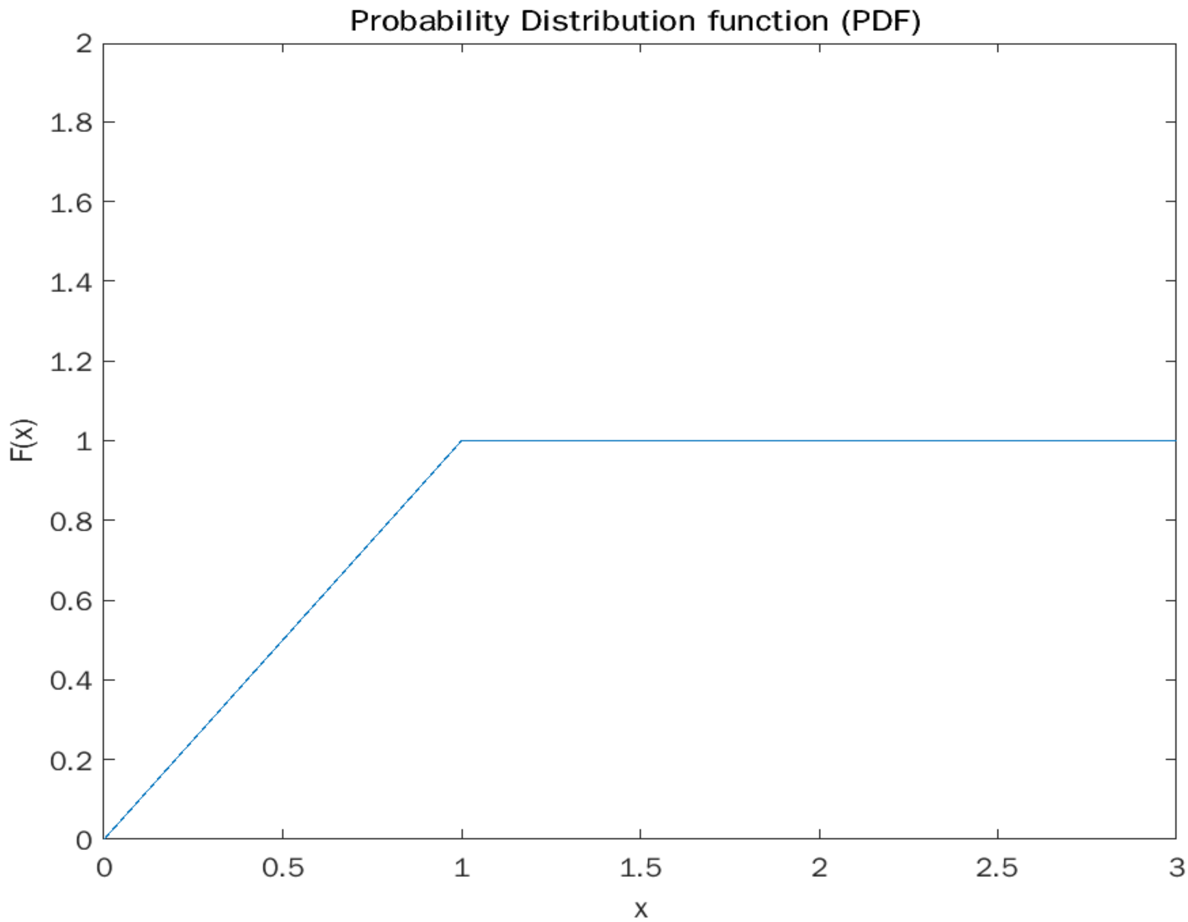
3.1.1 Problem 2.1

Code :-

```
%Problem 2.1  Generating values following Uniform Random
Distribution clc;
clearvars;
close all;
x = rand(1,1000);
%Probability Distribution Function y = 0:0.1:1;
xaxis = [ y 2:3];
y = [ y ones(1,2)];

figure(1);
histogram(x,10);
title('Histogram Plot (Problem 2.1)');

figure(2);
plot(xaxis,y);
title('Probability Distribution function (PDF) ');
xlim([0,3]);
ylim([0,2]);
xlabel('x');
ylabel('F(x)');
```



Histogram Plot (Problem 2.1)
120
100
80
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

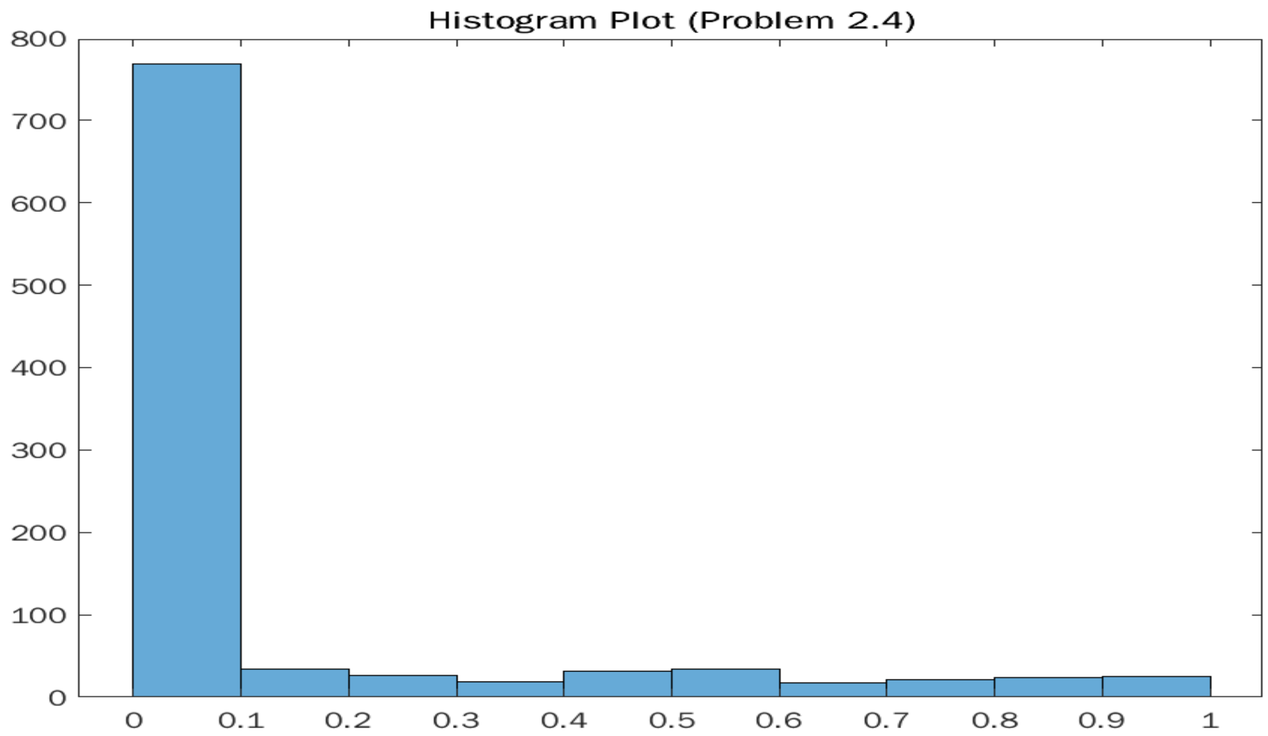


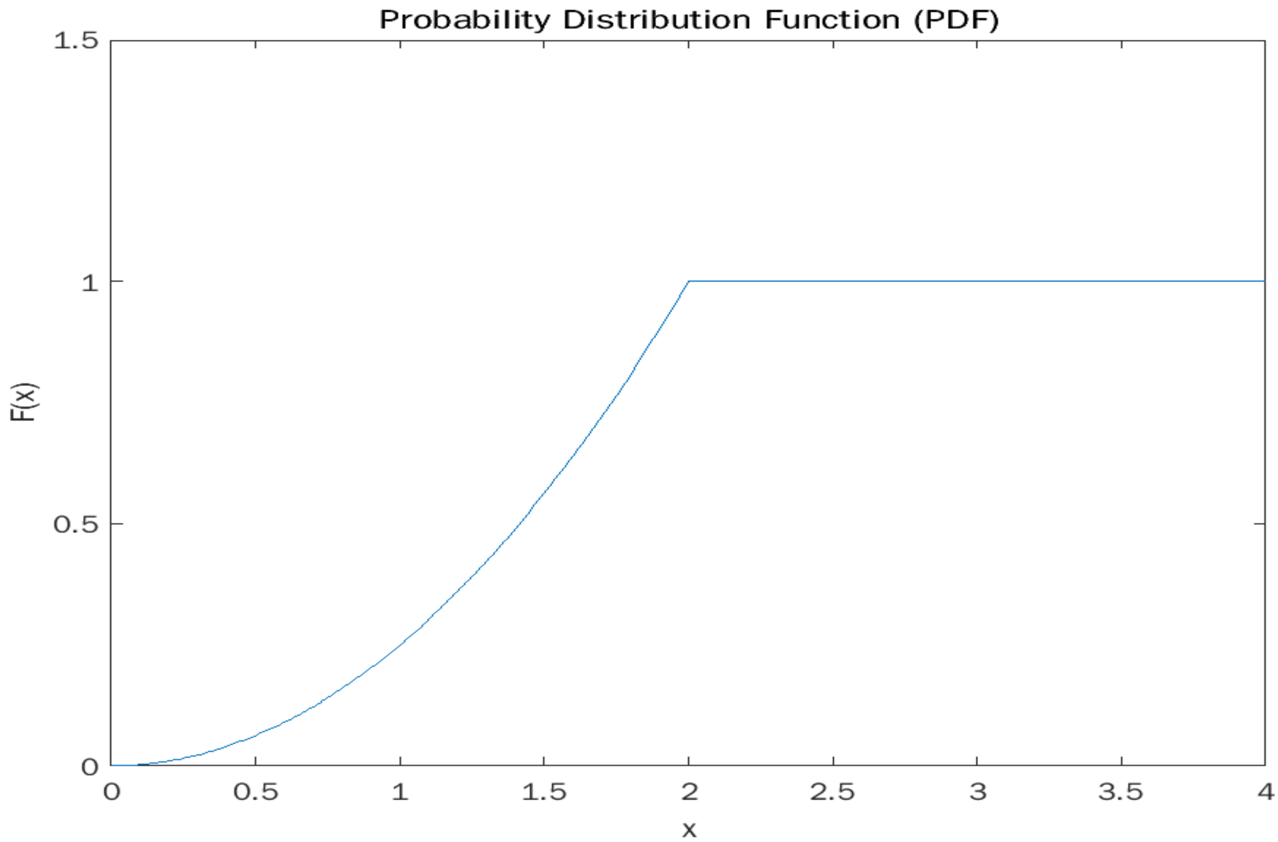
Probability Distribution function (PDF)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
F(x)
0
0.5
1
1.5
2
2.5
3
x

3.1.2 Problem 2.4

- **Code :-**

```
%% Problem 2.4 %Generating values following Uniform Random
Distribution
clc;
clearvars;
close all;
x = (8)*rand(1,1000) + (-4);
for i=1:length(x)
   if (x(i)>=0 && x(i)<=2)
       x(i) = x(i)/2;
   else
       x(i) = 0;
   end
end
ind = 1;
%Probability Distribution Function
for i=0:0.01:4
   if (i>=0 && i<=2)
      y(ind) = i^2/4;
   else
      y(ind) = 1;
end
xaxis(ind) = i;
ind = ind + 1;
end

figure(1);
histogram(x,10);
title('Histogram Plot (Problem 2.4)');

figure(2);
plot(xaxis,y);
title('Probability Distribution Function (PDF)');
xlabel('x');
ylabel('F(x)');
ylim([0,1.5]);
```


Histogram Plot (Problem 2.4)
800
700
600
500
400
300
200
100
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Probability Distribution Function (PDF)
1.5
1
0.5
0
F(x)
0
0.5
1
1.5
2
2.5
3
3.5
4
x

3.1.3 Problem 2.7

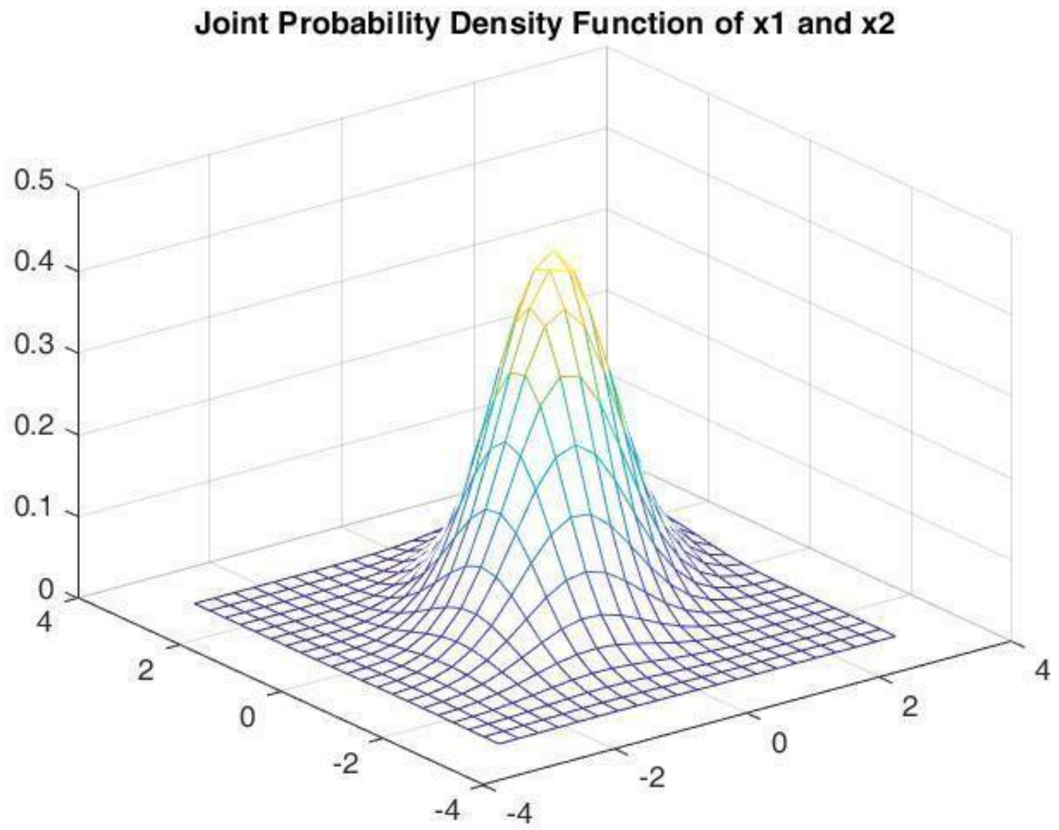
➢ **Code** :-

```
%% Problem 2.7
clear
echo on
 mx=[1/2 1/2]';
Cx=[1 1/2;1/2 1];

%Generating pairs (x1,x2)
%And calculating mean m1 and m2
x = zeros(2,1000);
m1 = 0;
m2 = 0;
for i=1:1000
        x(:,i) = multi_gp(mx,Cx);
        m1 = m1 + x(1,i);
        m2 = m2 + x(2,i);
        end
m1 = m1/1000;
m2 = m2/1000;

%Calculating Variance sigma1 and sigma2
%And covariance COVsigma1 = 0;
sigma2 = 0;
COV = 0;
for i=1:1000
        sigma1 = sigma1 + (x(1,i)-m1)^2;
         sigma2 = sigma2 + (x(2,i)-m2)^2;
         COV = COV + (x(1,i)-m1)*(x(2,i)-m2);
end
sigma1 = sigma1/1000;
sigma2 = sigma2/1000;
COV = COV/1000;

% Computation of the pdf of (x1,x2) follows.
delta=0.3;
x1=-3:delta:3;
x2=-3:delta:3;
```

```
 for i=1:length(x1)
     for j=1:length(x2)
         f(i,j)=(1/((2*pi)*det(Cx)^1/2))*exp((-1/2)*(([x1(i)
         x2(j)]-mx')*inv(Cx)*([x1(i);x2(j)]-mx)));
         echo off ;
     endend
% Plotting command for pdf follows.
figure(1);
mesh(x1,x2,f);
title('Joint Probability Density Function of x1 andx2');
function [x] = multi_gp(m,C)
%   [x]=multi_gp(m,C)
%   MULTI_GP   generates a multivariate Gaussianrandom
%   process with mean vector m (column vector)and covariance matrix C.
N=length(m);
for i=1:N
      y(i)=gngauss;
end
y=y.';
x=sqrtm(C)*y+m;
```

```
function [gsrv1,gsrv2]=gngauss(m,sgma)
%      [gsrv1,gsrv2]=gngauss(m,sgma)
%      [gsrv1,gsrv2]=gngauss(sgma)
%      [gsrv1,gsrv2]=gngauss
%      GNGAUSS   generates two independent Gaussianrandom variables with mean
%      m and standard deviation sgma. If one of theinput arguments is missing,
%      it takes the mean as 0.
%      If neither the mean nor the variance isgiven, it generates two standard
%      Gaussian random variables.
if nargin == 0
  m=0;
  sgma=1;
  elseif nargin == 1
  sgma=m; m=0;end
u=rand;                                  % a uniform random
variable in (0,1)
z=sgma*(sqrt(2*log(1/(1-u))));           % a Rayleighdistributed
```

```
random variable
u=rand;                                          % another uniform
random variable in (0,1)
gsrv1=m+z*cos(2*pi*u);
gsrv2=m+z*sin(2*pi*u);
end
```



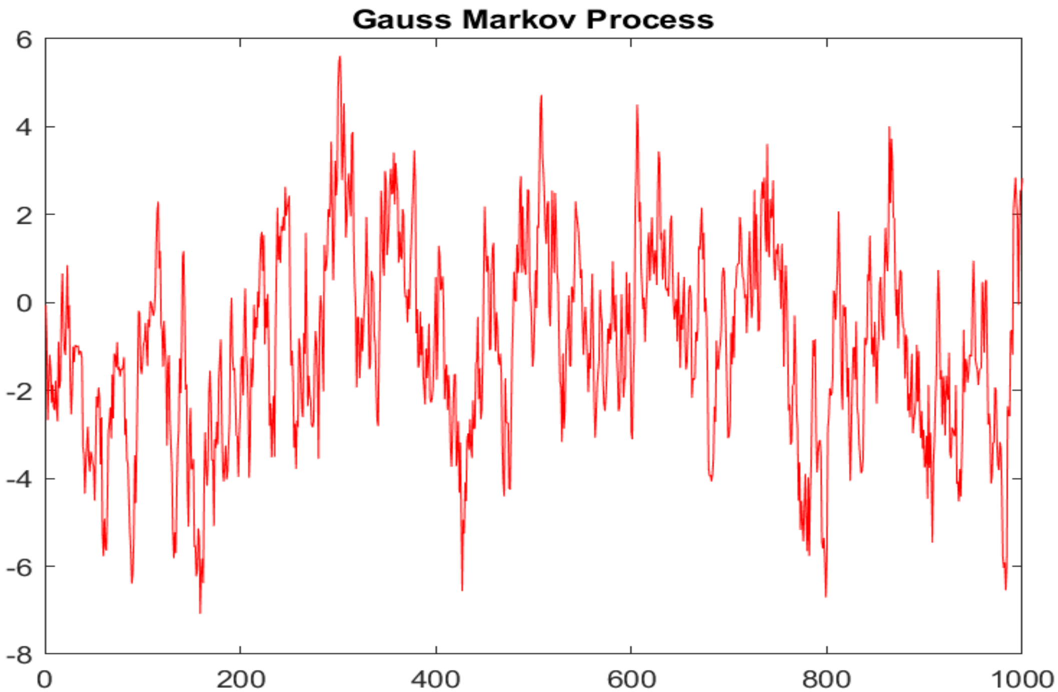
- **Observations :-**
  1. Mean of variable x1, m1 = 0.5168
  2. Mean of variable x2, m2 = 0.4865
  3. Variance of variable x1, Variance x1 = 1.061
  4. Variance of variable x2, Variance x2 = 1.043
  5. Covariance of x1 and x2, Covariance = 0.5503

3.1.4 Problem 2.8

- **Code :-**

```
%% Problem 2.8 Gauss Markov Process
clear
echo on
rho=0.9;
X0=0;
N=1000;
X=gaus_mar(X0,rho,N);

figure(1);
plot(X);
title('Gauss Markov Process');

function [X]=gaus_mar(X0,rho,N)

    % [X]=gaus_mar(X0,rho,N)
    % The noise process is taken to be white Gaussian
    % noise with zero mean and unit variance.
    for i=1:2:N
            [Ws(i) Ws(i+1)]=gngauss; % Generate the noise.
    end
    X(1)=rho*X0+Ws(1); %first element in the Gauss--Markov process
    for i=2:N
            X(i)=rho*X(i-1)+Ws(i); % the remaining elements
    end
end
```

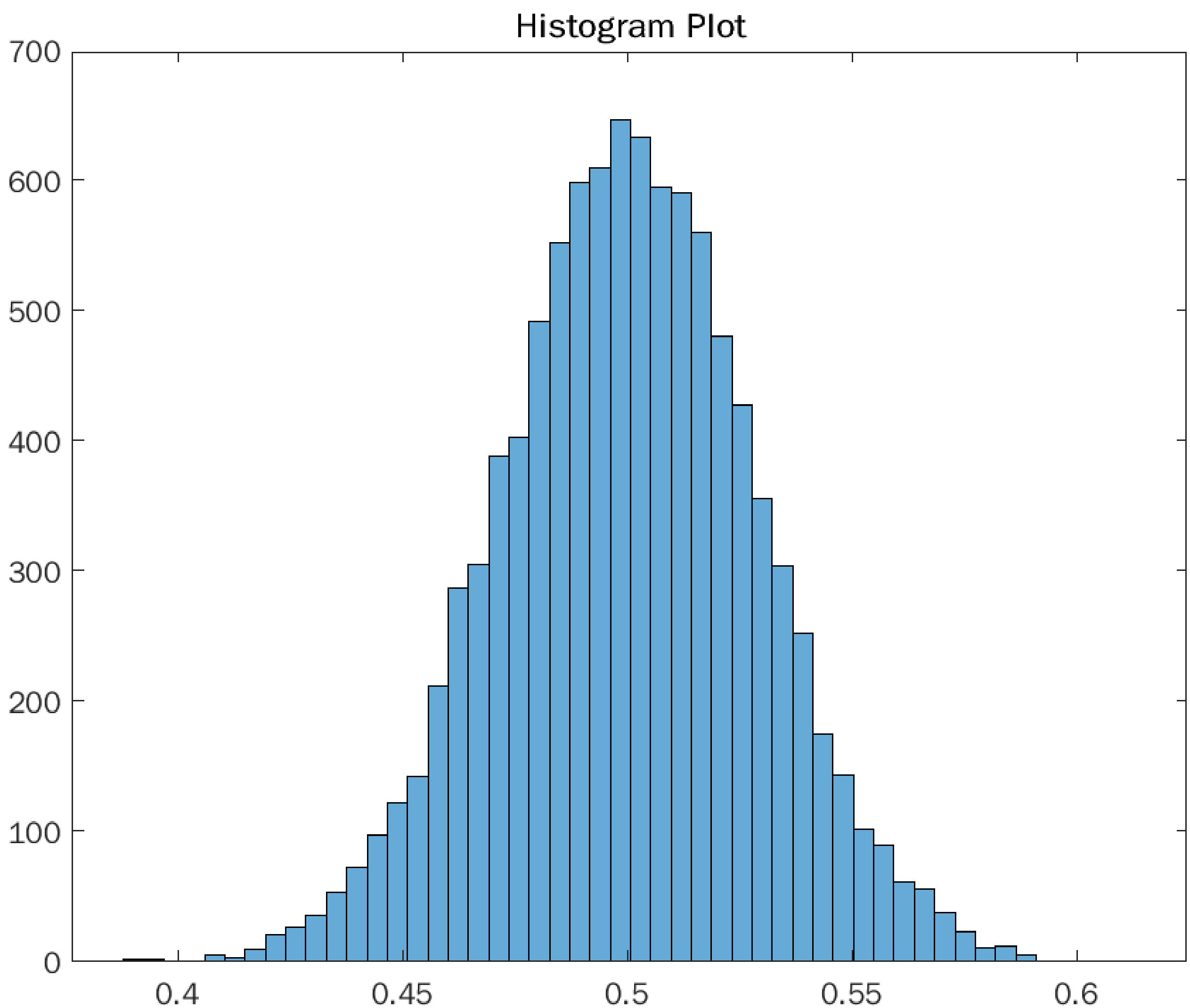
```
function  [gsrv1,gsrv2]=gngauss(m,sgma)
	%  [gsrv1,gsrv2]=gngauss(m,sgma)
	%  [gsrv1,gsrv2]=gngauss(sgma)
	%  [gsrv1,gsrv2]=gngauss
	% GNGAUSS generates two independent Gaussian random variables  with
	mean
	% m and standard deviation sgma. If one of the input arguments is missing,
	% it takes the mean as 0.
	% If neither the mean nor the variance is given, it generates two standard
	% Gaussian random variables.
	if nargin == 0
	   m=0;
	   sgma=1;
	elseif nargin == 1
	   sgma=m;
	   m=0;
	end

	u=rand; % a  uniform  random variable  in  (0,1)
	z=sgma*(sqrt(2*log(1/(1-u)))); % a Rayleighdistributed random variable
	u=rand; % another uniform random  variable  in  (0,1)
	gsrv1=m+z*cos(2*pi*u);
	gsrv2=m+z*sin(2*pi*u);
end
```



3.2 Question 2 :- Concept of Random Variable

➢ Code :-

```
%%Experiment 2
x = rand(1,1e6);
%   generate the sequence y oflength 100.
y = zeros(1,length(x)/100);
for i=1:length(x)/100
        y(i) = mean(x(100*(i-1)+1:100*i));
end

figure(1);
histogram(y,50);
title('Histogram Plot');
```



3.3 Observation :-

- The distribution curve of y is a Bell-shaped curve, as seen in the diagram above. This demonstrates that it has a Gaussian distribution.
- The central limit theorem states that if random variables X_i, $1 < i < n$, are independent and identically distributed, with finite mean and variance and n is large, then their average

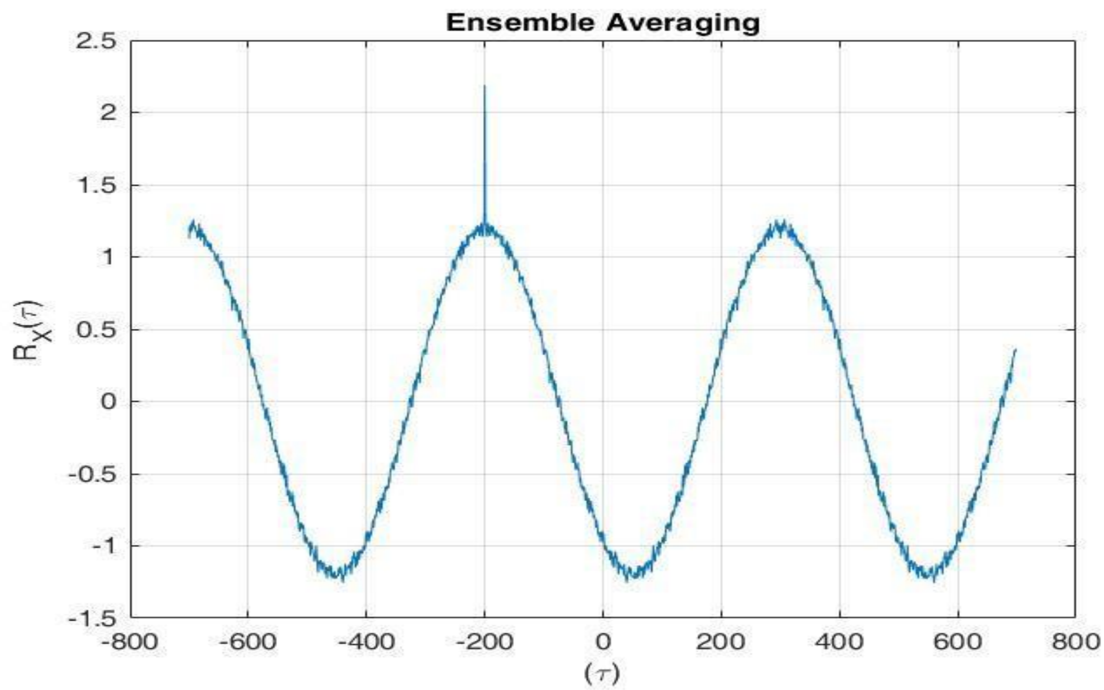
$$\text{(i.e., } Y = \frac{1}{N}\sum_{i=1}^{N} X_i\text{)}$$

  has roughly a Gaussian distribution.

- **Statistical Characterization of Random Process**

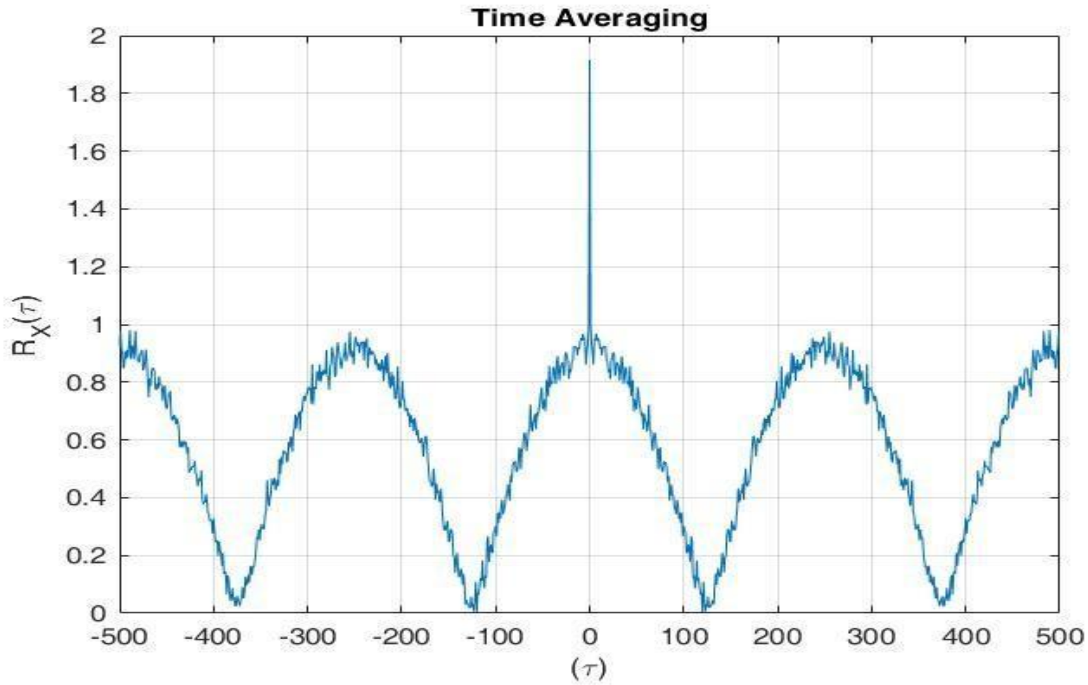
```
%% Experiment 3 (Ensemble Averaging)
clear all;
close all;
```

```
A = sqrt(2);
N=1000;
M=1;
SNRdb=0;
e_corrf_f = zeros(1,1400);
f_c=2/N; t=0:1:N-1;
for trial=1:M
    % signal
    s = cos(2*pi*f_c*t);
    %noise snr = 10^(SNRdb/10);
    wn = (randn(1,length(s)))/sqrt(snr)/sqrt(2);
    %signal plus noise
    s = s + wn; % autocorrelation
    [e_corrf] = en_corr(s,s,N);
    %Ensemble-averaged autocorrelation
    e_corrf_f = e_corrf_f + e_corrf;
end
%prints
figure(1);
plot(-700:700-1,e_corrf_f/M);
grid on;
title('Ensemble Averaging');
xlabel('(\tau)');
ylabel('R_X(\tau)');


function [corrf] = en_corr(u, v, N)
max_cross_corr = 0;
tt = length(u);

for m=0:tt
    shifted_u = [u(m+1:tt) u(1:m)];
    corr(m+1) = (sum(v.*shifted_u))/(N/2);
    if (abs(corr)>max_cross_corr)
        max_cross_corrs = abs(corr);
    end
end
corrl=flipud(corr);
corrf = [corrl(501:tt) corr(1:900)];
```




```
%% Experiment 3 (Time Averaging)
clear all; %close all;
A=sqrt(2);
N=1000;
SNRdb=0;
f_c=2/N;
t=0:1:N-1;
% signal
s = cos(2*pi*f_c*t);
%noise
snr = 10^(SNRdb/10);
wn = (randn(1,length(s)))/sqrt(snr)/sqrt(2);
%signal plus noise
s = s + wn;
% time - averaged autocorrelation
[e_corrf] = time_corr(s,N);
%prints

figure(2);
plot(-500:500-1,e_corrf);
grid on;
title('Time Averaging');
xlabel('(\tau)');
ylabel('R_X(\tau)');
```

```
function [corrf] = time_corr(s,N)
x=fft(s);
x1=fftshift((abs(x).^2)/(N/2));
corrf = fftshift(abs(ifft(x1)));
```



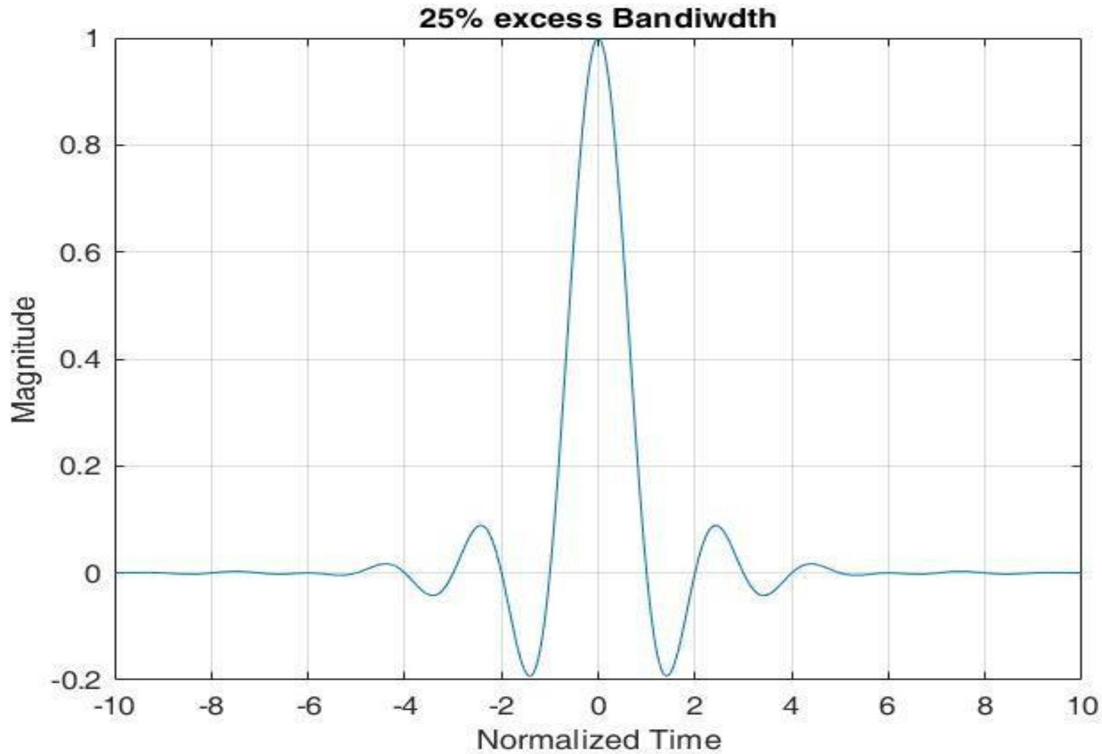
- **Observations**:

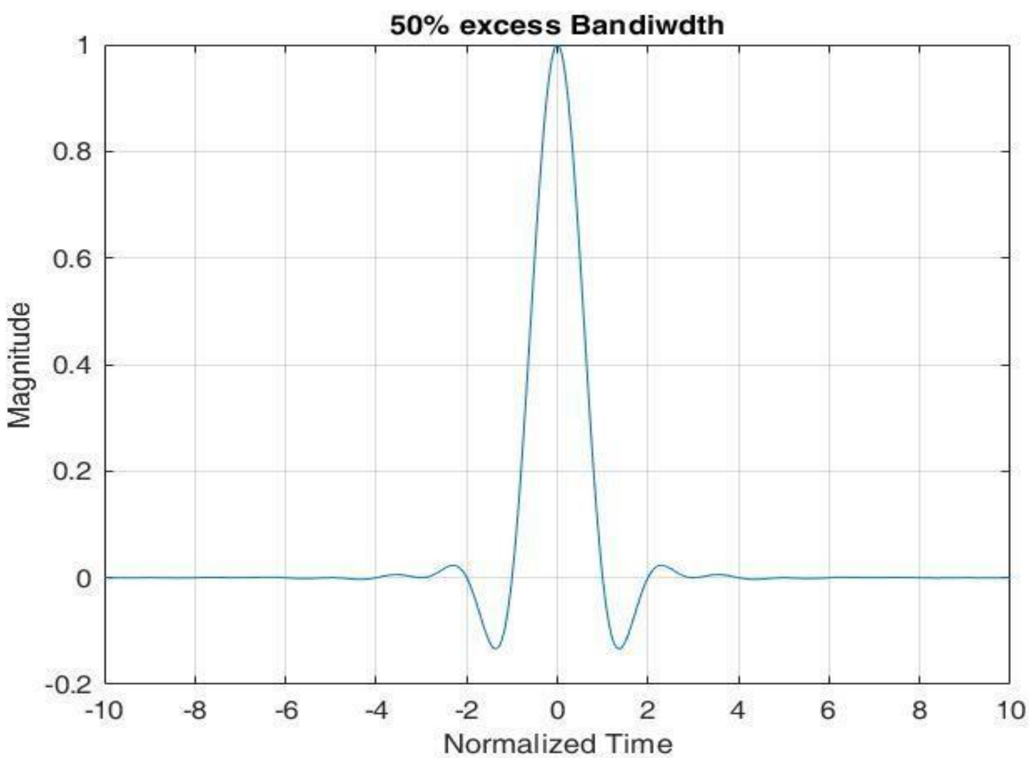
1. The ensemble-averaging and time-averaging approaches yield similar results for the autocorrelation function Rx(tau), signifying the fact that the random process X(t) described herein is indeed ergodic.
2. As the SNR is increased, the numerical accuracy of the estimation is improved, which is intuitively satisfying.

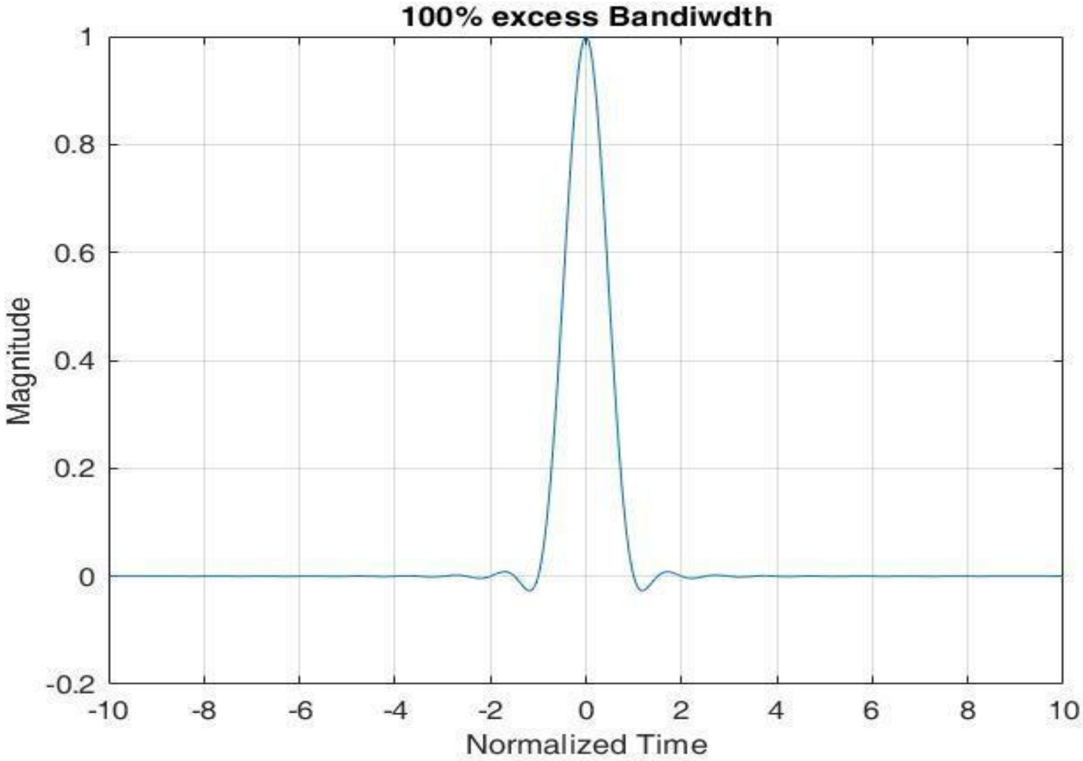
- **4.12(all parts) and 4.13(a and b)**

```
%% Experiment 4.12
%This can, for example, be used to generate a plot of
the raised cosine pulse,
% as follows, where we
%would typically oversample by a large factor ( % e.g.,
m = 32) in order to get a smooth plot.
%plot time domain raised cosine pulse
a = 0.25; % desired excess bandwidth
m = 32; %oversample by a lot to get smooth plot
```

```
length = 10; % where to truncate the time domain
response
%(one-sided, multiple of symbol time)
[rc,time] = raised_cosine(a,m,length);

figure(1);
plot(time,rc);
grid on;
title('25% excess Bandiwdth');
xlabel('Normalized Time');
ylabel('Magnitude');

%time domain pulse for raised cosine, together with
time vector to
% plot it against
%oversampling factor= how much faster than the symbol
rate we sample at
%length=where to truncate response (multiple of symbol
time)
% on each side of peak %a = excess bandwidth
function [rc,time_axis] = raised_cosine(a,m,length)
length_os = floor(length*m); %number of samples on each
side of peak %time vector (in units of symbol interval)
on one side of the peak
z = cumsum(ones(length_os,1))/m;
A= sin(pi*z)./(pi*z); %term 1
B= cos(pi*a*z); %term 2
C= 1 - (2*a*z).^2; %term 3
zerotest = m/(2*a); %location of zero in denominator
%check whether any sample coincides with zero location
if (zerotest == floor(zerotest))
    B(zerotest) = pi*a;
    C(zerotest) = 4*a;
    %alternative is to perturb around the sample
    %(find L'Hospital limit numerically)
    %B(zerotest) = cos(pi*a*(z(zerotest)+0.001));
    %C(zerotest) = 1-(2*a*(z(zerotest)+0.001))^2;
end
D = (A.*B)./C; %response to one side of peak
rc = [flipud(D);1;D]; %add in peak and other side
time_axis = [flipud(-z);0;z];
```


25% excess Bandiwdth
Magnitude
Normalized Time
1
0.8
0.6
0.4
0.2
0
-0.2
-10
-8
-6
-4
-2
0
2
4
6
8
10


50% excess Bandiwdth
Magnitude
Normalized Time
1
0.8
0.6
0.4
0.2
0
-0.2
-10
-8
-6
-4
-2
0
2
4
6
8
10



➢ Observations:

1. With the increase in excess bandwidth, the signal decays quickly.