SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

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

Department of Electronics and Communication Engineering

(15A02307) BASIC SIMULATION LAB -- R15

II B. Tech (ECE) I Semester 2018-2019

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

CERTIFICATE

ACADEMIC YEAR: 2018-2019

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 SIMULATION LAB.

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

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPURAMU.

B.Tech –II-I Sem (15A02307) BASIC SIMULATION LAB

LIST OF EXPERIMENTS Branch: ECE Regulation:R15

1	Basic operations on Matrices.
2	Generation of various signals and sequences (Periodic and aperiodic). Such as unit impulse, unit step, square, saw tooth, triangular, sinusoidal, ramp, sinc function.
3	Operation on signals and sequences such as addition, multiplication, scaling, shifting, folding, computation of energy and average power.
4	Convolution between signals and sequences.
5	Auto correlation and cross correlation between signals and sequences.
6	Verification of linearity and time invariance properties of a given continuous/discrete system.
7	Finding the Fourier transform of a given signal and plotting its magnitude and phase spectrum.
8	Waveform synthesis using Laplace Transform.
9	Generation of Gaussian noise (real and complex), computation of its mean, mean square values and its skew, kurtosis and PSD, probability distribution function.
10	Sampling theorem verification.
11	Removal of noise by auto correlation/cross correlation in a given signal corrupted by noise.
12	Impulse response of a raised cosine filter.
13	Checking a Random process for stationary in wide sense.

ECE DEPT VISION & MISSION PEOs and PSOs

Vision

To produce highly skilled, creative and competitive Electronics and Communication Engineers to meet the emergingneeds 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 problemsin the field of Electronics and develop sophisticated communication systems.

PEO2: Graduates embody a commitment to professional ethics, diversity and social awareness in theirprofessional 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 anengineering 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, andin 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 andin 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 introduce MATLAB and use it as a computation and visualization tool in the study of Signals & Systems.
- Students will then be exposed to the applications of MATLAB to signal analysis and system design.
- Synthesize Laplace transform and able to locate poles and zeros of a system.
- Compute various statistical properties of a random noise and verify whether it is stationary.
- Apply convolution and correlation operations on different signals.

V. COURSE OUTCOMES:

After the completion of the course students will be able to

Course Outcomes	Course Outcome statements	BTL
CO1	To study about signals and systems	L1
CO2	To do analysis of signals & systems (continuous and discrete) using time domain & frequency domain methods	L2
CO3	Students able to learn Various signals and Sequences (Periodic and Aperiodic.	L3
CO4	Students able to learn Various Operations on Signals,Auto correlation and Cross correlation	L4
CO5	Students able to learn Fourier Transform,Laplace Transform,Sampling Theorem ,Removal of Noise by Auto Correlation	L5

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

CourseTitle	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
BASIC SIMULATION LAB	3	3	2	3	3	2	2		2	2		2	3	2

VII. MAPPING OF COURSE OUTCOMES WITH PEO'S AND PO'S:

Course Title	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1	2	3	1	2	3	2	2	1	3	2	3	3	3	3
CO2	1	1	3	2	2	1		2	1	1	3	2	2	3
CO3	2	3	2	2	2	3	2	1	2	1	1	2	2	2
CO4	3	2		2		1		1		2		2	3	2
CO5	2		2	3	3	2	1	2	1	1	2			2

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 graphsheets if any for the lab classes without which the students will not be allowed for doing the experiment.

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

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 bookand 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.

I N D E X

Objective of Laboratory

The main objective of this lab is to learn MATLAB and know why it is an indispensable tool, especially for electronics and communication engineer.

Evaluation Procedure for Internal Laboratory Examinations

- **Day-to-day evaluation:**

The concerned teachers have to do necessary corrections with explanations and evaluate each lab experiment.

Concerned Lab Incharge should also enter the marks in index page of the record and observation book & also at the end of each experiment with signature.

- **Internal Laboratory examination:**

30 marks will be awarded for internal Lab exam, the division of the marks as given below:

1. Matlab program & Execution	: 12Marks
2. Observations and Graphs	: 6 Marks
3. Result	: 6 Marks
4. Viva voce	: 6 Marks

Internal lab exam will be conducted by the in-charge Faculty member along with Associate Faculty members

Evaluation Procedure for External Laboratory Examinations

This examination will be conducted by the teacher in-charge of the lab and another two faculty members of the same department (who have more knowledge in the concern lab), recommended by Head of the Department with the approval of Principal.The maximum marks for this examination is 70.

The distribution of marks for the evaluation is as follows.

1. Matlab program	: 25 Marks
2. Execution	: 10 Marks
3. Observations and Graphs	: 10 Marks
4. Result	: 15 Marks
5. Viva voce	: 10 Marks

1. MATLAB INTRODUCTION: BASIC OPERATIONS ON MATRICES

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.

Matrices:

1. Row matrix: Elements in a row are separated either by using white spaces or

Commas e.g.: a= [1 2 4 5]

2. Column matrix: Elements which differ by a column are separated by enter or

Semicolon e.g.: b= [1; 2; 3]

3. Square matrix: e.g.: c = [a, b; c, d] is a 2*2 matrix

Looking into matrix:

a(row, column) allows to look the particular element in the matrix “a”

Vectors:

d= [0:7]

”d” is a vector or row matrix with first element as 0 and last element as 7 and increment is by

Default 1.

The default increment can be changed (to 0.1) by using increment field in between as

e= [0:0.1:7].

d(1:2) allows to look into vector with increment 1

e(1:2:4) look with increment

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

Operations on matrices

a= [1 2 3];

b=a’; % Give the transpose of a

>> a = [1 2 3; 4 5 6; 7 8 9]

a (2 , 3) % Gives 6

a (3 , 2) % Gives 8

b = eye (3) % Creates 3 x 3 identity matrix

c = rand (4, 6) % Generates a 4 x 6 matrix of random numbers between 0 and 1

d = ones (4, 3) % Creates 4 x 3 matrix elements which are all ones

e = zeros (3, 6) % Creates 3 x 6 matrix of elements which are all zeros

[M, N]=size (e) % Gives M = 3, N= 6

f= magic (3) %Generates a 3 x3 magic square matrix

sum (f) % gives 15 15 15 by calculating the sum of elements in each column

g=fliplr (f) % Turn the matrix from left to right to convert anti diagonal to principal diagonal

Sub matrices

a = rand (4, 5) % Gives 4 x5 matrix of random elements

a(: , 2 : 4) %Extract the 2 nd , 3 rd , 4 th columns of matrix a

a (3 : 4 , :) % Extract 3 rd and 4 th rows and all columns of the matrix a

a (2 : 3 , 2: 4) % Takes out the elements in the 2 nd to 3 rd row and 2 nd to 4 th columns

a([1 4] , [2 5]) %Extract the elements in the 1 st and 4 th row of the 2 nd and 5 th columns

a(end, :) %Gives last row or column

a(: , 3) = [ ] % To delete the 3 rd column of matrix a

Determinant and Inverse of a matrix

A = [9, 7, 0 ; 0, 8, 6 ; 7, 1, -6] size (A), det(A), inv(A)

We can check our result by verifying that AA–1 = I and A–1A = I. , A*inv(A), inv(A)*A

Multiplication of two vectors

P= [4; 5; 6;] % P is a column vector of 3 elements

Q = [1 2 3] % Q is a row vector of 3 elements

R= P*Q %Result is assigned to R which will be a 3x3 matrix

Arithmetic operators

y = sqrt (17) a= 3+4*j;

b=abs (a) % Gives absolute value of a as 5

p =2; q =7;

r = complex (p , q) % Gives complex variable as r = 2.000 + 7.000 i

Row vector of arbitrary elements

a = [2 7 0 1 5]

a (3) % Gives a as 0

a(2) % Gives a as 7

a(6) = 3 % Gives a as 2 7 0 1 5 3

a(8) = 4 % Gives a as 2 7 0 1 5 3 0 4

c= [12 15]; d = [a c] % Gives d as 2 7 0 1 5 3 0 4 12 15

Note: similarly do the column vector of arbitrary elements by using transpose sign.

Row vector of equally spaced elements

b = 3 : 2 : 11 % Gives b as 3 5 7 9 11

c = 0 : 10 % Gives c as 0 1 2 3 4 5 6 7 8 9 10

n= 0.1 : 0.01 : 0.5;

t= 0.5 : -0.1 : 0.2; % Gives t as 0.5000 0.4000 0.3000 0.2000

Operations on vectors

a= [5 2 1 4 3] b= [8 6 7]

min(a) % Gives 1

max (b) % Gives 8

length (b) % Gives 3

sum (b) % Gives 21

prod (a) % Gives 120

mean (a) % Gives 3

std (a) % Gives 1.5811

median (b) % Gives 7

Sorting operation

a= [5 2 1 4 3];

sort (a , 'descend') % Gives 5 4 3 2 1

sort (a , 'ascending') % Gives 1 2 3 4 5

Addition operation

a= [5 2 1 4 3]; b= [8 6 7];

d = a + b % Gives d = 13 8 8 4 3

Array power operation

a = 1: 7; b = a. ^2 % Gives b= 1 4 9 16 25 36 49

Array multiplication operation

a= [5 2 1 4 3] d = [13 8 8 4 3] a.* d % Gives 65 16 8 16 9

RESULT:

MATLAB basic features and built in functions were practiced thoroughly.

2. GENERATION OF VARIOUS SIGNALS AND SEQUENCES

AIM:

To generate various periodic and aperiodic signals and sequences such as Unit Impulse, Unit step, Square, Saw Tooth, Triangular, Sinusoidal, Ramp, Sinc functions using MATLAB.

THEORY:

If x and y are two vectors of the same length then plot(x,y) plots x versus y. For example, to obtain the graph of y = cos(x) from – .. to .., we can first define the vector x with components equally spaced numbers between – .. and .., with increment, say 0.01.

» x=-pi:0.01:pi;

We placed a semicolon at the end of the input line to avoid seeing the (long) output. Note that the smallest the increment, the “smoother” the curve will be. Next, we define the vector y

» y=cos(x);

(using a semicolon again) and we ask for the plot

» plot(x,y)

It is good practice to label the axis on a graph and if applicable indicate what each axis represents. 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')

Various line types, plot symbols and colors can be used. If these are not specified (as in the case above) MATLAB will assign (and cycle through) the default ones as given in the table below.

y yellow . point

m magenta o circle

c cyan x x-mark

r red + plus

g green - solid

b blue * star

w white : dotted

k black -. dash dot

-- dashed

So, to obtain the same graph but in green, we type

» 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:’)

Multiple curves can appear on the same graph. If for example we define another vector

» z = sin(x);

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

Common Sequences: Unit Impulse, Unit Step, and Unit Ramp

Common Periodic Waveforms

The toolbox provides functions for generating widely used periodic waveforms: saw tooth generates a saw tooth wave with peaks at ±1 and a period of 2pi. An optional width parameter specifies a fractional multiple of 2pi at which the signal's maximum occurs. Square generates a square wave with a period of 2pi. An optional parameter specifies duty cycle, the percent of the period for which the signal is positive.

Common Aperiodic Waveforms

The toolbox also provides functions for generating several widely used aperiodic waveforms:

gauspuls generates a Gaussian-modulated sinusoidal pulse with a specified time, center frequency, and fractional bandwidth. Optional parameters return in-phase and quadrature pulses, the RF signal envelope, and the cutoff time for the trailing pulse envelope. chirp generates a linear, log, or quadratic swept-frequency cosine signal. An optional parameter specifies alternative sweep methods. An optional parameter phi allows initial phase to be specified in degrees.

The pulstran Function

The pulstran function generates pulse trains from either continuous or sampled prototype pulses. The following example generates a pulse train consisting of the sum of multiple delayed interpolations of a Gaussian pulse.

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.

The Dirichlet Function

The toolbox function diric computes the Dirichlet function, sometimes called the periodic sinc or aliased sinc function, for an input vector or matrix x. The Dirichlet function is where N is a user-specified positive integer. For N odd, the Dirichlet function has a period of 2p; for N even, its

period is 4p. The magnitude of this function is (1/N) times the magnitude of the discrete-time Fourier transforms of the N-point rectangular window.

Different continuous time signals and sequences

t = (0:0.001:1)';

% 1001-element row vector that represents time running from 0 to 1 s in steps of 1 ms.

imp= [1; zeros(99,1)]; % Impulse

unit_step = ones (100, 1); % Step (with 0 initial cond.)

ramp_sig= t; % Ramp

quad_sig=t. ^2; % Quadratic

sq_wave = square (4*pi*t); % Square wave with period 0.5

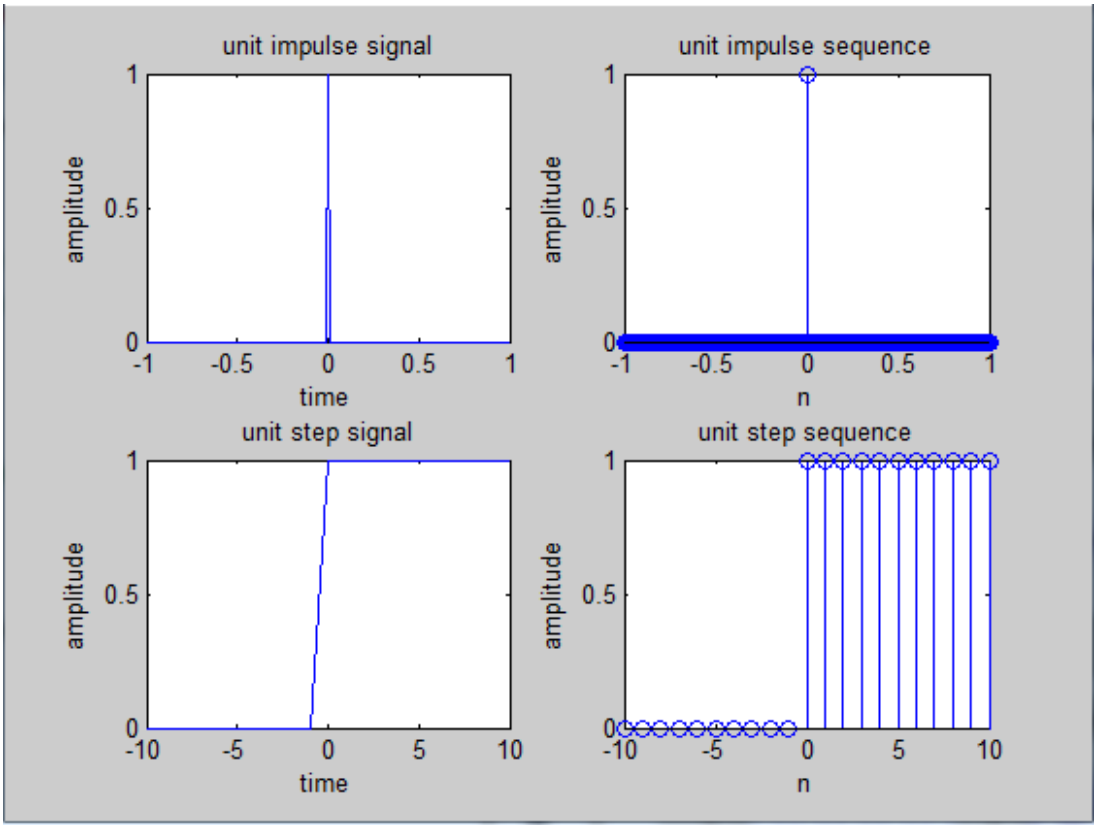
```
% Generation of signals and sequences
clc;
clear all;
close all;
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%generation of unit impulse signal
t1=-1:0.01:1
y1=(t1==0);
subplot(2,2,1);
plot(t1,y1);
xlabel('time');
ylabel('amplitude');
title('unit impulse signal');
%generation of impulse sequence
subplot(2,2,2);
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
```

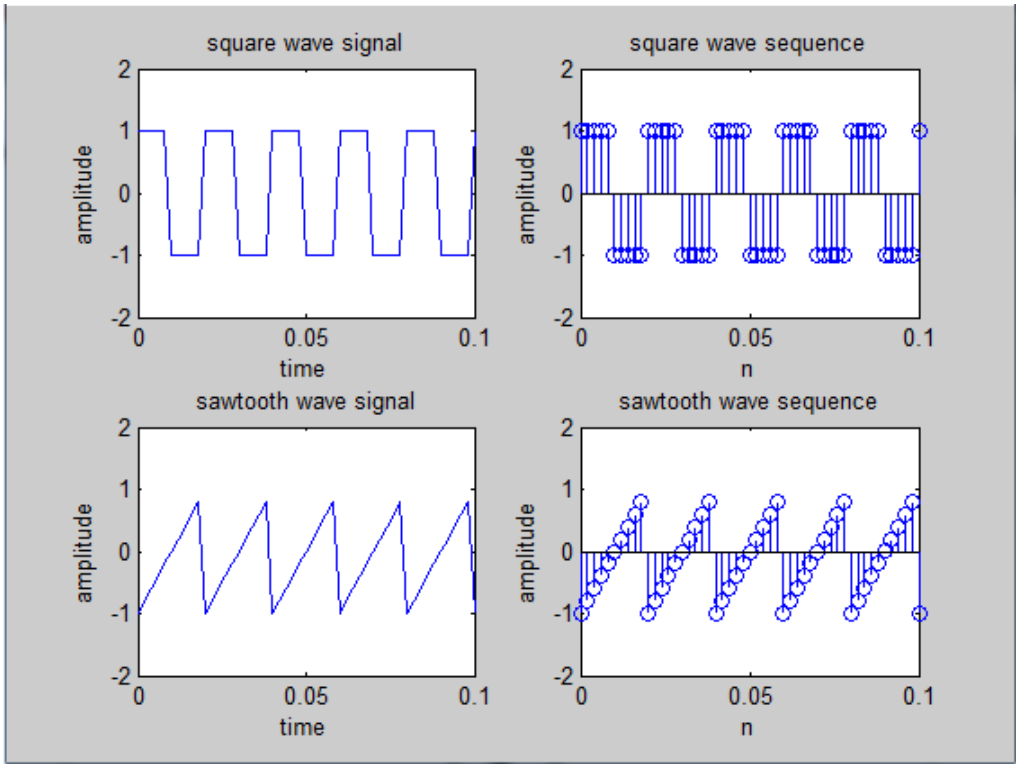
```
subplot(2,2,4);
stem(t2,y2);
xlabel('n');
ylabel('amplitude');
title('unit step sequence');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%generation of square wave signal
t=0:0.002:0.1;
y3=square(2*pi*50*t);
figure;
subplot(2,2,1);
plot(t,y3);
axis([0 0.1 -2 2]);
xlabel('time');
ylabel('amplitude');
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');
```

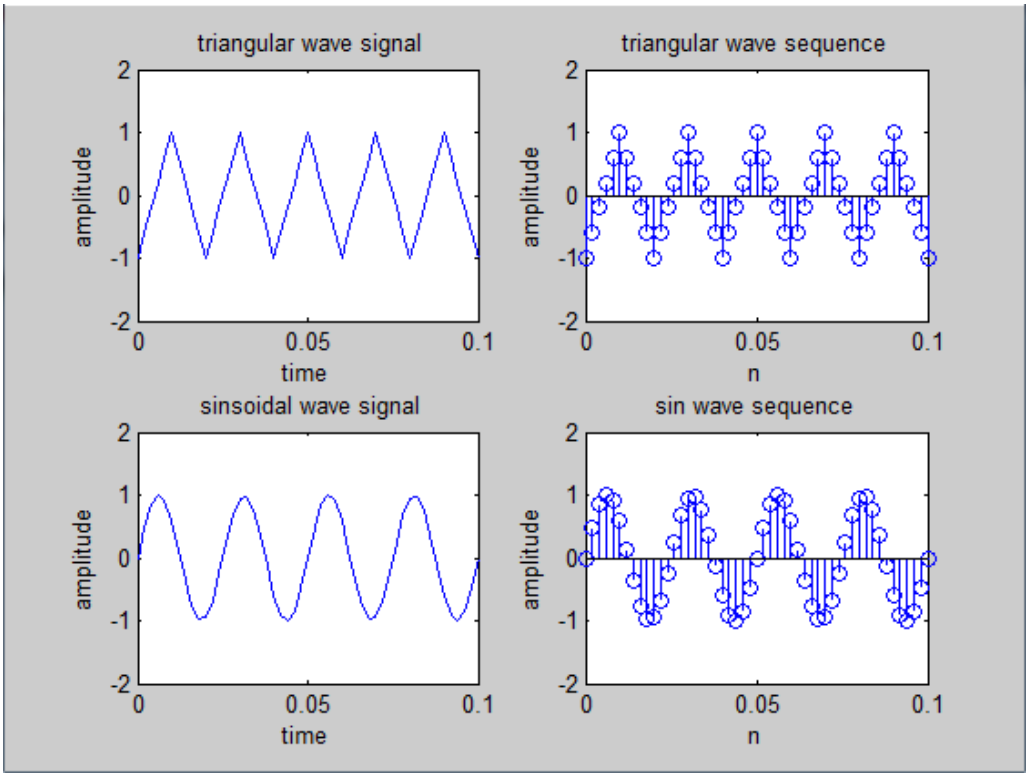
```
ylabel('amplitude');
title(' triangular wave signal');
%generation of triangular wave sequence
subplot(2,2,2);
stem(t,y5);
axis([0 0.1 -2 2]);
xlabel('n');
ylabel('amplitude');
title('triangular wave sequence');
%generation of sinsoidal wave signal
y6=sin(2*pi*40*t);
subplot(2,2,3);
plot(t,y6);
axis([0 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');
```

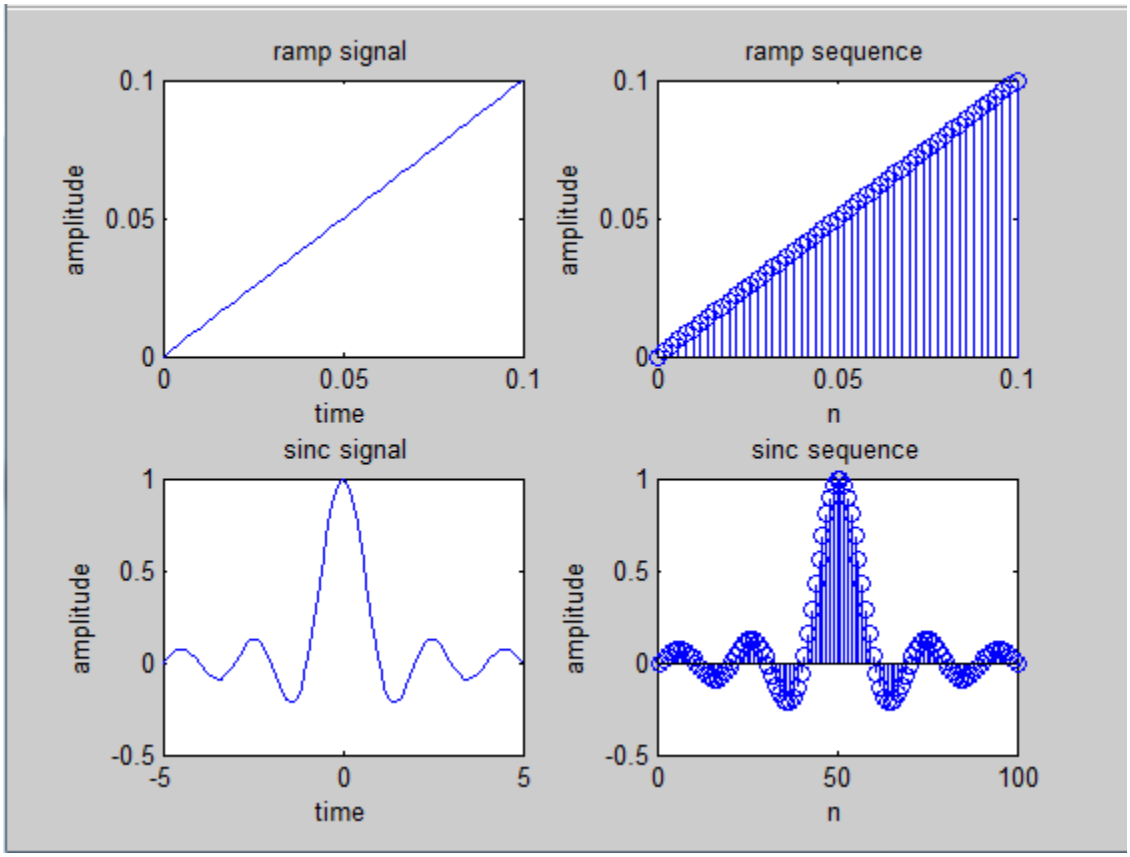
```
title(' sinc signal');
%generation of sinc sequence
subplot(2,2,4);
stem(y8);
xlabel('n');
ylabel('amplitude');
title('sinc sequence');
```

Result: Various signals & sequences generated using Matlab software.




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


triangular wave signal
amplitude
time
triangular wave sequence
amplitude
n
sinsoidal wave signal
amplitude
time
sin wave sequence
amplitude
n



Results and discussion : Periodic and Aperiodic, Unit Impulse, Unit Step, Square, Saw tooth, Triangular, Sinusoidal, Ramp, Sinc function are generated and plotted.

3. OPERATIONS ON SIGNALS AND SEQUENCES

AIM:

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

THEORY:

Signal energy and power:

The energy of a signal is the area under the squared signal.

Power is a time average of energy (energy per unit time). This is useful when the energy of the signal goes to infinity.

Energy vs. Power

1. "Energy signals" have finite energy.

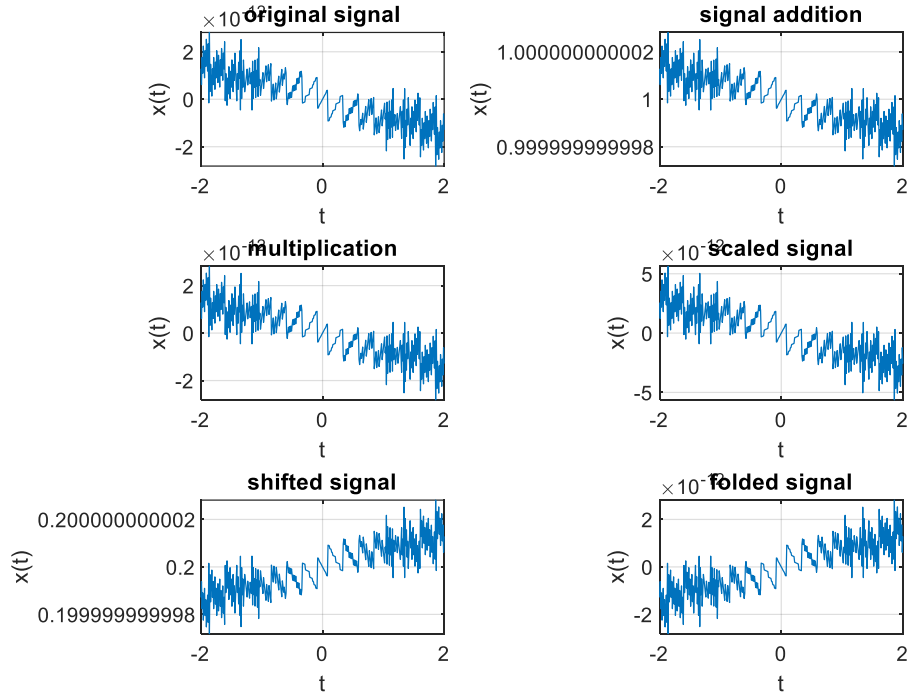
2. "Power signals" have finite and non-zero power.

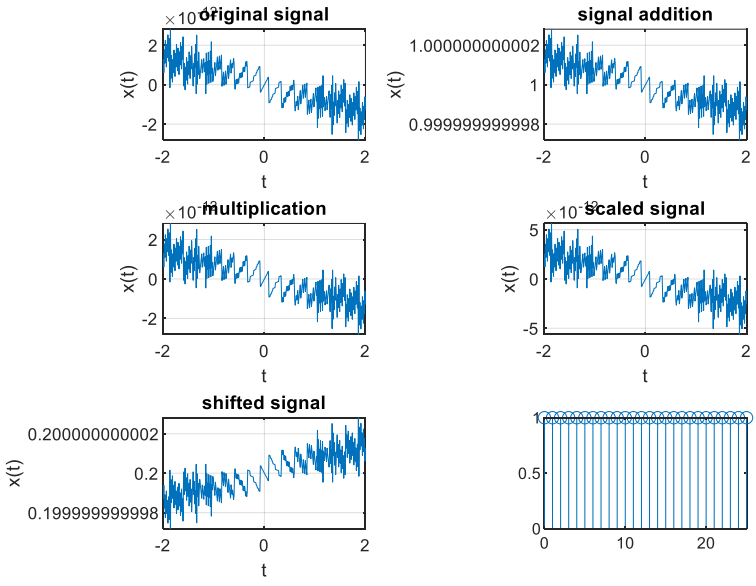
Program for Addition, Multiplication, Scaling, Shifting, Folding

```
clc; clear all;

t=-2:0.01:2;

f=input ('enter the fundamental frequency :');

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

x1=cos(2*pi*f*t);
x_sum=x+x1;

x_pro=x.*x1;

x_sca=2*x;

x_shi=0.2-x;

x_fol=-x;

subplot (3,2,1)

plot (t,x);grid on;

title ('original signal');
```

```
xlabel('t');
ylabel('x(t)');
subplot (3,2,2);
plot (t,x_sum);grid on;
title ('signal addition');
xlabel('t');
ylabel('x(t)');
subplot (3,2,3);
plot (t,x_pro);grid on;
title ('multiplication');
xlabel('t');
ylabel('x(t)');
subplot (3,2,4);
plot (t,x_sca); grid on;
title ('scaled signal');
xlabel('t');
ylabel('x(t)');
subplot (3,2,5);
plot (t,x_shi);grid on;
title ('shifted signal');
xlabel('t');
ylabel('x(t)');
subplot (3,2,6);
plot (t,x_fol);grid on;
title ('folded signal');
```

xlabel('t');ylabel('x(t)');





Computation of Energy of a signal

```
clc; clear all;
n=0:1:50;
x =cos (2*pi*n);
stem(n , x );
axis([0 25 0 1 ]);
disp('The calculated energy E of the signal is ' );
E= sum (abs(x).^2)
```

Computation of power of a signal

```
N= input ('Enter a value for N: ');
t= -N: 0.0001: N
x= cos (2 *pi *50 *t);
disp('The calculated power of the signal is :')
P= sum (abs(x). ^2)/length (x);
```

RESULT: various functions on signals and sequences such as addition, multiplication, scaling, shifting, folding, computation of energy and power were performed in MATLAB

4. Linear and Circular Convolution of two Sequences/ Signals.

AIM:

To perform linear and Circular Convolution of two Sequences/ Signals.

THEORY:

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.

Program for Convolution

```
clc;  clear all;  close all;

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

h=input ('enter the impulse response');

n1=length(x); disp (n1)

n2=length (h); disp (n2)

n3=n1+n2-1;

disp('the resultant length is:');n3

y=conv2(x,h);

subplot (3,1,1);

t1=0: 1: (n1-1);

stem(t1,x);  xlabel('n');  ylabel('x(n)');  title ('input sequence');

subplot (3,1,2);

t2=0: 1: (n2-1);

stem (t2,h);

xlabel('n');  ylabel ('h(n)');  title ('impulse response');

subplot (3,1,3);

t3=0 : 1 : ( n3-1);
```

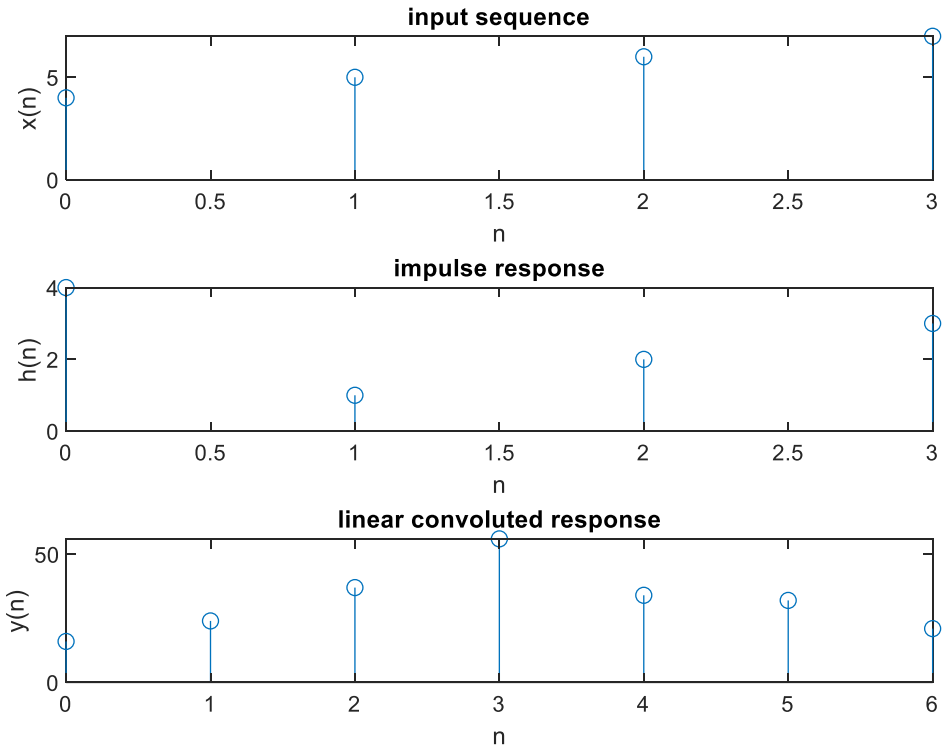
stem (t3,y); xlabel('n'); ylabel('y(n)'); title('linear convoluted response');

disp('the resultant signal is:');

out put : enter the input sequence[4 5 6 7]

enter the impulse response[4 1 2 3]

the resultant signal is:



Program for Convolution between signals

clc; clear all; close all;

t= -pi: 0.01: pi;

f=input ('enter the fundamental freq');

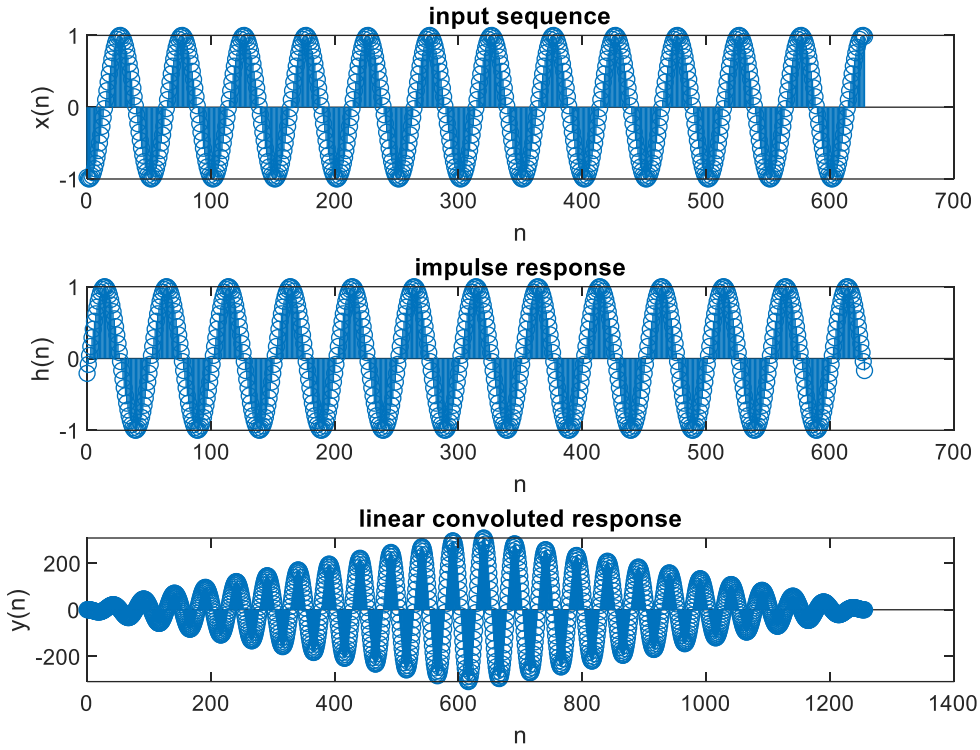
x=sin(2*pi*f*t);

h=cos(2*pi*f*t);

```
n1=length(x);
disp(n1)
n2=length(h);
disp(n2)
n3=n1+n2-1;
disp('the resultant length is:');n3
y=conv2(x,h);
subplot (3,1,1);
t1=0:1 :( n1-1);
stem (t1,x);
xlabel('n');
ylabel('x(n)');
title ('input sequence');
subplot (3,1,2);
t2=0:1 :( n2-1);
stem(t2,h);
xlabel('n');
ylabel('h(n)');
title ('impulse response');
subplot (3,1,3);
t3=0:1 :( n3-1);
stem (t3,y);
xlabel('n');
ylabel('y(n)');
title('linear convoluted response');
```

```
disp('the resultant signal is:');y
```

RESULT:



RESULT: Linear and Circular Convolution of Sequences/signals were verified using MATLAB

5. AUTO CORRELATION AND CROSS CORRELATION

AIM:

To Verify Auto and Cross Correlation of Sequences / Signals Using MATLAB

Program for Cross correlation of sequences:

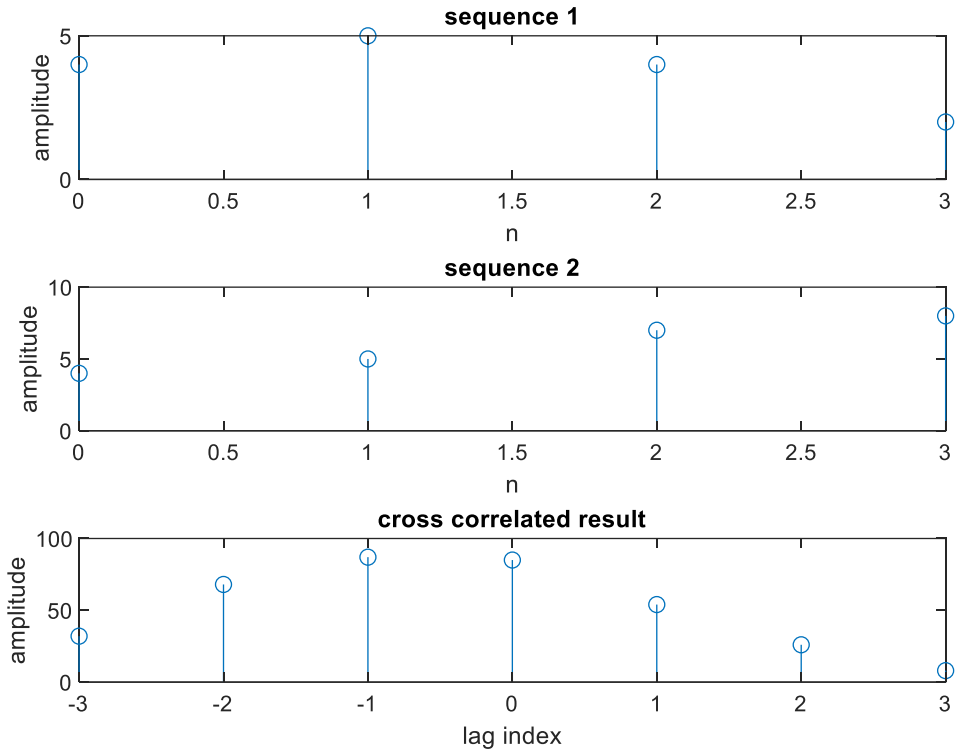
```
clc;  clear all;  close all;
x=input ('enter the first seq');
y=input ('enter the second seq');
n1= (length(y)-1)
n2= (length(x)-1)
k= (-n1):n2';
subplot (3,1,1);
t1=0:1:n2;
stem (t1,x);
 xlabel('n');  ylabel('amplitude');  title ('sequence 1');
subplot (3,1,2);
t2=0:1:n1;
stem(t2,y);
 xlabel('n');  ylabel('amplitude');  title('sequence 2');
r=xcorr(x,y);
subplot (3,1,3);
stem (k , r);
 xlabel('lag index');  ylabel('amplitude');
title ('cross correlated result');
disp('cross correlated result is:');
```

OUTPUT:

enter the first seq[4 5 4 2]

enter the second seq[4 5 7 8]

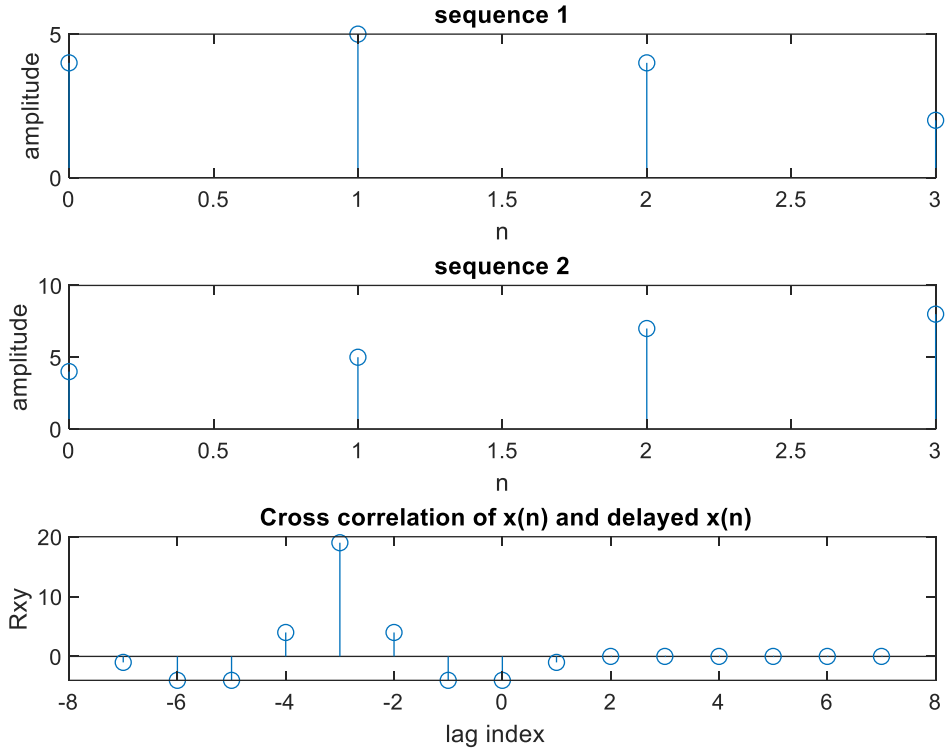
cross correlated result is:



Program for cross correlation of signal x(n) and delayed x(n)

```
x = [1 2 3 -2 -1];
D= input (' Enter the delay ');
xd = [zeros(1,D),x];
[r,lag] = xcorr(x,xd);
stem(lag,r);
title(' Cross correlation of x(n) and delayed x(n)');
xlabel('lag index  '); ylabel ('Rxy');
```

Enter the delay 3

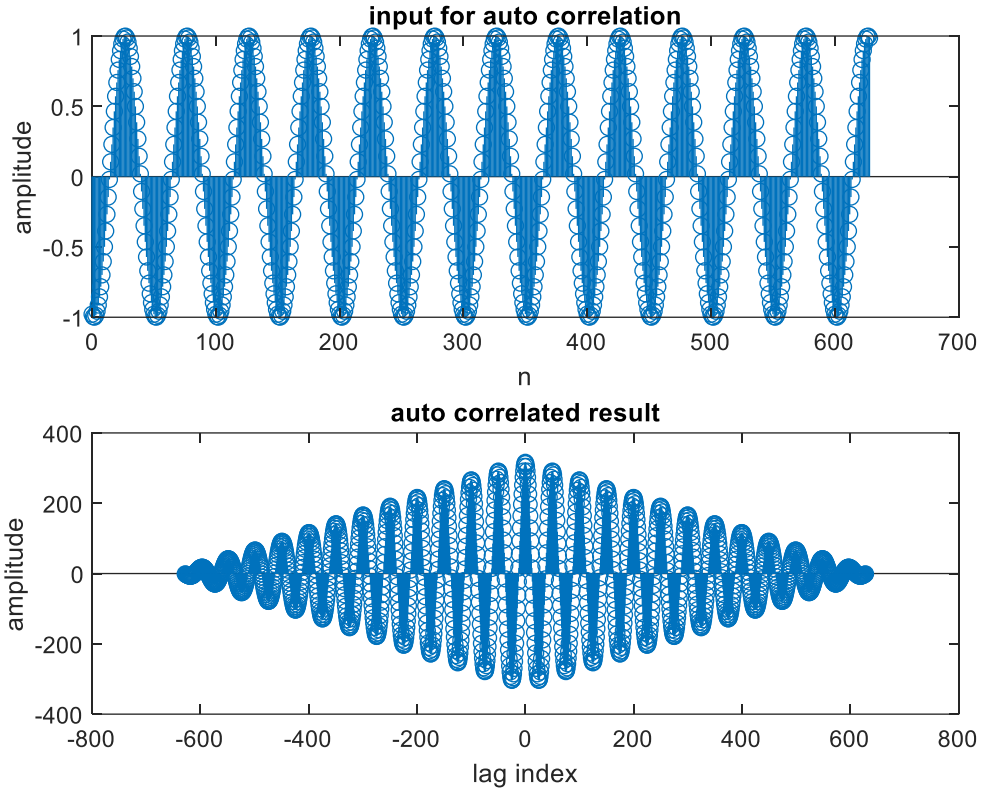


Program for Auto Correlation of a signal

```
clc;  clear all;  close all;

t= -pi: 0.01: pi;

f=input ('enter the fundamental freq:');

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

n2= (length(x)-1)

k= (-n2):n2';

subplot (2,1,1);

t1=0:1:n2;

stem (t1,x);

xlabel('n');       ylabel('amplitude');

title ('input for auto correlation');
```

```
r=xcorr(x,x);
subplot (2,1,2);
stem (k,r);      xlabel('lag index');      ylabel('amplitude');      title ('auto correlated result');
```

OUTPUT : enter the fundamental freq:2



RESULT: Auto and Cross Correlation of Sequences/ Signal was verified using MATLAB

6. VERIFICATION OF LINEARITY AND TIME INVARIANCE

AIM:

To Verify the Linearity and Time Invariance Properties of a given system using MATLAB

Program for Linearity check

```
x1=input('Enter the samples of x1'); x2= input('Enter the samples of x2');
if (length(x1) ~= length (x2))
disp('ERROR : Lengths of x1 & x2 are different');
return; end;
h= input('Enter the samples of h');
N= length(x1) + length (h) -1;
disp('length of the output signal will be '); disp(N);
a1=input(' The scale factor a1 is '); a2=input(' The scale factor a2 is ');
x= a1 *x1 + a2* x2;
y01 = conv2(x,h);
y1 = conv2(x1,h);
y1s = a1 * y1;
y2 = conv2(x2,h);
y2s = a2 * y2;
y02 = y1s +y2s;
disp('input signal x1 is '); disp(x1); disp('input signal x2 is '); disp(x2);
disp('output sequence y01 is  '); disp(y01); disp('output sequence y02 is  '); disp(y02);
if (y01==y02)
disp(' y01=y02 .Hence the LTI system is LINEAR ');
end;
```

OUTPUT :

Enter the samples of x1[2 5 4 6]

Enter the samples of x2[4 5 6 7]

Enter the samples of h[5 6 4 7]

length of the output signal will be 7

The scale factor a1 is 1

The scale factor a2 is 3

input signal x1 is

2 5 4 6

input signal x2 is

4 5 6 7

output sequence y01 is

70 184 286 445 390 262 189

output sequence y02 is

70 184 286 445 390 262 189

y01=y02 .Hence the LTI system is LINEAR

Program for verification of Time invariance property of a discrete time system

```
x=input('Enter the samples of x(n)'); h= input('Enter the samples of h(n)');
y=conv2(x, h);
disp( 'Enter a positive number for delay');
d = input (' Desired delay of the signal is ');
xd = [zeros(1,d),x];
nxd = 0 : length (xd)-1;
yd = conv2( xd,h);
nyd = 0 : length (yd)-1;
disp('Original input signal x(n) is '); disp(x); disp(' Delayed input signal xd(n) is  '); disp(xd);
```

```
disp('Original output signal y(n) is '); disp(y); disp(' Delayed output signal yd(n) is  '); disp(yd);
xp= [ x, zeros(1,d)];
figure
subplot(2,1,1); stem(nxd,xp);grid; xlabel('Time index n '); ylabel('x (n)');
title(' Original input sequence x(n)');
subplot(2,1,2); stem(nxd,xd);grid; xlabel('Time index n '); ylabel('x d(n)');
title(' Delayed input signal xd(n)');
yp= [ y, zeros(1,d)];
figure
subplot(2,1,1); stem(nyd,yp);grid; xlabel('Time index n '); ylabel('y (n)');
title(' Original output signal y(n)');
subplot (2,1,2); stem(nyd,yd);grid; xlabel('Time index n '); ylabel('yd(n)');
title(' Delayed output signal yd(n)');
```

OUTPUT :

Enter the samples of x(n)[5 6 4 7]

Enter the samples of h(n)[4 5 6 7]

Enter a positive number for delay

Desired delay of the signal is 3

Original input signal x(n) is

5 6 4 7

Delayed input signal xd(n) is

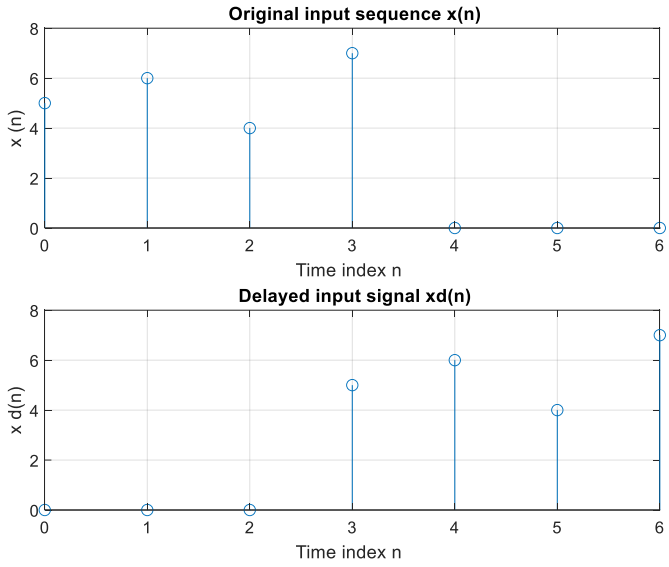
0 0 0 5 6 4 7

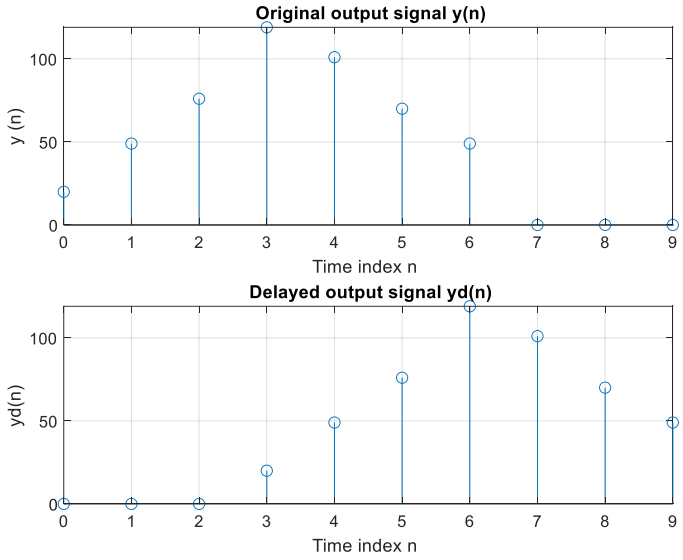
Original output signal y(n) is

20 49 76 119 101 70 49

Delayed output signal yd(n) is

0 0 0 20 49 76 119 101 70 49





RESULT: The Linearity and Time Invariance Properties using MATLAB are verified.

7. FOURIER TRANSFORM

AIM:

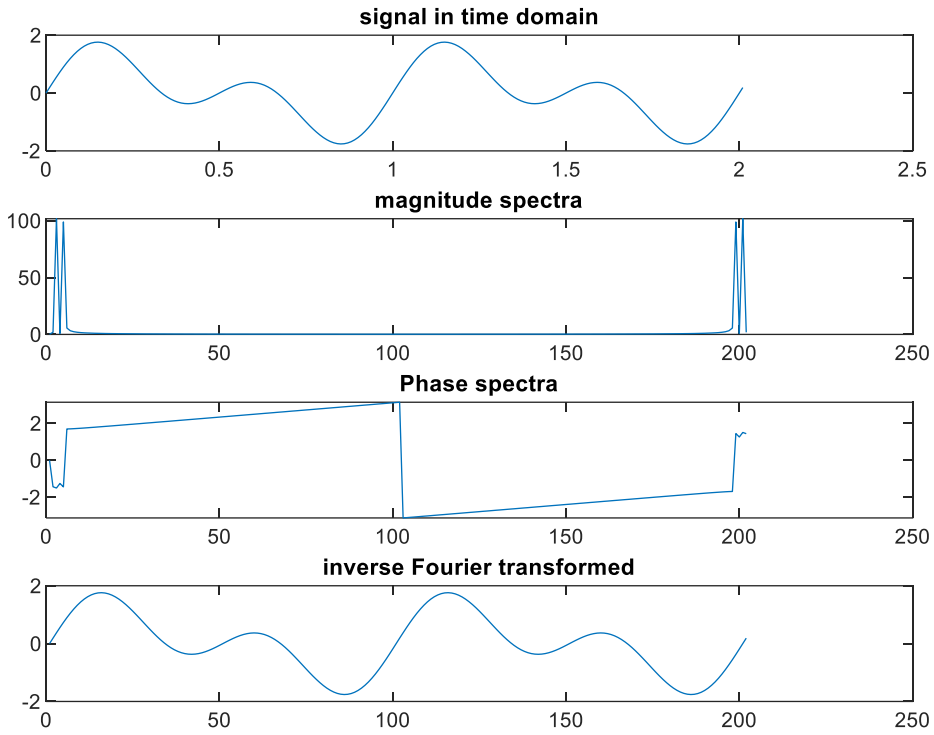
To obtain Fourier Transform and Inverse Fourier Transform of a given signal / sequence and to plot its Magnitude and Phase Spectra.

Program for Fourier Transform and Inverse FT of a Signal

```
N=input ('enter the length of input signal');

%Prepare to sample a signal for two seconds, at a rate of 100 samples per second.

Fs = 100; % Sampling rate

t = [0:2*Fs+1]'/Fs % Time points for sampling

x = sin (2*pi*t) + sin(4*pi*t);

subplot (4,1,1);  plot (t,x);

title ('signal in time domain');

XK=fft(x,N);

disp(XK);

m=abs(XK);

subplot (4,1,2);  plot(m);

title('magnitude spectra');

a=angle (XK);

subplot (4,1,3);  plot(a);

title ('Phase spectra');

xn=ifft(XK,N);

disp(xn);

subplot(4,1,4);  plot (xn);

title('inverse Fourier transformed')
```

OUTPUT :

enter the length of input signal[4 5 6 1]



PROGRAM TO FFT AND IFFT OF A COMPLEX SIGNAL

```
f1 = input ('enter the frequency of signal 1 :')

f2 = input ('enter the frequency of signal 2 :')

fs = 5*max(f1,f2)

N = fs %input('enter the size of fft')

t= 0:1/fs:1-1/fs

x = sin (2*pi*f1*t) + sin (2*pi*f2*t);

subplot (4,1,1);  plot(t,x);

title ('signal in time domain');

y=fft(x,N);

disp(y);
```

```
m=abs(y);
f = (0: N-1)/ (N/fs)
subplot (4,1,2);  plot(f,m);
title ('magnitude spectra');
a=angle(y);
subplot (4,1,3);  plot(f,a);
title ('Phase spectra');
xn=ifft(y);
disp(xn);
subplot (4,1,4);  plot(t,xn);
title ('inverse Fourier transformed')
```

OUTPUT : enter the frequency of signal 1 :10

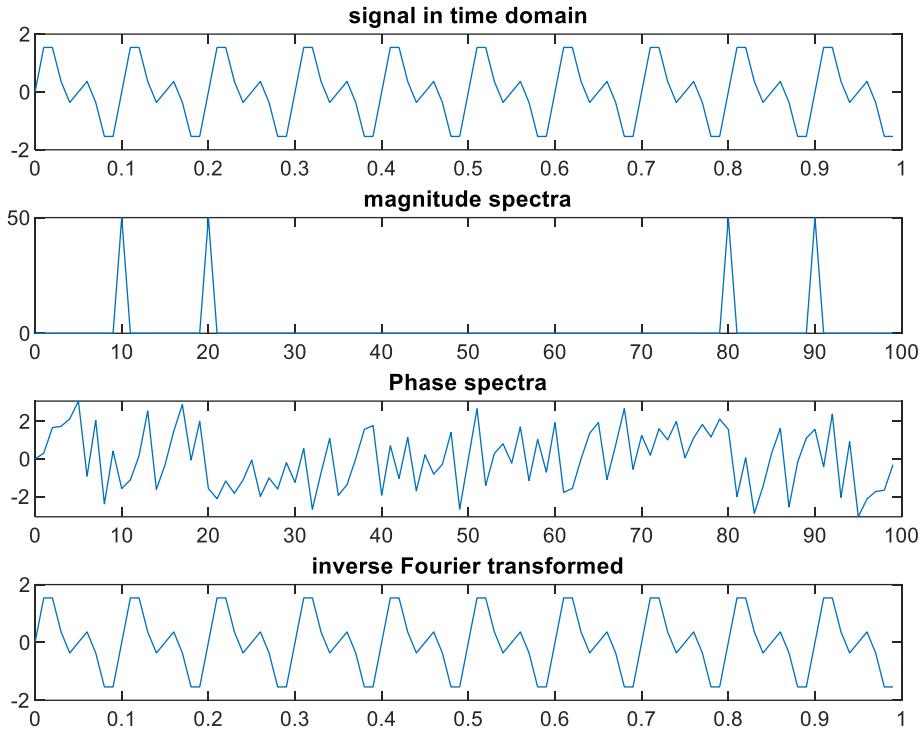
f1 = 10

enter the frequency of signal 2 :20

f2 = 20

fs = 100

N = 100



RESULT: The Magnitude and Phase Spectra of Fourier Transformed Signal was Plotted

8. WAVEFORM SYNTHESIS USING LAPLACE TRANSFORM

AIM: To obtain Laplace and Inverse Laplace Transforms of different functions.

Program for symbolic variables and expressions

```
syms a

b=a/(1-a)^2

c=subs(b,3)

syms d

f=d^2

g=subs(b,f)

subs(g,2)

ezplot(b,[-50 , 50])
```

Program for finding Laplace transform

```
syms t s % specify that variables t and s are symbolic ones

f=-1.25+3.5*t*exp (-2*t) +1.25*exp (-2*t)

F=laplace(f,t,s)

simplify (F)

pretty (ans)
```

Program for finding inverse Laplace transform

```
a=(s-5)/(s*(s+2)*2);

f=ilaplace(a)

simplify(f)

pretty(f)
```

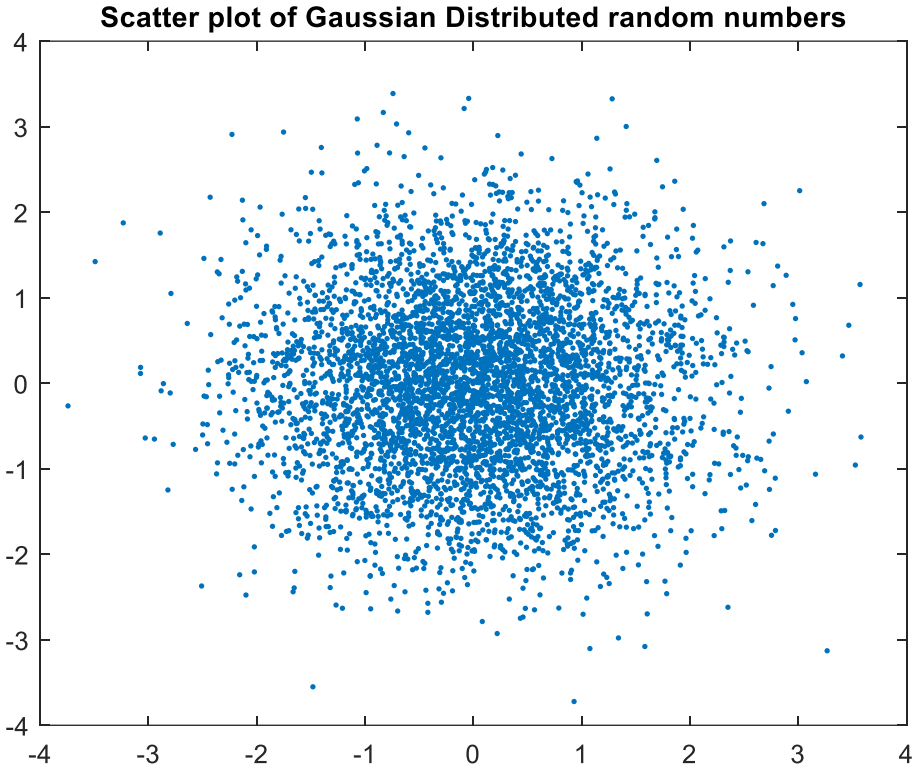
RESULT: Laplace and inverse Laplace transforms were verified using MATLAB functions.

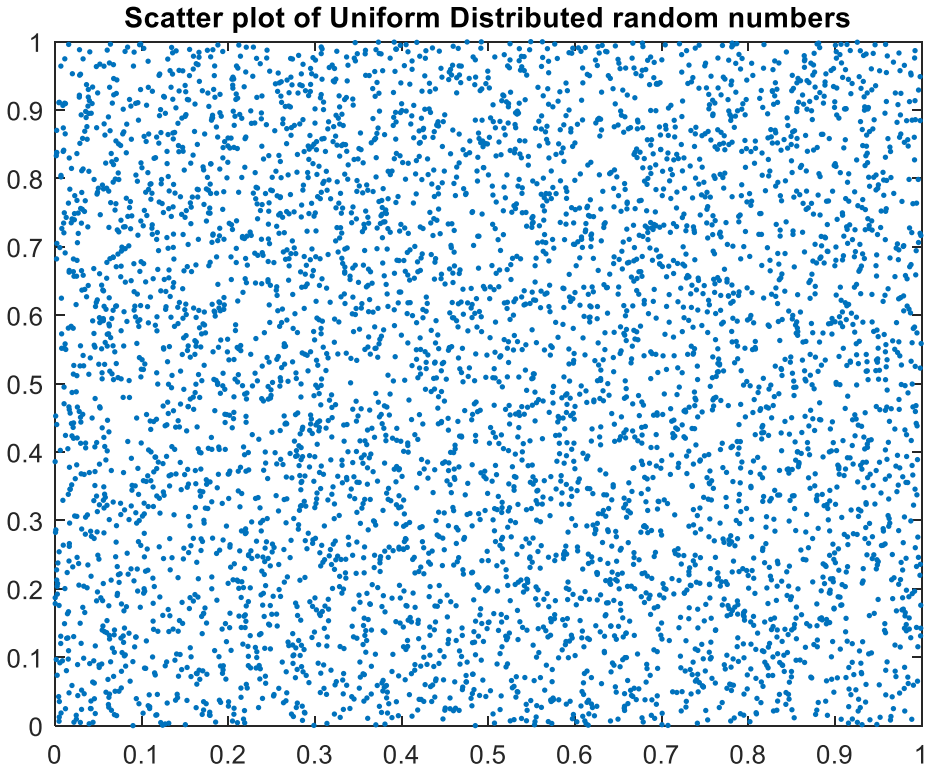
9. GENERATION OF GAUSSIAN NOISE

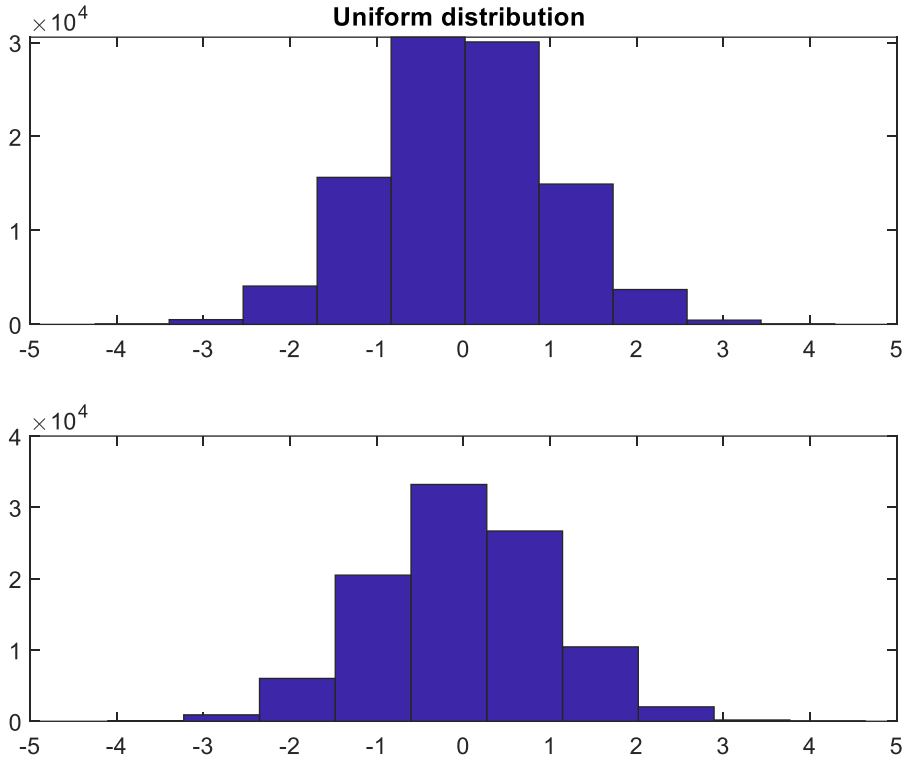
AIM: To Generate Gaussian Noise and to compute its Mean, M.S. Values, Skew, kurtosis, PSD and PDF

Program Generate Gaussian noise signal

```
clc;clear all; x1=randn(1,5000); x2=randn(1,5000);

figure;

plot(x1,x2,'.'); title(' Scatter plot of Gaussian Distributed random numbers ');

x1=rand(1,5000); x2=rand(1,5000);

figure;

plot(x1,x2,'.');

title (' Scatter plot of Uniform Distributed random numbers ');

x3=randn(1,100000);

figure;

subplot(2,1,1); hist(x3); title('Uniform distribution ');

y=randn(1,100000);

subplot(2,1,2); hist(y);

ymu=mean(y)

ymsq=sum(y.^2)/length(y)

ysigma=std(y)

yvar=var(y)

yskew=skewness(y)

ykurt=kurtosis(y)
```


Scatter plot of Gaussian Distributed random numbers
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4


Scatter plot of Uniform Distributed random numbers
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



RESULT: Additive White Gaussian noise was generated and its PSD and PDF were plotted and its Mean, Standard Deviation, Kurtosis, Skew were Computed using MATLAB functions.

10. SAMPLING THEOREM VERIFICATION.

AIM:

To Demonstrate Sampling Theorem and aliasing effect using MATLAB.

THEORY:

The theorem shows that a band limited analog signal that has been sampled can be perfectly reconstructed from an infinite sequence of samples if the sampling rate exceeds 2B samples per second, where B is the highest frequency in the original signal. If a signal contains a component at exactly B hertz, then samples spaced at exactly 1/ (2B) seconds do not completely determine the signal, Shannon's statement notwithstanding.

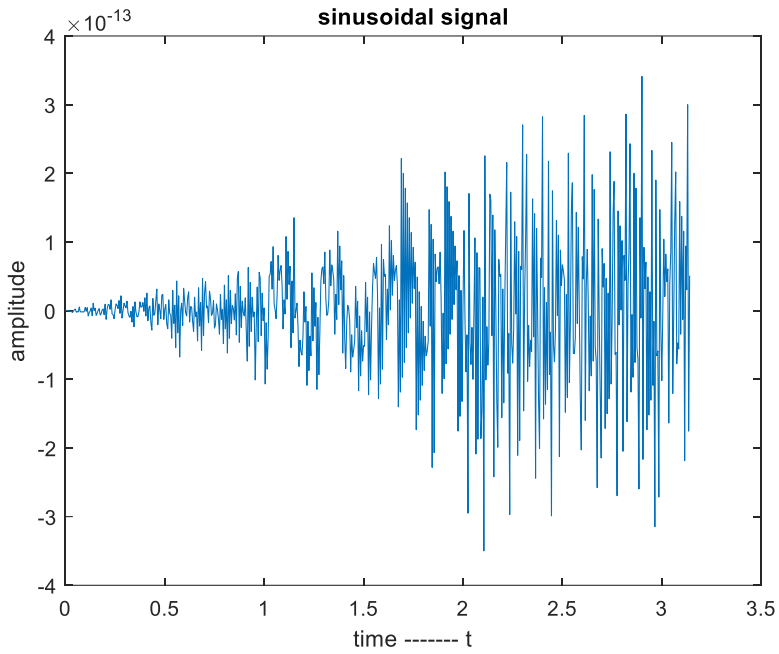
Program for sampling theorem verification

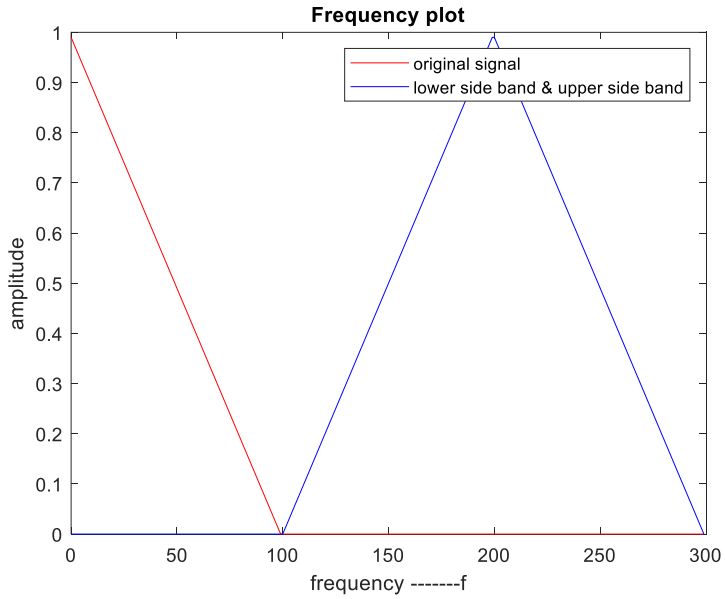
```
f= input ('enter the frequency')

fs= input('enter the sampling frequency')

t=0:1:fs+f-1;

if fs >= 2*f

disp('sampling theorem is satisfied ')

%ploting of sinusoidal

 n=0:1/fs:pi;

x=sin(2*pi*f*n);

figure(1)

plot(n,x)

title('sinusoidal signal')

xlabel('time ------- t')

ylabel('amplitude')  % to plot the frequency spectrum (indirect method)

 s1=0:1/f:1-1/f;

s2=(1-1/f)-s1;

 w1=[s2,zeros(1,(fs))];
```

```
w2=[zeros(1,fs-f),s1,s2];
figure (2)
plot(t,w1,'r',t,w2,'b')
title('Frequency plot')
xlabel('frequency -------f')
ylabel('amplitude')
legend ('original signal', 'lower side band & upper side band')
else
 disp('sampling theorem is not satisfied')
% to plot the frequency spectrum (indirect method)
 s1=0:1/f:1-1/f;
s2=(1-s1);
 w1=[s2,zeros(1,(fs))];
w2=[zeros(1,fs-f),s1,s2];
figure (3)
plot(t,w1,'r',t,w2,'b')
 title('Frequency plot')
 xlabel('frequency -------f')
 ylabel('amplitude')
 legend ('original signal', 'lower side band & upper side band')
end
```

OUTPUT :

```
nter the frequency100
enter the sampling frequency200
sampling theorem is satisfied
```





RESULT: Sampling theorem was verified and aliasing was identified when the sampling frequency is less than the nyquist frequency using MATLAB.

11. REMOVAL OF NOISE BY AUTO CORRELATION/CROSS CORRELATION IN A GIVEN SIGNAL CORRUPTED BY NOISE

AIM:

To generate a periodic sequence, corrupt it with zero mean White noise and extract the sequence using periodic circular cross correlation.

Program for extracting a periodic signal from noise

```
clear all

clc

rand('state',1000)                    % Construct signals x and y

N = 256;

k = 0 : N-1;

x = cos(32*pi*k/N) + sin(48*pi*k/N); % periodic signal which include

%two sin components

a = 0.5;

y = f_randu(1,N,-a,a) + x; % + white noise uniformly distributed over

%interval(-.5 , 0.5)

figure

plot (k,y)

f_labels ('A noisy periodic signal','\it{k}','\it{y(k)}')

f_wait

M = N/8;

L = floor(N/M); %no. of complete cycles of x(k) in y(k)

% generation of N-point Periodic impulse train with period M…such that

% M<<N

d = zeros(1,N);

for i = 1 : N
```

```
if rem(i-1,M) == 0
d(i) = 1;
end
end
% Compute cross-correlation with periodic pulse train
c_yd = f_corr (y,d,1,0); % fast cross correlation.
% Plot portions of x and rho
figure
m = 0 : N/4;
plot (m,x(m+1),m,(N/L)*c_yd(m+1))
f_labels ('Comparison of {\itx(k)} and {\it(N/L)c_{y\delta_M}(k)}','\it{k}','signals')
```

RESULT: using Cross-correlation the periodic signal from noise was estimated using MATLAB.

12. IMPULSE RESPONSE OF RAISED COSINE FILTER

AIM:

To Design an FIR Raised Cosine Filter and to plot its Magnitude, Phase and Impulse responses using MATLAB.

PROGRAM FOR IMPULSE RESPONSE OF RAISED COSINE FILTER

```
Fd=input('enter Fd the sampling frequency of digital i/p signal:');

disp('The ratio Fs/Fd must be a positive integer greater than 1');

% Define filter-related parameters.

Fs=input('enter Fs the sampling frequency for the filter:');

filtorder = 40; % Filter order

delay = filtorder/(Fs*2); % Group delay (# of input samples)

rolloff = 0.25; % Rolloff factor of filter

rcfilter = rcosine(Fd,Fs,'fir',rolloff,delay);

H=tf(rcfilter)

% Plot impulse response.

%figure; impz(rcfilter,1)

fvtool(rcfilter)
```

OUTPUT :

```
enter Fd the sampling frequency of digital i/p signal:10

The ratio Fs/Fd must be a positive integer greater than 1

enter Fs the sampling frequency for the filter:20

H =

  From input 1 to output:

  3.675e-17

  From input 2 to output:
```

0.6274

From input 3 to output:1

From input 4 to output:

0.6274

From input 5 to output:

3.675e-17

Static gain.

RESULT:

The Impulse Response of a Raised Cosine Filter was plotted using fvtool in MATLAB.

13. CHECKING A RANDOM PROCESS FOR STATIONARY IN WIDESENSE

AIM:

To generate a Random process and to check for its Wide Sense Stationary using MATLAB

THEORY:

A random process is wide sense stationary (WSS) if the mean function and autocorrelation function are invariant to a time shift. In particular, this implies that

All strict sense stationary random processes are also WSS, provided that the mean and autocorrelation function exist. The converse is not true. A WSS process does not necessarily need to be stationary in the strict sense. We refer to a process that is not WSS as non stationary.

Many of the processes we deal with are WSS and hence have a constant mean function and an autocorrelation function that depends only on a single time variable. Hence, in the remainder of the text, when a process is known to be WSS or if we are assuming it to be WSS, then we will represent its autocorrelation function by RXX (t). If a process is non stationary or if we do not know if the process is WSS, then we will explicitly write the autocorrelation function as a function of two variables, RXX (t, t + t).

PROGRAM:

```
clc;

clear all;

close all;

syms pi a wo t t1 t2 theta

l=input('Enter the lower limit');

u=input('Enter the upper limit');

x=input('Enter the PDF');

f=a*cos((wo*t)+theta);

f1=f*x;

y1=int(f1,theta,l,u);

disp('The Expectation is:');
```

```
disp(y1);

f2=a*cos((wo*t1)+theta);

f3=a*cos((wo*t2)+theta);

f4=f2*f3*x;

y2=int(f4,theta,l,u);

disp('The AutoCorrelation is:');

disp(y2);
```

Manual Calculation

1) Consider the random process X(t)=Acos(.0t+T) Where “T” is real-valued random variable and is uniformly distributed over [0, p]. Check if the process is wide sense stationary.

2) Consider the random process X(t)=A cos(.0t+T) Where “T” is real-valued random variable and is uniformly distributed over [0, 2p]. Check if the process is wide sense stationary.

RESULT:

Thus the given random processes are identified whether they are WSS or not through calculating the mean and auto correlation functions manually and verified these results through MATLAB simulation.