Channabasaveshwara InstituteTechnology

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(NAAC Accredited & ISO 9001:2015 Certified Institution)

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

PRINCIPLES OF COMMUNICATION SYSTEMS

BEC402

2023-24

Prepared by:

VEENA KUMARI H M

Associate Professor

Dept. of ECE

MANOHARA T N

Assistant Professor

Dept. of ECE

GAGANA R

Assistant Professor

Dept. of ECE

Approved by

Dr. Thejaswini R

Dept. of ECE

VISION OF THE INSTITUTE

"To create centres of excellence in education and to serve the society by enhancing the quality of life through value based professional leadership"

MISSION OF THE INSTITUTE

- **To provide high quality technical and professionally relevant education in a diverse learning environment.**
- **To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society.**
- **To prepare the next generation of skilled professionals to successfully compete in the diverse global market.**
- **To promote a campus environment that welcomes and honors women and men of all races, creeds and cultures, values and intellectual curiosity, pursuit of knowledge and academic integrity and freedom.**
- **To offer a wide variety of off-campus education and training programmes to individuals and groups.**
- **To stimulate collaborative efforts with industry, universities, government and professional societies.**
- **To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.**

Channabasaveshwara Institute of Technology

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(NAAC Accredited & ISO 9001:2015 Certified Institution)

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

PRINCIPLES OF COMMUNICATION SYSTEMS

COURSE OBJECTIVES:

This course will enable students to

- Understand and analyze concepts of Analog Modulation schemes viz; AM, FM.
- Design and analyze the electronic circuits for AM and FM modulation and demodulation.
- Understand the concepts of random variable and random process to model communication systems..
- Understand and analyze the concepts of digitization of signals.
- Evolve the concept of SNR in the presence of channel induced noise

COURSE OUTCOMES:

On the completion of this laboratory course, the students will be able to:

- Understand the principles of analog communication systems and noise modelling.
- Identify the schemes for analog modulation and demodulation and compare their performance.
- Design of PCM systems through the processes sampling, quantization and encoding..
- Describe the ideal condition, practical considerations of the signal representation for baseband transmission of digital signals.
- identify and associate the random variables and random process in Communication system design.

digital signals.

'Instructions to the candidates'

- Student should come with through preparation for the experiment to be conducted.

- Student should take prior permission from the concerned faculty before availing the leave.

- Student should come with proper dress code and to be present on time in the laboratory.

- Student will not be permitted to attend the laboratory unless they bring the practical record fully completed in all respects pertaining to the experiment conducted in the previous class.

- Student will not be permitted to attend the laboratory unless they bring the observation book fully completed in all respects pertaining to the experiment to be conducted in present class.

- Experiment should be started conducting only after the staff-in-charge has checked the circuit diagram.

- All the calculations should be made in the observation book. Specimen calculations for one set of readings have to be shown in the practical record.

- Wherever graphs to be drawn, A-4 size graphs only should be used and the same should be firmly attached in the practical record.

- Practical record and observation book should be neatly maintained.

- Student should obtain the signature of the staff-in-charge in the observation book after completing each experiment.

- Theory related to each experiment should be written in the practical record before procedure in your own words with appropriate references.

Channabasaveshwara Institute of Technology

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(NAAC Accredited & ISO 9001:2015 Certified Institution)

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

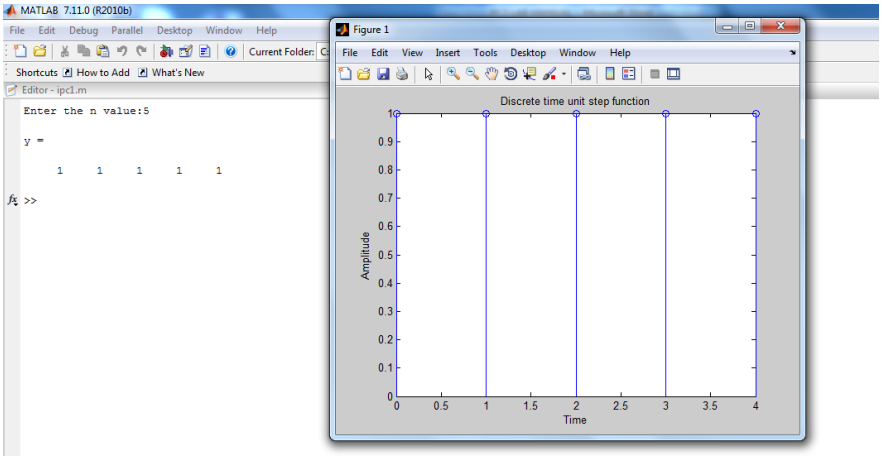
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Experiment 1. Aim: Basic Signals and Signal Graphing: a) Unit Step, b) Rectangular, c) Standard triangle d) sinusoidal and e) Exponential signal.

1a) GENERATION OF UNIT STEP SIGNAL

```
clc;
clear all;
close all;
n=input('Enter the n value:');
 t=0:1:n-1;
y=ones(1,n)
stem(t,y);
 xlabel('Time');
ylabel('Amplitude');
 title('Discrete time unit step function');
```

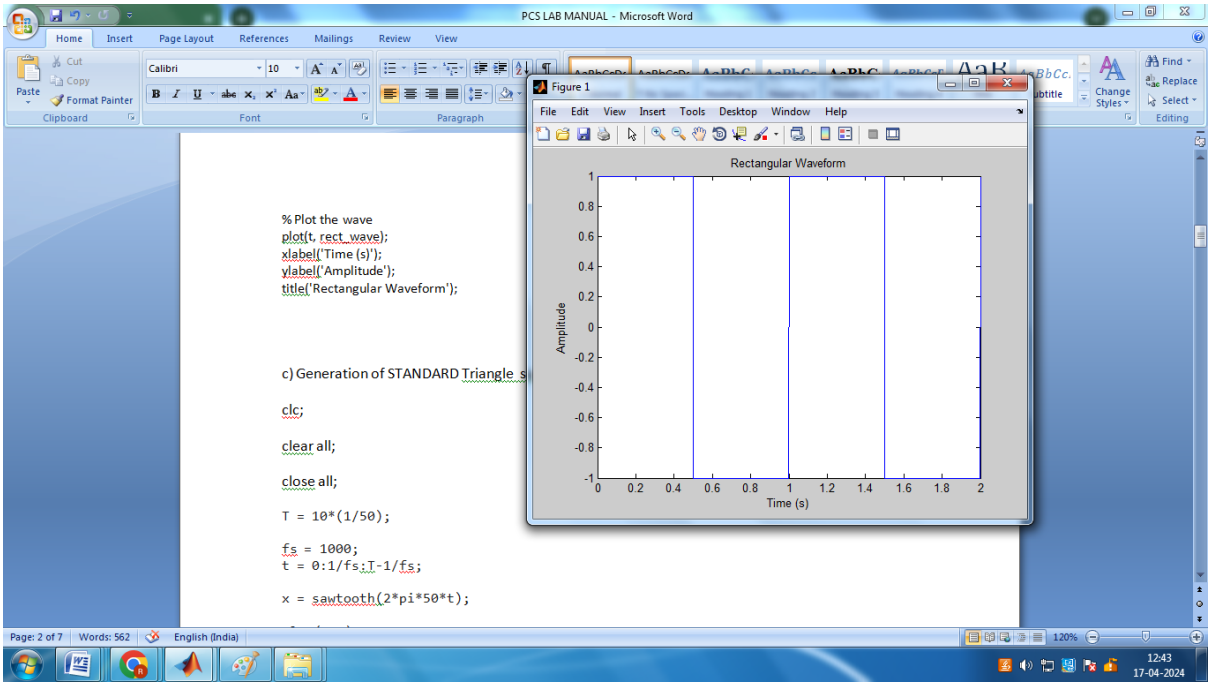
Output:



1b) Generation of Rectangular pulse.

```
% Define parameters
frequency = 1; % Frequency of the rectangular wave in Hz
amplitude = 1; % Amplitude of the rectangular wave
duration = 1; % Duration of the wave in seconds
sampling_rate = 1000; % Sampling rate in Hz

% Generate time vector
t = 0:1/sampling_rate:duration;

% Generate rectangular wave
rect_wave = amplitude * square(2*pi*frequency*t);

% Plot the wave
plot(t, rect_wave);
xlabel('Time (s)');
ylabel('Amplitude');
title('Rectangular Waveform');
```

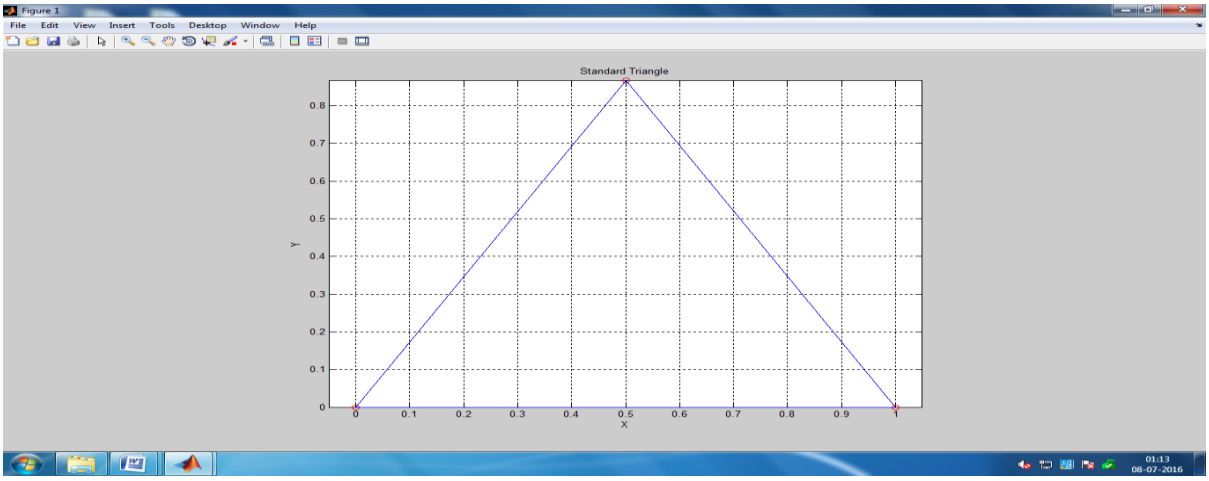
Output:



1c) Generation of Standard Triangle signal.

```
A = [0, 0];
B = [1, 0];
C = [0.5, sqrt(3)/2];
% Plot the vertices
plot(A(1), A(2), 'ro'); % Vertex A
hold on;
plot(B(1), B(2), 'ro'); % Vertex B
plot(C(1), C(2), 'ro'); % Vertex C
% Connect the vertices to form the triangle
plot([A(1), B(1)], [A(2), B(2)], 'b'); % Line AB
plot([B(1), C(1)], [B(2), C(2)], 'b'); % Line BC
plot([C(1), A(1)], [C(2), A(2)], 'b'); % Line CA
% Set plot properties
axis equal; % Make the axes scale equally
grid on; % Show grid
xlabel('X');
ylabel('Y');
title('Standard Triangle');
```

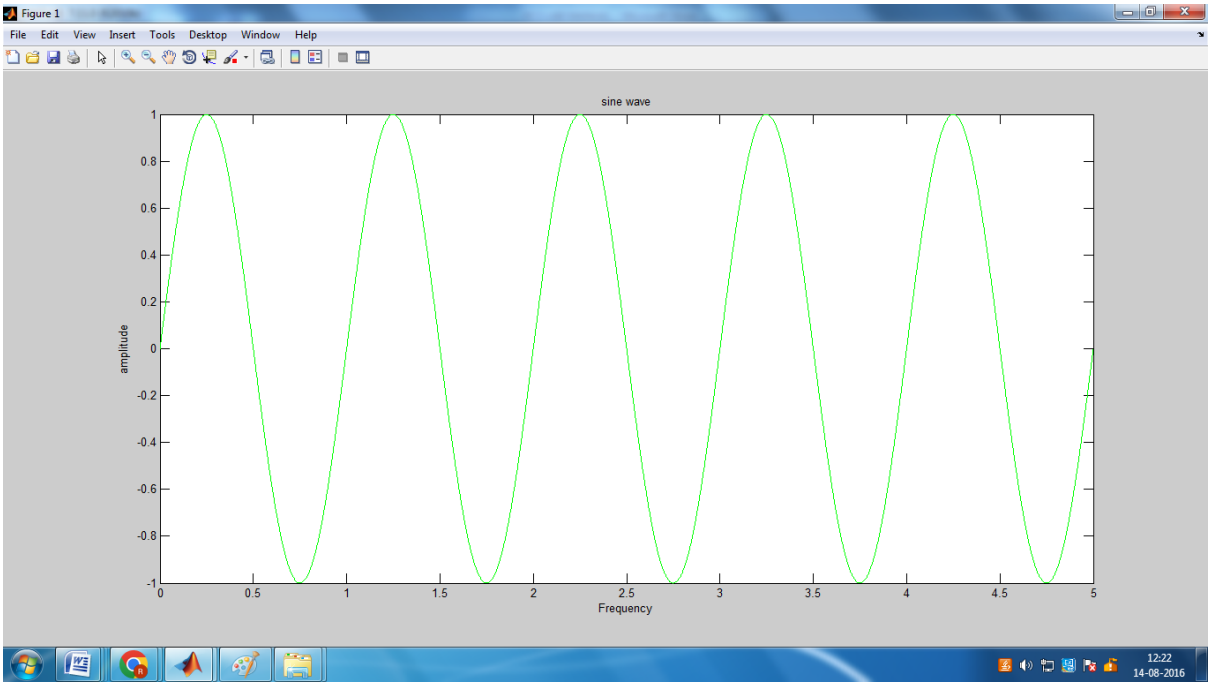
Output:



1d) Generation of sine wave y (n) =sin (2πfn)

```
clc;
clear all;
close all;
t=0:0.001:5;
f=1;
y=sin(2*pi*f*t);
plot(t,y,'g');
xlabel('Frequency');
ylabel('amplitude');
title('sine wave');
```

Output:

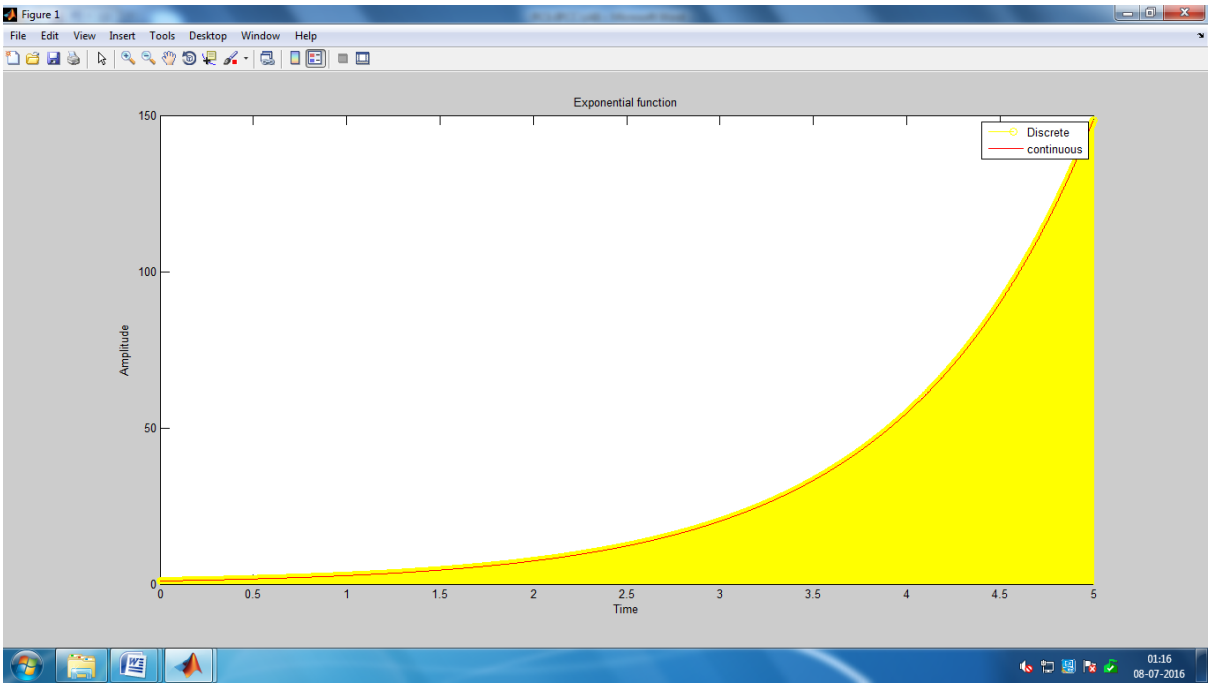


1e) Generation of exponential signal.

```
clc;
clear all;
 close all;
 n=input('enter the N value:');
t=0:0.001:n;
 a=input('enter the A value:');
y=exp(a*t);
 stem(t,y,'y');
hold on;
plot(t,y,'r');
xlabel('Time');
```

```
ylabel('Amplitude');

 title('Exponential function');

 legend('Discrete','continuous');
```

Output:



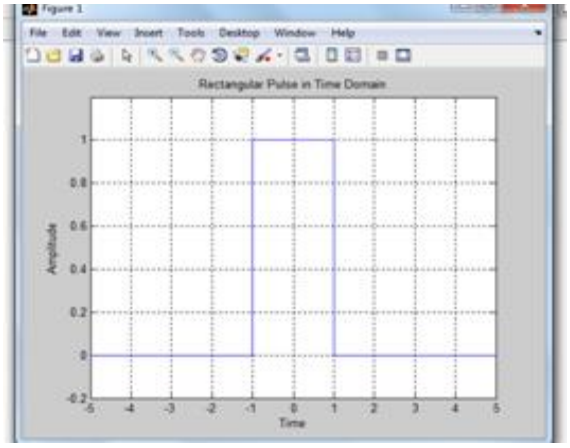
Experiment 2: Aim: Illustration of signal representation in time and frequency domains for a rectangular pulse.

2a) Signal representation of rectangular pulse in time domain

```
% Define the parameters
t = -5:0.01:5;  % time vector
width = 2;      % width of the rectangular pulse
% center of the pulse

% Generate the rectangular pulse using rectpuls
x = rectpuls(t, width);

% Plot the rectangular pulse
plot(t, x);
xlabel('Time');
ylabel('Amplitude');
title('Rectangular Pulse in Time Domain');
axis([-5 5 -0.2 1.2]);  % Set axis limits
grid on;

% Define the parameters
t = -5:0.01:5;  % time vector
```

```
width = 2;      % width of the rectangular pulse
center = 0;     % center of the pulse

% Generate the rectangular pulse
x = (abs(t-center) <= width/2);
% Plot the rectangular pulse
plot(t, x);
xlabel('Time');
ylabel('Amplitude');
title('Rectangular Pulse in Time Domain');
axis([-5 5 -0.2 1.2]);  % Set axis limits
grid on;
```

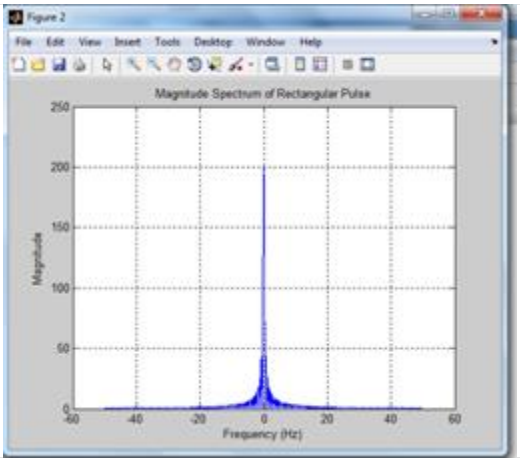
Output:



2. b) signal representation of rectangular pulse in frequency domain

```
% Define the parameters
t = -5:0.01:5;  % time vector for the rectangular pulse
T = 2;          % width of the rectangular pulse
center = 0;     % center of the pulse

% Generate the rectangular pulse
x = (abs(t-center) <= T/2);

% Compute the Fourier Transform
X = fft(x);

% Frequency vector
fs = 1/(t(2)-t(1));  % Sampling frequency
f = (-fs/2:fs/length(t):fs/2-fs/length(t));  % Frequency vector

% Plot the magnitude of the Fourier Transform
figure;
plot(f, fftshift(abs(X)));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Magnitude Spectrum of Rectangular Pulse');
grid on;
```

Output:

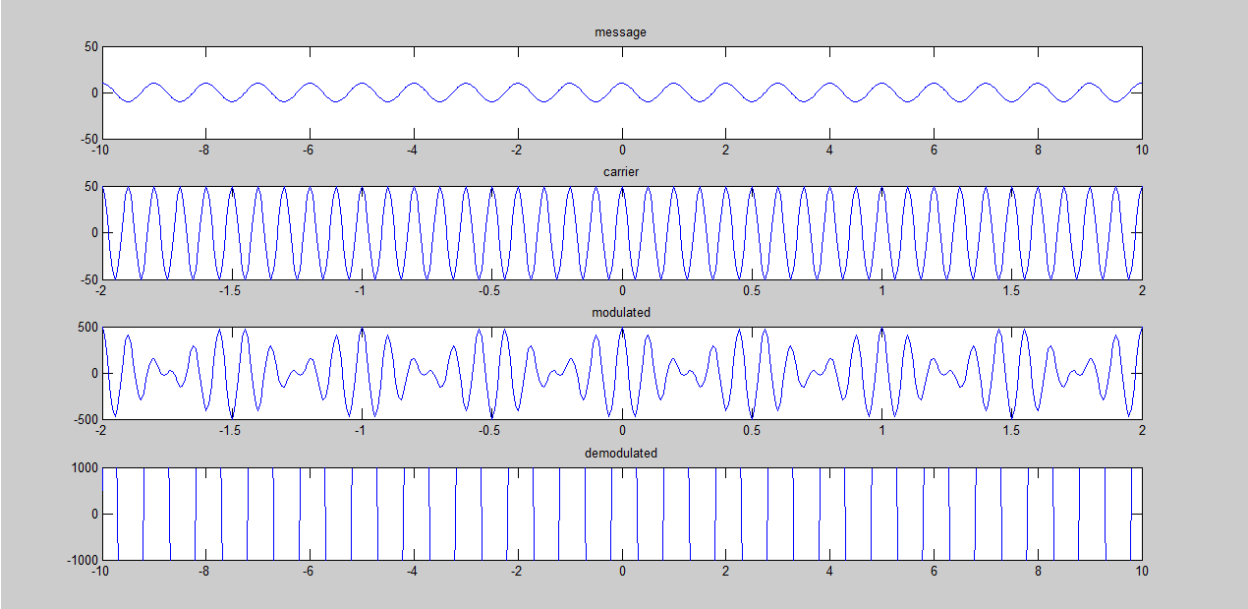


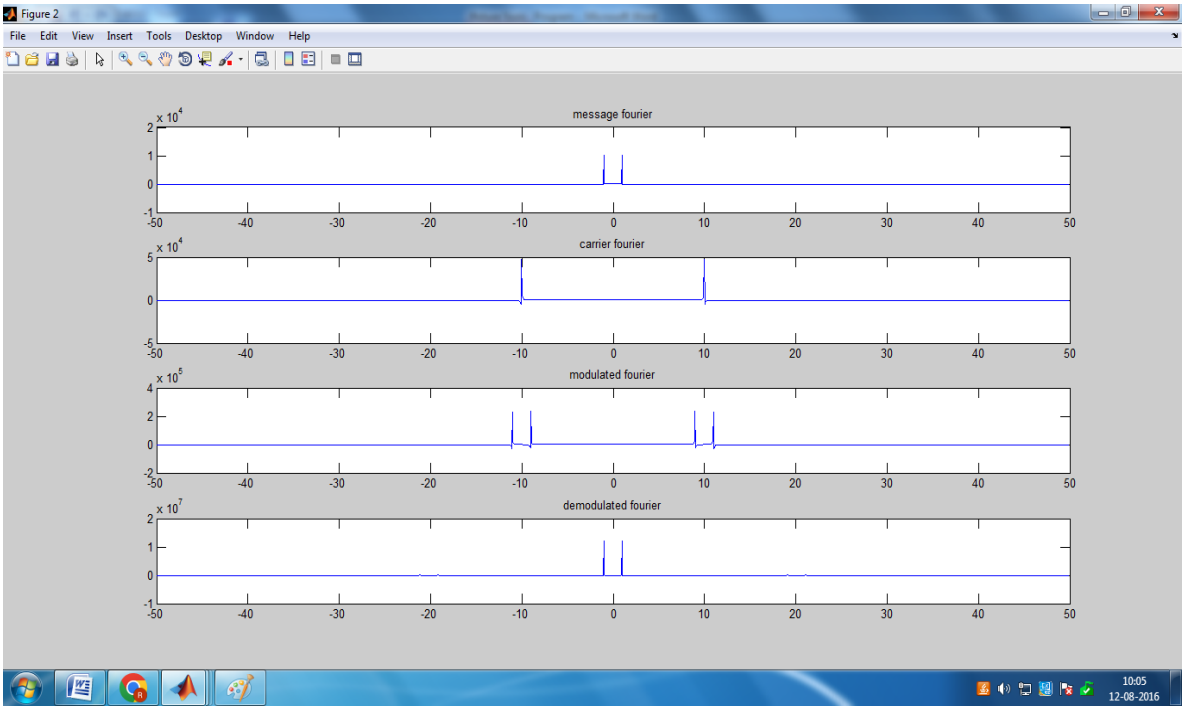
Experiment 3. Aim: Amplitude Modulation and demodulation: Generation and display the relevant signals and its spectrums.

```
clc;
close all;
vm=10;
vc=50;
fm=1;
fc=10;
t=-10:0.01:10;
%time domain%
m=vm*cos(2*pi*fm*t);
c=vc*cos(2*pi*fc*t);
subplot(4,1,1);
plot(t,m);
axis([-10 10 -50 50]);
title('message');
subplot(4,1,2);
plot(t,c);
axis([-2 2 -50 50]);
title('carrier');
fs=100;
s=m.*c;
```

```
subplot(4,1,3);
plot(t,s);
axis([-2 2 -500 500]);
title('modulated');
[B A]=butter(2,0.1,'low');
d=s.*c;
fi=filter(B,A,d);
subplot(4,1,4);
plot(t,fi);
axis([-10 10 -1000 1000]);
title('demodulated');
%frequency domain%
f= -(fs/2):fs/length(t):(fs/2)-(fs/length(t))/2;
figure;
subplot(4,1,1);
plot(f,fftshift(fft(m)));
title('message fourier');
subplot(4,1,2);
plot(f,fftshift(fft(c)));
title('carrier fourier');
subplot(4,1,3);
plot(f,fftshift(fft(s)));
title('modulated fourier');
subplot(4,1,4);
m=m.*5;
plot(f,fftshift(fft(fi)));
title('demodulated fourier');
```

output:



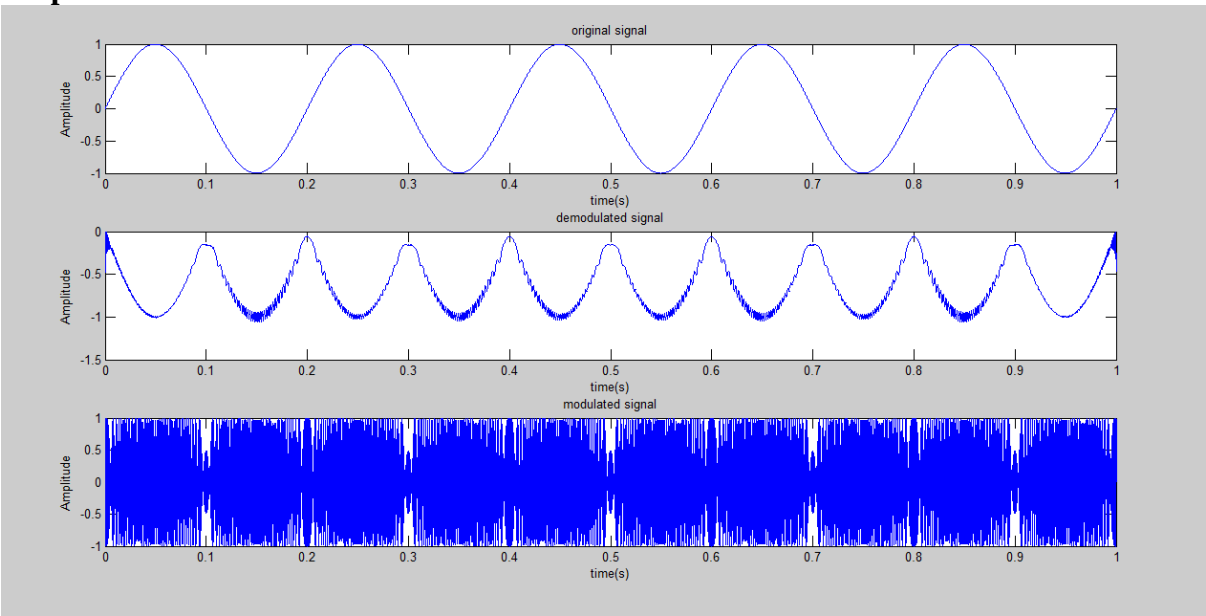


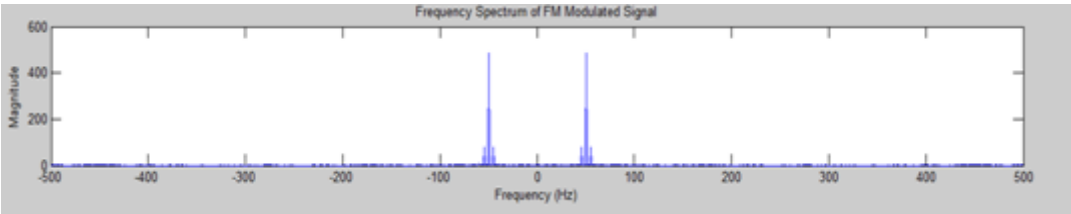
Experiment 4. Aim: Frequency Modulation and demodulation: Generation and display the relevant signals and its spectrums.

```
Clc;close all;

fc=3000;

fs= 2*fc;

t=0:1/fs:1;
fm=5;
kf=500
m=sin(2*pi*fm*t);
y=fmmod(m,fc,fs,kf);
demodulated=fmdemod(y,fc,fs,kf);
subplot (3,1,1)
plot (t,m)
title('original signal')
xlabel('time(s)');
title('original signal');
xlabel('time(s)');
ylabel('Amplitude');
subplot (3,1,2)
```

```
plot (t,demodulated)
title('demodulated signal')
xlabel('time(s)');
ylabel('Amplitude');
subplot (3,1,3)
plot (t,y)
title('modulated signal')
xlabel('time(s)');
ylabel('Amplitude');
```

Output:



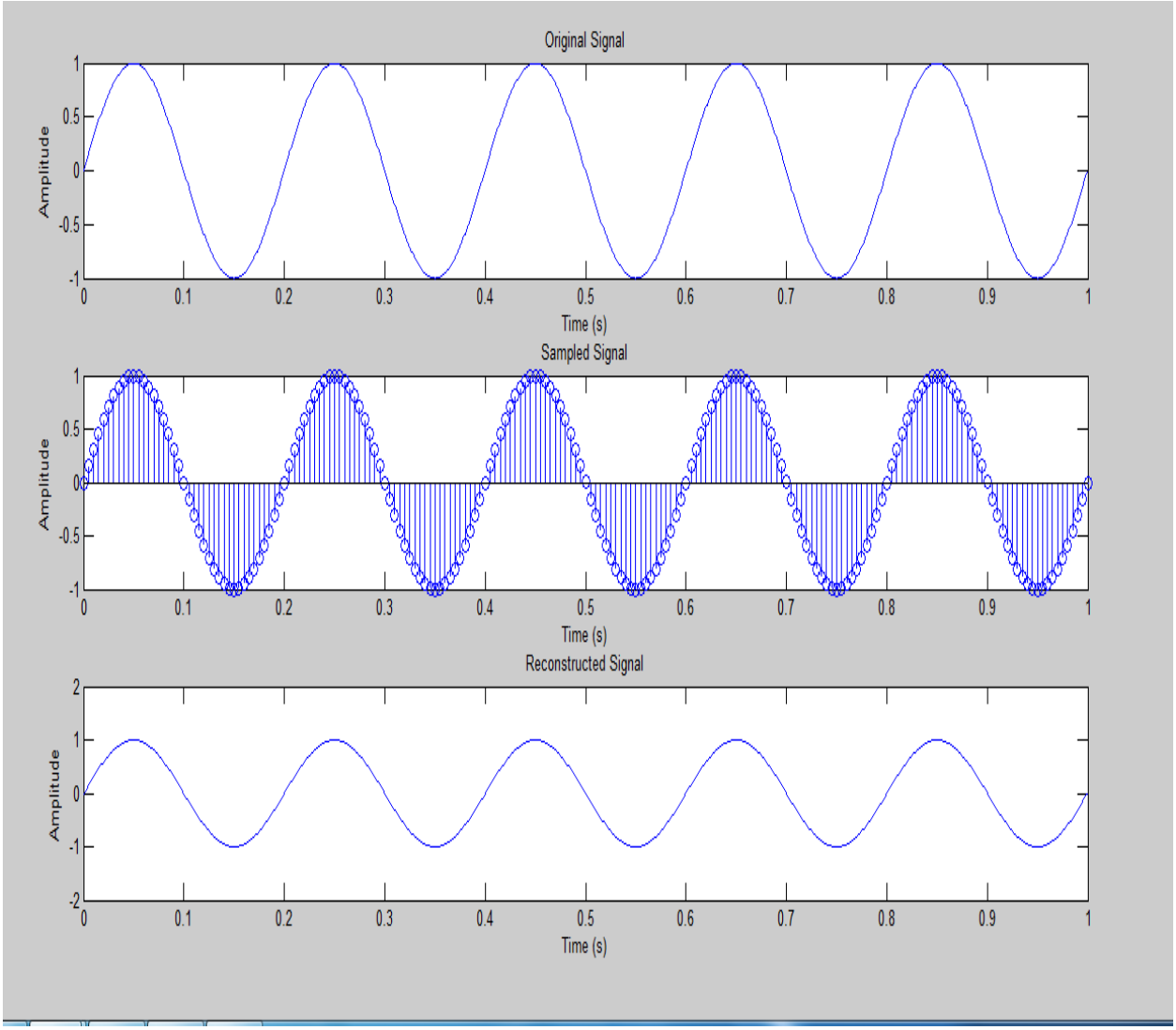


Experiment 5. Aim: Sampling and reconstruction of low pass signals. Display the signals and its spectrum.

```
clc;
clear all;
close all;% Define parameters
% Define parameters
Fs = 1000;           % Sampling frequency (Hz)
T = 1/Fs;            % Sampling period (s)
t = 0:T:1;           % Time vector from 0 to 1 second
f1 = 5;              % Frequency of the original signal (Hz)

% Generate a low-pass signal (sine wave)
x = sin(2*pi*f1*t);

% Plot the original signal
subplot(3,1,1);
plot(t,x);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Sampling the signal
Fs_new = 200;        % New sampling frequency (Hz), lower than the Nyquist rate
Ts_new = 1/Fs_new;   % New sampling period (s)
n = 0:Ts_new:1;      % New time vector
xn = sin(2*pi*f1*n);% Sampled signal
```

```
% Plot the sampled signal
subplot(3,1,2);
stem(n,xn);
title('Sampled Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Reconstruction of the signal using sinc interpolation
x_reconstructed = zeros(size(t)); % Initialize the reconstructed signal
for i = 1:length(n)
    x_reconstructed = x_reconstructed + sinc((t-n(i))/Ts_new) * xn(i);
end

% Plot the reconstructed signal
subplot(3,1,3);
plot(t,x_reconstructed);
title('Reconstructed Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

Output:

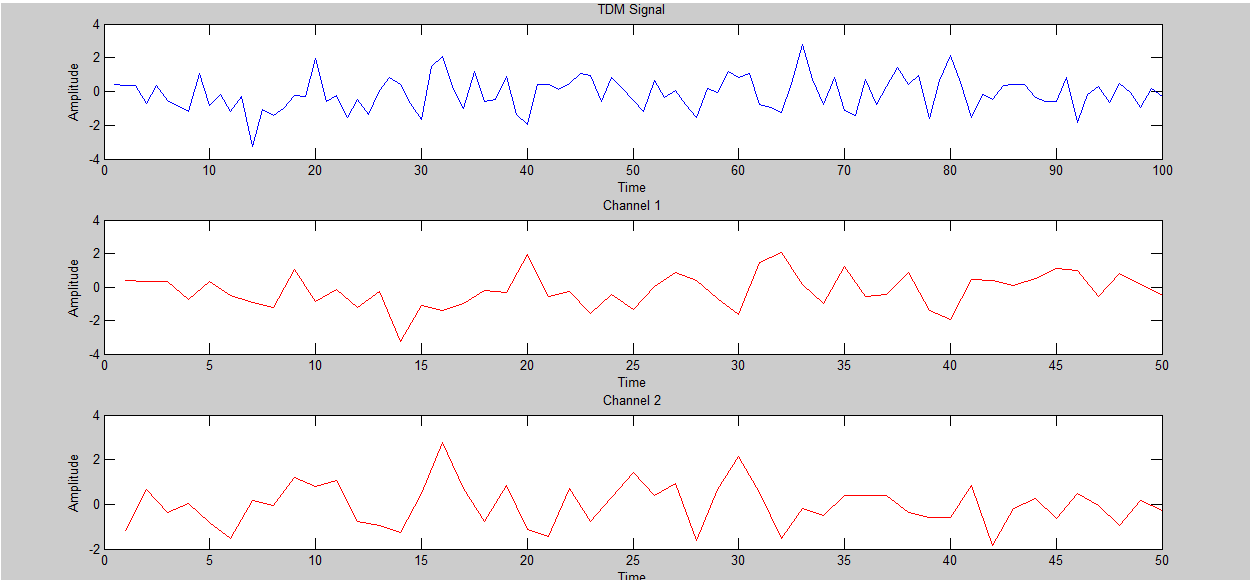


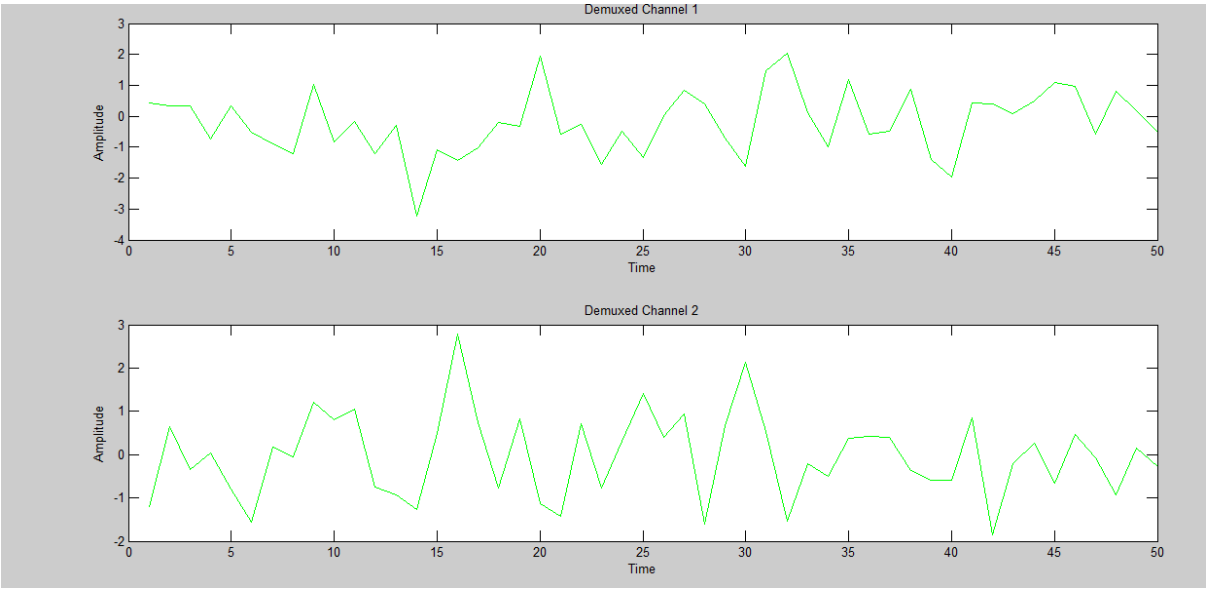
Experiment 6. Aim: Time Division Multiplexing and demultiplexing.

```
% Parameters
numChannels = 2; % Number of channels
numSamples = 50; % Number of samples per channel
timeSlot = 5; % Time slot for each channel (arbitrary)

% Generate sample data for each channel
channels = cell(1, numChannels);
for i = 1:numChannels
    channels{i} = randn(1, numSamples);
end

% Time Division Multiplexing
TDM_signal = zeros(1, numChannels * numSamples);
for i = 1:numChannels
    TDM_signal((i-1)*numSamples + 1:i*numSamples) = channels{i};
end

% Display original channels and TDM signal
figure;
subplot(numChannels + 1, 1, 1);
plot(TDM_signal, 'b');
title('TDM Signal');
xlabel('Time');
ylabel('Amplitude');
for i = 1:numChannels
    subplot(numChannels + 1, 1, i + 1);
    plot(channels{i}, 'r');
    title(['Channel ', num2str(i)]);
    xlabel('Time');
    ylabel('Amplitude');
end

% Demultiplexing
demuxed_channels = cell(1, numChannels);
for i = 1:numChannels
```

```
    demuxed_channels{i} = TDM_signal((i-1)*numSamples + 1:i*numSamples);
end

% Display demultiplexed channels
figure;
for i = 1:numChannels
    subplot(numChannels, 1, i);
    plot(demuxed_channels{i}, 'g');
    title(['Demuxed Channel ', num2str(i)]);
    xlabel('Time');
    ylabel('Amplitude');
end
```

output:



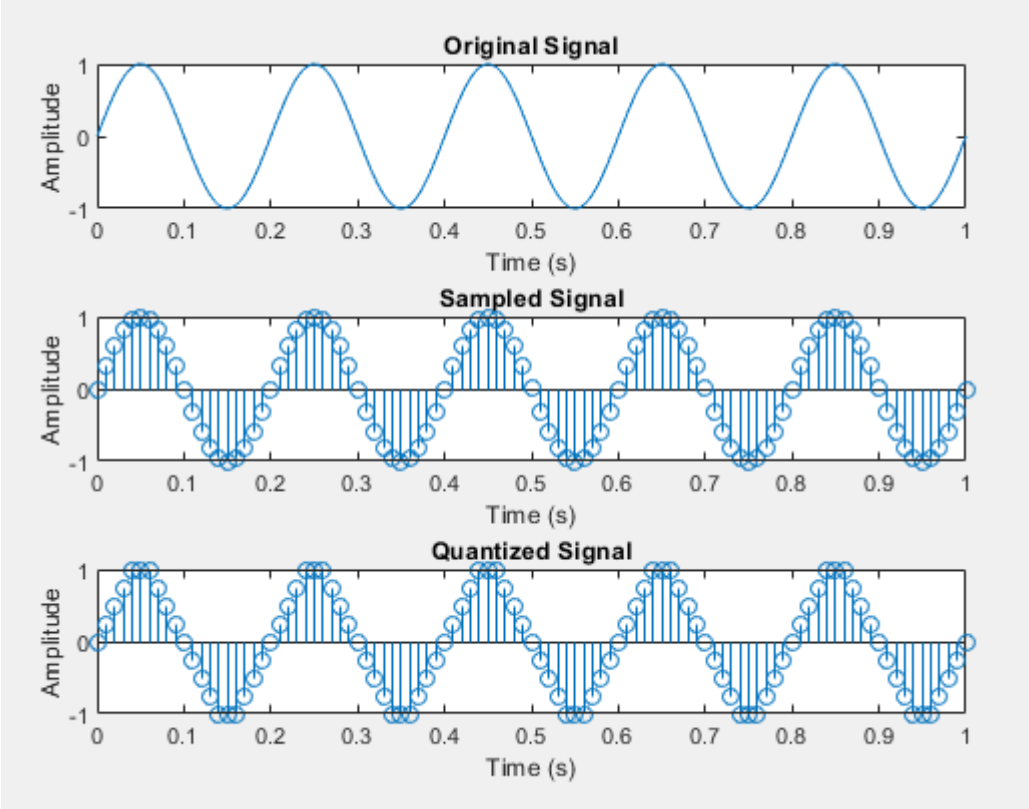


Experiment 7. PCM Illustration: Sampling, Quantization and Encoding

```
clc; close all;
% Parameters
Fs = 1000;       % Sampling frequency (Hz)
t = 0:1/Fs:1;    % Time vector (1 second duration)
f = 5;           % Frequency of the input sinusoid (Hz)
A = 1;           % Amplitude of the sinusoid

% Generate a sinusoidal signal
x = A*sin(2*pi*f*t);

% Plot the original signal
subplot(3,1,1);
plot(t, x);
xlabel('Time (s)');
ylabel('Amplitude');
title('Original Signal');

% Sampling
Fs_new = 100;            % New sampling frequency (Hz)
t_new = 0:1/Fs_new:1;    % New time vector
x_sampled = A*sin(2*pi*f*t_new);

% Plot the sampled signal
```

```
subplot(3,1,2);
stem(t_new, x_sampled);
xlabel('Time (s)');
ylabel('Amplitude');
title('Sampled Signal');

% Quantization (uniform quantization)
bits = 3;                % Number of bits for quantization
max_value = max(abs(x_sampled)); % Maximum absolute value of the signal
step_size = (2*max_value) / (2^bits); % Step size for quantization
quantized_signal = round(x_sampled / step_size) * step_size;

% Plot the quantized signal
subplot(3,1,3);
stem(t_new, quantized_signal);
xlabel('Time (s)');
ylabel('Amplitude');
title('Quantized Signal');

% PCM encoding (convert quantized samples to binary)
% First, convert quantized samples to integers
quantized_integers = round(quantized_signal / step_size);
% Then, convert integers to binary
binary_signal = de2bi(quantized_integers);

% Display the binary signal
disp('PCM Encoded Signal (Binary):');
disp(binary_signal);
```

Output:

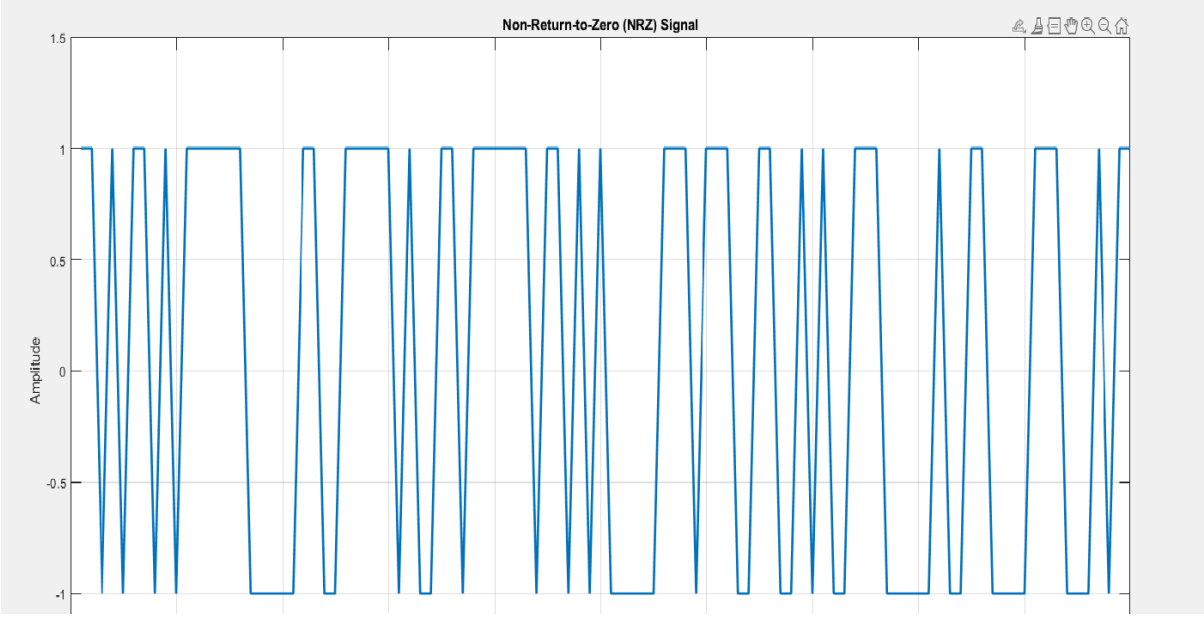


Experiment 8. **Generate a) NRZ, RZ and Raised cosine pulse, b) Generate and plot eye diagram**

% Generate NRZ signal

```
bitStream = randi([0, 1], 1, 100);  % Generate a random bit stream of length
100

% NRZ Encoding
nrzSignal = 2 * bitStream - 1;  % Convert 0s to -1s and 1s to 1s

% Plot the NRZ signal
figure;
plot(nrzSignal, 'LineWidth', 2);
grid on;
axis([0 length(nrzSignal) -1.5 1.5]);
xlabel('Time');
ylabel('Amplitude');
title('Non-Return-to-Zero (NRZ) Signal');
```

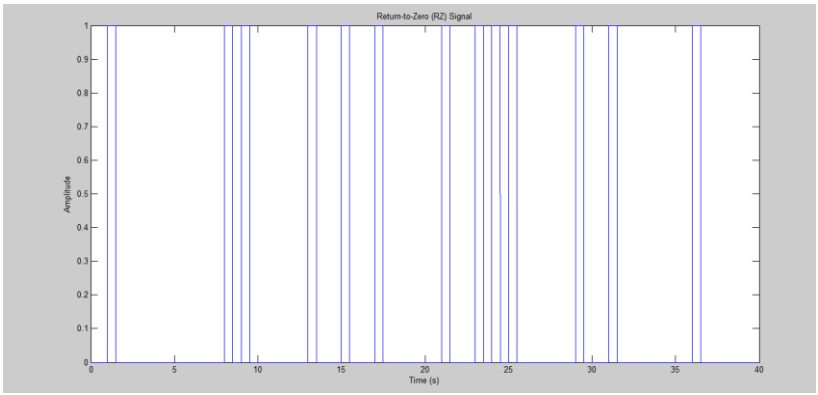
output:



% Generate RZ signal

```
rz_signal = zeros(1, length(bits) * sampling_rate * bit_duration);
for i = 1:length(bits)
    if bits(i) == 1
        rz_signal((i-1)*sampling_rate*bit_duration + 1 : (i-
1)*sampling_rate*bit_duration + sampling_rate*bit_duration/2) = 1;
    end
end

% Plot RZ signal
plot(t, rz_signal);
xlabel('Time (s)');
ylabel('Amplitude');
title('Return-to-Zero (RZ) Signal');
```

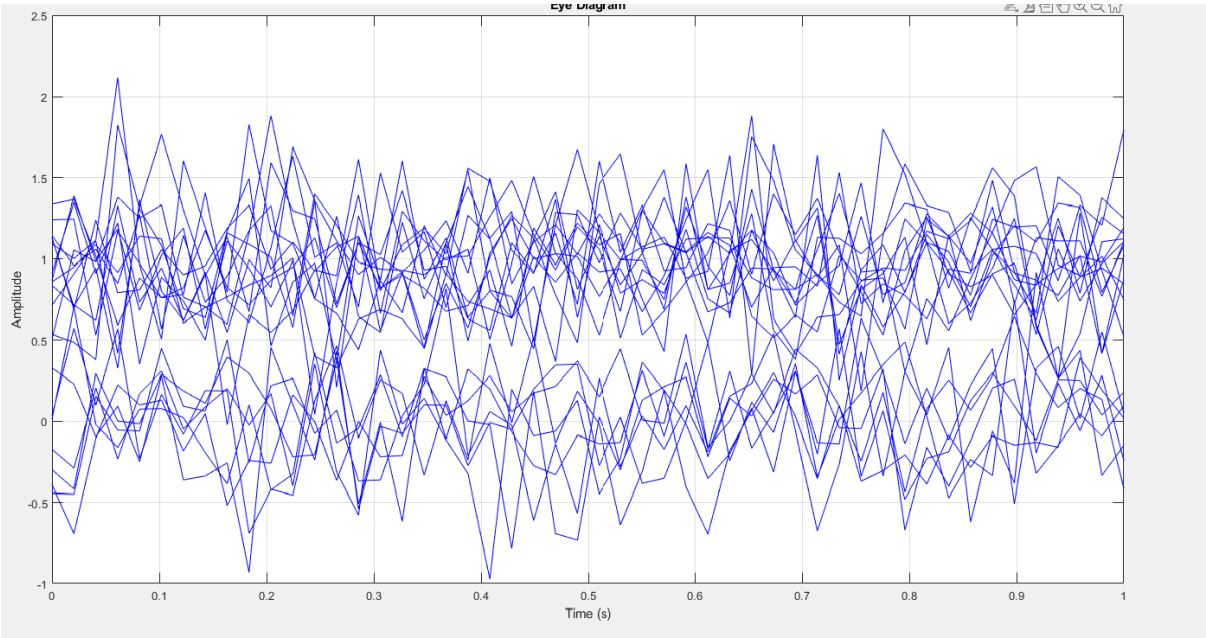
Output:



eye diagram

```
% Parameters

bit_duration = 1; % Duration of each bit in seconds
bit_rate =5; % Bit rate in bits per second
sampling_rate = 10 * bit_rate; % Sampling rate in samples per second
num_bits = 20; % Number of bits
noise_power = 0.1; % Noise power

% Generate random bits
bits = randi([0,1], 1, num_bits);

% Generate NRZ signal
nrz_signal = repmat(bits, [sampling_rate * bit_duration, 1]);
nrz_signal = nrz_signal(:)';

% Add noise to the signal
noisy_signal = nrz_signal + sqrt(noise_power) * randn(size(nrz_signal));

% Reshape the noisy signal to create an eye diagram
eye_matrix = reshape(noisy_signal, [sampling_rate * bit_duration, num_bits]);

% Plot the eye diagram
t = linspace(0, bit_duration, sampling_rate * bit_duration);
figure;
plot(t, eye_matrix, 'b');
xlabel('Time (s)');
ylabel('Amplitude');
title('Eye Diagram');
grid on;
```

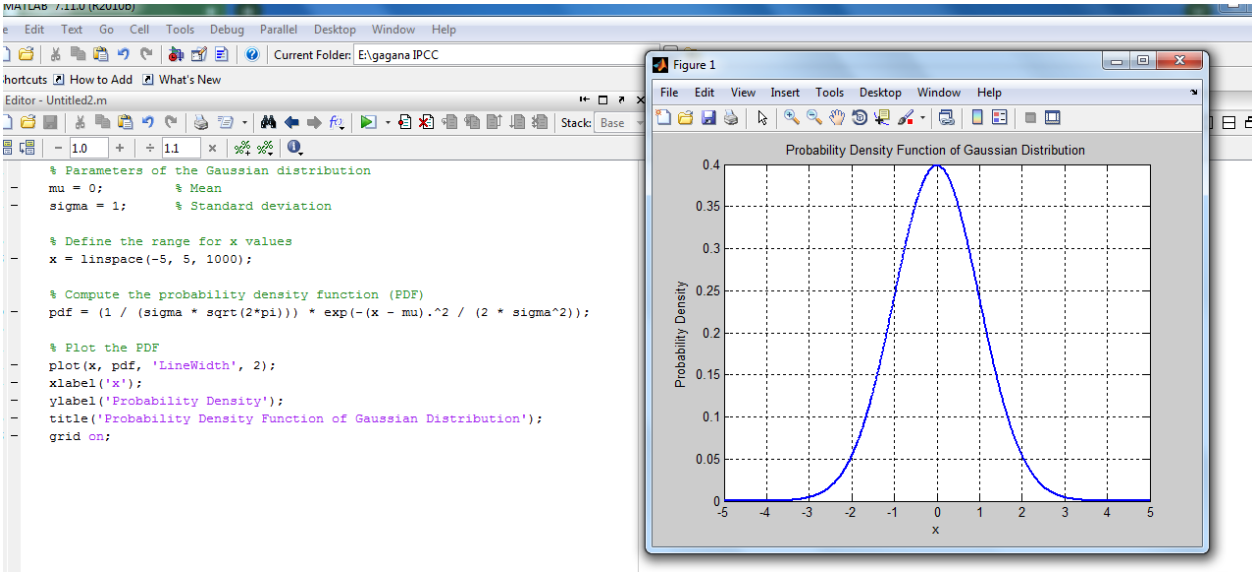
Output:



Experiment 9. Generate the Probability density function of Gaussian distribution function.

```
% Parameters of the Gaussian distribution
```

```
mu = 0;          % Mean
sigma = 1;       % Standard deviation

% Define the range for x values
x = linspace(-5, 5, 1000);

% Compute the probability density function (PDF)
pdf = (1 / (sigma * sqrt(2*pi))) * exp(-(x - mu).^2 / (2 * sigma^2));

% Plot the PDF
plot(x, pdf, 'LineWidth', 2);
xlabel('x');
ylabel('Probability Density');
title('Probability Density Function of Gaussian Distribution');
grid on;
```



Experiment 10. Display the signal and its spectrum of an audio signal.

```
filename = 'your_audio_file.wav';  % Change 'your_audio_file.wav' to the name
of your audio file
```

```
[y, Fs] = audioread(filename);

% Plot the waveform of the audio signal
t = (0:length(y)-1) / Fs;  % Time vector
figure;
subplot(2,1,1);
plot(t, y);
xlabel('Time (s)');
ylabel('Amplitude');
title('Audio Signal Waveform');

% Compute the Fourier Transform of the audio signal
Y = fft(y);

% Compute the frequency vector
f = Fs*(0:(length(y)/2))/length(y);

% Plot the magnitude spectrum
subplot(2,1,2);
plot(f, 2*abs(Y(1:length(y)/2+1)));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Magnitude Spectrum of Audio Signal');
grid on;
```

Output:

References:

- "Principles of Digital Audio" by Ken C. Pohlmann: Covers the fundamentals of digital audio, including PCM.
- "Digital Signal Processing: Principles, Algorithms, and Applications" by John G. Proakis and Dimitris G. Manolakis: Provides a comprehensive introduction to digital signal processing, which is essential for understanding PCM.