

**Metropolitan University**

**Department of Computer Science and Engineering**

**Lab Report**

**Course Title: Digital Signal Processing Lab**

**Course Code: CSE-442**

**Submitted to:**

**Sadman Sakib**

**Lecturer**

Department of Electrical and Electronics Engineering

**Submitted by:**

**Name : Moumita Datta Sristy**

**ID: 221-115-062**

**Batch: 56th (B)**

Department of Computer Science and Engineering

Metropolitan University ,Sylhet

**Date of Submission:** 21 March, 2025

**Lab 01**

**1. Unit Impulse Signal**

**Code:**

clc;

clear all;

close all;

t = -3:1:3;

y = [zeros(1,3),ones(1,1),zeros(1,3)];

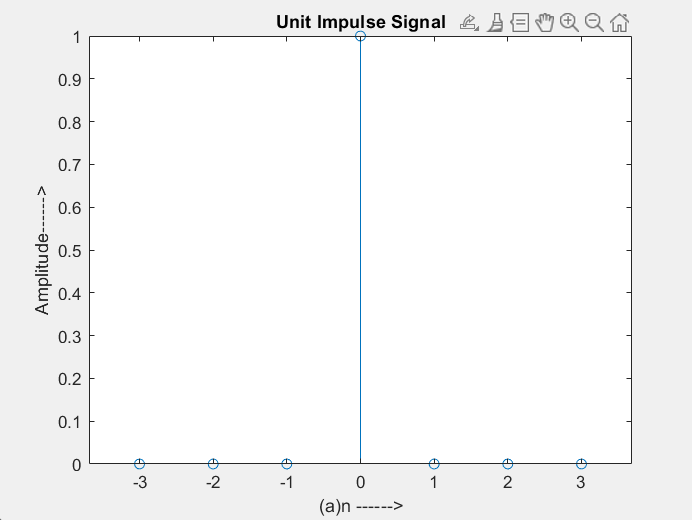
stem(t,y);

ylabel('Amplitude------>');

xlabel('(a)n ------>');

title ('Unit Impulse Signal');

**OUTPUT:**



**2. Unit Step Sequence [u(n)-u(n-N)]**

**Code:**

n=input ('enter the N value ');

t=0:1:n-1;

y=ones(1,n) ;

stem(t,y);

ylabel ('amplitude ---->');

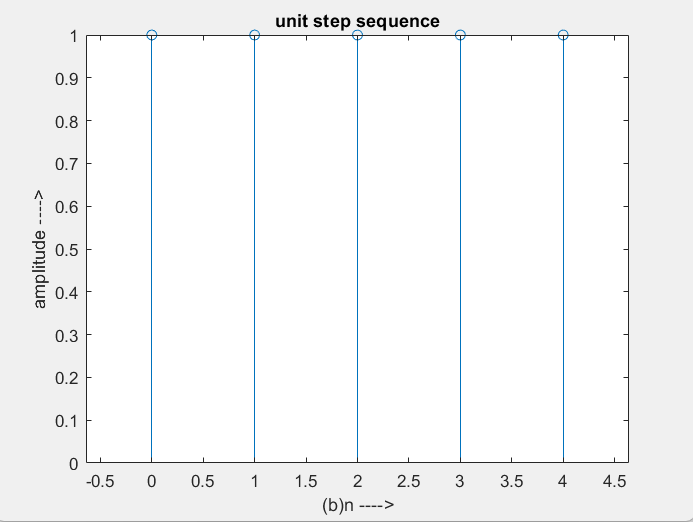
xlabel('(b)n ---->');

title (' unit step sequence ');

**OUTPUT:**

Enter the N value 5

**EXPECTED GRAPH:**



**3. Ramp sequence**

**Code:**

n=input('enter the length of ramp sequence ');

t=0:n-1 ;

stem(t,t) ;

ylabel('amplitude ---->') ;

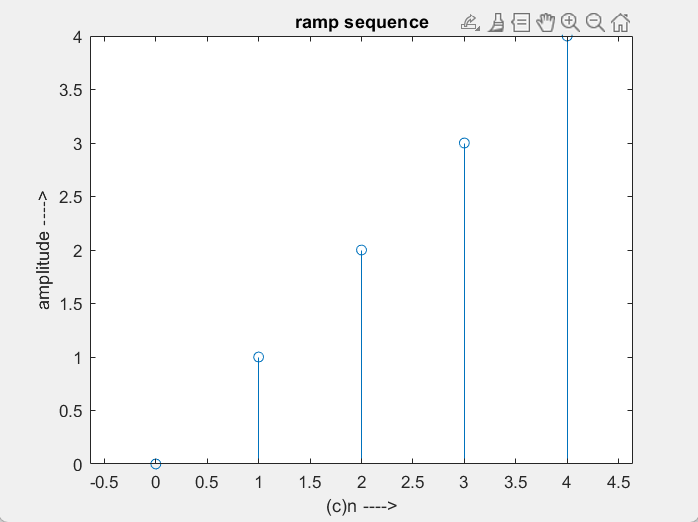
xlabel('(c)n ---->') ;

title(' ramp sequence ') ;

**OUTPUT:**

Enter the length of ramp sequence 5

**EXPECTED GRAPH:**

****

**4. Exponential sequence**

**Code:**

n=input('enter the length of exponential sequence') ;

t=0:n;

a=input('enter the value of a');

y=exp(a\*t);

stem(t,y);

ylabel('amplitude ---->') ;

xlabel('(d)n ---->') ;

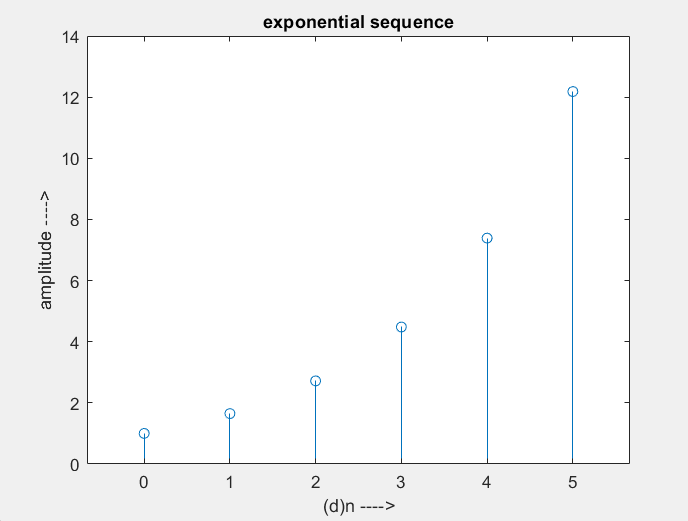
title(' exponential sequence ');

**OUTPUT:**

Enter the length of exponential sequence 5

Enter the value of a 0.5

**EXPECTED GRAPH:**



**5. Sine sequence**

**Code:**

t=0:0.05:pi ;

y=sin(2\*pi\*t) ;

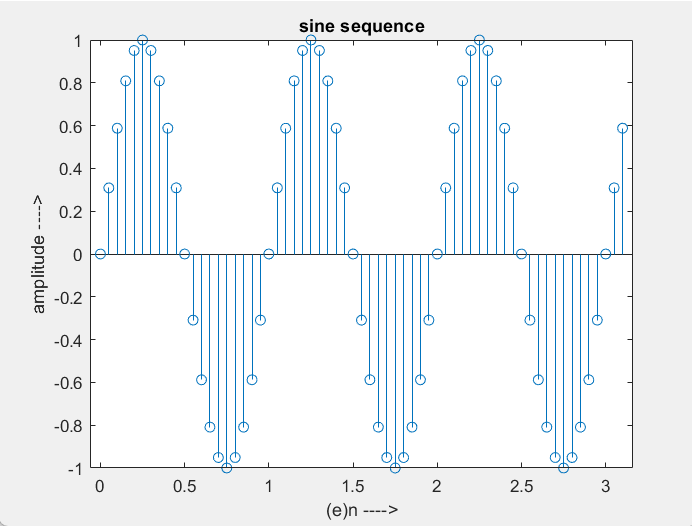
stem(t,y) ;

ylabel('amplitude ---->') ;

xlabel('(e)n ---->') ;

title(' sine sequence ') ;

**EXPECTED GRAPH:**

****

**6. Cosine sequence**

**Code:**

t=0:0.05:pi ;

y=cos(2\*pi\*t) ;

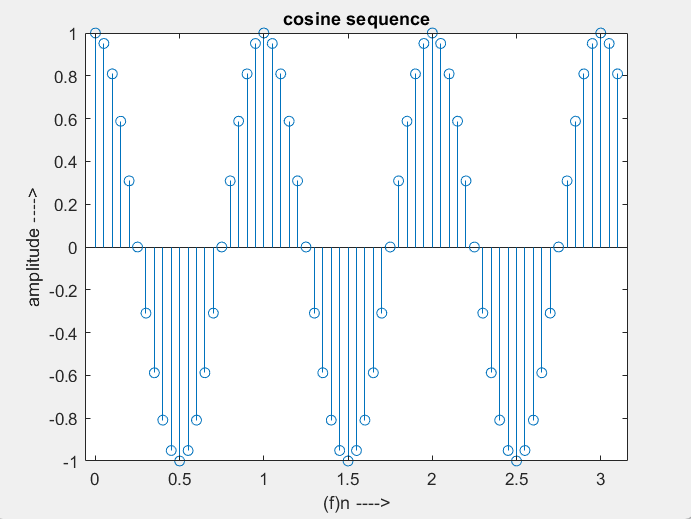
stem(t,y) ;

ylabel('amplitude ---->') ;

xlabel('(f)n ---->') ;

title(' cosine sequence ') ;

**EXPECTED GRAPH:**

****

**Lab 02**

**1.Scaling:**

**Code:**

clc;

clear all;

close all;

x1=input('enter the scaling value value');

n=input('enter the N value');

t=0:1:n-1 ;

x=ones(1,n) ;

y=x1\*x;

subplot(2,2,1);

stem(t,x) ;

ylabel('amplitude ---->') ;

xlabel('(b)n ---->') ;

title(' unit step sequence ') ;

subplot(2,2,2);

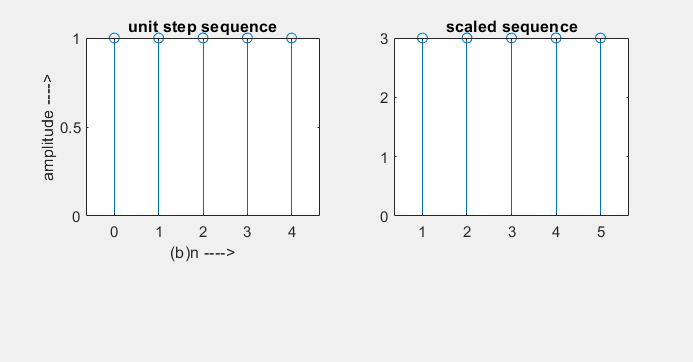
stem(y);

title('scaled sequence');

**OUTPUT:**

Enter the scaling value value 3

Enter the N value 5

**EXPECTED GRAPH: **

**2.Addition:** This is a sample-by-sample addition given by and the length of x1(n) and x2(n) must be the same.

**Code:**

clear all;

close all;

x1=0;

y1=0;

x=input('ENTER THE FIRST SEQUENCE: ');

subplot(3,1,1);

stem(x);

title('X');

y=input('ENTER THE SECOND SEQUENCE: ');

subplot(3,1,2);

stem(y);

title('Y');

l1= length(x);

l2=length(y);

l3=l1-l2;

if l3>0

y= [y,zeros(1,l3)];

elseif l3<0

x=[x,zeros(1,abs(l3))];

else

disp('doable')

end

z=x+y;

disp(z)

subplot(3,1,3);

stem(z);

title('Z=X+Y');

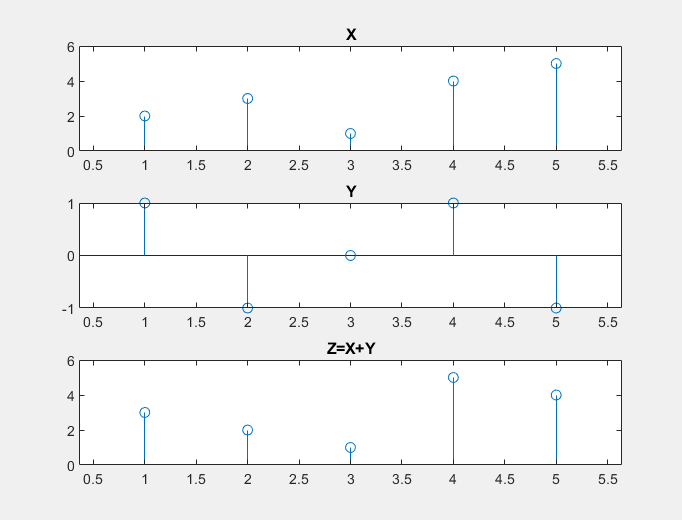
**OUTPUT:**

Enter the first sequence: [2 3 1 4 5]

Enter the second sequence: [1 -1 0 1 -1]

Result: 3 2 1 5 4

**EXPECTED GRAPH:**

****

**3.Shifting**

**y(n)= {x(n-k)}**

**Code:**

clc;

clear all;

close all;

n1=input('Enter the amount to be delayed');

n2=input('Enter the amount to be advanced');

n11=input('ENTER THE LOWER BOUNDARY OF THE FIRST SEQUENCE:');

n12=input('ENTER THE UPPER BOUNDARY OF THE FIRST SEQUENCE:');

n=n11:n12;

x=input('ENTER THE SEQUENCE');

subplot(3,1,1);

stem(n,x);

title('Signal x(n)');

m=n+n1;

y=x;

subplot(3,1,2);

stem(m,y);

title('Delayed signal x(n-n1)');

t=n-n2;

z=x;

subplot(3,1,3);

stem(t,z);

title('Advanced signal x(n+n2)');

**OUTPUT:**

Enter the amount to be delayed 3

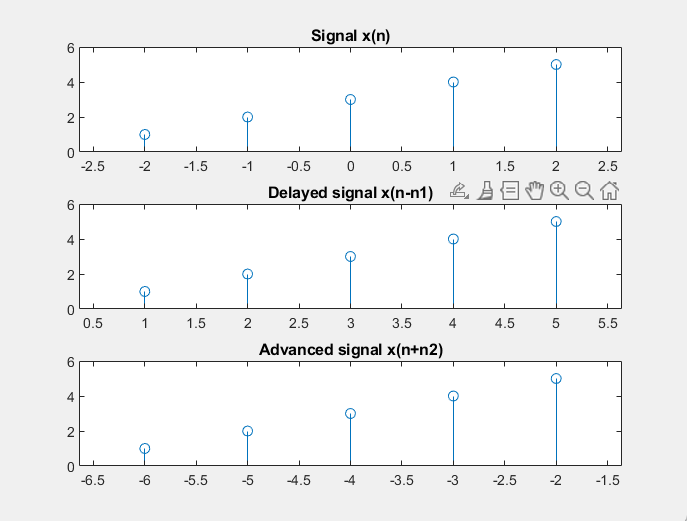
Enter the amount to be advanced 4

ENTER THE LOWER BOUNDARY OF THE FIRST SEQUENCE: -2

ENTER THE UPPER BOUNDARY OF THE FIRST SEQUENCE: 2

ENTER THE SEQUENCE [1 2 3 4 5]

**EXPECTED GRAPH:**

****

**4.Folding or Reversing:**

**y(n)={x(-n)}**

**Code:**

clc;

clear all;

close all;

n11=input('ENTER THE LOWER BOUNDARY OF THE FIRST SEQUENCE:');

n12=input('ENTER THE UPPER BOUNDARY OF THE FIRST SEQUENCE:');

n=n11:n12;

x=input('ENTER THE SEQUENCE');

subplot(2,1,1)

stem(n,x);

axis([-3 3 -5 5]);

title('Signal x(n)');

c=fliplr(x);

y=fliplr(-n);

disp('FOLDED SEQUENCE')

disp(c)

subplot(2,1,2);

stem(y,c);

axis([-3 3 -5 5]);

title('Reversed Signal x(-n)');

**OUTPUT:**

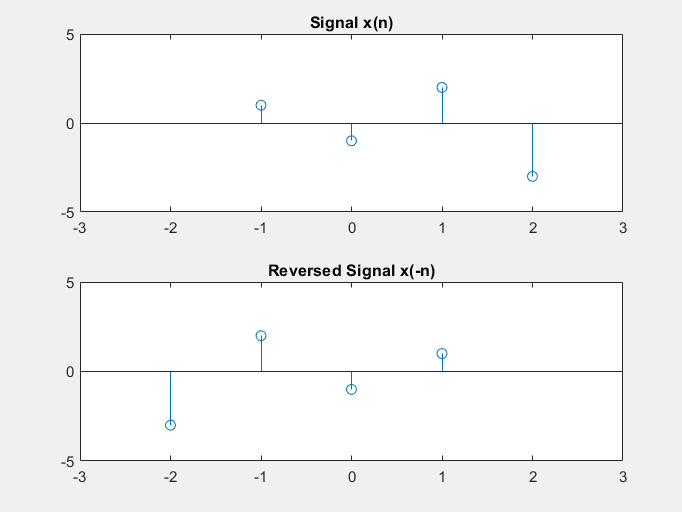
ENTER THE LOWER BOUNDARY OF THE FIRST SEQUENCE: -1

ENTER THE UPPER BOUNDARY OF THE FIRST SEQUENCE: 2

ENTER THE SEQUENCE [1 -1 2 -3]

FOLDED SEQUENCE -3 2 -1 1

**EXPECTED GRAPH:**

****

**Lab 03**

**Convolution**

**a) Linear convolution using conv function**

**Code:**

clc;

clear all;

close all;

x1=input('Enter the first sequence x1(n) = ');

x2=input('Enter the second sequence x2(n) = ');

L=length(x1);

M=length(x2);

N=L+M-1;

yn=conv(x1,x2);

disp('The values of y(n) are= ');

disp(yn);

n1=0:L-1;

subplot(311);

stem(n1,x1);

grid on;

xlabel('n1--->');

ylabel('amplitude--->');

title('First sequence');

n2=0:M-1; subplot(312);

stem(n2,x2);

grid on;

xlabel('n2--->');

ylabel('amplitude--->');

title('Second sequence');

n3=0:N-1;

subplot(313);

stem(n3,yn);

grid on;

xlabel('n3--->');

ylabel('amplitude--->');

title('Convolved output');

**OUTPUT:**

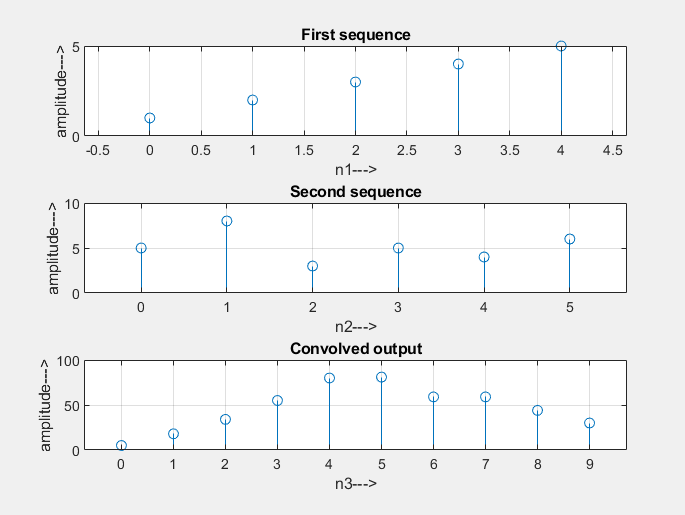
Enter the first sequence x1(n) = [1 2 3 4 5]

Enter the second sequence x2(n) = [5 8 3 5 4 6]

The values of y(n) are=

5 18 34 55 80 81 59 59 44 30

**EXPECTED GRAPH:**

****

**b)Circular convolution**

**Code:**

clc;

clear all;

close all;

a = input('enter the sequence x(n) = ');

b = input('enter the sequence h(n) = ');

n1=length(a);

n2=length(b);

N=max(n1,n2);

x= [a,zeros(1,(N-n1))];

h=[b,zeros(1,N - n2)];

H = zeros(N, N);

for i = 1:N

H(i, :) = circshift(h, i - 1);

end

y = x \* H';

disp('The output sequence is y(n) = ');

disp(y);

% Plot the circular convolution result

stem(0:N-1, y, 'filled');

title('Circular Convolution');

xlabel('n');

ylabel('y(n)');

**OUTPUT:**

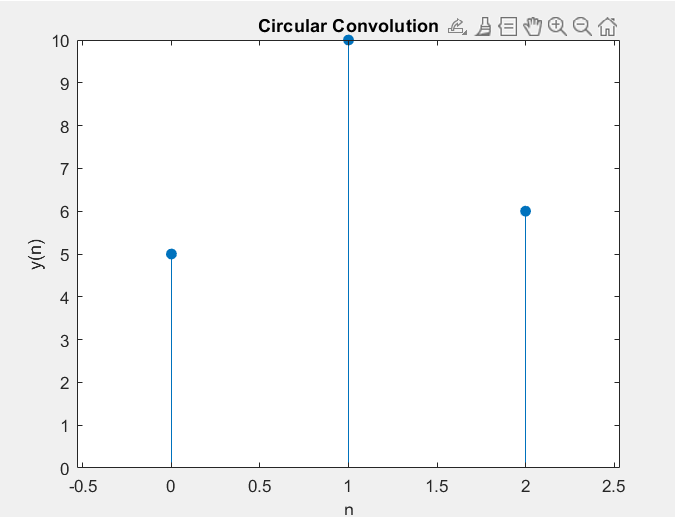
enter the sequence x(n) = [1 2 4]

enter the sequence h(n) = [1 2]

The output sequence is y(n) =

5 10 6

**EXPECTED GRAPH:**

****

**Lab 4: Discrete Time Fourier Transform**

**Example-1:**

**● Compute DTFT of x(n) = (0.5)n u(n).**

**code:**

clc;

% Define the signal

n = 0:50;

x = (0.9).^n; % Right-sided exponential signal

% Define frequency range for DTFT

omega = linspace(-pi, pi, 1000);

X = zeros(size(omega)); % Initialize DTFT result

% Compute DTFT using summation formula

for k = 1:length(omega)

X(k) = sum(x .\* exp(-1j \* omega(k) \* n)); % DTFT formula

end

% Plot magnitude and phase response

figure;

subplot(2,1,1);

plot(omega, abs(X), 'r', 'LineWidth', 1.5);

xlabel('\omega (rad/sample)'); ylabel('|X(\omega)|');

title('Magnitude Spectrum of DTFT');

grid on;

subplot(2,1,2);

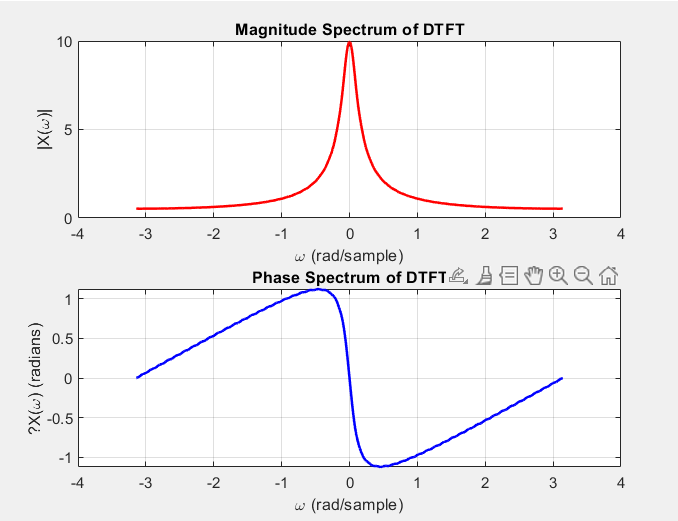
plot(omega, angle(X), 'b', 'LineWidth', 1.5);

xlabel('\omega (rad/sample)'); ylabel('?X(\omega) (radians)');

title('Phase Spectrum of DTFT');

grid on;

**EXPECTED GRAPH:**

****

**Example-2:** Compute the DTFT for different signal lengths:

* x(n) =(0.9)^n for n=0 to **10, 20, and 50.**

**code:**

clc;

N\_values = [10, 20, 50]; % Different signal lengths

omega = linspace(-pi, pi, 1000); % Frequency range

figure;

for i = 1:length(N\_values)

N = N\_values(i);

n = 0:N; % Time index

x = (0.9).^n; % Exponential signal

% Compute DTFT

X = zeros(size(omega));

for k = 1:length(omega)

X(k) = sum(x .\* exp(-1j \* omega(k) \* n));

end

% Plot magnitude response

subplot(3,1,i);

plot(omega, abs(X), 'LineWidth', 1.5);

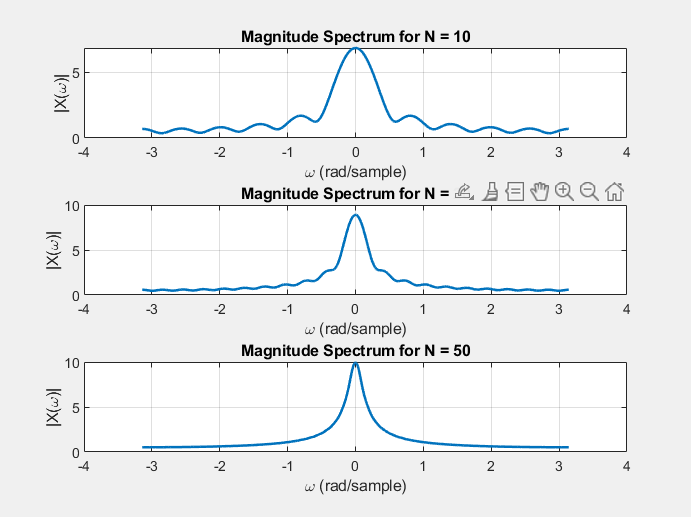
xlabel('\omega (rad/sample)'); ylabel('|X(\omega)|');

title(['Magnitude Spectrum for N = ', num2str(N)]);

grid on;

end

**EXPECTED GRAPH:**

****

**Example-3:**

* **DTFT** of a Sinusoidal Signal: x(n)=cos(0.5πn)

**code:**

clc;

clear;

close all;

n = 0:50;

x = cos(0.5 \* pi \* n);

omega = linspace(-pi, pi, 1000);

X = zeros(size(omega));

% Compute DTFT

for k = 1:length(omega)

X(k) = sum(x .\* exp(-1j \* omega(k) \* n));

end

% Plot magnitude response

figure;

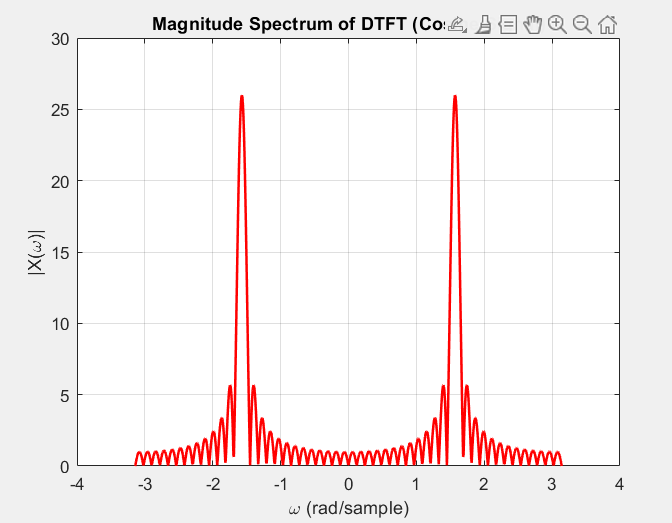
plot(omega, abs(X), 'r', 'LineWidth', 1.5);

xlabel('\omega (rad/sample)'); ylabel('|X(\omega)|');

title('Magnitude Spectrum of DTFT (Cosine Signal)');

grid on;

**EXPECTED GRAPH:**

****

**Lab 5: Digital Image Processing**

**● Grayscale Conversion and Edge Detection**

**Code:**

clc;

clear all;

close all;

% Load the image

img = imread('sample.jpg'); % Use any image file you have

% Convert the image to grayscale

gray\_img = rgb2gray(img);

% Display the original and grayscale images

subplot(1, 3, 1);

imshow(img);

title('Original Image');

subplot(1, 3, 2);

imshow(gray\_img);

title('Grayscale Image');

% Apply edge detection using Sobel filter

edges\_img = edge(gray\_img, 'Sobel');

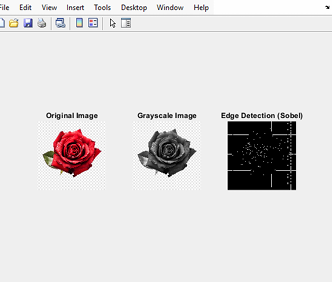
% Display the edges

subplot(1, 3, 3);

imshow(edges\_img);

title('Edge Detection (Sobel)');

**EXPECTED GRAPH:**

****

**Lab Task**

**1.** **Generate a sinusoidal signal, add random noise to it and Plot the signal.(noise can be generated by matlab different random functions).**

**code:**

clc;

clear all;

close all;

% Define parameters

fs = 1000;

t = 0:1/fs:1;

f = 5;

x = sin(2 \* pi \* f \* t);

noise\_uniform = rand(size(t)) - 0.5;

noise\_gaussian = 0.2 \* randn(size(t));

y\_uniform = x + noise\_uniform;

y\_gaussian = x + noise\_gaussian;

subplot(3,1,1);

plot(t, x, 'b', 'LineWidth', 1.5);

title('Original Sinusoidal Signal');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

subplot(3,1,2);

plot(t, y\_uniform, 'r');

title('Sinusoidal Signal with Uniform Noise');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

subplot(3,1,3);

plot(t, y\_gaussian, 'g');

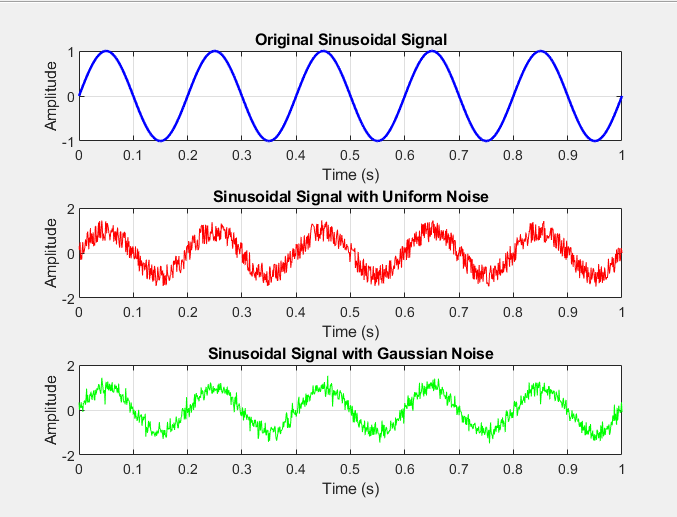
title('Sinusoidal Signal with Gaussian Noise');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

**EXPECTED GRAPH:**

****

**2. Generate a signal: x = [2, -1, 4, 7, -3, 5, 0]. (value 7 is in n=0 position) Compute the even and odd part. Plot x[n], xeven[n], xodd[n] and verify that x[n] = xeven[n] + xodd[n]**

**code:**

clc;

clear all;

close all;

x = [2, -1, 4, 7, -3, 5, 0];

n = -3:3;

x\_even = (x + fliplr(x)) / 2;

x\_odd = (x - fliplr(x)) / 2;

x\_reconstructed = x\_even + x\_odd;

subplot(4,1,1);

stem(n, x);

title('Original Signal x[n]');

xlabel('n');

ylabel('Amplitude');

grid on;

subplot(4,1,2);

stem(n, x\_even);

title('Even Part of x[n]');

xlabel('n');

ylabel('Amplitude');

grid on;

subplot(4,1,3);

stem(n, x\_odd);

title('Odd Part of x[n]');

xlabel('n');

ylabel('Amplitude');

grid on;

subplot(4,1,4);

stem(n, x\_reconstructed);

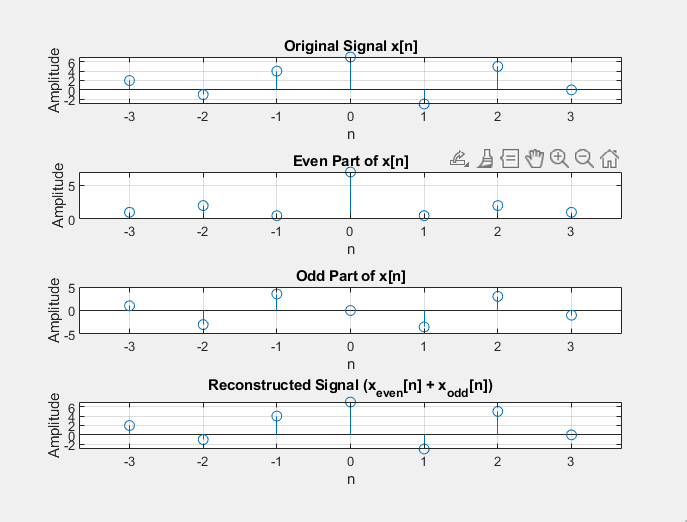
title('Reconstructed Signal (x\_{even}[n] + x\_{odd}[n])');

xlabel('n');

ylabel('Amplitude');

grid on;

**EXPECTED GRAPH:**

****

**3. Generate a decaying exponential signal: x[n] = (0.5)^n u[n].Compute and show the total energy and Power of the signal. Verify whether the signal is an energy or power signal.**

**code:**

clc;

clear all;

close all;

n = 0:50;

x = (0.5).^n;

energy = sum(abs(x).^2);

power = mean(abs(x).^2);

fprintf('Total Energy of the Signal: %.4f\n', energy);

fprintf('Average Power of the Signal: %.4f\n', power);

if isinf(energy)

fprintf('The signal is a Power Signal.\n');

elseif energy > 0 && power == 0

fprintf('The signal is an Energy Signal.\n');

else

fprintf('The signal does not strictly fall into either category.\n');

end

figure;

stem(n, x);

title('Decaying Exponential Signal x[n] = (0.5)^n u[n]');

xlabel('n');

ylabel('x[n]');

grid on;

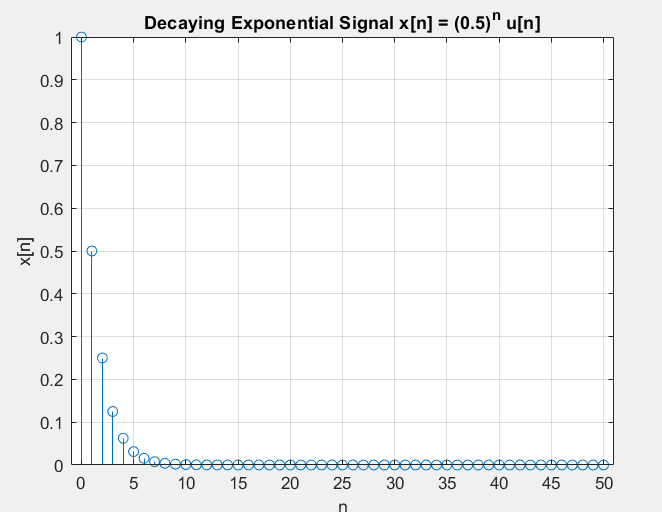
**OUTPUT:**

Total Energy of the signal: 1.3333

Power of the signal: 0.026144

The given signal is an ENERGY signal.

**EXPECTED GRAPH:**

****

**4. Define a rectangular pulse: x[n] = u[n] − u[n−20].**

**Compute and plot its magnitude spectrum and phase spectrum.**

**Code:**

clc;

clear all;

close all

n = -50:50;

x = (n>=0 & n<20);

omega = linspace(-pi, pi, 1000);

X = zeros(size(omega));

for k = 1:length(omega)

X(k) = sum(x .\* exp(-1j \* omega(k) \* n));

end

subplot(2, 1, 1);

plot(omega, abs(X), 'r');

xlabel('|X(\omega)|');

ylabel('\omega (rad/sample)');

title('Magnitude Spectrum of Rectangular Pulse');

grid on;

subplot(2, 1, 2);

plot(omega, angle(X), 'b');

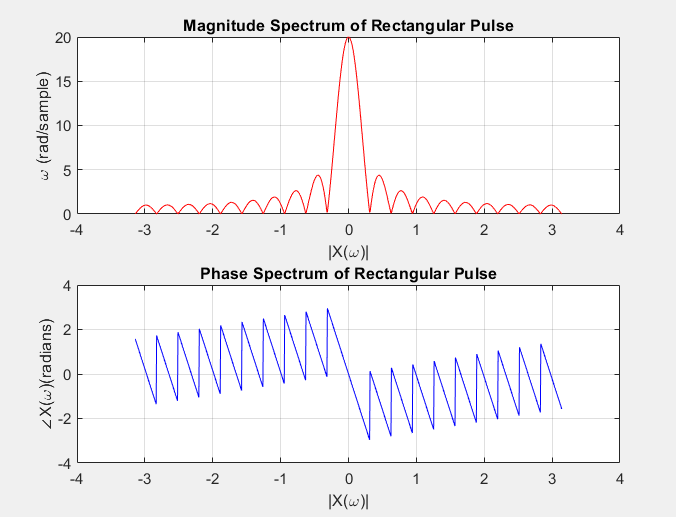
xlabel('|X(\omega)|');

ylabel('\angleX(\omega)(radians)');

title('Phase Spectrum of Rectangular Pulse');

grid on;

**EXPECTED GRAPH:**

****

**5. Create a signal: x[n] = sin(0.1πn) + 0.5sin(0.3πn).**

**Compute and plot the DTFT magnitude response.**

**Code:**

clc;

clear all;

close all

n = 0:50;

x = sin(0.1\*pi\*n) + 0.5.\*sin(0.3\*pi\*n);

omega = linspace(-pi, pi, 1000);

X = zeros(size(omega));

for k = 1:length(omega)

X(k) = sum(x.\*exp(-1j \* omega(k) \* n));

end

plot(omega, abs(X));

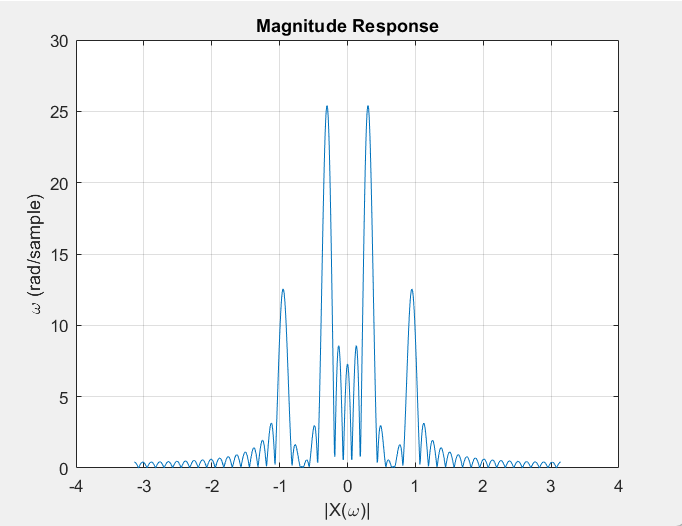
xlabel('|X(\omega)|');

ylabel('\omega (rad/sample)');

title('Magnitude Response');

grid on;

**EXPECTED GRAPH:**

****