SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyalurmetta, Nandyal – 518503. Website: www.svrec.ac.in

Department of Electronics and Communication Engineering

BASIC SIMULATIONLABORATORY

II B.Tech (ECE) I Semester

2020-21

STUDENT NAME	
ROLL NUMBER	
SECTION	

SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyalurmetta, Nandyal – 518503. Website: www.svrec.ac.in

Department of Electronics and Communication Engineering

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

ACADEMIC YEAR: 2020-21

This is to certify that the bonafide record work done by

*Mr./Ms.*____ *bearing*

*H.T.No.*__________________ *of II B.Tech I Semester in the* **Basic simulationLaboratory.**

Faculty In-Charge **Head of the Department**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B. Tech – II-I Sem L T P C 2 0 0 1

19A04303 -- BASIC SIMULATION LAB

Course Objectives: To provide practical exposure with generation and simulation of basic signals using standardized tools.

To teach analyzing signals and sequences using Fourier, Laplace and Z-transforms.

To enable to write programs for signal processing applications.

List of Experiments:

1. Write a program to generate various Signals and Sequences: Periodic and Aperiodic, Unit Impulse, Unit Step, Square, Saw tooth, Triangular, Sinusoidal, Ramp, Sinc function.

2. Perform operations on Signals and Sequences: Addition, Multiplication, Scaling, Shifting, Folding, Computation of Energy and Average Power.

3. Write a program to find the trigonometric & exponential Fourier series coefficients of a rectangular periodic signal. Reconstruct the signal by combining the Fourier series coefficients with appropriate weightages- Plot the discrete spectrum of the signal.

4. Write a program to find Fourier transform of a given signal. Plot its amplitude and phase spectrum.

5. Write a program to convolve two discrete time sequences. Plot all the sequences.

6. Write a program to find autocorrelation and cross correlation of given sequences.

7. Write a program to verify Linearity and Time Invariance properties of a given Continuous/Discrete System.

8. Write a program to generate discrete time sequence by sampling a continuous time signal. Show that with sampling rates less than Nyquist rate, aliasing occurs while reconstructing the signal.

9. Write a program to find magnitude and phase response of first order low pass and high pass filter. Plot the responses in logarithmic scale.

10. Write a program to find response of a low pass filter and high pass filter, when a speech signal is passed through these filters.

11. Write a program to generate Complex Gaussian noise and find its mean, variance, Probability Density Function (PDF) and Power Spectral Density (PSD).

12. Generate a Random data (with bipolar) for a given data rate (say 10kbps). Plot the same for a time period of 0.2 sec.

13. To plot pole-zero diagram in S-plane/Z-plane of given signal/sequence and verify its stability.

Note: All the experiments are to be simulated using MATLAB or equivalent software.

ECE DEPT VISION & MISSION PEOs and PSOs

Vision

To produce highly skilled, creative and competitive Electronics and Communication Engineers to meet the emerging needs of the society.

Mission

- Impart core knowledge and necessary skills in Electronics and Communication Engineering through innovative teaching and learning.
- Inculcate critical thinking, ethics, lifelong learning and creativity needed for industry and society
- Cultivate the students with all-round competencies, for career, higher education and self-employability

I. PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1:	Graduates apply their knowledge of mathematics and science to identify, analyze and solve problems in the field of Electronics and develop sophisticated communication systems.
PEO2:	Graduates embody a commitment to professional ethics, diversity and social awareness in their professional career.
PEO3:	Graduates exhibit a desire for life-long learning through technical training and professional activities.

II. PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1:	Apply the fundamental concepts of electronics and communication engineering to design a variety of components and systems for applications including signal processing, image processing, communication, networking, embedded systems, VLSI and control system
PSO2:	Select and apply cutting-edge engineering hardware and software tools to solve complex Electronics and Communication Engineering problems.

III. PROGRAMME OUTCOMES (PO'S)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

IV. COURSE OBJECTIVES

- To familiarize the students with basic analog communication systems.
- Integrate theory with experiments so that the students appreciate the knowledge gained from the theory course.
- Understand all types of analog modulation / demodulation principles.
- Substantiate pulse modulation techniques.
- To design and implement different modulation and demodulation techniques.
- To write and execute programs in MATLAB to implement various modulation techniques.

V. COURSE OUTCOMES

After the completion of the course students will be able to

Course Outcomes	Course Outcome statements	BTL
CO1	Understand the basic concepts of programming in MATLAB and explain use of built-in functions to perform assigned task.	3
CO2	Generate signals and sequences, Input signals to the systems to perform various operations	3
CO3	Analyze signals using Fourier, Laplace and Z-transforms	3
CO4	Compute Fourier transform of a given signal and plot its magnitude and phase spectrum	3
CO5	Verify Sampling theorem, Determine Convolution and Correlation between signals and sequences.	3

VI.COURSE MAPPING WITH PO'S AND PEO'S

Course Title	P01	P02	P03	P04	P05	P06	P07	P08	P09	P010	P011	P012	PE01	PE02
Analog communications Lab	2.60	2.4	2.6	2.6	2.8	2.4	2.2	2	2.20	2.8	2.4	2	2.4	2.8

V MAPPING OF COURSE OUTCOMES WITH PEO'S AND PO'S

Course Title	P01	P02	P03	P04	P05	P06	P07	P08	P09	P010	P011	P012	PE01	PE02
CO1	3	3	2	3	3	2	2	3	3	3	3	3	3	3
CO2	1	3	3	2	2	3	3	2	3	2	3	1	2	3
CO3	3	3	3	3	3	3	2	2	2	3	3	2	1	3
CO4	3	2	3	2	3	2	3	1	1	3	1	1	3	2
CO5	3	1	2	3	3	2	1	2	2	3	2	3	3	3

LABORATORY INSTRUCTIONS

1. While entering the Laboratory, the students should follow the dress code. (Wear shoes and White apron, Female Students should tie their hair back).

2. The students should bring their observation book, record, calculator, necessary stationery items and graph sheets if any for the lab classes without which the students will not be allowed for doing the experiment.

3. All the Equipments and components should be handled with utmost care. Any breakage or damage will be charged.

4. If any damage or breakage is noticed, it should be reported to the concerned in charge immediately.

5. The theoretical calculations and the updated register values should be noted down in the observation book and should be corrected by the lab in-charge on the same day of the laboratory session.

6. Each experiment should be written in the record note book only after getting signature from the lab in-charge in the observation notebook.

7. Record book must be submitted in the successive lab session after completion of experiment.

8. 100% attendance should be maintained for the laboratory classes.

Precautions.

1. Check the connections before giving the supply
2. Observations should be done carefully

INDEX

9	Write a program to find magnitude and phase response of first order low pass and high pass filter. Plot the responses in logarithmic scale.	43-44				
10	Write a program to find response of a low pass filter and high pass filter, when a speech signal is passed through these filters.	45				
11	Write a program to generate Complex Gaussian noise and find its mean, variance, Probability Density Function (PDF) and Power Spectral Density (PSD).	46-47				
12	Generate a Random data (with bipolar) for a given data rate (say 10kbps). Plot the same for a time period of 0.2 sec.	48				
13	To plot pole-zero diagram in S-plane/Z-plane of given signal/sequence and verify its stability.	49-51				

MATLAB INTRODUCTION:

MATLAB, which stands for Matrix Laboratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation.

As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

MATLAB is case-sensitive, which means that a + B is not the same as a + b. The MATLAB prompt (») in command window is where the commands are entered.

Operators:

1. + addition
2. -subtraction
3. * multiplication
4. ^ power
5. ' transpose
6. \ left division
7. / right division

Remember that the multiplication, power and division operators can be used in conjunction with a period to specify an element-wise operation.

Built in Functions:

1. Scalar Functions:

Certain MATLAB functions are essentially used on scalars, but operate element-wise when applied to a matrix (or vector). They are summarized below.

1. sin -trigonometric sine

2. cos -trigonometric cosine

3. tan -trigonometric tangent

4. asin -trigonometric inverse sine (arcsine)

5. acos -trigonometric inverse cosine (arccosine)

6. atan -trigonometric inverse tangent (arctangent)

7. exp -exponential

8. log -natural logarithm

9. abs -absolute value 10. sqrt -square root

11. rem -remainder 12. round -round towards nearest integer

13. floor -round towards negative infinity 14. ceil -round towards positive infinity

2. Vector Functions:

Other MATLAB functions operate essentially on vectors returning a scalar value. Some of these functions are given below.

1. max largest component : get the row in which the maximum element lies

2. min smallest component

3. length length of a vector

4. sort sort in ascending order

5. sum sum of elements

6. prod product of elements

7. median median value

8. mean mean value std standard deviation

3. Matrix Functions:

Much of MATLAB's power comes from its matrix functions. These can be further separated into two sub-categories. The first one consists of convenient matrix building functions, some of which are given below.

1. eye -identity matrix

2. zeros -matrix of zeros

3. ones -matrix of ones

4. diag -extract diagonal of a matrix or create diagonal matrices

5. triu -upper triangular part of a matrix

6. tril -lower triangular part of a matrix

7. rand -randomly generated matrix

eg: diag([0.9092;0.5163;0.2661])

```
ans =

0.9092 0 0

0 0.5163 0

0 0 0.2661
```

Commands in the second sub-category of matrix functions are

1. size size of a matrix

2. det determinant of a square matrix

3. inv inverse of a matrix

4. rank rank of a matrix

5. rref reduced row echelon form

6. eig eigenvalues and eigenvectors

1. Write a program to generate various Signals and Sequences: Periodic and Aperiodic, Unit Impulse, Unit Step, Square, Saw tooth, Triangular, Sinusoidal, Ramp, Sinc function.

Aim: Generate various signals and sequences (Periodic and aperiodic), such as Unit Impulse,Unit Step, Square, Saw tooth, Triangular, Sinusoidal, Ramp, Sinc.

Software Required: Matlab software

Theory: If the amplitude of the signal is defined at every instant of time then it is calledcontinuous time signal. If the amplitude of the signal is defined at only at some instants oftime then it is called discrete time signal. If the signal repeats itself at regular intervals then itis called periodic signal. Otherwise they are called aperiodic signals.

EX: ramp,Impulse,unit step, sinc- Aperiodic signalssquare,sawtooth,triangular sinusoidal – periodic signals.

```
» plot(x,y)
```

It is good practice to label the axis on a graph and if applicable indicate what each axisrepresents. This can be done with the xlabel and ylabel commands.

```
» xlabel('x')
```

```
» ylabel('y=cos(x)')
```

Inside parentheses, and enclosed within single quotes, we type the text that we wish to be displayed along the x and y axis, respectively. We could even put a title on top using

```
» title('Graph of cosine from -pi to pi')
```

```
» plot (x,y,’g’)
```

Where the third argument indicating the color, appears within single quotes. We could get a dashed line instead of a solid one by typing

```
» plot (x,y,’--’)
```

or even a combination of line type and color, say a blue dotted line by typing

```
» plot (x,y,’b:’)
```

We can get both graphs on the same axis, distinguished by their line type, using

```
» plot(x,y,'r--',x,z,'b:')
```

When multiple curves appear on the same axis, it is a good idea to create a legend to label and distinguish them. The command legend does exactly this.

» legend ('cos(x)','sin(x)')

The text that appears within single quotes as input to this command, represents the legend labels. We must be consistent with the ordering of the two curves, so since in the plot command we asked for cosine to be plotted before sine, we must do the same here.

At any point during a MATLAB session, you can obtain a hard copy of the current plot by either issuing the command print at the MATLAB prompt, or by using the command menus on the plot window. In addition, MATLAB plots can by copied and pasted (as pictures) in your favorite word processor (such as Microsoft Word). This can be achieved using the Edit menu on the figure window. Another nice feature that can be used in conjunction with plot is the command grid, which places grid lines to the current axis (just like you have on graphing paper). Type help grid for more information. Other commands for data visualization that exist in MATLAB include subplot create an array of (tiled) plots in the same window log log plot using log-log scales semi logx plot using log scale on the x-axis semi logy plot using log scale on the y-axis

The Sinc Function :The sinc function computes the mathematical sinc function for an input vector or matrix x.

Viewed as a function of time, or space, the sinc function is the inverse Fourier transform of the rectangular pulse in frequency centered at zero of width 2p and height

The sinc function has a value of 1 when x is equal to zero, and a value of for all other elements of x.

```
% Generation of signals and sequences

clc;

clear all;

close all;

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%generation of unit impulse signal

t1=-1:0.01:1

y1=(t1==0);

subplot(2,2,1);

plot(t1,y1);

xlabel('time');

ylabel('amplitude');

title('unit impulse signal');

%generation of impulse sequence

subplot(2,2,2);
```

```
stem(t1,y1);

xlabel('n');

ylabel('amplitude');

title('unit impulse sequence');

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%generation of unit step signal

t2=-10:1:10;

y2=(t2>=0);

subplot(2,2,3);

plot(t2,y2);

xlabel('time');

ylabel('amplitude');

title('unit step signal');

%generation of unit step sequence

subplot(2,2,4);

stem(t2,y2);

xlabel('n');

ylabel('amplitude');

title('unit step sequence');

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%generation of square wave signal

t=0:0.002:0.1;

y3=square(2*pi*50*t);

figure;

subplot(2,2,1);

plot(t,y3);

axis([0 0.1 -2 2]);

xlabel('time');

ylabel('amplitude');

title('square wave signal');
```

```
%generation of square wave sequence

subplot(2,2,2);

stem(t,y3);

axis([0 0.1 -2 2]);

xlabel('n');

ylabel('amplitude');

title('square wave sequence');

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%generation of sawtooth signal

y4=sawtooth(2*pi*50*t);

subplot(2,2,3);

plot(t,y4);

axis([0 0.1 -2 2]);

xlabel('time');

ylabel('amplitude');

title('sawtooth wave signal');

%generation of sawtooth sequence

subplot(2,2,4);

stem(t,y4);

axis([0 0.1 -2 2]);

xlabel('n');

ylabel('amplitude');

title('sawtooth wave sequence');

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%generation of triangular wave signal

y5=sawtooth(2*pi*50*t,.5);

figure;

subplot(2,2,1);

plot(t,y5);

axis([0 0.1 -2 2]);
```

```
xlabel('time');

ylabel('amplitude');

title(' triangular wave signal');

%generation of triangular wave sequence

subplot(2,2,2);

stem(t,y5);

axis([0 0.1 -2 2]);

xlabel('n');

ylabel('amplitude');

title('triangular wave sequence');

%generation of sinsoidal wave signal

y6=sin(2*pi*40*t);

subplot(2,2,3);

plot(t,y6);

axis([0 0.1 -2 2]);

xlabel('time');

ylabel('amplitude');

title(' sinsoidal wave signal');

%generation of sin wave sequence

subplot(2,2,4);

stem(t,y6);

axis([0 0.1 -2 2]);

xlabel('n');

ylabel('amplitude');

title('sin wave sequence');

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%generation of ramp signal

y7=t;

figure;

subplot(2,2,1);
```

```
plot(t,y7);

xlabel('time');

ylabel('amplitude');

title('ramp signal');

%generation of ramp sequence

subplot(2,2,2);

stem(t,y7);

xlabel('n');

ylabel('amplitude');

title('ramp sequence');

%generation of sinc signal

t3=linspace(-5,5);

y8=sinc(t3);

subplot(2,2,3);

plot(t3,y8);

xlabel('time');

ylabel('amplitude');

title(' sinc signal');

%generation of sinc sequence

subplot(2,2,4);

stem(y8);

xlabel('n');

ylabel('amplitude');

title('sinc sequence');
```

VIVA VOCE --

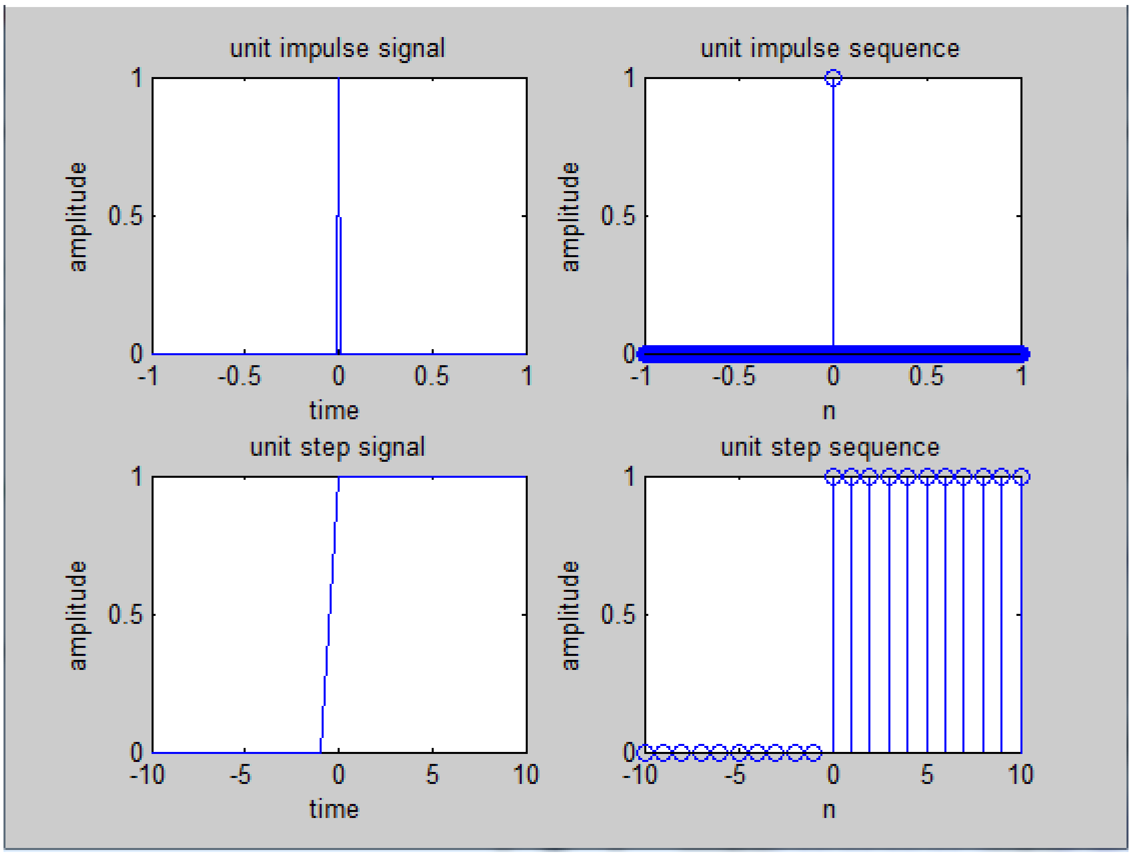
1. Define Continuous-time Signal?

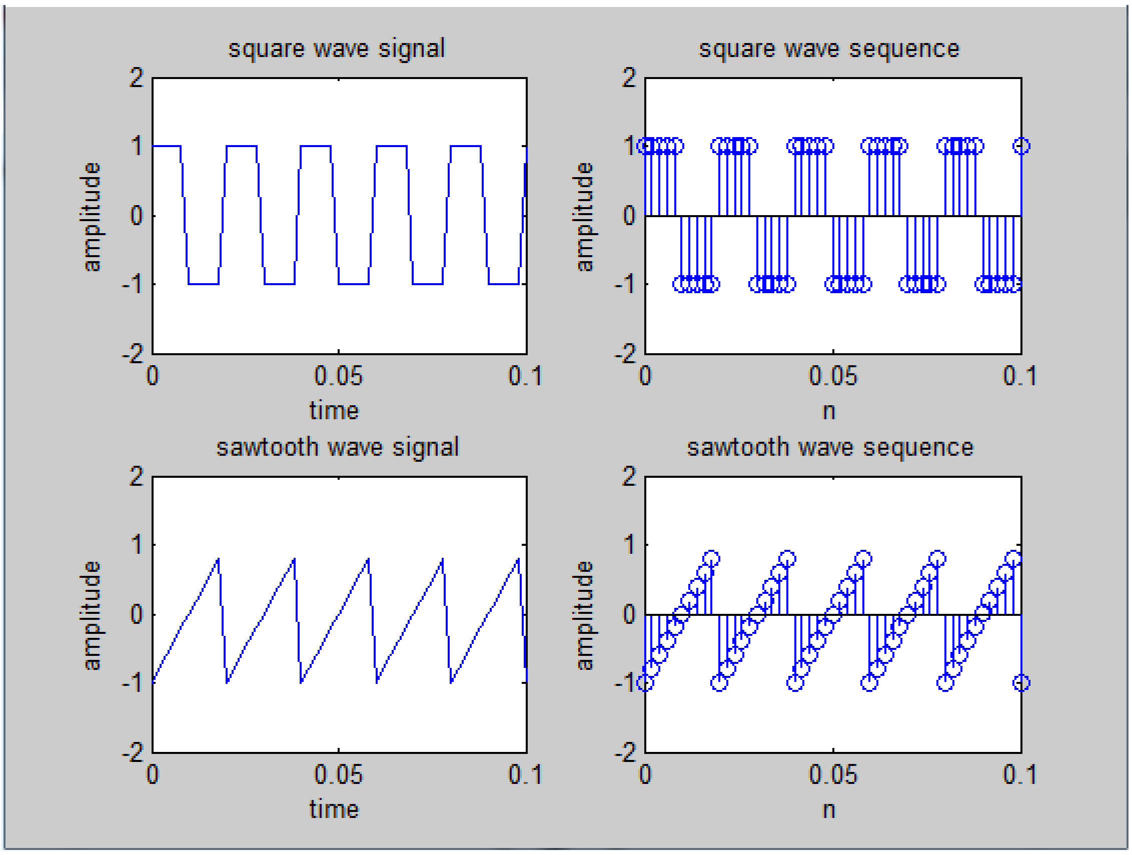
2. Define Signal?

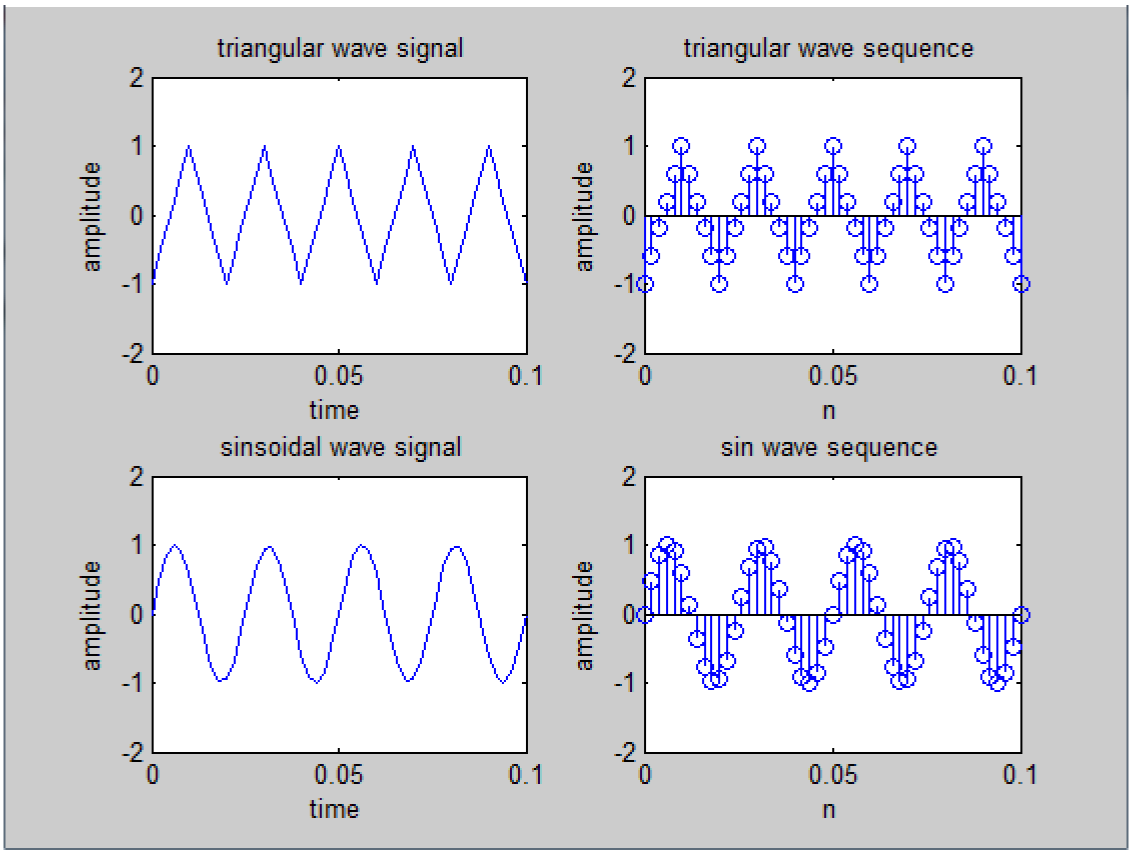
3. What Are The Major Classifications Of The Signal?

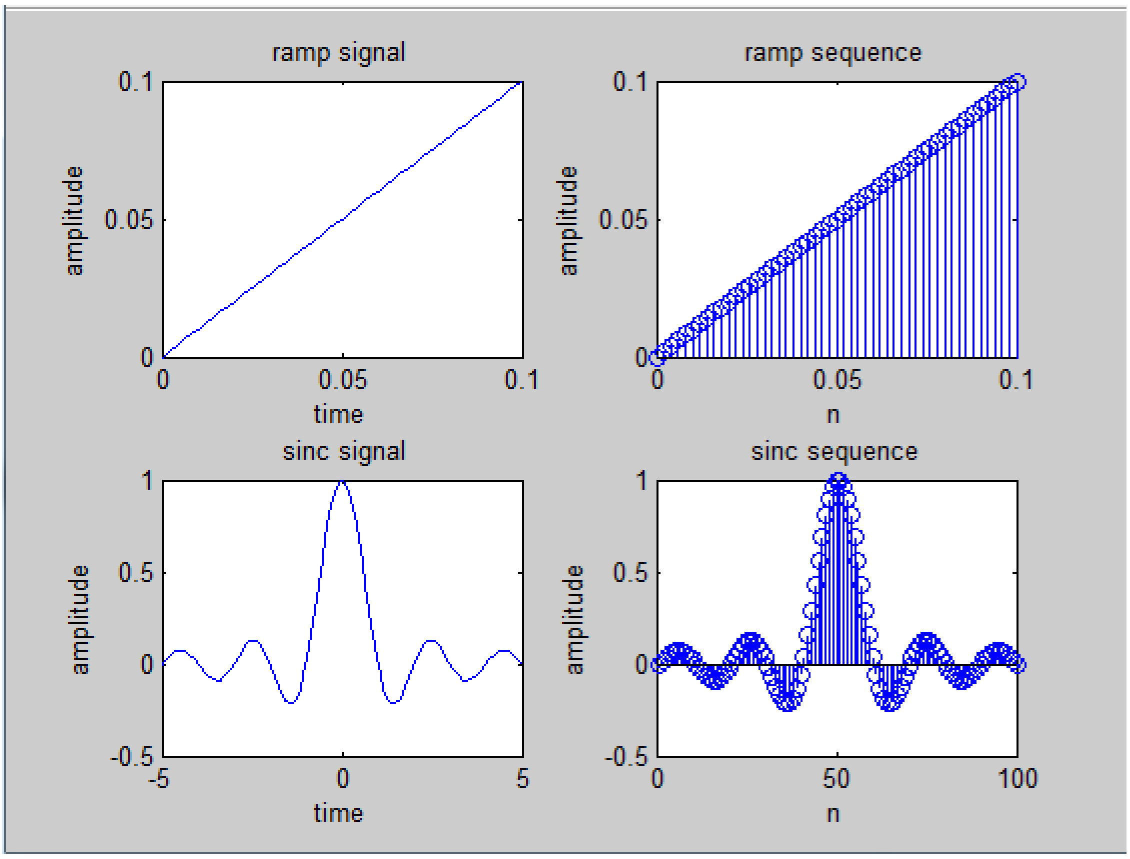
4. What Are The Classification Of Continuous Time Signals? Name Them?

5. Classify Discrete Time Signal?


unit impulse signal
amplitude
time
unit impulse sequence
n
unit step signal
unit step sequence


square wave signal
amplitude
time
square wave sequence
n
sawtooth wave signal
sawtooth wave sequence





Result: Various signals & sequences generated using Matlab software.

2. Perform operations on Signals and Sequences: Addition, Multiplication, Scaling, Shifting, Folding, Computation of Energy and Average Power.

Aim: To performs functions on signals and sequences such as addition, multiplication, scaling, shifting, folding, computation of energy and average power.

Theory:

Signal Addition

Addition: any two signals can be added to form a third signal,

z (t) = x (t) + y (t)

Multiplication:

Multiplication of two signals can be obtained by multiplying their values at every instant. z

z(t) = x (t) y (t)

Time reversal/Folding:

Time reversal of a signal x(t) can be obtained by folding the signal about t=0.

Y(t)=y(-t)

Signal Amplification/Scaling : Y(n)=ax(n) if a < 1 attenuation

a >1 amplification

Time shifting: The time shifting of x(n) obtained by delay or advance the signal in time by using y(n)=x(n+k)

If k is a positive number, y(n) shifted to the right i e the shifting delays the signal

If k is a negative number, y(n) it gets shifted left. Signal Shifting advances the signal

Program :

```
clc;
clear all;
close all;
% generating two input signals
t=0:.01:1;
x1=sin(2*pi*4*t);
x2=sin(2*pi*8*t);
subplot(2,2,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('input signal 1');
subplot(2,2,2);
plot(t,x2);
xlabel('time');
ylabel('amplitude');
```

```
title('input signal 2');
% addition of signals
y1=x1+x2;
subplot(2,2,3);
plot(t,y1);
xlabel('time');
ylabel('amplitude');
title('addition of two signals');
% multiplication of signals
y2=x1.*x2;
subplot(2,2,4);
plot(t,y2);
xlabel('time');
ylabel('amplitude');
title('multiplication of two signals');
% scaling of a signal1
A=2;
y3=A*x1;
figure;
subplot(2,2,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('input signal')
subplot(2,2,2);
plot(t,y3);
xlabel('time');
ylabel('amplitude');
title('amplified input signal');
% folding of a signal1
h=length(x1);
nx=0:h-1;
subplot(2,2,3);
plot(nx,x1);
xlabel('nx');
ylabel('amplitude');
title('input signal')
y4=fliplr(x1);
nf=-fliplr(nx);
subplot(2,2,4);
plot(nf,y4);
xlabel('nf');
ylabel('amplitude');
title('folded signal');
%shifting of a signal 1
figure;
subplot(3,1,1);
plot(t,x1);
xlabel('time t');
ylabel('amplitude');
title('input signal');
```

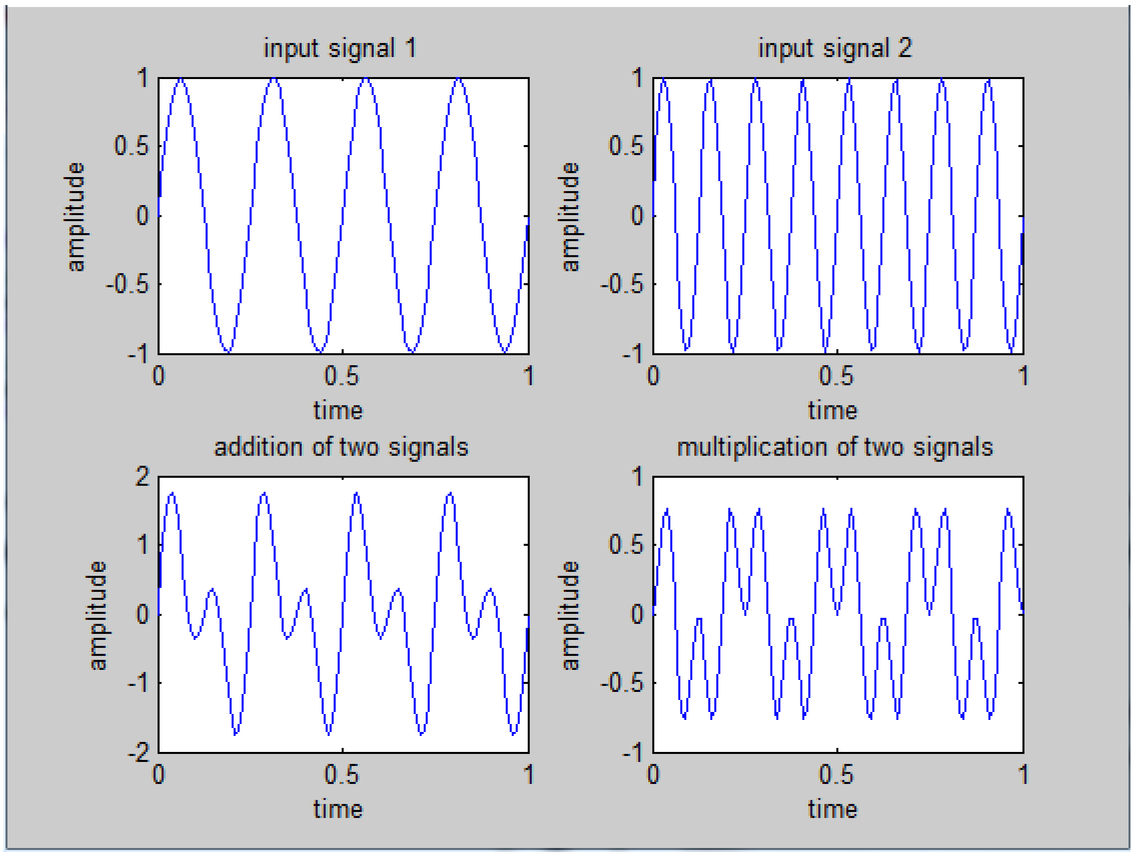
```
subplot(3,1,2);
plot(t+2,x1);
xlabel('t+2');
ylabel('amplitude');
title('right shifted signal');
subplot(3,1,3);
plot(t-2,x1);
xlabel('t-2');
ylabel('amplitude');
title('left shifted signal');
%operations on sequences
n1=1:1:9;
s1=[1 2 3 0 5 8 0 2 4];
figure;
subplot(2,2,1);
stem(n1,s1);
xlabel('n1');
ylabel('amplitude');
title('input sequence1');
s2=[1 1 2 4 6 0 5 3 6];
subplot(2,2,2);
stem(n1,s2);
xlabel('n2');
ylabel('amplitude');
title('input sequence2');
% addition of sequences
s3=s1+s2;
subplot(2,2,3);
stem(n1,s3);
xlabel('n1');
ylabel('amplitude');
title('sum of two sequences');
% multiplication of sequences
s4=s1.*s2;
subplot(2,2,4);
stem(n1,s4);
xlabel('n1');
ylabel('amplitude');
title('product of two sequences');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% program for energy of a sequence
z1=input('enter the input sequence');
e1=sum(abs(z1).^2);
disp('energy of given sequence is');e1
% program for energy of a signal
t=0:pi:10*pi;
z2=cos(2*pi*50*t).^2;
e2=sum(abs(z2).^2);
disp('energy of given signal is');e2
% program for power of a sequence
p1= (sum(abs(z1).^2))/length(z1);
```

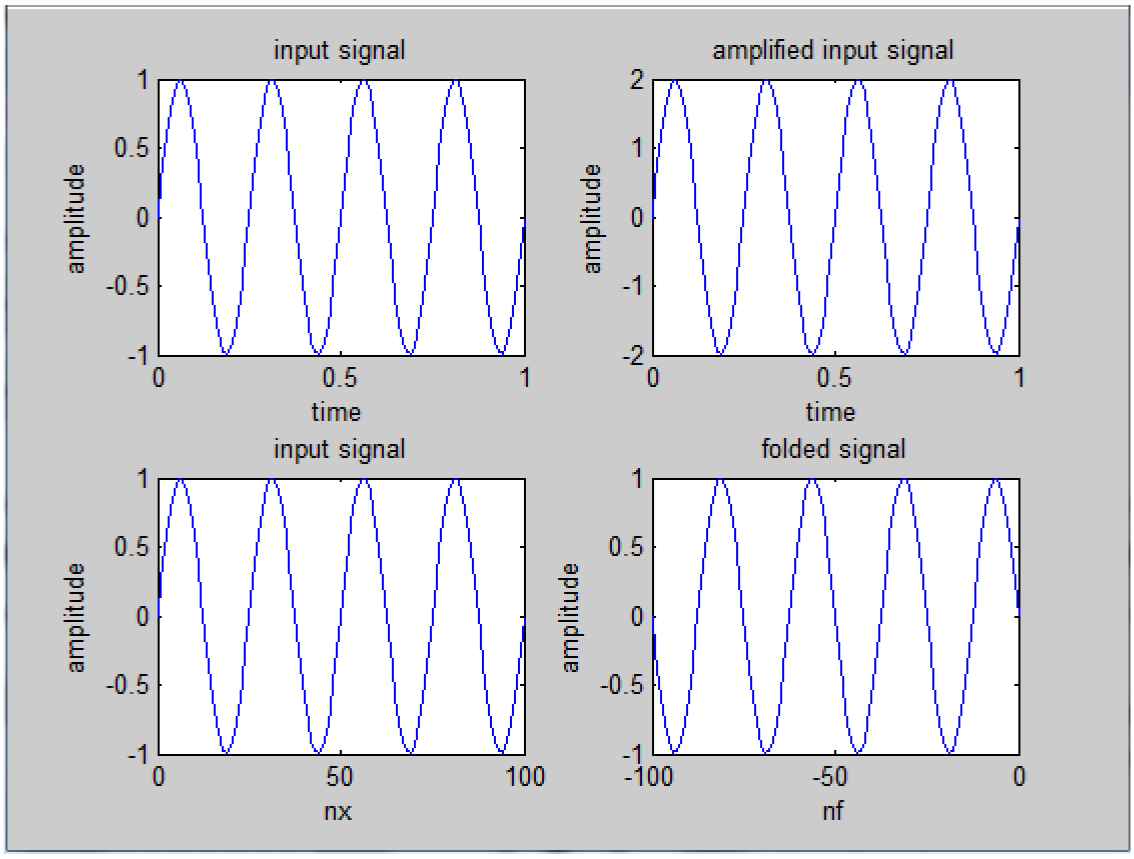
```
disp('power of given sequence is');p1
% program for power of a signal
p2=(sum(abs(z2).^2))/length(z2);
disp('power of given signal is');
```

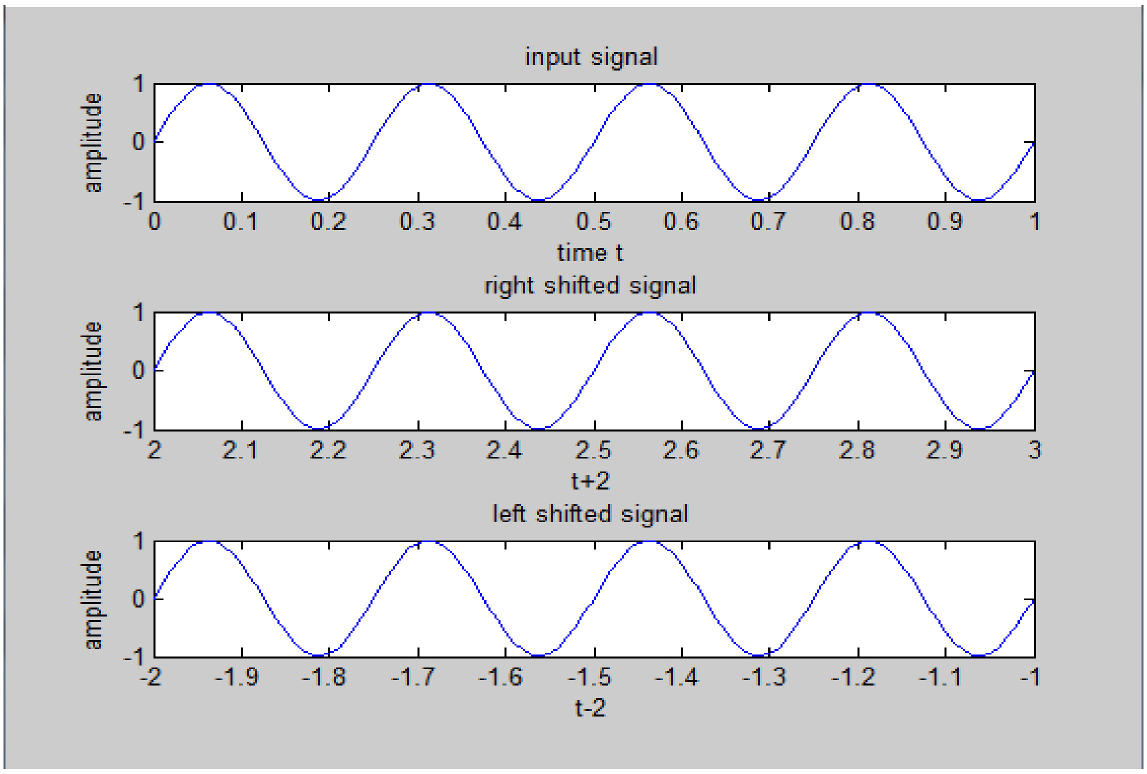
Output:

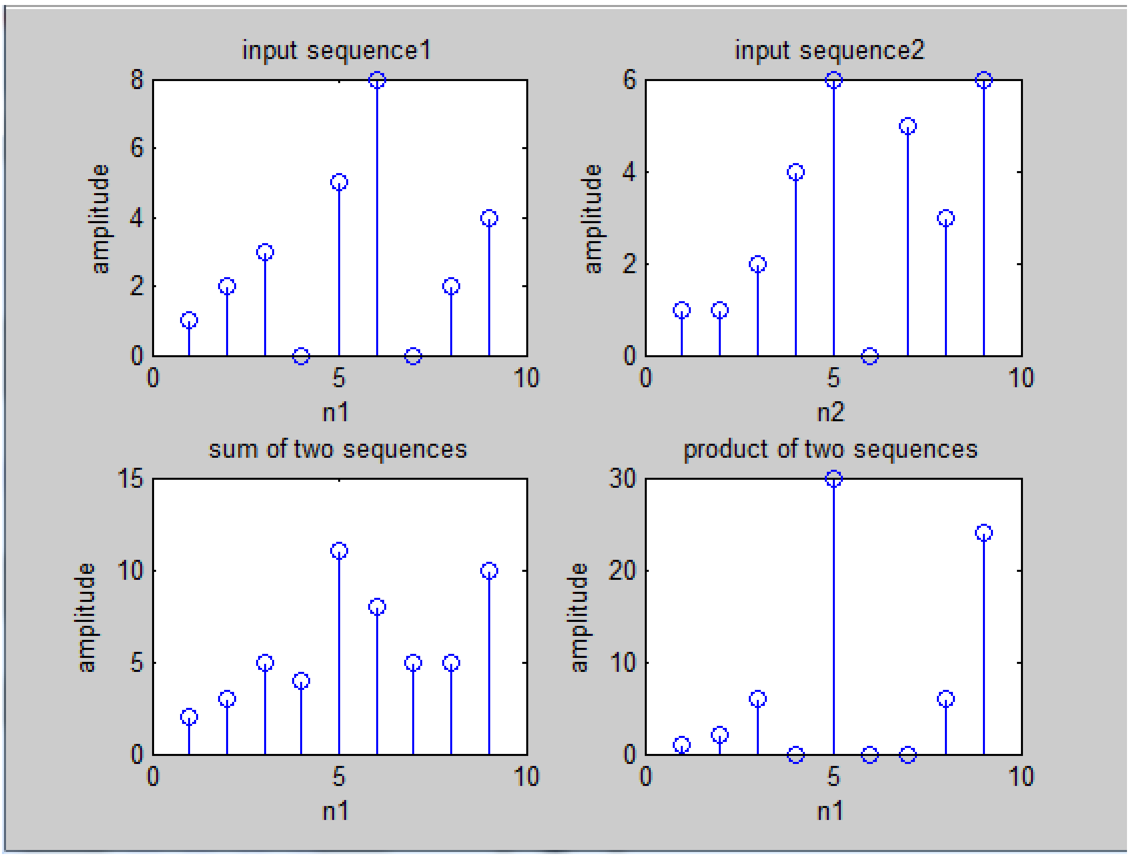
```
enter the input sequence[1 3 2 4 1]
energy of given sequence is
e1 = 31
energy of given signal is
e2 = 4.0388
power of given sequence is
p1 = 6.2000
power of given signal is
p2 = 0.3672
```

OUTPUT --




input signal
amplified input signal
input signal
folded signal
amplitude
time
nx
nf


input signal
right shifted signal
left shifted signal
amplitude
time t
t+2
t-2



Result: Various operations on signals and sequences are performed

VIVA VOCE--

1) Amplitude scaling refers to multiplication of a constant with the given signal.
It is given by y (t) = a x (t). It can be both increase in amplitude or decrease in amplitude.

2) y (t) = x (at), comparing this with the given expression we get a = 1/5. If 0<a<1 then it is expanded (stretched) version of x (t)

3) Time scaling is an example for operations performed on independent variable time.
It is given by y (t) = x (at).

4) AM radio signal is an example for y (t) = x1 (t) * x2 (t) where, x1 (t) consists of an audio signal plus a dc component and x2 (t) is a sinusoidal signal called carrier wave.

5) Audio mixer is a device which combines music and voice signals. It is given by
Y (t) = x1 (t) + x2 (t).

3. Write a program to find the trigonometric & exponential Fourier series coefficients of a rectangular periodic signal. Reconstruct the signal by combining the Fourier series coefficients with appropriate weightages- Plot the discrete spectrum of the signal.

Aim: To find the trigonometric & exponential Fourier series coefficients of a rectangular periodic signal. Reconstruct the signal by combining the Fourier series coefficients with appropriate weightages- Plot the discrete spectrum of the signal.

Theory: to compute the trigonometric fourier series coefficients of a periodic square wave time signal that has a value of 2 from time 0 to 3 and a value of -12 from time 3 to 6. It then repeats itself. I am trying to calculate in MATLAB the fourier series coefficients of this time signal and am having trouble on where to begin.

The equation is x(t) = a0 + sum(bk*cos(2*pi*f*k*t)+ck*sin(2*pi*f*k*t))

The sum is obviously from k=1 to k=infinity.

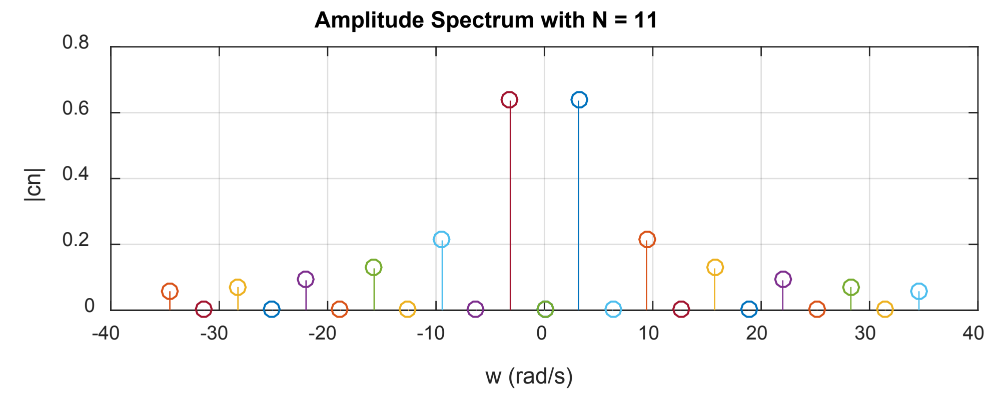
a0, bk, and ck are the coefficients

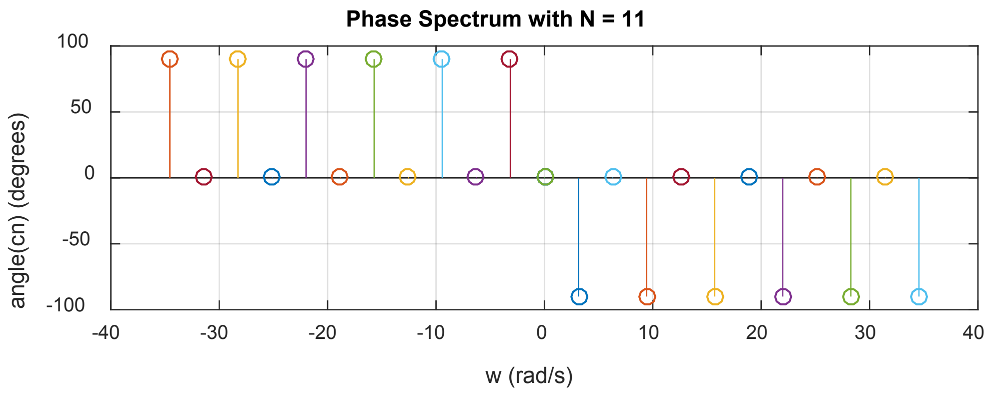
Program:

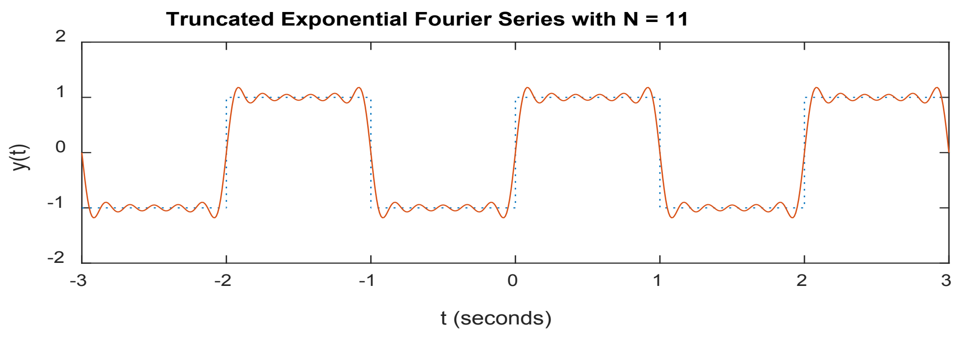
```
% Description: This M-file plots the truncated Fourier Series
%              representation of a square wave as well as its
%              amplitude and phase spectrum.

clear;                             % clear all variables
clf;                               % clear all figures

N = 11;                            % summation limit (use N odd)
wo = pi;                           % fundamental frequency (rad/s)
c0 = 0;                            % dc bias
t = -3:0.01:3;                     % declare time values

figure(1)                          % put first two plots on figure 1

% Compute yce, the Fourier Series in complex exponential form

yce = c0*ones(size(t));            % initialize yce to c0

for n = -N:2:N,                    % loop over series index n (odd)
  cn = 2/(j*n*wo);                 % Fourier Series Coefficient
  yce = yce + real(cn*exp(j*n*wo*t)); % Fourier Series computation
end

subplot(2,1,1)
plot([-3 -2 -2 -1 -1  0 0 1  1  2 2 3],...    % plot original y(t)
     [-1 -1  1  1 -1 -1 1 1 -1 -1 1 1], ':');
hold;
```

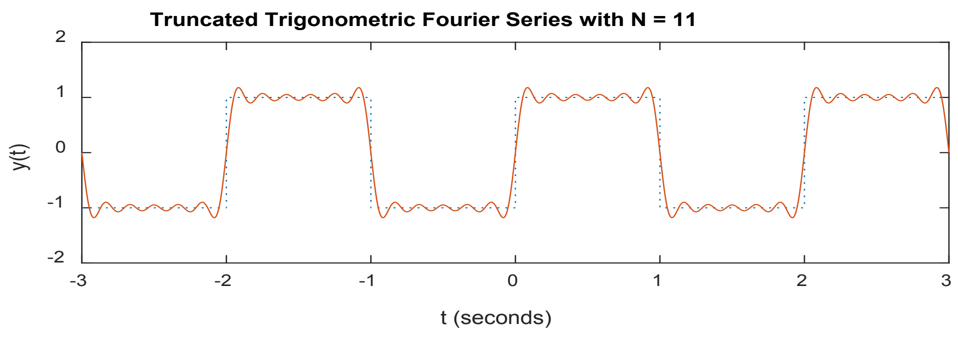
```
plot(t,yce);                  % plot truncated exponential FS
xlabel('t (seconds)'); ylabel('y(t)');
ttle = ['EE341.01: Truncated Exponential Fourier Series with N = ',...
         num2str(N)];
title(ttle);
hold;

% Compute yt, the Fourier Series in trigonometric form

yt = c0*ones(size(t));               % initialize yt to c0

for n = 1:2:N,                       % loop over series index n (odd)
  cn = 2/(j*n*wo);                   % Fourier Series Coefficient
  yt = yt + 2*abs(cn)*cos(n*wo*t+angle(cn));  % Fourier Series computation
end

subplot(2,1,2)
plot([-3 -2 -2 -1 -1  0 0 1  1  2 2 3],...    % plot original y(t)
     [-1 -1  1  1 -1 -1 1 1 -1 -1 1 1], ':');
hold;                                 % plot truncated trigonometric FS
plot(t,yt);
xlabel('t (seconds)'); ylabel('y(t)');
ttle = ['EE341.01: Truncated Trigonometric Fourier Series with N = ',...
         num2str(N)];
title(ttle);
hold;

% Draw the amplitude spectrum from exponential Fourier Series

figure(2)                            % put next plots on figure 2

subplot(2,1,1)
stem(0,c0);                          % plot c0 at nwo = 0

hold;
for n = -N:2:N,                      % loop over series index n
  cn = 2/(j*n*wo);                   % Fourier Series Coefficient
  stem(n*wo,abs(cn))                 % plot |cn| vs nwo
end
for n = -N+1:2:N-1,                  % loop over even series index n
  cn = 0;                            % Fourier Series Coefficient
  stem(n*wo,abs(cn));                % plot |cn| vs nwo
end

xlabel('w (rad/s)')
ylabel('|cn|')
ttle = ['EE341.01: Amplitude Spectrum with N = ',num2str(N)];
title(ttle);
grid;
hold;

% Draw the phase spectrum from exponential Fourier Series

subplot(2,1,2)
stem(0,angle(c0)*180/pi);            % plot angle of c0 at nwo = 0

hold;
for n = -N:2:N,                      % loop over odd series index n
  cn = 2/(j*n*wo);                   % Fourier Series Coefficient
  stem(n*wo,angle(cn)*180/pi);      % plot |cn| vs nwo
end
for n = -N+1:2:N-1,                 % loop over even series index n
  cn = 0;                           % Fourier Series Coefficient
  stem(n*wo,angle(cn)*180/pi);      % plot |cn| vs nwo
end
xlabel('w (rad/s)')
ylabel('angle(cn) (degrees)')
ttle = ['EE341.01: Phase Spectrum with N = ',num2str(N)];
title(ttle);
grid;
hold;
```

OUTPUT--









Result: Trigonometric & exponential Fourier series coefficients of a rectangular periodic signals are plotted.

VIVA VOCE--

1) Trigonometric F S

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x).dx \qquad a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx.dx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nx.dx.$$

Exponential F S

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}, \text{ where } c_n = \frac{1}{2\pi}\int_{c}^{c+2\pi} f(x)e^{-inx}dx.$$

4. Write a program to find Fourier transform of a given signal. Plot its amplitude and phase spectrum.

Aim: To find the Fourier Transform of a given signal and plotting its magnitude and phasespectrum.

Software Required: Matlab software

Theory:

Fourier Transform:

The Fourier transform as follows. Suppose that f is a function which is zero outside of someinterval $[-L/2, L/2]$. Then for any $T \geq L$ we may expand f in a Fourier series on the interval$[-T/2,T/2]$, where the "amount" of the wave $e2\pi inx/T$ in the Fourier series of f is given byBy definition Fourier Transform of signal f(t) is defined as

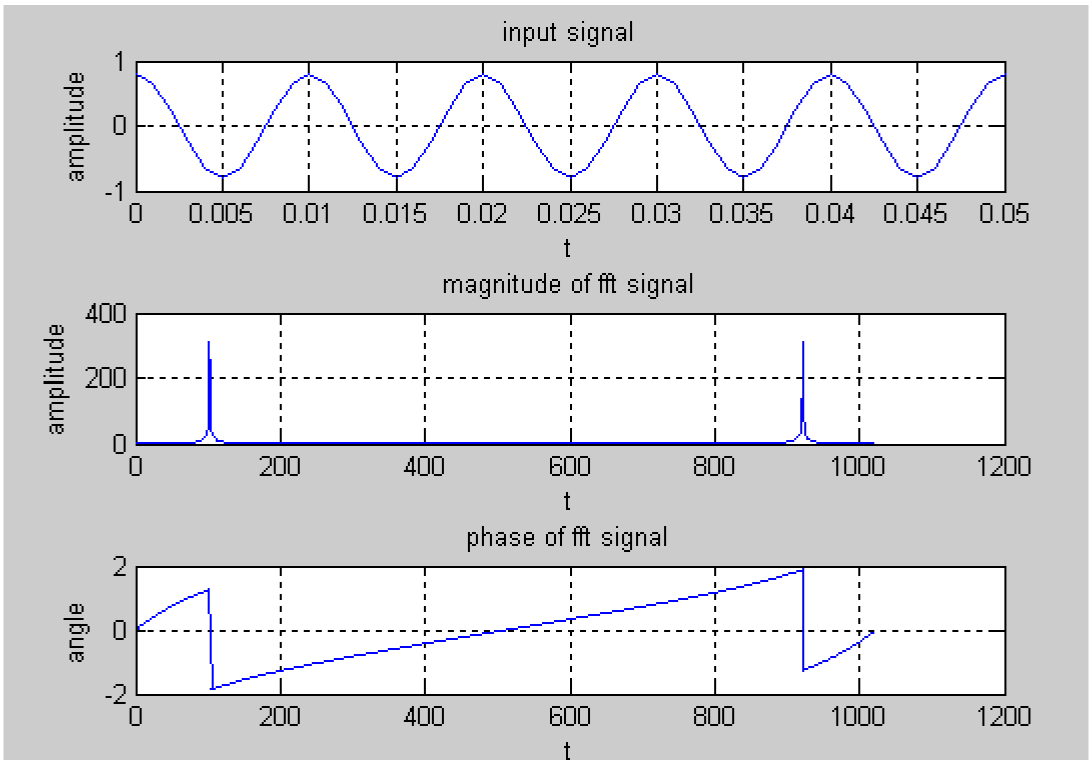
$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt$$

Program:

```
clc;
clear all;
close all;
fs=1000;
N=1024; % length of fft sequence
t=[0:N-1]*(1/fs);
% input signal
x=0.8*cos(2*pi*100*t);
subplot(3,1,1);
plot(t,x);
axis([0 0.05 -1 1]);
grid;
xlabel('t');
ylabel('amplitude');
title('input signal');
% Fourier transform of given signal
x1=fft(x);
% magnitude spectrum
k=0:N-1;
Xmag=abs(x1);
subplot(3,1,2);
plot(k,Xmag);
grid;
xlabel('t');
ylabel('amplitude');
title('magnitude of fft signal')
%phase spectrum
Xphase=angle(x1);
subplot(3,1,3);
plot(k,Xphase);
grid;
```

```
xlabel('t');
ylabel('angle');
title('phase of fft signal');
```

OUTPUT--



Result: Magnitude and phase spectrum of FFT of a given signal is plotted.

VIVA VOCE--

1)Given $f(t)= e_{-at}\ u(t)$

We know that $u(t)=\{01t<0t>0$

Fourier transform,

$F(\omega)=\int_{\infty-\infty}f(t)e_{-j\omega t}dt=\int_{\infty-\infty}e_{-at}u(t)e_{-j\omega t}dt=\int_{\infty 0}e_{-(a+j\omega)t}dt$

$F(\omega) = 1a+j\omega,\ a>0.$

2)The given two-sided exponential function $f(t) = e_{-a|t|}$, $a>0$ can be expressed as

$f(t)=\{e_{-at}e_{at}t\geq 0t\leq 0$

The Fourier transform is

$F(\omega)=\int_{\infty-\infty}f(t)e_{-j\omega t}dt=\int_{0-\infty}f(t)e_{-j\omega t}dt+\int_{\infty 0}f(t)e_{-j\omega t}dt$

$F(\omega)=1a+j\omega+1a-j\omega=2aa2+\omega 2.$

5. Write a program to convolve two discrete time sequences. Plot all the sequences

Aim: Write the program for convolution between two signals and also between two sequences.

Software Required: Matlab software

Theory:

Convolution involves the following operations.

1. Folding
2. Multiplication
3. Addition
4. Shifting

Convolution is an integral concatenation of two signals. It is used for the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system. Note that convolving two signals is equivalent to multiplying the Fourier transform of the two signals.

These operations can be represented by a Mathematical Expression as follows:

x[n]= Input signal Samples

h[n-k]= Impulse response co-efficient.

y[n]= Convolution output.

n = No. of Input samples

h = No. of Impulse response co-efficient.

Example : X(n)={1 2 -1 0 1}, h(n)={ 1,2,3,-1}

Program:

```
clc;
close all;
clear all;
%program for convolution of two sequences
x=input('enter input sequence: ');
h=input('enter impulse response: ');
y=conv(x,h);
subplot(3,1,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input sequence')
subplot(3,1,2);
stem(h);
xlabel('n');
ylabel('h(n)');
```

```
title('impulse response sequence')
subplot(3,1,3);
stem(y);
xlabel('n');
ylabel('y(n)');
title('linear convolution')
disp('linear convolution y=');
disp(y)
%program for signal convolution
t=0:0.1:10;
x1=sin(2*pi*t);
h1=cos(2*pi*t);
y1=conv(x1,h1);
figure;
subplot(3,1,1);
plot(x1);
xlabel('t');
ylabel('x(t)');
title('input signal')
subplot(3,1,2);
plot(h1);
xlabel('t');
ylabel('h(t)');
title('impulse response')
subplot(3,1,3);
plot(y1);
xlabel('n');
ylabel('y(n)');
title('linear convolution');
```

Output:

```
enter input sequence: [1 3 4 5]
enter impulse response: [2 1 4]
linear convolution y=
2 7 15 26 21 20
```

VIVA VOCE--

1)The auto correlation of $x(t) = e_{-at}u(t)$ is ______

$R(\lambda) = \int_{\infty-\infty} x(t)x(t\pm\lambda)dt$

$= \int_{\infty-\infty} e_{-at}u(t)e_{-a(t-\lambda)}u(t-\lambda)dt$

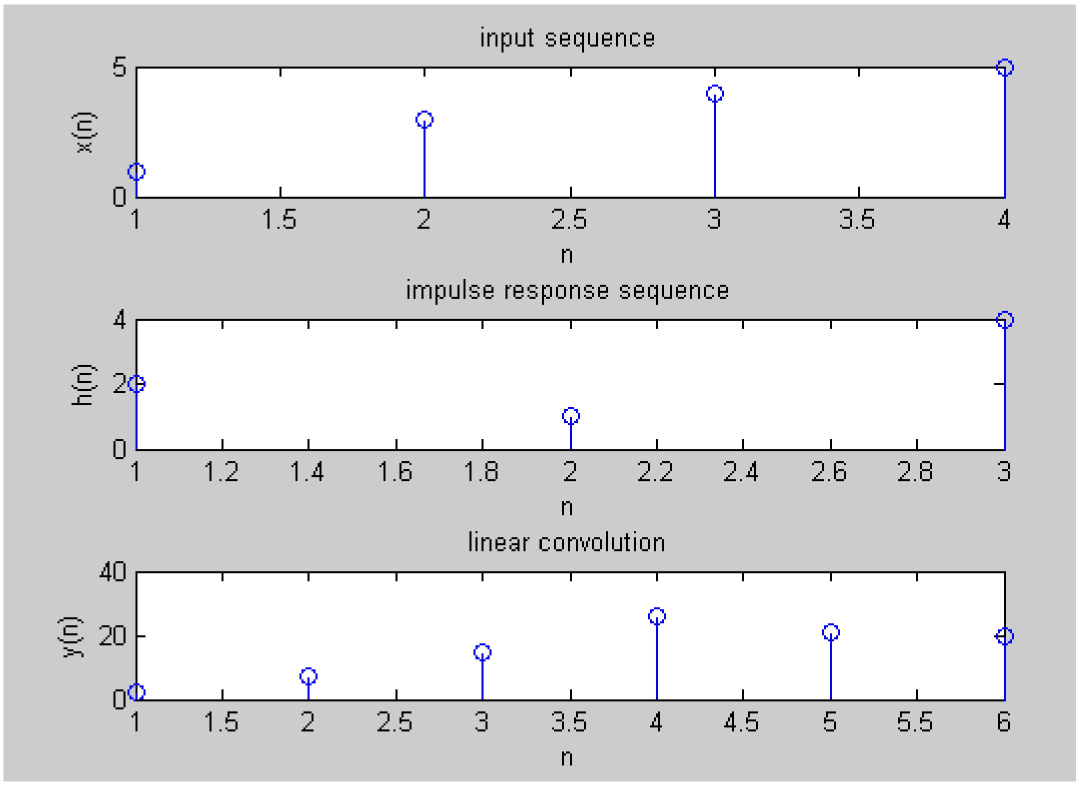
$= \int_{\infty\lambda} e_{-2at}e_{a\lambda}dt$

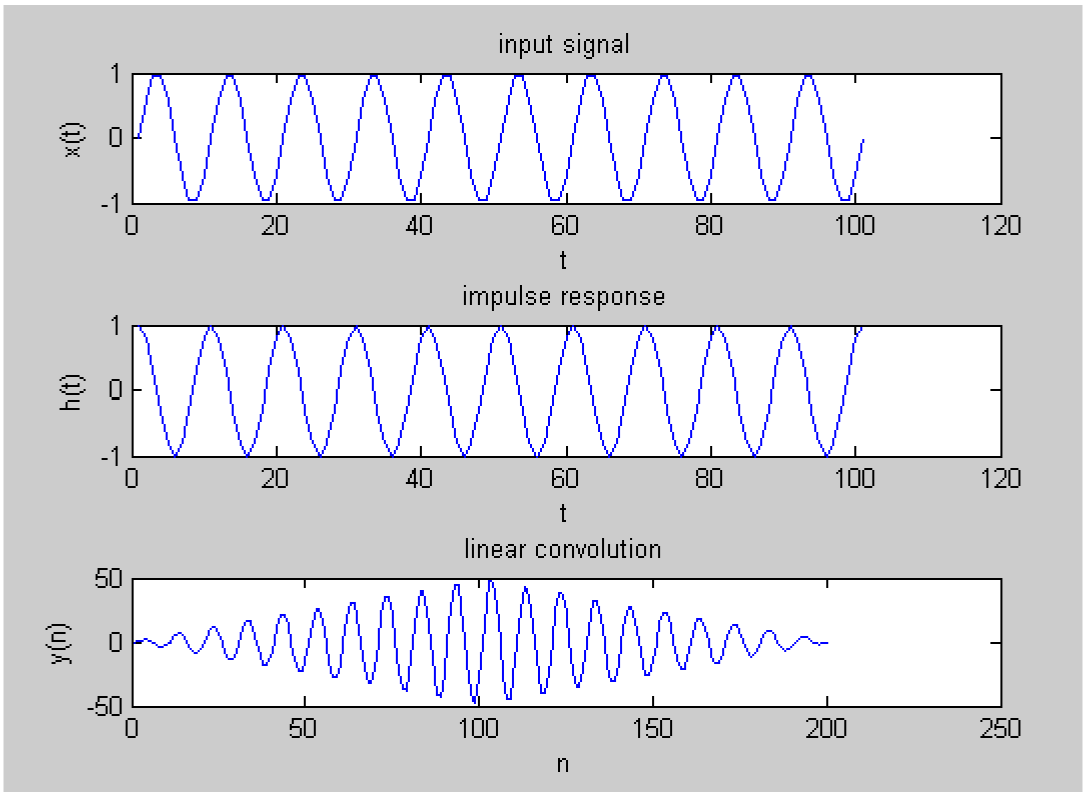
$= e_{a\lambda}-2a[0-e_{-2a\lambda}]$

$= e_{-a\lambda}2a.$

2) The resulting signal when a continuous time periodic signal x(t) having period T, is convolved with itself is ______

The solution lies with the definition of convolution. Given a periodic signal x (t) having period T. When convolution of a periodic signal with period T occurs with itself, it will give the same period T.





RESULT: convolution between signals and sequences is computed.

6. Write a program to find autocorrelation and cross correlation of given sequences.

Aim: To compute Auto correlation and Cross correlation between signals and sequences.

Software Required: Mat lab software

Theory:

Correlations of sequences:

It is a measure of the degree to which two sequences are similar. Given two real-valued sequences $x(n)$ and $y(n)$ of finite energy,

Convolution involves the following operations.

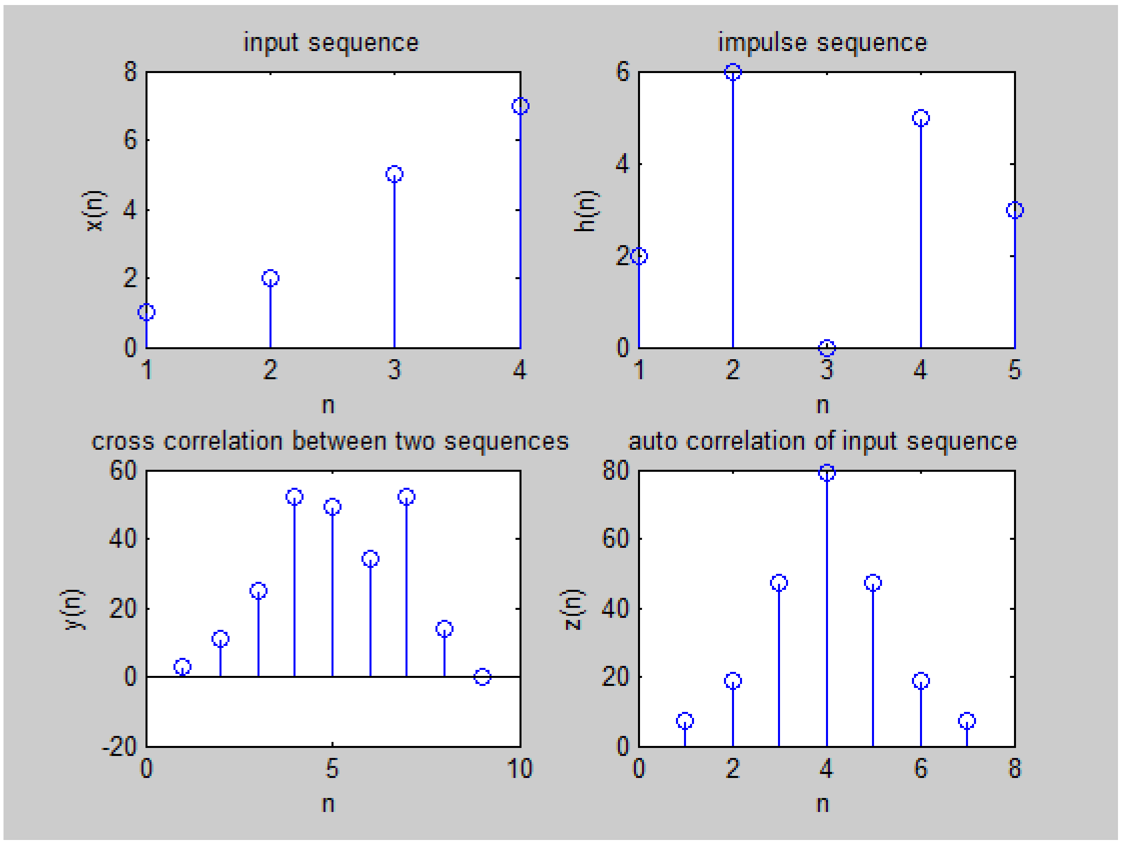
1. Shifting
2. Multiplication
3. Addition

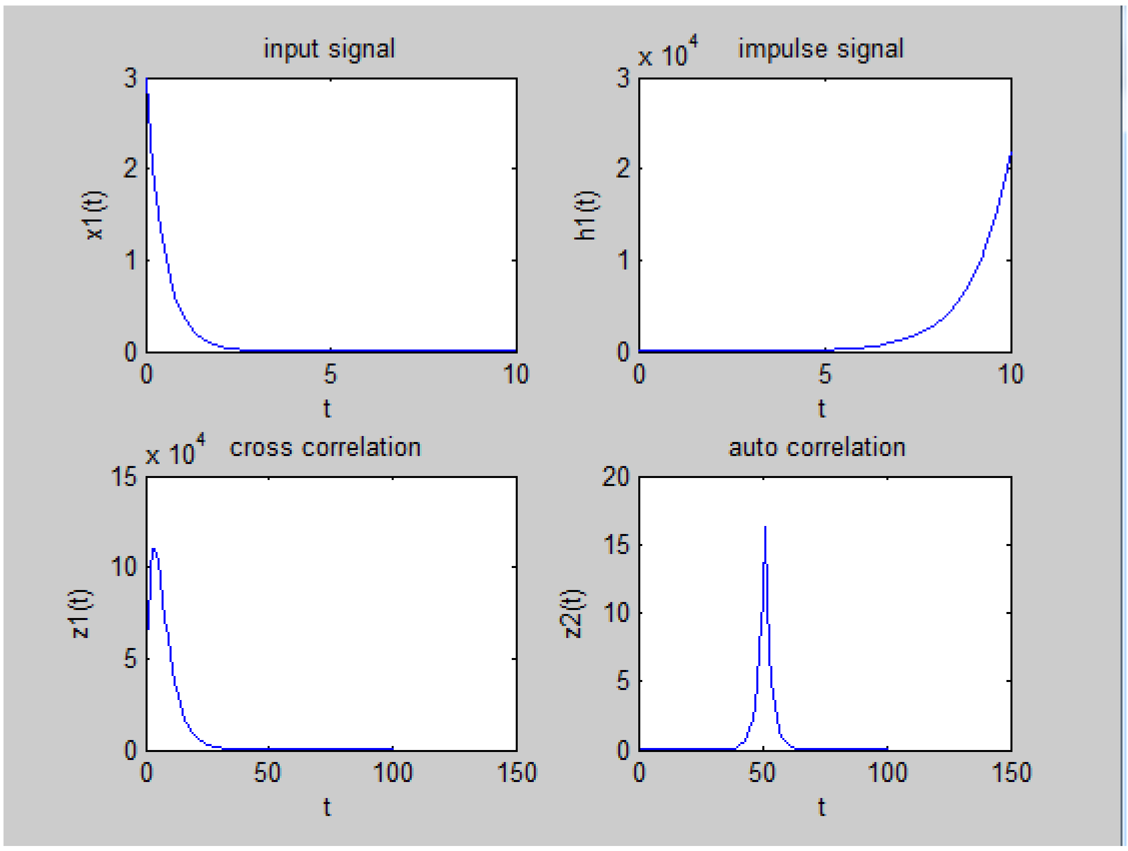
Program:

```
clc;
close all;
clear all;
% two input sequences
x=input('enter input sequence');
h=input('enter the impulse suquence');
subplot(2,2,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input sequence');
subplot(2,2,2);
stem(h);
xlabel('n');
ylabel('h(n)');
title('impulse sequence');
% cross correlation between two
y=xcorr(x,h);
subplot(2,2,3);
stem(y);
xlabel('n');
ylabel('y(n)');
title(' cross correlation between two sequences ');
% auto correlation of input sequence
z=xcorr(x,x);
subplot(2,2,4);
stem(z);
```

```
xlabel('n');
ylabel('z(n)');
title('auto correlation of input sequence');
% cross correlation between two signals
% generating two input signals
t=0:0.2:10;
x1=3*exp(-2*t);
h1=exp(t);
figure;
subplot(2,2,1);
plot(t,x1);
xlabel('t');
ylabel('x1(t)');
title('input signal');
subplot(2,2,2);
plot(t,h1);
xlabel('t');
ylabel('h1(t)');
title('impulse signal');
% cross correlation
subplot(2,2,3);
z1=xcorr(x1,h1);
plot(z1);
xlabel('t');
ylabel('z1(t)');
title('cross correlation ');
% auto correlation
subplot(2,2,4);
z2=xcorr(x1,x1);
plot(z2);
xlabel('t');
ylabel('z2(t)');
title('auto correlation ');
```

Output: enter input sequence [1 2 5 7]
enter the impulse sequence [2 6 0 5 3]





Result: Auto correlation and Cross correlation between signals and sequences is computed.

7.Write a program to verify Linearity and Time Invariance properties of a given Continuous/Discrete System.

Aim: Verify the Linearity of a given Discrete System.

Software Required:Mat lab software 7.0 and above

Theory:

LINEARITY PROPERTY: Any system is said to be linear if it satisfies the superposition principal. Superpositionprincipal state that Response to a weighted sum of input signal equal to the correspondingweighted sum of the outputs of the system to each of the individual input signals.If x(n) is a input signal and y(n) is a output signal theny(n)=T[x(n)]

y1(n)=T[x1(n)] and y2(n)=T[x2(n)]

x3=[a*x1(n) +b *x2(n)]

Y3(n)= T [x3(n)]

T [a*x1(n)+b*x2(n)] = a y1(n)+ b y2(n)

Program (A) :

```
% Verification of Linearity of a given System.
% a) y(n)=nx(n) b) y=x^2(n)
clc;
clear all;
close all;
n=0:40;
a1=input('enter the scaling factor a1=');
a2=input('enter the scaling factor a2=');
x1=cos(2*pi*0.1*n);
x2=cos(2*pi*0.4*n);
x3=a1*x1+a2*x2;
%y(n)=n.x(n);
y1=n.*x1;
y2=n.*x2;
y3=n.*x3;
yt=a1*y1+a2*y2;
yt=round(yt);
y3=round(y3);
if y3==yt
disp('given system [y(n)=n.x(n)]is Linear');
else
disp('given system [y(n)=n.x(n)]is non Linear');
end
%y(n)=x(n).^2
x1=[1 2 3 4 5];
x2=[1 4 7 6 4];
x3=a1*x1+a2*x2;
y1=x1.^2;
y2=x2.^2;
y3=x3.^2;
yt=a1*y1+a2*y2;
if y3==yt
disp('given system [y(n)=x(n).^2 ]is Linear');
else
disp('given system is [y(n)=x(n).^2 ]non Linear');
end
```

Result: The Linearity of a given Discrete System is verified.

Output:

```
enter the scaling factor a1=3
enter the scaling factor a2=5
given system [y(n)=n.x(n)]is Linear
given system is [y(n)=x(n).^2 ]non Linear
```

Program (B) :

```
% Verification of Time Invariance of a Discrete System
% a)y=x^2(n) b) y(n)=nx(n)
clc;
clear all;
close all;
n=1:9;
x(n)=[1 2 3 4 5 6 7 8 9];
d=3; % time delay
xd=[zeros(1,d),x(n)];%x(n-k)
y(n)=x(n).^2;
yd=[zeros(1,d),y];%y(n-k)
disp('transformation of delay signal yd:');disp(yd)
dy=xd.^2; % T[x(n-k)]
disp('delay of transformation signal dy:');disp(dy)
if dy==yd
disp('given system [y(n)=x(n).^2 ]is time invariant');
else
disp('given system is [y(n)=x(n).^2 ]not time invariant');
end
y=n.*x;
yd=[zeros(1,d),y(n)];
disp('transformation of delay signal yd:');disp(yd);
n1=1:length(xd);
dy=n1.*xd;
disp('delay of transformation signal dy:');disp(dy);
if yd==dy
disp('given system [y(n)=nx(n)]is a time invariant');
else
disp('given system [y(n)=nx(n)]not a time invariant');
end
```

Output:

```
transformation of delay signal yd:

0 0 0 1 4 9 16 25 36 49 64 81

delay of transformation signal dy:

0 0 0 1 4 9 16 25 36 49 64 81

given system [y(n)=x(n).^2 ]is time invariant

transformation of delay signal yd:

0 0 0 1 4 9 16 25 36 49 64 81

delay of transformation signal dy:

0 0 0 4 10 18 28 40 54 70 88 108

given system [y(n)=nx(n)]not a time invariant
```

Result: The Time Invariance of a given Discrete System is verified.

8. Write a program to generate discrete time sequence by sampling a continuous time signal. Show that with sampling rates less than Nyquist rate, aliasing occurs while reconstructing the signal.

Aim: Verify the sampling theorem.

Software Required: Matlab software

Theory:

Sampling Theorem:

\A bandlimited signal can be reconstructed exactly if it is sampled at a rate atleast twice the maximum frequency component in it."

The maximum frequency component of g(t) is fm. To recover the signal g(t) exactly from its samples it has to be sampled at a rate fs ≥ 2fm.

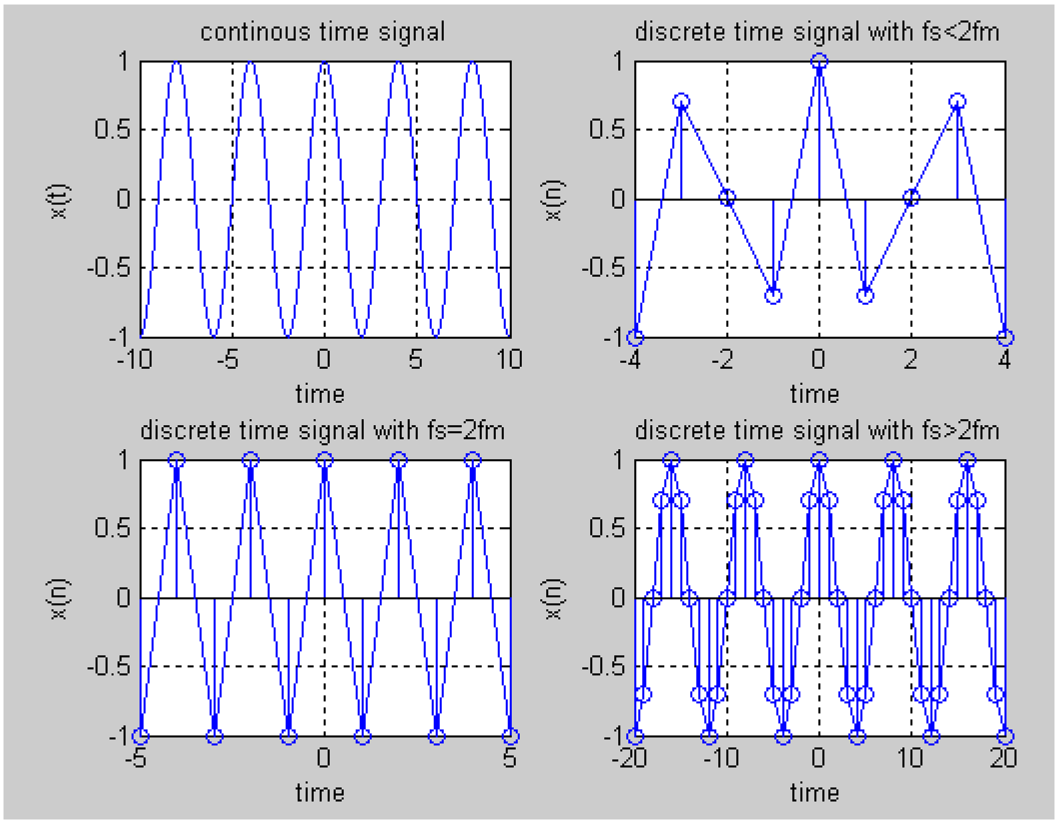
The minimum required sampling rate fs = 2fm is called ' Nyquist rate

Program:

```
clc;
clear all;
close all;
t=-10:.01:10;
T=4;
fm=1/T;
x=cos(2*pi*fm*t);
subplot(2,2,1);
plot(t,x);
xlabel('time');
ylabel('x(t)');
title('continous time signal');
grid;
n1=-4:1:4;
fs1=1.6*fm;
fs2=2*fm;
fs3=8*fm;
x1=cos(2*pi*fm/fs1*n1);
subplot(2,2,2);
stem(n1,x1);
xlabel('time');
ylabel('x(n)');
title('discrete time signal with fs<2fm');
hold on;
subplot(2,2,2);
plot(n1,x1);
grid;
n2=-5:1:5;
x2=cos(2*pi*fm/fs2*n2);
subplot(2,2,3);
stem(n2,x2);
xlabel('time');
ylabel('x(n)');
title('discrete time signal with fs=2fm');
hold on;
subplot(2,2,3);
```

```
plot(n2,x2)
grid;
n3=-20:1:20;
x3=cos(2*pi*fm/fs3*n3);
subplot(2,2,4);
stem(n3,x3);
xlabel('time');
ylabel('x(n)');
title('discrete time signal with fs>2fm')
hold on;
subplot(2,2,4);
plot(n3,x3)
grid;
```

OUTPUT



Result: Sampling theorem is verified.

9. Write a program to find magnitude and phase response of first order low pass and high pass filter. Plot the responses in logarithmic scale.

Aim: To find magnitude and phase response of first order low pass and high pass filter. Plot the responses in logarithmic scale.

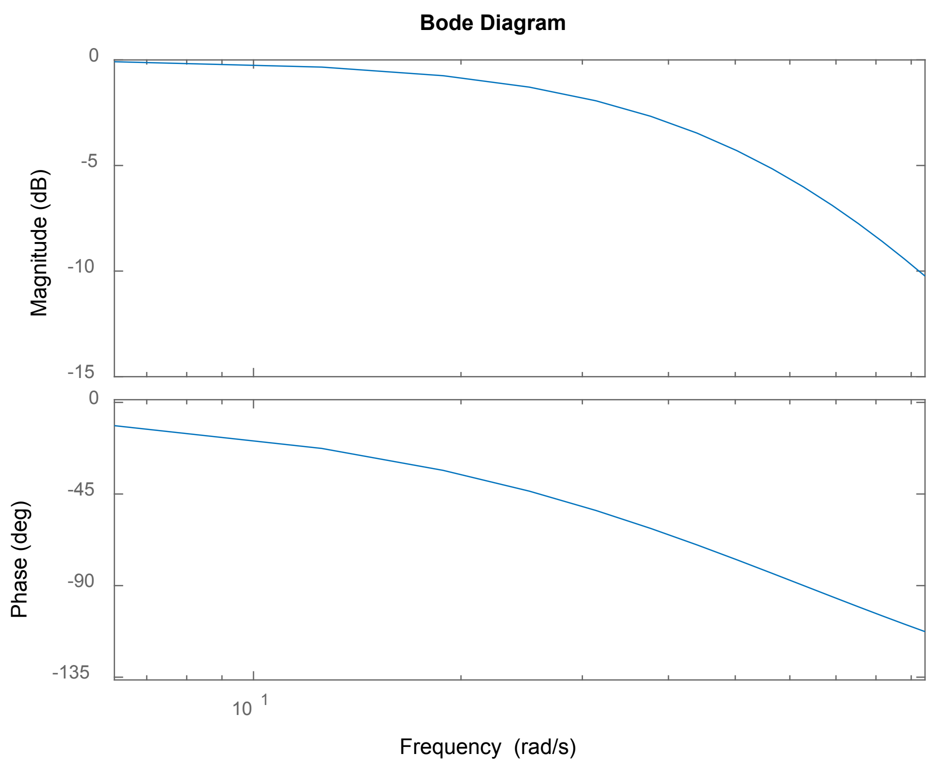
Theory: To get the phase response in the time domain you need to estimate the delay between the input and the output at the test frequency and then convert to phase. For sure, taking the angle of the RMS ratio will not yield that delay. If you really want to estimate the delay in the time domain, there is a function called finddelay that may be of use.

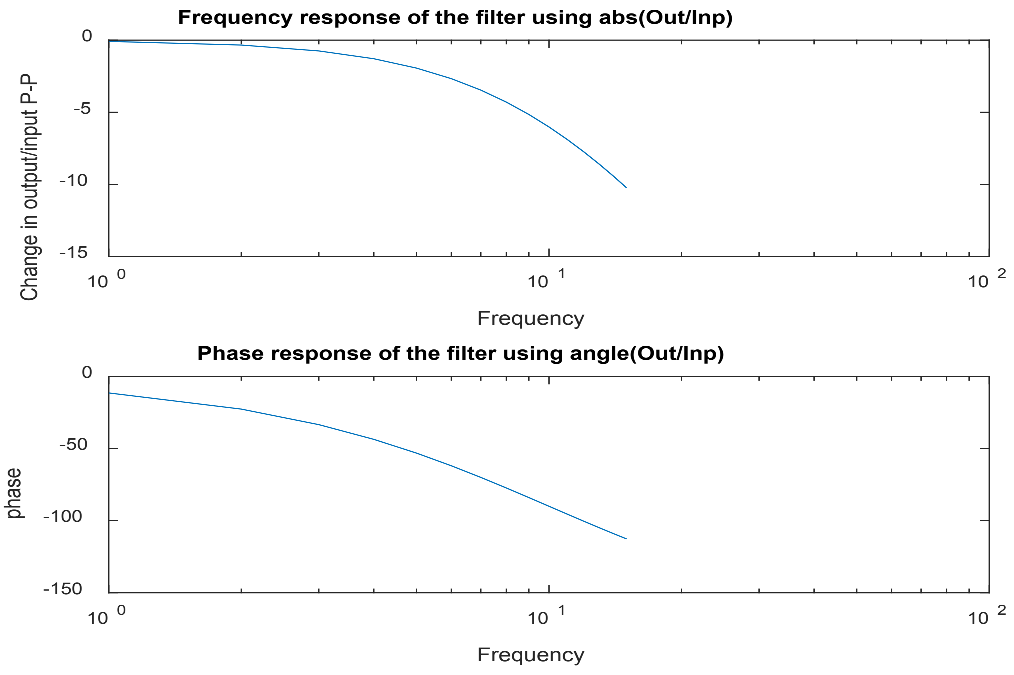
However, the alternative approach is to work in the frequncy domain, i.e., use the ratio of the frequency response of the output to the frequency resposne of the input to directly stimate the magnitude and phase response of the system. The code that follows shows how to do that for a simple low pass filter. Keep in mind, that any approach you use may begin to suffere as you test frequency gets close to the Nyquist frequency. You can experiment with this code to see how close you can get before this simple approach begins to break down.

Program :

```
f=1:15;
% second order filter with 10 Hz pass band
H   = @(f,s) 1./(1./(2*pi*f).^2*s.^2 + 2./(2*pi*f)*s + 1);
H10 = @(s) H(10,s);
% magnitude estimation
magn = abs(H10(1j*2*pi*f));
% phase estimation
phase = rad2deg( angle(H10(1j*2*pi*f)) );
figure;
subplot(2,1,1);
semilogx(f,mag2db(magn));
title('Frequency response of the filter using abs(Out/Inp)')
xlabel('Frequency');
ylabel ('Change in output/input P-P' );
subplot(2,1,2);
semilogx(f,phase);
title('Phase response of the filter using angle(Out/Inp)')
xlabel('Frequency');
ylabel ('phase' );
% check against inbuilt functions
H_tf = @(f) tf(1,[1./(2*pi*f).^2  2./(2*pi*f) 1]);
figure;
bodeplot(H_tf(10),2*pi*f)
```

OUTPUT--





Result: The magnitude and phase response of LPF plotted.

10. Write a program to find response of a low pass filter and high pass filter, when a speech signal is passed through these filters.

Aim: To find response of a low pass filter and high pass filter, when a speech signal is passed through these filters.

Theory:

`y = lowpass(x,wpass)` filters the input signal x using a lowpass filter with normalized passband frequency `wpass` in units of π rad/sample. `lowpass` uses a minimum-order filter with a stopband attenuation of 60 dB and compensates for the delay introduced by the filter. If x is a matrix, the function filters each column independently.

`y = lowpass(x,fpass,fs)` specifies that x has been sampled at a rate of `fs` hertz. `fpass` is the passband frequency of the filter in hertz.

`y = lowpass(xt,fpass)` lowpass-filters the data in timetable `xt` using a filter with a passband frequency of `fpass` hertz. The function independently filters all variables in the timetable and all columns inside each variable.

`y = lowpass(__,Name,Value)` specifies additional options for any of the previous syntaxes using name-value pair arguments. You can change the stopband attenuation, the transition band steepness, and the type of impulse response of the filter.

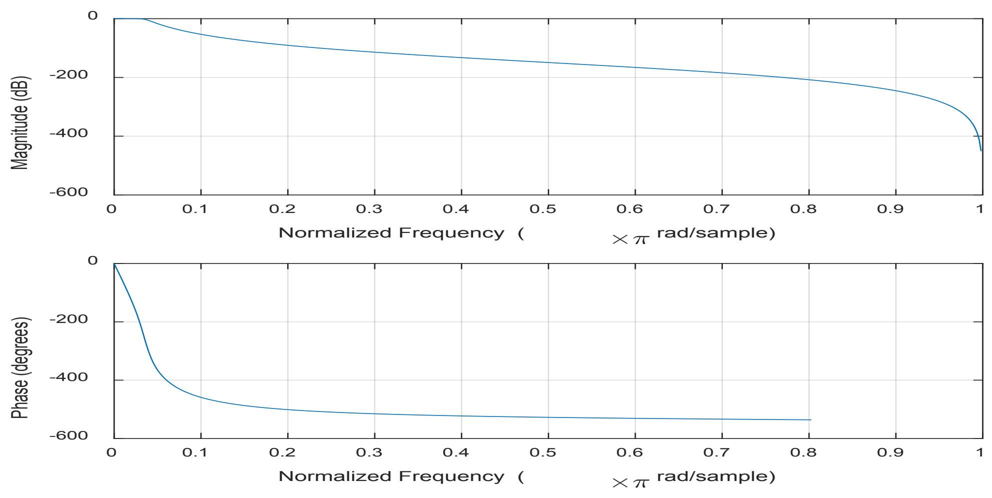
`[y,d] = lowpass(__)` also returns the `digitalFilter` object `d` used to filter the input.

`lowpass(__)` with no output arguments plots the input signal and overlays the filtered signal.

Program : `% Read standard sample tune that ships with MATLAB.`

```
[dataIn, Fs] = audioread('guitartune.wav');
% Filter the signal
fc = 800; % Make higher to hear higher frequencies.
% Design a Butterworth filter.
[b, a] = butter(6,fc/(Fs/2));
freqz(b,a)
% Apply the Butterworth filter.
filteredSignal = filter(b, a, dataIn);
% Play the sound.
player = audioplayer(filteredSignal, Fs);
play(player);
```

OUTPUT--



Result: Response of a low pass filter and high pass filter, when a speech signal studied.

11 .Write a program to generate Complex Gaussian noise and find its mean, variance, Probability Density Function (PDF) and Power Spectral Density.

Aim: Write the program for generation of Gaussian noise and computation of its mean, mean square value, standard deviation, variance, and skewness.

Software Required: Matlab software

Theory:

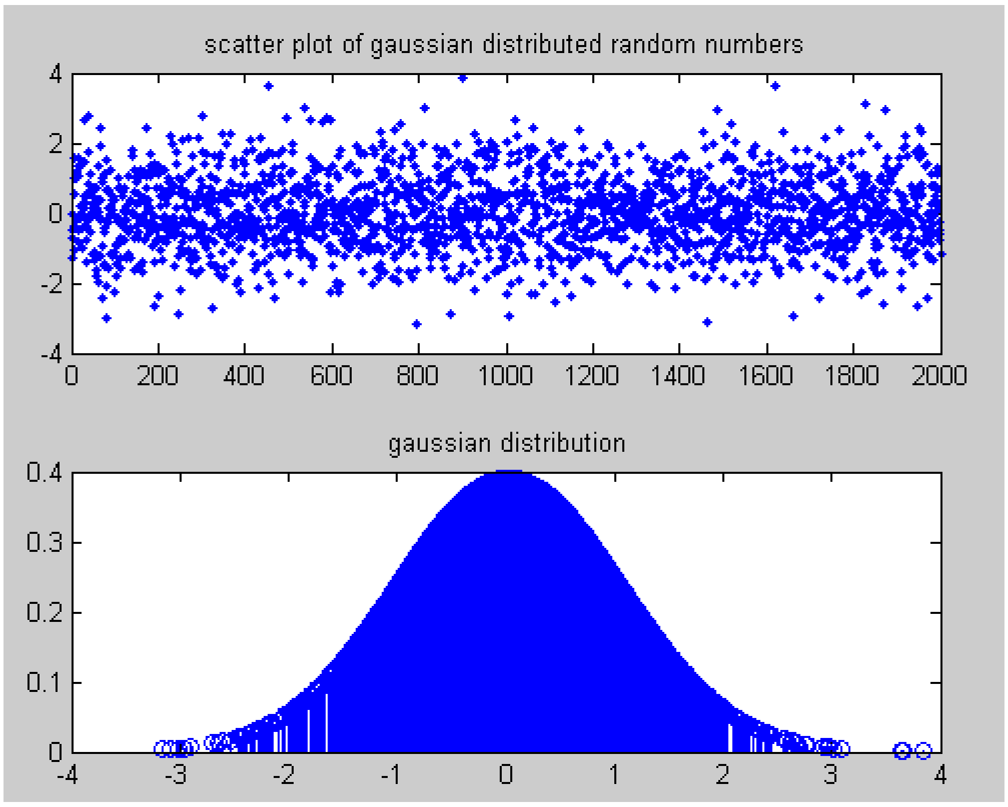
Gaussian noise is statistical noise that has a probability density function (abbreviated pdf) of the normal distribution (also known as Gaussian distribution). In other words, the valuestha the noise can take on are Gaussian-distributed. It is most commonly used as additive white noise to yield additive white Gaussian noise (AWGN).Gaussian noise is properly defined as the noise with a Gaussian amplitude distribution. says nothing of the correlation of the noise in time or of the spectral density of the noise. Labeling Gaussian noise as 'white' describes the correlation of the noise. It is necessary to use the term "white Gaussian noise" to be correct. Gaussian noise is sometimes misunderstood to be white Gaussian noise, but this is not the case.

Program:

```
clc;
clear all;
close all;
%generates a set of 2000 samples of Gaussian distributed random numbers
x=randn(1,2000);
%plot the joint distribution of both the sets using dot.
subplot(211)
plot(x,'.');
title('scatter plot of gaussian distributed random numbers');
ymu=mean(x)
ymsq=sum(x.^2)/length(x)
ysigma=std(x)
yvar=var(x)
yskew=skewness(x)
p=normpdf(x,ymu,ysigma);
subplot(212);
stem(x,p);
title(' gaussian distribution');
```

Output:

```
ymu = 0.0403
ymsq = 0.9727
ysigma = 0.9859
yvar = 0.9720
yskew = 0.0049
```



Result: Gaussian noise and its characteristics are studied.

12. Generate a Random data (with bipolar) for a given data rate (say 10kbps). Plot the same for a time period of 0.2 sec.

Aim: To Generate a Random data (with bipolar) for a given data rate (say 10kbps). Plot the same for a time period of 0.2 sec.

Theory:

X = rand returns a single uniformly distributed random number in the interval (0,1).

X = rand(n) returns an n-by-n matrix of random numbers.

X = rand(sz1,...,szN) returns an sz1-by-...-by-szN array of random numbers where sz1,...,szN indicate the size of each dimension. For example, rand(3,4) returns a 3-by-4 matrix.

X = rand(sz) returns an array of random numbers where size vector sz specifies size(X). For example, rand([3 4]) returns a 3-by-4 matrix.

X = rand(___,typename) returns an array of random numbers of data type typename. The typename input can be either 'single' or 'double'. You can use any of the input arguments in the previous syntaxes.

Program :

```
clc;
clear all;
a=-3
b=3
x=rand
c=a+(b-a)*x
y=c^2
z=y
for i=1:1000
    x1=rand
    c1=a+(b-a)*x1
    y1=c1^2

if y1<z
        z=y1
else
        z;
end
end
z
```

Result: Random data generated and studied.

13. To plot pole-zero diagram in S-plane/Z-plane of given signal/sequence and verify its stability

Aim: Write the program for locating poles and zeros and plotting pole-zero maps in s-plane and z-plane for the given transfer function.

Software Required: Matlab software

Theory:

Z-transforms

The Z-transform, like many other integral transforms, can be defined as either a *one-sided* or *two-sided* transform.

transform is defined as

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=0}^{\infty} x[n] z^{-n}$$

In signal processing, this definition is used when the signal is causal.

where $z = r.e^{j\omega}$

$$X(z) = \frac{P(z)}{Q(z)}$$

The roots of the equation P(z) = 0 correspond to the 'zeros' of X(z)
The roots of the equation Q(z) = 0 correspond to the 'poles' of X(z)

Example:

$$H(z) = \frac{z+1}{(z-\frac{1}{2})(z+\frac{3}{4})}$$

The zeros are: $\{-1\}$

The poles are: $\left\{ \frac{1}{2}, -\left(\frac{3}{4}\right) \right\}$

Program

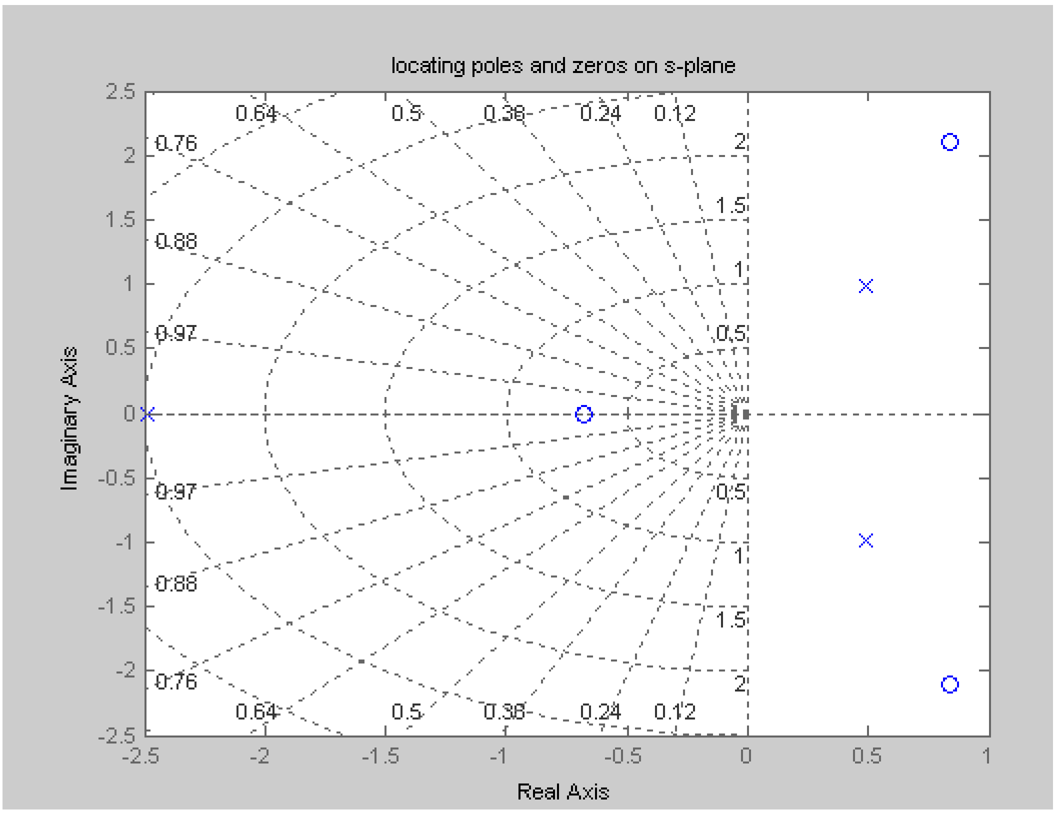
```
clc;
clear all;
close all;
%enter the numerator and denamenator cofficients in square brackets
num=input('enter numerator co-efficients');

den=input('enter denominator co-efficients');
% find poles and zeros
poles=roots(den)
```

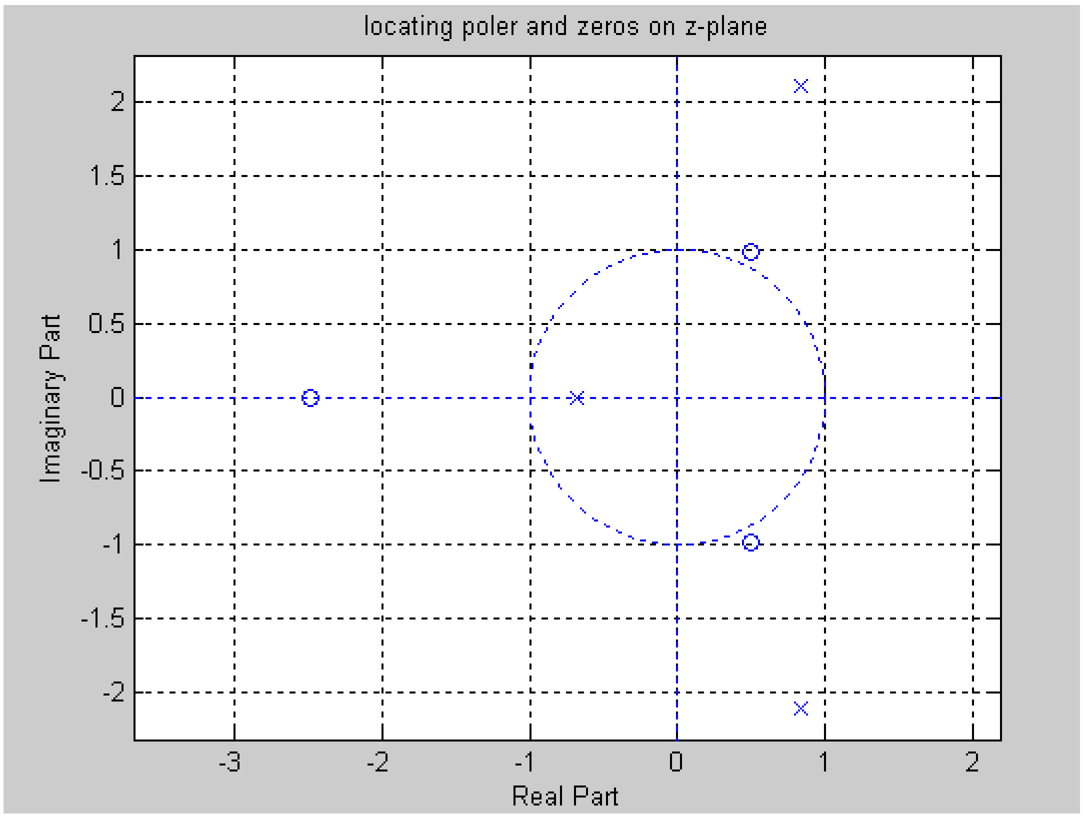
```
zeros=roots(num)
% find transfer function H(s)
h=tf(num,den);
% plot the pole-zero map in s-plane
sgrid;
pzmap(h);
grid on;
title('locating poles and zeros on s-plane');
%plot the pole zero map in z-plane
figure
zplane(poles,zeros);
grid on;
title('locating poler and zeros on z-plane');
```

Result: Pole-zero maps are plotted in s-plane and z-plane for the given transfer function.

Output:

```
enter numerator co-efficients[1 -1 4 3.5]
enter denominator co-efficients[2 3 -2.5 6]
poles =
-2.4874
0.4937 + 0.9810i
0.4937 - 0.9810i
zeros =
0.8402 + 2.1065i
0.8402 - 2.1065i
-0.6805
```





Result: Stability test verified for given pole zero plot.