Lab Code:20ECL303
Signals and Systems
Lab Manual

Department of Electronics & Communication Engineering

Bapatla Engineering College :: Bapatla

(Autonomous)

G.B.C. Road, Mahatmajipuram, Bapatla-522102, Guntur (Dist.)

Andhra Pradesh, India.

E-Mail:bec.principal@becbapatla.ac.in

Web:www.becbapatla.ac.in

Contents

Bapatla Engineering College :: Bapatla
(Autonomous)

Vision

- To build centers of excellence, impart high quality education and instill high standards of ethics and professionalism through strategic efforts of our dedicated staff, which allows the college to effectively adapt to the ever changing aspects of education.

- To empower the faculty and students with the knowledge, skills and innovative thinking to facilitate discovery in numerous existing and yet to be discovered fields of engineering, technology and interdisciplinary endeavors.

Mission

- Our Mission is to impart the quality education at par with global standards to the students from all over India and in particular those from the local and rural areas.

- We continuously try to maintain high standards so as to make them technologically competent and ethically strong individuals who shall be able to improve the quality of life and economy of our country.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Vision

To produce globally competitive and socially responsible Electronics and Communication Engineering graduates to cater the ever changing needs of the society.

Mission

- To provide quality education in the domain of Electronics and Communication Engineering with advanced pedagogical methods.

- To provide self learning capabilities to enhance employability and entrepreneurial skills and to inculcate human values and ethics to make learners sensitive towards societal issues.

- To excel in the research and development activities related to Electronics and Communication Engineering.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Educational Objectives (PEO's)

PEO-I: Equip Graduates with a robust foundation in mathematics, science and Engineering Principles, enabling them to excel in research and higher education in Electronics and Communication Engineering and related fields.

PEO-II: Impart analytic and thinking skills in students to develop initiatives and innovative ideas for Start-ups, Industry and societal requirements.

PEO-III: Instill interpersonal skills, teamwork ability, communication skills, leadership, and a sense of social, ethical, and legal duties in order to promote lifelong learning and Professional growth of the students.

Program Outcomes (PO's)

Engineering Graduates will be able to:

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. 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.

PO3. 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.

PO4. 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.

PO5. 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.

PO6. 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.

PO7.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.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and Teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. 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.

PO11. 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.

PO12. 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.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Specific Outcomes (PSO's)

PSO1: Develop and implement modern Electronic Technologies using analytical methods to meet current as well as future industrial and societal needs.

PSO2: Analyze and develop VLSI, IoT and Embedded Systems for desired specifications to solve real world complex problems.

PSO3: Apply machine learning and deep learning techniques in communication and signal processing.

SIGNALS AND SYSTEMS LAB
II B.Tech. III Semester (Code: 20ECL303)

Lectures	0	Tutorial	0	Practical	3	Credits	1.5
Continuous Internal Assessment			30	Semester End Examination (3 Hours)			70

Prerequisites: None

Course Objectives: Students will

- Describe the MATLAB syntax, functions and programming.
- Simulate various continuous and discrete time signals using MATLAB
- Perform basic operations on signals and sequences by using MATLAB
- Compute convolution, correlation between signals and sequences

Course Outcomes: After studying this course, the students will be able to

CO1	Demonstrate the MATLAB syntax, functions and programming.
CO2	Generate and characterize various continuous and discrete time signals by using MATLAB
CO3	Examine basic operations on signals and sequences by using MATLAB
CO4	Analyze LTI systems by using convolution and correlation

Mapping of Course Outcomes with Program Outcomes & Program Specific Outcomes

	PO's												PSO's		
CO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CO1	2				3				3						3
CO2	2				3				3						3
CO3	2				3				3						3
CO4	2				3				3						3
AVG	2				3				3						3

LIST OF PROGRAMS

1. Basic Operations on Matrices.
2. Program to show how to create a variety of 2-D plots in MATLAB.
3. Generation of basic continuous time signals namely unit impulse, step, ramp, exponential and Sinusoidal signals.
4. Generation of basic discrete time signals namely unit impulse, step,

ramp, exponential and Sinusoidal signals.

5. Operations on Signals and Sequences such as Addition, Multiplication, Scaling, Shifting, Folding, Computation of Energy and Average Power.
6. Finding the Even and Odd Parts of Signal or Sequence and Real and Imaginary Parts of Signal.
7. Verification of linearity and time invariance properties of a given Continuous/discrete system.
8. Convolution between Signals and Sequences.
9. Auto-correlation and Cross-correlation between Signals and Sequences.
10. Sampling Theorem Verification.

NOTE: All the programs have to be performed and recorded by the candidate to attain eligibility for Semester End Examination

Program 1: Basic Operations on Matrices

Aim: To perform basic operations on matrices such as addition, multiplication, transpose, length of the matrix, size of the matrix, mean of the matrix, inverse of the matrix etc.

Software Required: MATLAB

Theory:

length(X)	length(X) returns the length of vector X. It is equivalent to max(size(X)) for non-empty arrays and 0 for empty ones.
transpose(a)	transpose (a) is called for the syntax a.' when a is an object. a.' computes the non-conjugate transpose of matrix a
+ Plus	X + Y adds matrices X and Y. X and Y must have the same dimensions unless one is a scalar (a 1-by-1 matrix). C = plus(A,B) is called for the syntax 'A + B' when A or B is an object.
* Matrix multiply	X*Y is the matrix product of X and Y. Any scalar (a 1-by-1 matrix) may multiply anything. Otherwise, the number of columns of X must equal the number of rows of Y. C = mtimes(A,B) is called for the syntax 'A * B' when A or B is an object.
.*	X.*Y denotes element-by-element multiplication.
Array multiply.	X and Y must have the same dimensions unless one is a scalar. A scalar can be multiplied into anything. C = times(A,B) is called for the syntax 'A .* B' when A or B is an object.
Size	D = size(X), for M-by-N matrix X, returns the two-element row vector.
mean	mean(X) is the mean value of the elements in X. For matrices, mean(X) is a row vector containing the mean value of each column.
inv	inv(X) is the inverse of the square matrix X.

Matlab Program:

```
A=[1 2 3; 1 0 1; 2 1 4];

B=[1 1 3; 0 5 7; 2 7 4];

C=A+B; % Addition of two matrices

D=A*B   % Multiplication of two matrices

E=A.*B   % vector multiplication

F=transpose(A) % transpose of a matrix

G=length(B)      % length of a matrix

[H I]=size(A)     % size of the matrix

J=mean(A)       % calculating mean of the matrix

colvec=[13;20;5] % creating a column vector

rowvec=[1 2 3] %Creating a row vector

sub_matrix=A(2:3,2:3)   %Extracting sub matrix from matrix

col_vector=A(:,2) % extracting column vector from matrix row_vector=A(3,:) %

extracting row vector from matrix

K=inv(A)    % inverse of the matrix

L=plus(A,B)

M=mtimes(A,B)

N=times(A,B)

O=A.'

P=max(size(B))
```

OUTPUTS:

```
C =
     2     3     6
     1     5     8
     4     8     8

D =
     7    32    29
     3     8     7
    10    35    29

E =
     1     2     9
     0     0     7
     4     7    16

F =
     1     1     2
     2     0     1
     3     1     4

G =
     3

H =
     3

I= 3

J =
    1.3333    1.0000    2.6667

colvec =13
20
5

rowvec =
     1     2     3

sub_matrix =0   1
     1     4
```

```
col_vector = 2
     0
     1

row_vector = 2  1     4

  K =
    0.5000    2.5000   -1.0000
    1.0000    1.0000   -1.0000
   -0.5000   -1.5000    1.0000

  L =
     2     3     6
     1     5     8
     4     8     8


  M =
     7    32    29
     3     8     7
    10    35    29


  N =
     1     2     9
     0     0     7
     4     7    16

  O =
     1     1     2
     2     0     1
     3     1     4

  P =
     3
```

Program 2: Program to show how to create a variety of 2-D plots in MATLAB

Aim: To understand the use of different functions used in MATLAB forrepresenting 2D Plots.

Software Required: MATLAB

Theory:

1 Line Plots: The plot function creates simple line plots of *x* and *y* values.

2 Stem Plots: The stem function draws a marker for each *x* and *y* valuewith a vertical line connected to a common baseline.

3 Scatter Plots: The scatter function draws a scatter plot of *x* and *y* values.

4 Bar Plots: The bar function creates vertical bar charts. The barh functioncreates horizontal bar charts.

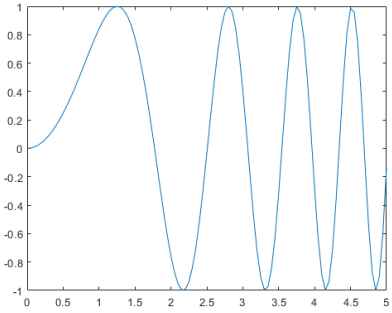
5 Stairstep Plots: The stairs function creates a stairstep plot. It can create astairstep plot of Y values only or a stairstep plot of *x* and *y* values.

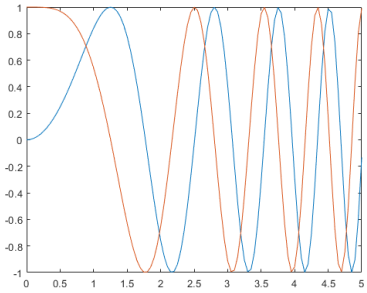
6 Errorbar Plots: The errorbar function draws a line plot of *x* and *y* values and superimposes a vertical error bar on each observation. To specify thesize of the error bar, pass an additional input argument to
the errorbar function

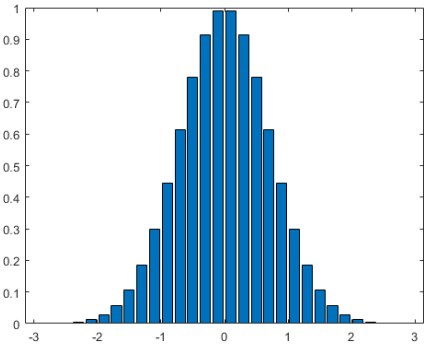
7 Polar Plots: The polarplot function draws a polar plot of the angle valuesin theta (in radians) versus the radius values in rho.

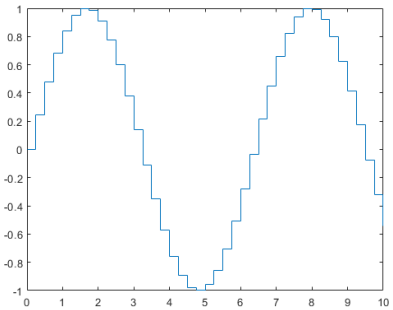
Program:

S.No.	Code	Output

1	x = 0:0.05:5;
	y = sin(x.^2);
	figure
	plot(x,y)
		

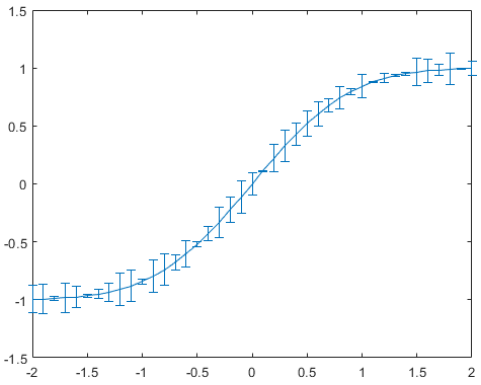
2	y1 = sin(x.^2);
	y2 = cos(x.^2);
	plot(x,y1,x,y2)
		

3	x = -2.9:0.2:2.9;
	y = exp(-x.*x);
	bar(x,y)
		

4	x = 0:0.25:10;
	y = sin(x);
	stairs(x,y)
		

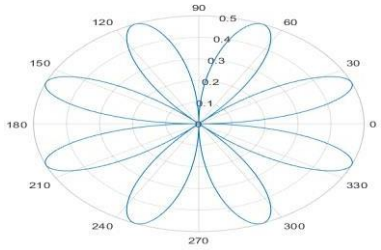
5

```
x = -2:0.1:2;
y = erf(x);
eb = rand(size(x))/7;
errorbar(x,y,eb)
```



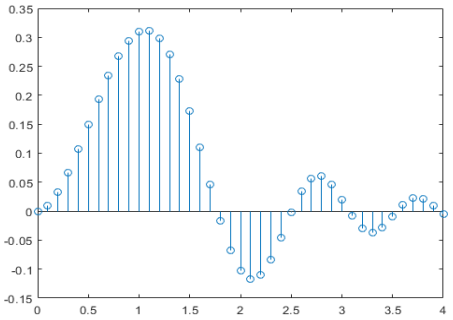
1.

```
theta = 0:0.01:2*pi;      % angle
rho =
abs(sin(2*theta).*cos(2*theta));
% radius
polarplot(theta,rho)
```



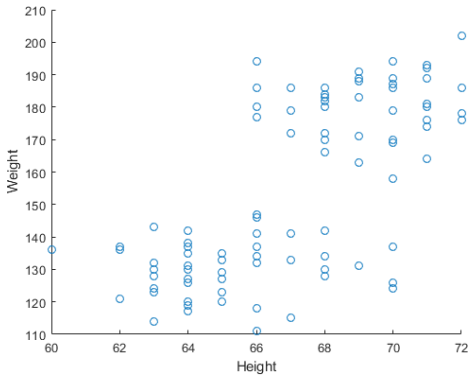
7.

```
x = 0:0.1:4;
y = sin(x.^2).*exp(-x);
stem(x,y)
```



8.

```
load patients Height Weight
Systolic    % load data
scatter(Height,Weight)
% scatter plot of Weight vs.
Height
xlabel('Height')
ylabel('Weight')
```



Program 3: Generation of basic continuous time signals namely unit impulse, step, ramp, exponential and Sinusoidal signals.

Aim: To generate basic continuous time signals namely unit impulse, step, ramp, exponential and Sinusoidal signals.

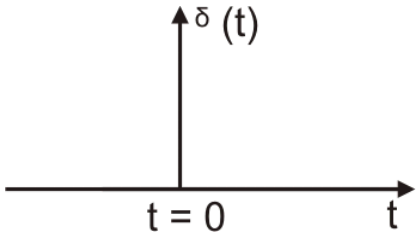
Software Required: MATLAB

Theory:

A continuous-time signal is a signal that can be defined at every instant of time. A continuous-time signal contains values for all real numbers along the X-axis.

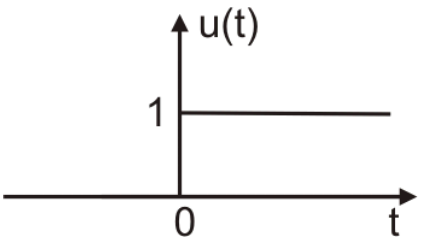
Unit Impulse An ideal impulse function is a function that is zero everywhere but at the origin, where it is infinitely high. However, the *area* of the impulse is finite.

$$\delta(t) = \begin{cases} 0, & t \neq 0 \\ \textit{undefined}, & t = 0 \end{cases}$$



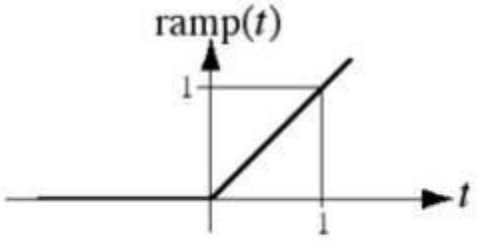
Unit Step

$$u(t) = \begin{cases} 0 \quad \text{for} \quad t < 0 \\ 1 \quad \text{for} \quad t \geq 0 \end{cases}$$



The unit step function, also known as the Heaviside function

Ramp



$$\text{ramp}(t) = \begin{Bmatrix} t & , & t > 0 \\ 0 & , & t \leq 0 \end{Bmatrix} = \int_{-\infty}^{t} u(\lambda)\, d\lambda = t\, u(t)$$

The ramp function is a unary real function, easily computable as the mean of the independent variable and its absolute value.

Exponential signal

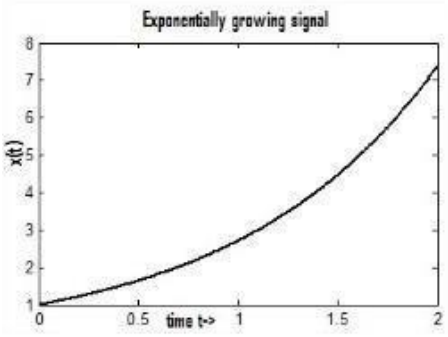
Exponential signal is of two types. These two types of signals are real exponential signal and complex exponential signal which are given below.

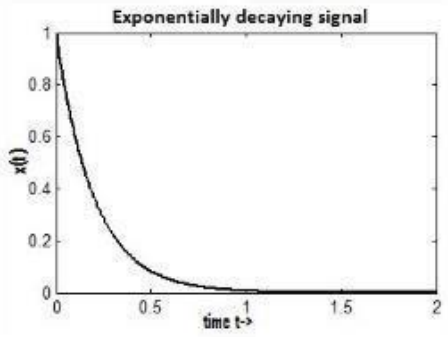
Real Exponential Signal:

A real exponential signal is defined as

$$x(t) = Ae^{\sigma t}$$

Where both *"A"* and *"σ"* are real. Depending on the value of "σ" the signals will be different. If *"σ"* is positive the signal *x(t)* is a growing exponential and if *"σ"* is negative then the signal *x(t)* is a decaying exponential. For σ=0, signal *x(t)* will be constant.





Complex exponential Signal:

The complex exponential signal is given by

$$x(t) = Ae^{st}$$

Where "s" is a complex variable and it is defined as

$$s = \sigma + j\omega$$

A complex exponential signal cannot be plot in a two dimensional (2D) graph, it should be plot in a three dimensional graph.

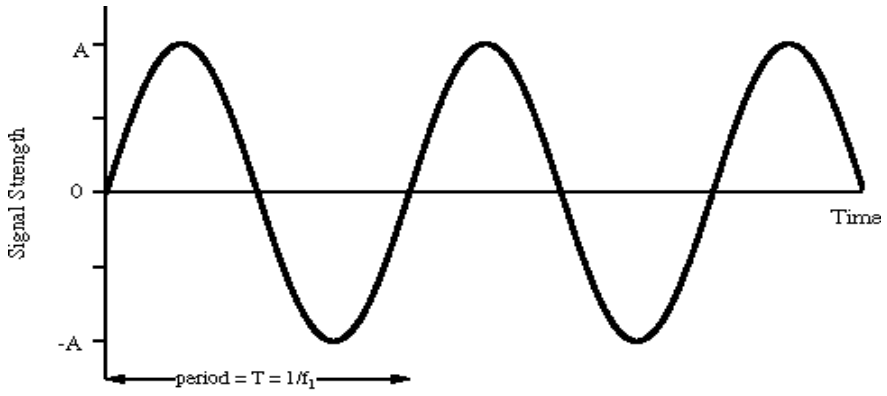
Sinusoidal signal

A sine wave or sinusoid is a mathematical curve that describes a smooth repetitive oscillation.

$$y(t) = A\sin(2\pi ft + \varphi) = A\sin(\omega t + \varphi)$$

where:

- A = the amplitude, the peak deviation of the function from zero.
- f = the ordinary frequency, the number of oscillations (cycles) that occur each second of time.
- ω = 2πf, the angular frequency, the rate of change of the function argument in units of radians per second
- φ = the phase, specifies (in radians) where in its cycle the oscillation is at t = 0.



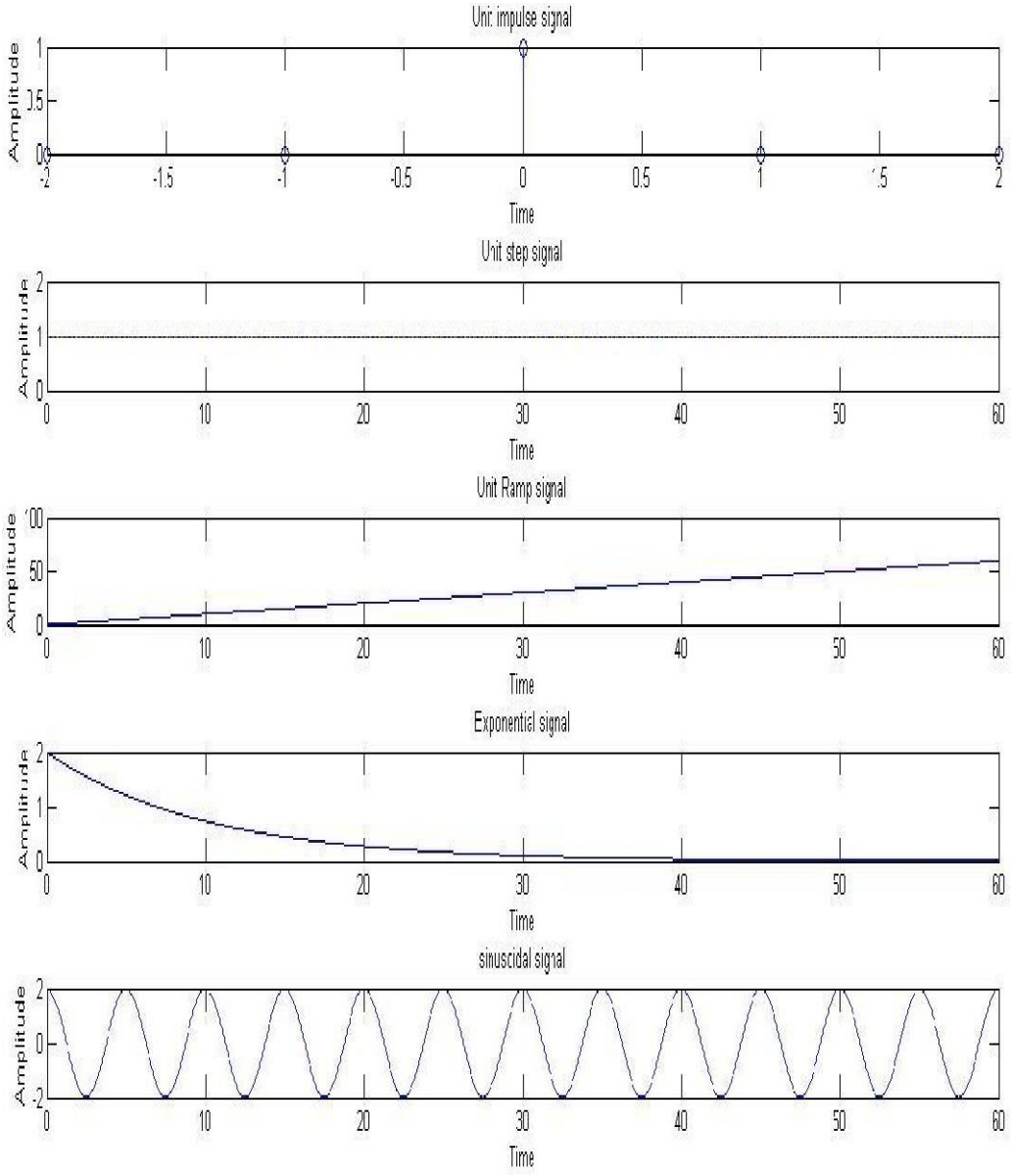
(a) Sine Wave

MATLAB PROGRAM:

```
%(i)unit impulse signal
clc;
clear all;
close all;
n=-2:2;
d=[zeros(1,2),1,zeros(1,2)];
subplot(5,1,1);
stem(n,d);
xlabel('Time');
ylabel('Amplitude');
title('Unit impulse signal');

%(ii)unit step signal
L=input('Length of step signal, L=');
t=0:0.01:L;
u=t/t;
subplot(5,1,2);
plot(t,u);
xlabel('Time');
ylabel('Amplitude');
title('Unit step signal');

%(iii)unit ramp signal
L=input('Length of ramp signal, L=');
t=0:0.01:L;
subplot(5,1,3);
plot(t,t);
xlabel('Time');
ylabel('Amplitude');
title('Unit Ramp signal');

%(iv)exponential signal
L=input('Length of exponential signal, L=');
B=input('Amplitude of exponential signal, B=');
a=input('Value of a=');
t=0:0.01:L;
x=B*exp(a*t);
subplot(5,1,4);
plot(t,x);
xlabel('Time');
ylabel('Amplitude');
title('Exponential signal');
```

```
%(v)sinusoidal signal
L=input('Length of sinusoidal signal, L=');
T=input('Time period of sinusoidal signal, T=');
A=input('Amplitude of sinusoidal signal, A=');
phi=input('Initial phase of sinusoidal signal, phi(in radian)=');
t=0:0.01:L;
x=A*cos((2*pi*t/T)+phi);
subplot(5,1,5);
plot(t,x);
xlabel('Time');
ylabel('Amplitude');
title('sinusoidal signal');
```

OUTPUT :


Unit impulse signal
Amplitude
Time
Unit step signal
Amplitude
Time
Unit Ramp signal
Amplitude
Time
Exponentiel signal
Amplitude
Time
sinuscidal signal
Amplitude
Time

Program 4: Generation of basic discrete time signals namely unit impulse, step, ramp, exponential and Sinusoidal signals.

Aim: To generate basic discrete time signals namely unit impulse, step, ramp, exponential and Sinusoidal signals.

Software Required: MATLAB

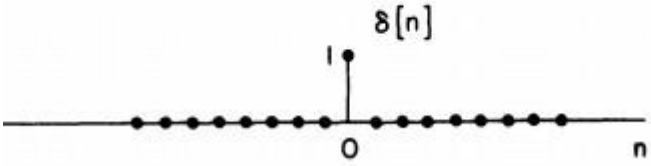
Theory:

Signals that are discrete in time but continuous in amplitude are referred to as discrete-time signals.

Unit Impulse

An ideal impulse function is a function that is zero everywhere but at the origin, where it is infinitely high. However, the *area* of the impulse is finite.

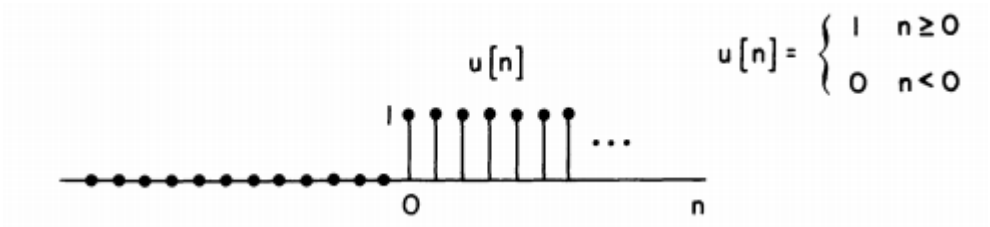
$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$



Unit Step

The unit step function, also known as the Heaviside function, is defined as

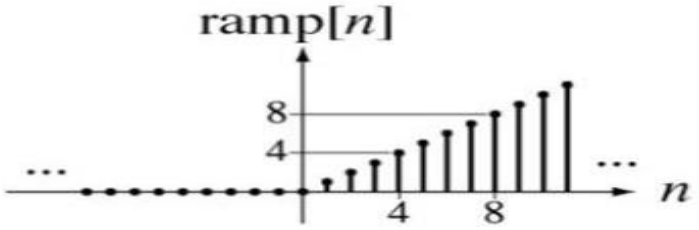
$$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$



Ramp

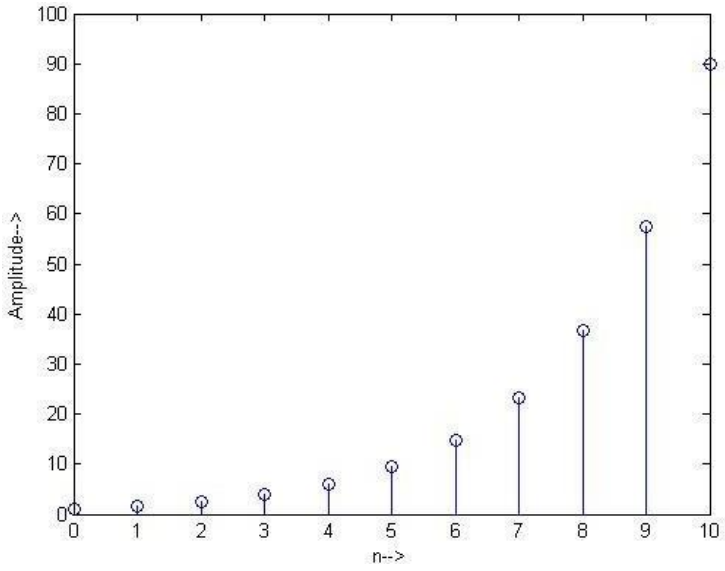
The ramp function is a unary real function, easily computable as the mean of the independent variable and its absolute value.

$$\text{ramp}[n] = \begin{Bmatrix} n & , & n \geq 0 \\ 0 & , & n < 0 \end{Bmatrix} = \sum_{m=-\infty}^{n} u[m-1]$$



Exponential sequence

Exponential signal is represented by :



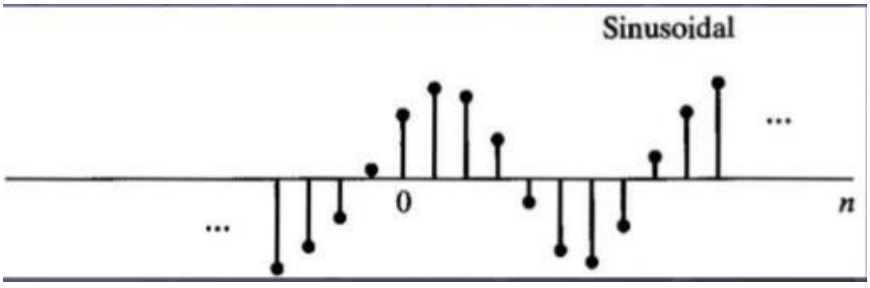
Sinusoidal sequence

A sine wave or sinusoid is a mathematical curve that describes a smooth repetitive oscillation.

$$x[n] = A\cos(w\,n + \emptyset)$$

where:

- A = the amplitude, the peak deviation of the function from zero.
- f = the ordinary frequency, the number of oscillations (cycles) that occur each second of time.
- ω = 2πf, the angular frequency, the rate of change of the function argument in units of radians per second
- Φ = the phase, specifies (in radians) where in its cycle the oscillation is at t = 0.

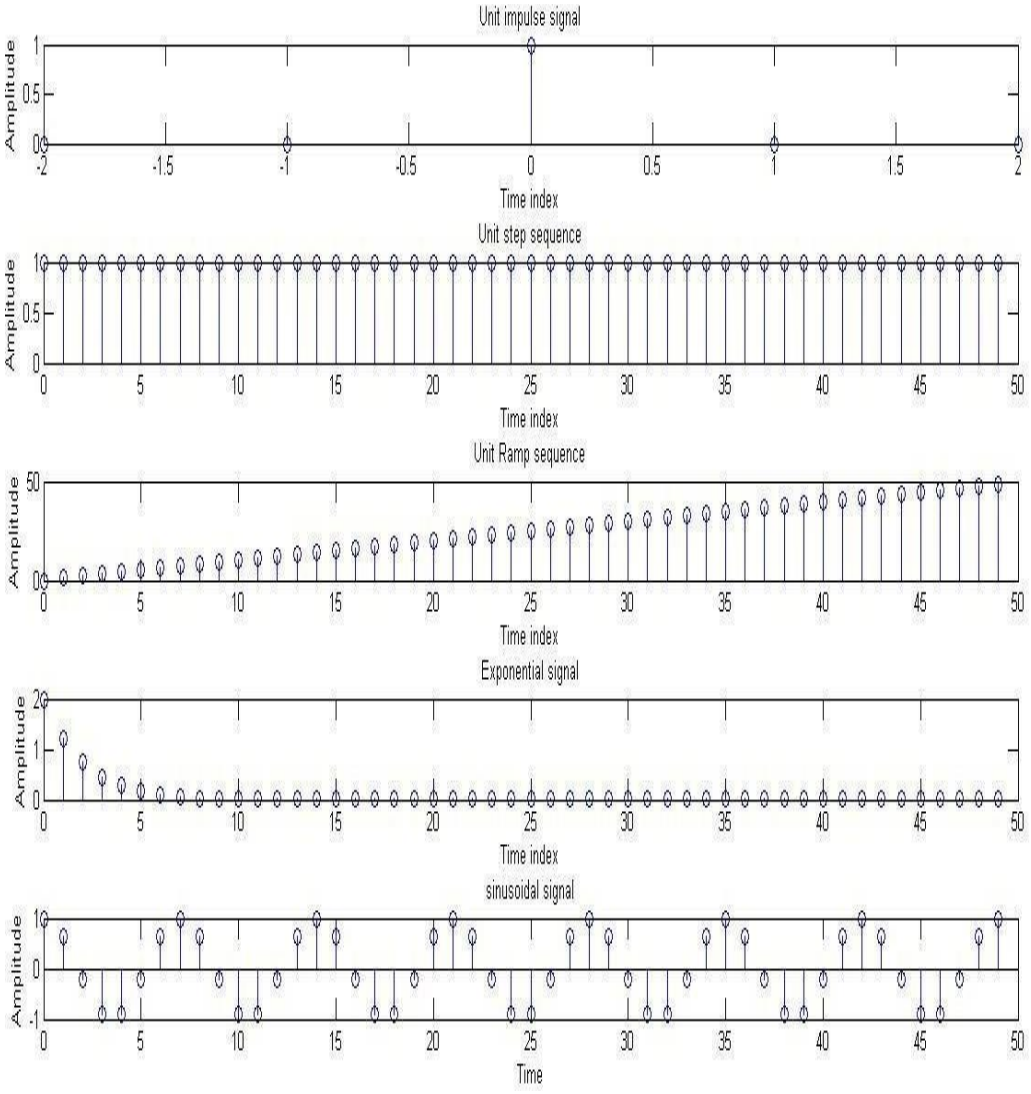


MATLAB PROGRAM:

```
%(i)unit impulse sequence
clc;
clear all;
close all;
n=-2:2;
d=[zeros(1,2),1,zeros(1,2)];
subplot(5,1,1);
stem(n,d);
xlabel('Time index');
ylabel('Amplitude');
title('Unit impulse sequence ');

%(ii)unit step sequence
L=input('Length of step sequence, L=');
n=0:L-1;
u=ones(1,L);
subplot(5,1,2);
stem(n,u);
xlabel('Time index');
ylabel('Amplitude');
title('Unit step sequence');

%(iii)unit ramp sequence
L=input('Length of ramp sequence, L=');
n=0:L-1;
subplot(5,1,3);
stem(n,n);
xlabel('Time index');
ylabel('Amplitude');
title('Unit Ramp sequence');

%(iv)exponential sequence
L=input('Length of exponential sequence, L=');
n=0:L-1;
B=input('Amplitude of exponential sequence, B=');
a=input('Value of a=');
e=exp(a*n);
x=B*e;
subplot(5,1,4);
stem(n,x);
xlabel('Time index');
ylabel('Amplitude');
```

```
title('Exponential sequence ');

%(v)sinusoidal sequence
L=input('Length of sinusoidal sequence, L=');
N=input('Time period of sinusoidal sequence , N=');
A=input('Amplitude of sinusoidal sequence , A=');
phi=input('Initial phase of sinusoidal sequence, phi(in radian)=');
n=0:L-1;
x=A*cos((2*pi*n/N)+phi);
subplot(5,1,5);
stem(n,x);
xlabel('Time');
ylabel('Amplitude');
title('sinusoidal sequence');
```

OUTPUT:



Program 5: Operations on Signals and Sequences such as addition, Multiplication, Scaling, Shifting, Folding, Computation of Energy and Average Power.

Aim: Matlab programs to explain the Operations on Signals and Sequences suchas Addition, Multiplication, Scaling, Shifting, Folding, Computation of Energy and Average Power.

Software Required: MATLAB

Theory:

1. Addition: Addition can be carried out using the ' + ' symbol and plotting will give youthe result.
2. Subtraction: With the same program code as of addition replacing arithmeticoperator ' – ' we can perform subtraction in signals.
3. Multiplication: By using ' * ' (asterisk) operator we can perform multiplicationof signals.
4. Reversing a Signal: The inbuilt function **fliplr()** function can be used to performreversing or folding of a signal.

MATLAB PROGRAM:

```
clc;
clear all;
close all;

t=0:.01:1;

% generating two input signals
x1=sin(2*pi*4*t);
x2=sin(2*pi*8*t);
subplot(2,2,1);
plot(t,x1);
xlabel('time'); ylabel('amplitude');
title('signal1:sine wave of frequency 4Hz');
subplot(2,2,2);
plot(t,x2);
xlabel('time');
ylabel('amplitude');
title('signal2:sine wave of frequency 8Hz');
```

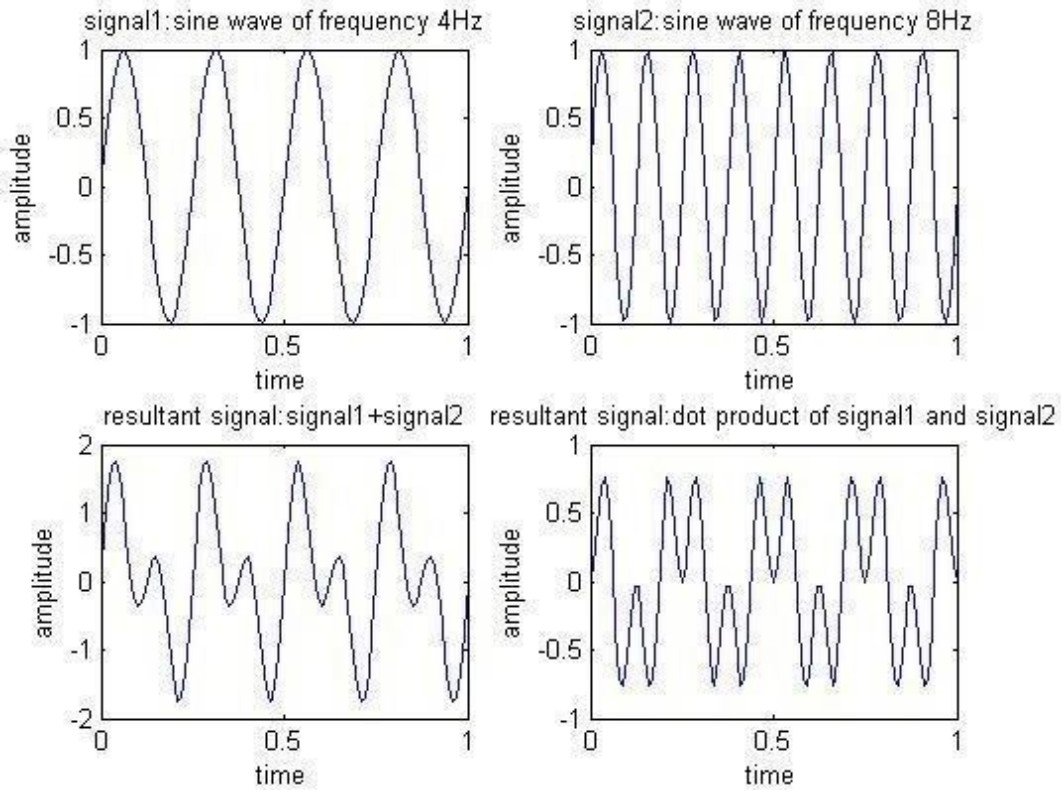
```
% addition of signals
y1=x1+x2;
subplot(2,2,3);
plot(t,y1);
xlabel('time');
ylabel('amplitude');
title('resultant signal:signal1+signal2');

% multiplication of signals
 y2=x1.*x2;
subplot(2,2,4);
plot(t,y2);
xlabel('time');
ylabel('amplitude');
title('resultant signal:dot product of signal1 and signal2');

% scaling of a signal1
A=10;
y3=A*x1;
figure;
subplot(2,2,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('sine wave of frequency 4Hz')
subplot(2,2,2);
plot(t,y3);
xlabel('time');
ylabel('amplitude');
title('amplified input signal1 ');

% folding of a signal1
l1=length(x1);
nx=0:l1-1;
subplot(2,2,3);
plot(nx,x1);
xlabel('nx');
ylabel('amplitude');
title('sine wave of frequency 4Hz')
y4=fliplr(x1);
nf=-fliplr(nx);
subplot(2,2,4);
plot(nf,y4);
```

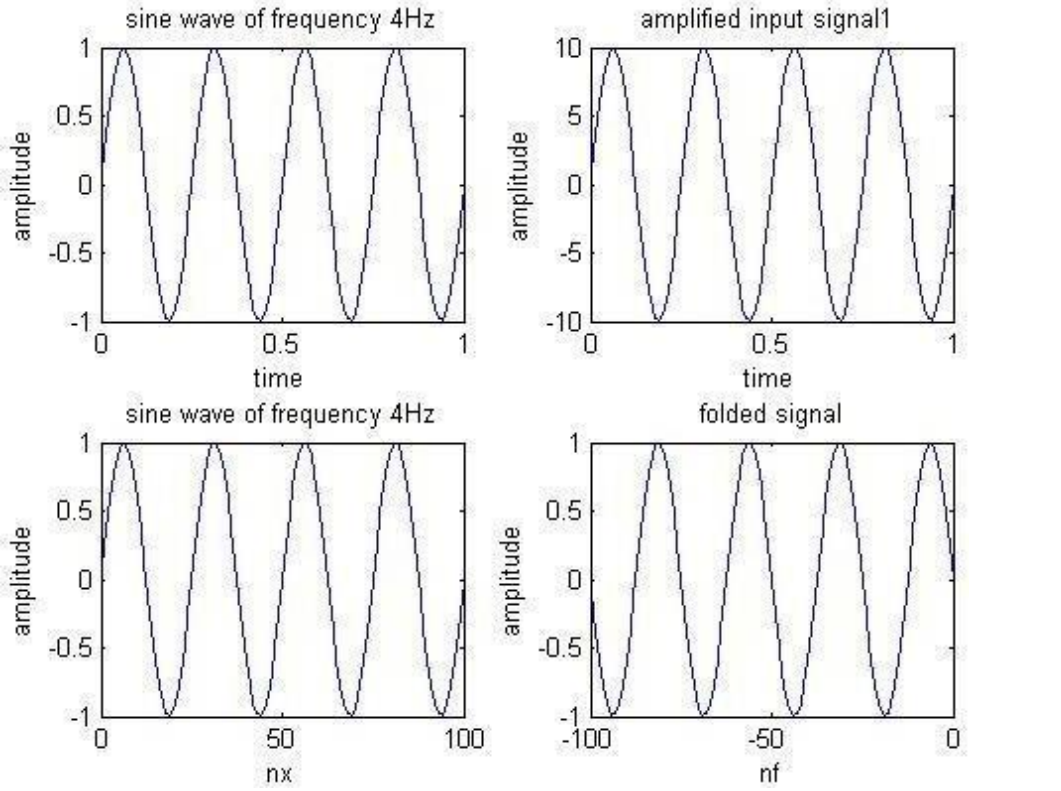
```
xlabel('nf');
ylabel('amplitude');
title('folded signal');

%shifting of a signal
 figure;
t1=0:.01:pi;
x3=8*sin(2*pi*2*t1);
subplot(3,1,1);
plot(t1,x3);
xlabel('time t1');
ylabel('amplitude');
title('sine wave of frequency 2Hz');
subplot(3,1,2);
 plot(t1+10,x3);
xlabel('t1+10');
ylabel('amplitude');
title('right shifted signal');
subplot(3,1,3);
plot(t1-10,x3);
xlabel('t1-10');
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');
n2=-2:1:6;
s2=[1 1 2 4 6 0 5 3 6];
subplot(2,2,2);
stem(n2,s2);
xlabel('n2');
ylabel('amplitude');
title('input sequence2');
```

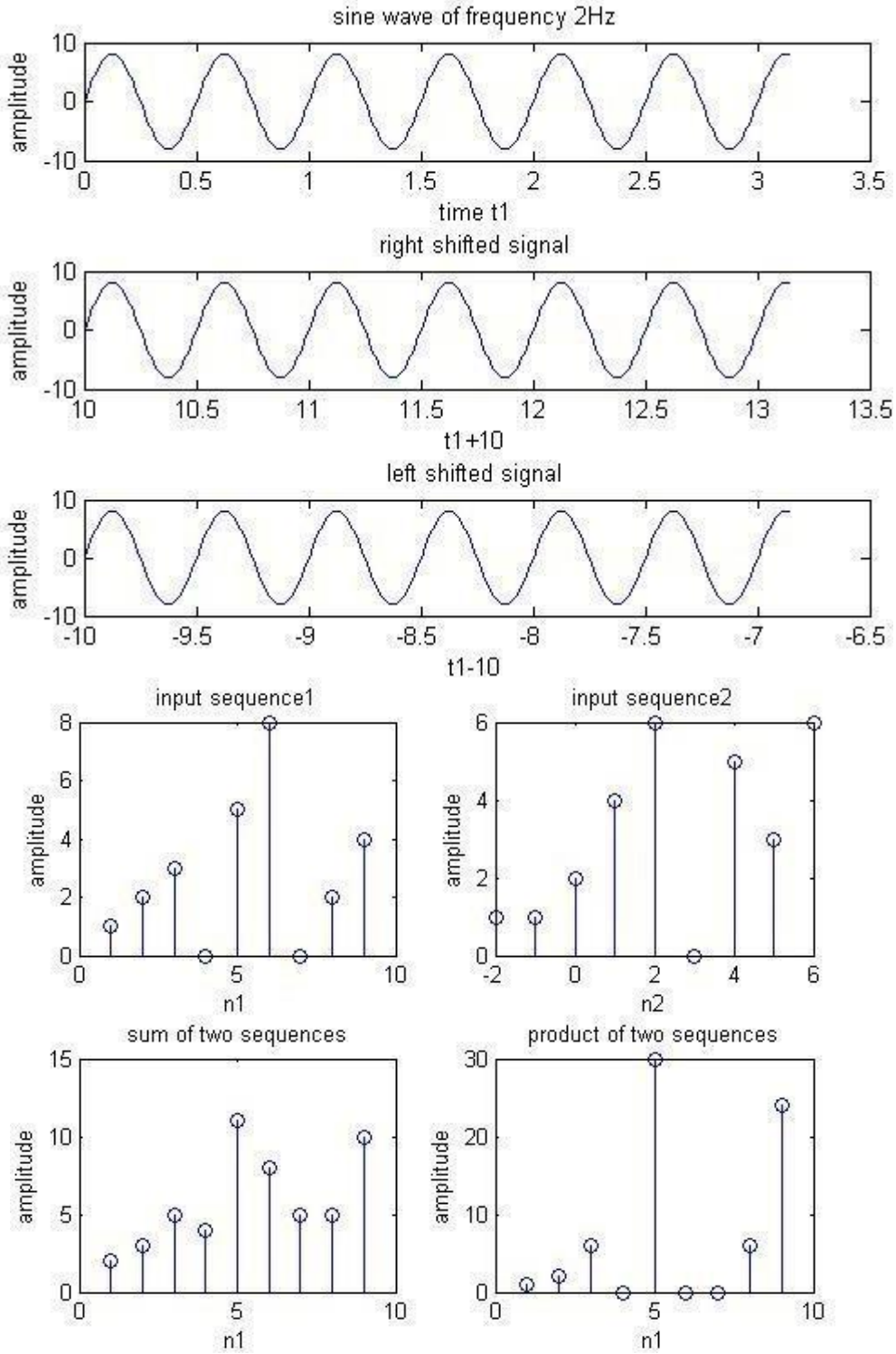
```
% addition of sequences
 s3=s1+s2;
 subplot(2,2,3);
stem(n1,s3);
xlabel('n1');
ylabel('amplitude');
 title('sum of two sequences');

% multiplication of sequences
s4=s1.*s2;
subplot(2,2,4);
stem(n1,s4);
xlabel('n1');
ylabel('amplitude');
title('product of two sequences');


% scaling of a sequence
 figure;
subplot(2,2,1);
stem(n1,s1);
xlabel('n1');
ylabel('amplitude');
title('input sequence1');
s5=4*s1;
subplot(2,2,2);
stem(n1,s5);
xlabel('n1');
ylabel('amplitude');
title('scaled sequence1');

% shifting of a sequence
subplot(2,2,3);
stem(n1-2,s1);
xlabel('n1');
ylabel('amplitude');
title('left shifted sequence1');
subplot(2,2,4);
stem(n1+2,s1);
xlabel('n1');
ylabel('amplitude');
title('right shifted sequence1');
```

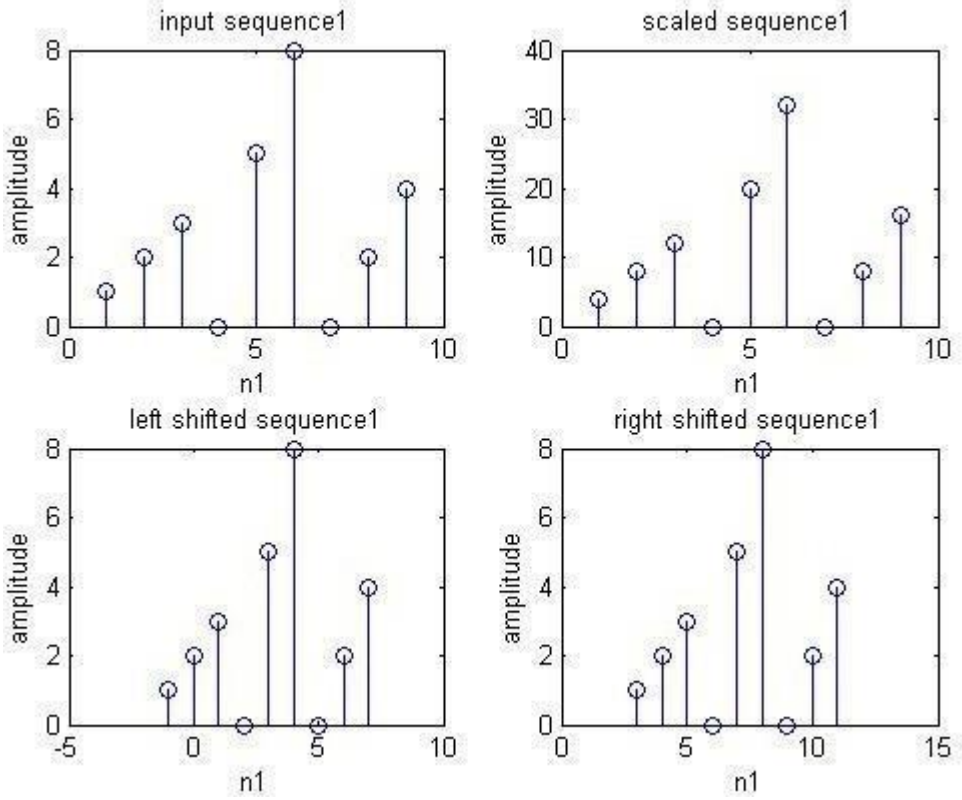
```
% folding of a sequence
l2=length(s1);
nx1=0:l2-1;
figure;
subplot(2,1,1);
stem(nx1,s1);
xlabel('nx1');
ylabel('amplitude');
title('input sequence1');
s6=fliplr(s1);
nf2=-fliplr(nx1);
subplot(2,1,2);
stem(nf2,s6);
xlabel('nf2');
ylabel('amplitude');
title('folded sequence1');


% program for energy of a sequence
z1=input('enter the input sequence');
e1=sum(abs(z1).^2);
e1

% program for energy of a signal
 t=0:pi:10*pi;
z2=cos(2*pi*50*t).^2;
e2=sum(abs(z2).^2);
 e2

% program for power of a sequence
p1= (sum(abs(z1).^2))/length(z1);
p1

% program for power of a signal
p2=(sum(abs(z2).^2))/length(z2);
p2
```

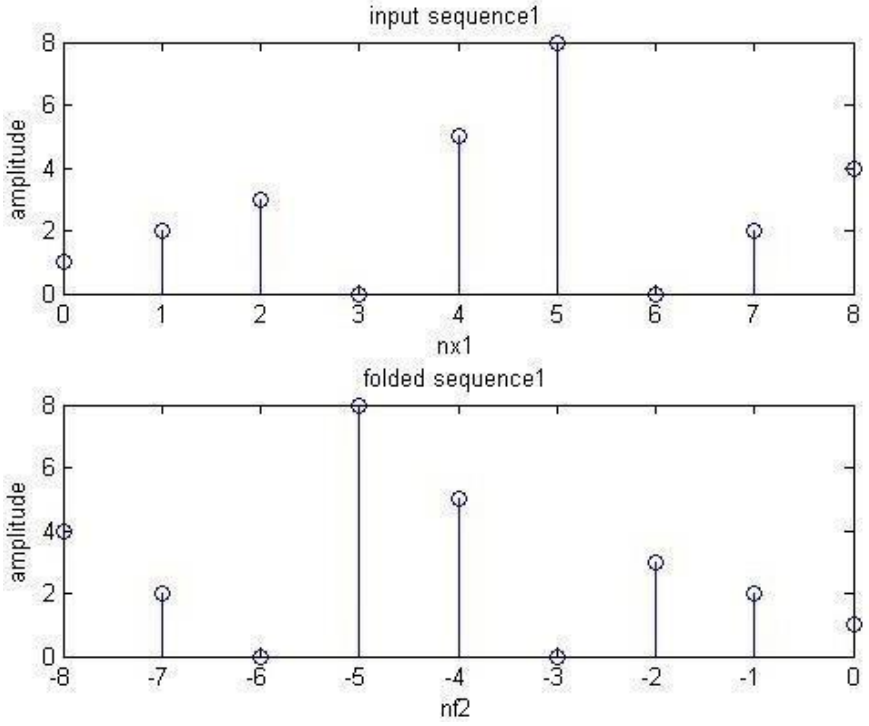
OUTPUT :


signal1:sine wave of frequency 4Hz
signal2:sine wave of frequency 8Hz
resultant signal:signal1+signal2
resultant signal:dot product of signal1 and signal2
amplitude
time


sine wave of frequency 4Hz
amplified input signal1
sine wave of frequency 4Hz
folded signal
amplitude
time
nx
nf


sine wave of frequency 2Hz
amplitude
time t1
right shifted signal
t1+10
left shifted signal
t1-10
input sequence1
n1
input sequence2
n2
sum of two sequences
product of two sequences


input sequence1
amplitude
n1
scaled sequence1
amplitude
n1
left shifted sequence1
amplitude
n1
right shifted sequence1
amplitude
n1


input sequence1
amplitude
nx1
folded sequence1
amplitude
nf2

Program 6: Finding the Even and Odd Parts of Signal or Sequence and Real and Imaginary Parts of Signal.

Aim: Matlab programs to Finding the Even and Odd Parts of Signal or Sequence and Realand Imaginary Parts of Signal.

Software Required: MATLAB

Theory:

A signal x(t) is said to be, even if, $x(t)=x(-t)$ (1)

Odd if, $x(t)=-x(-t)$ (2)

Any signal x(t) can be written as the sum of an even signal and odd signal.

$$x(t)=x_e(t)+x_o(t)$$

where, $x_e(t)$ is the even part and $x_o(t)$ is the odd part.

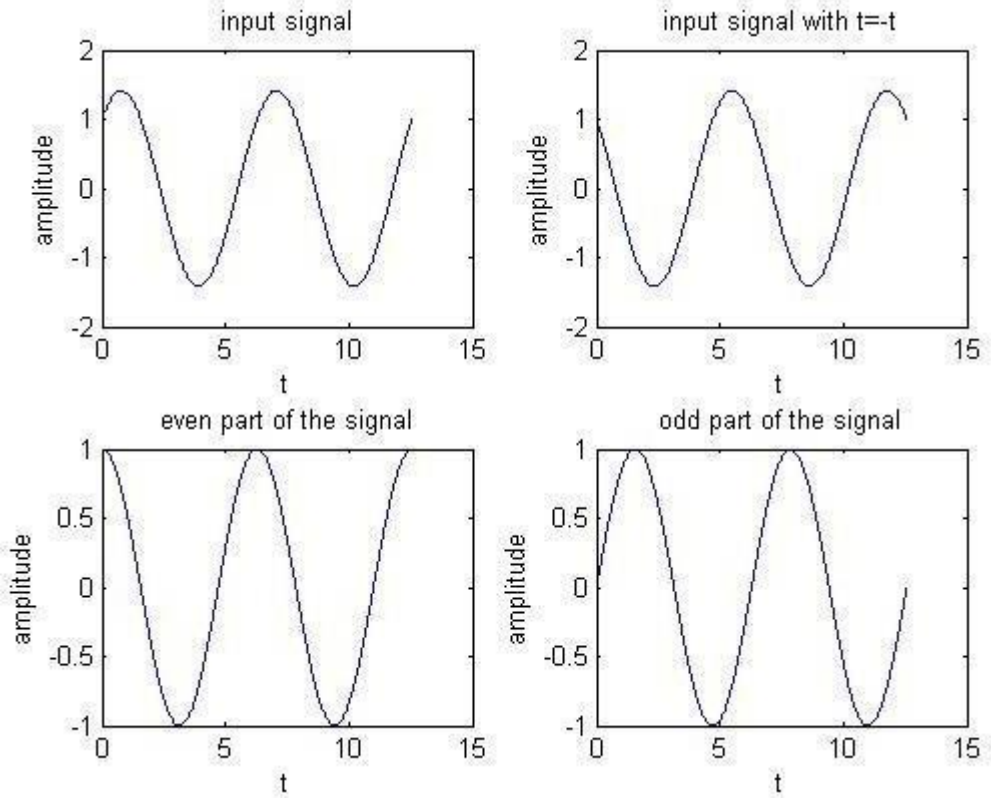
MATLAB PROGRAM:

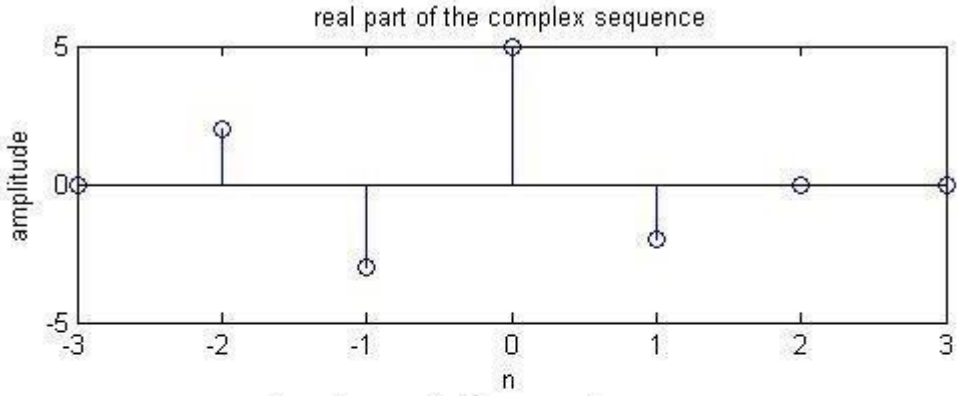
```
clc;
close all;
clear all;

% Even and odd parts of a signal
t=0:.005:4*pi;
x=sin(t)+cos(t); % x(t)=sint(t)+cos(t)
subplot(2,2,1)
plot(t,x)
xlabel('t');
ylabel('amplitude')
title('input signal')
y=sin(-t)+cos(-t) % y=x(-t)
subplot(2,2,2)
plot(t,y)
xlabel('t');
ylabel('amplitude')
title('input signal with t=-t')
z=x+y
subplot(2,2,3)
plot(t,z/2)
xlabel('t');
```

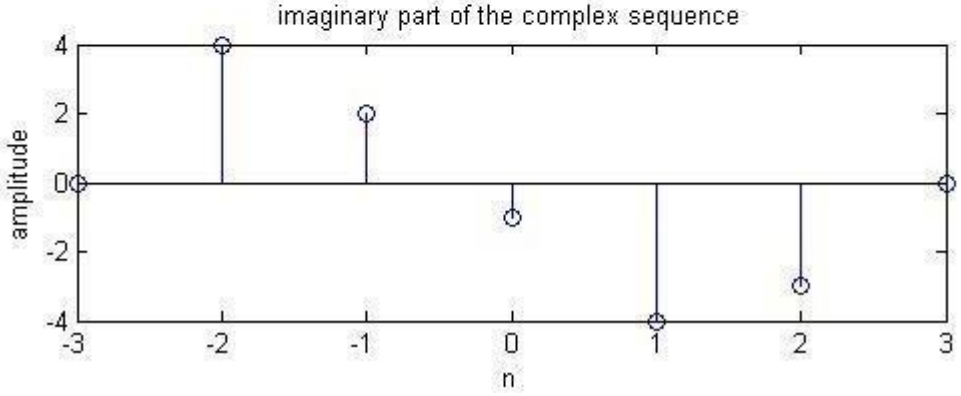
```
ylabel('amplitude')
title('even part of the signal')
p=x-y
subplot(2,2,4)
plot(t,p/2)
xlabel('t');
ylabel('amplitude');
title('odd part of the signal');

% Even and odd parts of a sequence
z=[0,2+j*4,-3+j*2,5-j*1,-2-j*4,-j*3,0];
n=-3:3;

% plotting real and imaginary parts of the sequence
figure;
subplot( 2,1,1);
stem(n,real(z));
xlabel('n');
ylabel('amplitude');
title('real part of the complex sequence');
subplot( 2,1,2);
stem(n,imag(z));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the complex sequence');

% complex conjugate of a signal
zc=conj(z);
zc_folded= fliplr(zc);
zc_even=.5*(z+zc_folded);
zc_odd=.5*(z-zc_folded);

% plotting even and odd parts of the sequence
figure;
subplot( 2,2,1);
stem(n,real(zc_even));
xlabel('n');
ylabel('amplitude');
title('real part of the even sequence');
subplot( 2,2,2);
stem(n,imag(zc_even));
xlabel('n');
ylabel('amplitude');
```

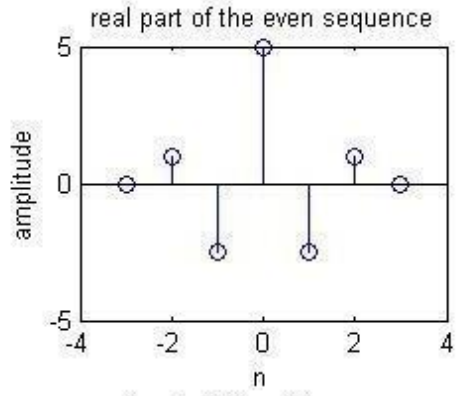
```
title('imaginary part of the even sequence');
subplot( 2,2,3);
stem(n,real(zc_odd));
xlabel('n');
ylabel('amplitude');
title('real part of the odd sequence');
subplot( 2,2,4);
stem(n,imag(zc_odd));
xlabel('n');
ylabel('amplitude');
title('imaginary part of the odd sequence');
```

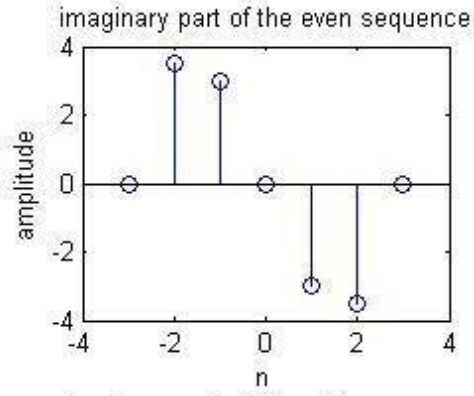
OUTPUT:

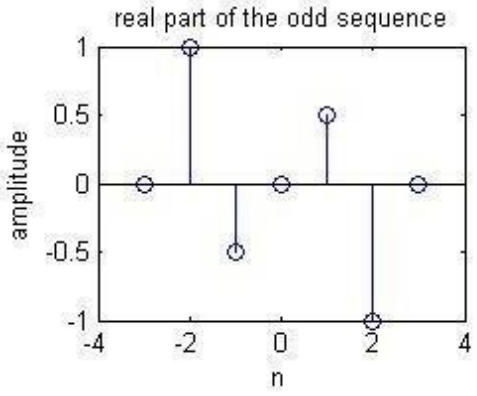


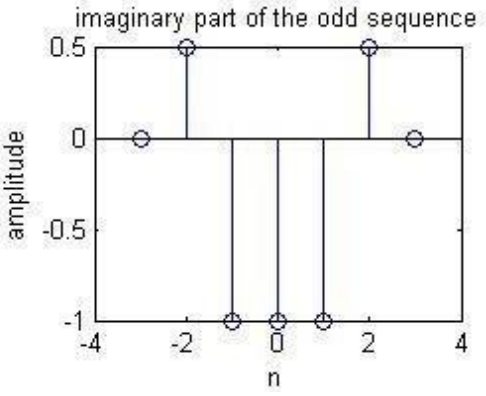

real part of the complex sequence
amplitude
n


imaginary part of the complex sequence
amplitude
n


real part of the even sequence
amplitude
n


imaginary part of the even sequence
amplitude
n


real part of the odd sequence
amplitude
n


imaginary part of the odd sequence
amplitude
n

Program 7: Verification of linearity and time invariance properties of a given Continuous /discrete system.

Aim: Matlab programs to Verify linearity and time invariance properties of a givenContinuous /discrete system.

Software Required: MATLAB

Theory:

If a system is linear, this means that when an input to a given system is scaled by a value, the output of the system is scaled by the same amount. A linear system also obeys the principle of superposition. This means that if two inputs are added together and passed through a linear system, the output will be the sum of the individual inputs' outputs.

A time-invariant system has the property that a certain input will always give the same output (up to timing), without regard to when the input was applied to the system.

MATLAB PROGRAM:

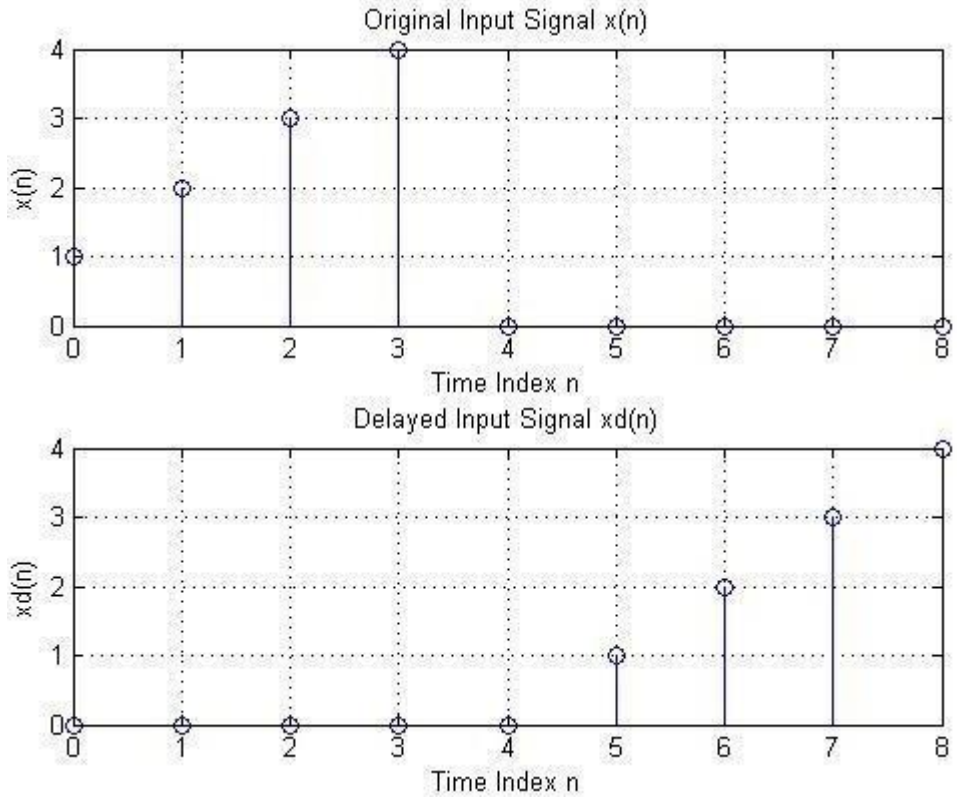
```
% Verification of Linearity

clc;
clear all;
close all;

% entering two input sequences and impulse sequence
x1 = input (' type the samples of x1 ');
x2 = input (' type the samples of x2 ');
if(length(x1)~=length(x2))
disp('error: Lengths of x1 and x2 are different');
return;
end;
h = input (' type the samples of h ');

% length of output sequence
N = length(x1) + length(h) -1;
disp('length of the output signal will be ');
disp(N);

% entering scaling factors
a1 = input (' The scale factor a1 is ');
a2 = input (' The scale factor a2 is ');
x = a1 * x1 + a2 * x2;
```

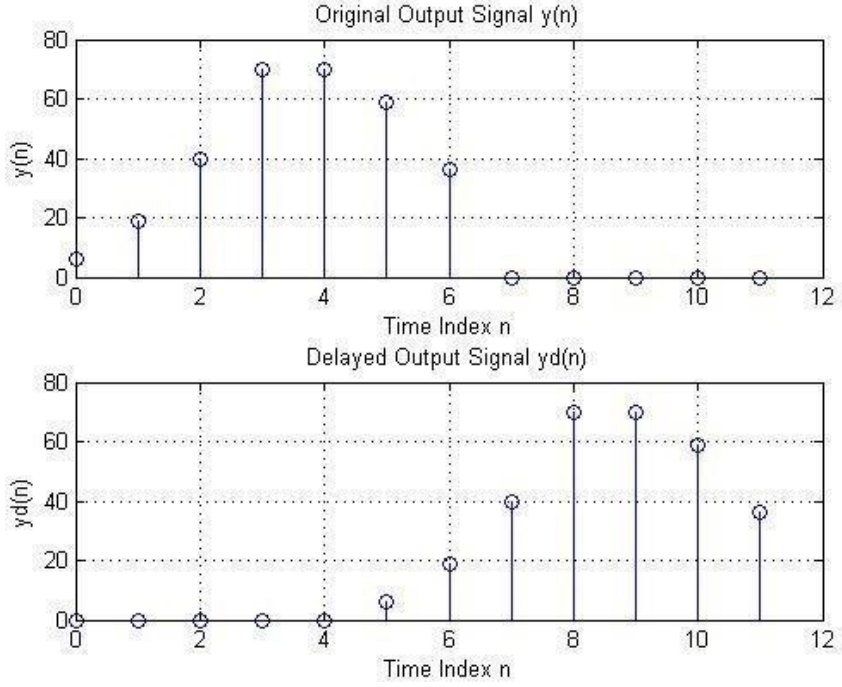
```
% response of x and x1
yo1 = conv(x,h);
y1 = conv(x1,h);

% scaled response of x1
y1s = a1 * y1;

% response of x2
y2 = conv(x2,h);

% scaled response of x2
y2s = a2 * y2;
yo2 = y1s + y2s;
disp ('Input signal x1 is ');
 disp(x1);
disp ('Input signal x2 is ');
 disp(x2);
disp ('Output Sequence yo1 is ');
disp(yo1);
disp ('Output Sequence yo2 is ');
 disp(yo2);
if ( yo1 == yo2 )
disp(' yo1 = yo2. Hence the LTI system is LINEAR ')
end;


% verification of time invariance

clc;
clear all;
close all;

% entering two input sequences
x = input( ' Type the samples of signal x(n) ' );
h = input( ' Type the samples of signal h(n) ' );

% original response
y = conv(x,h);
disp( ' Enter a POSITIVE number for delay ' );
d = input( ' Desired delay of the signal is ' );
```

```
% delayed input
xd = [zeros(1,d), x];
nxd = 0 : length(xd)-1;

%delayed output
yd = conv(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)];
subplot(2,1,1);
stem(nxd,xp);
grid;
xlabel( ' Time Index n ' );
ylabel( ' x(n) ' );
title( ' Original Input Signal x(n) ' );
subplot(2,1,2);
stem(nxd,xd);
grid;
xlabel( ' Time Index n ' );
ylabel( ' xd(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:

For Linearity the output will be displayed as Linear or Non Linear System.





Program 8: Convolution between Signals and Sequences.

Aim: Matlab programs to perform Convolution between Signals and Sequences.

Software Required: MATLAB

Theory:

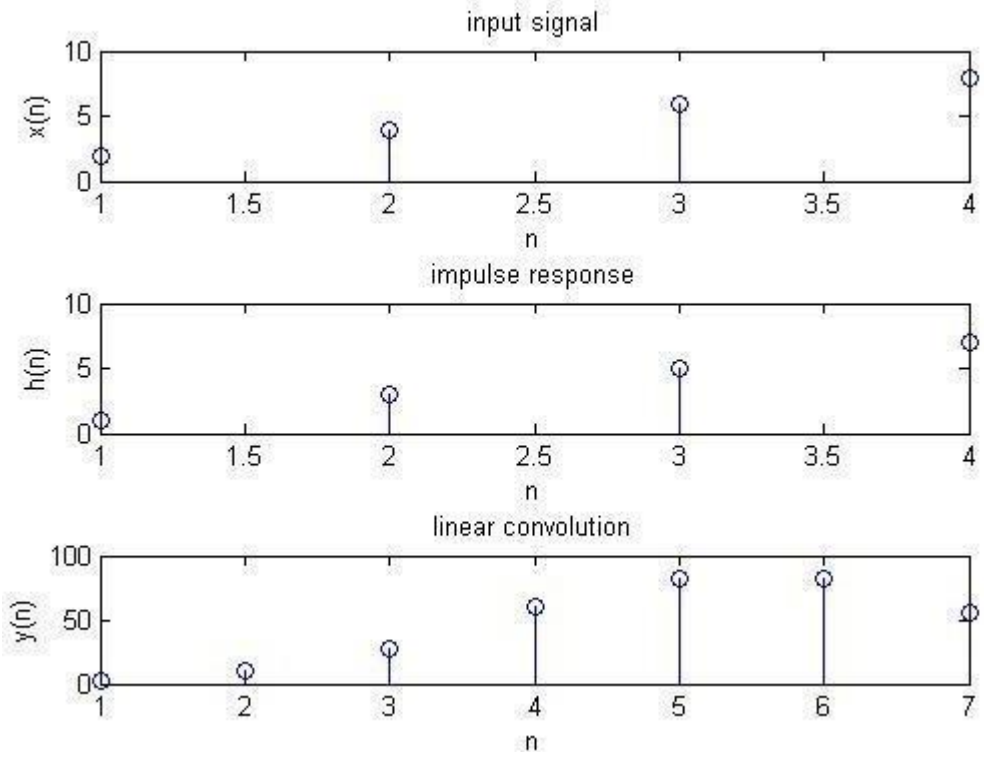
Convolution is a mathematical operation on two functions (*f* and *g*) that produces a third function expressing how the shape of one is modified by theother. The term *convolution* refers to both the result function and to the process of computing it. It is defined as the integral of the product of the two functions afterone is reversed and shifted.

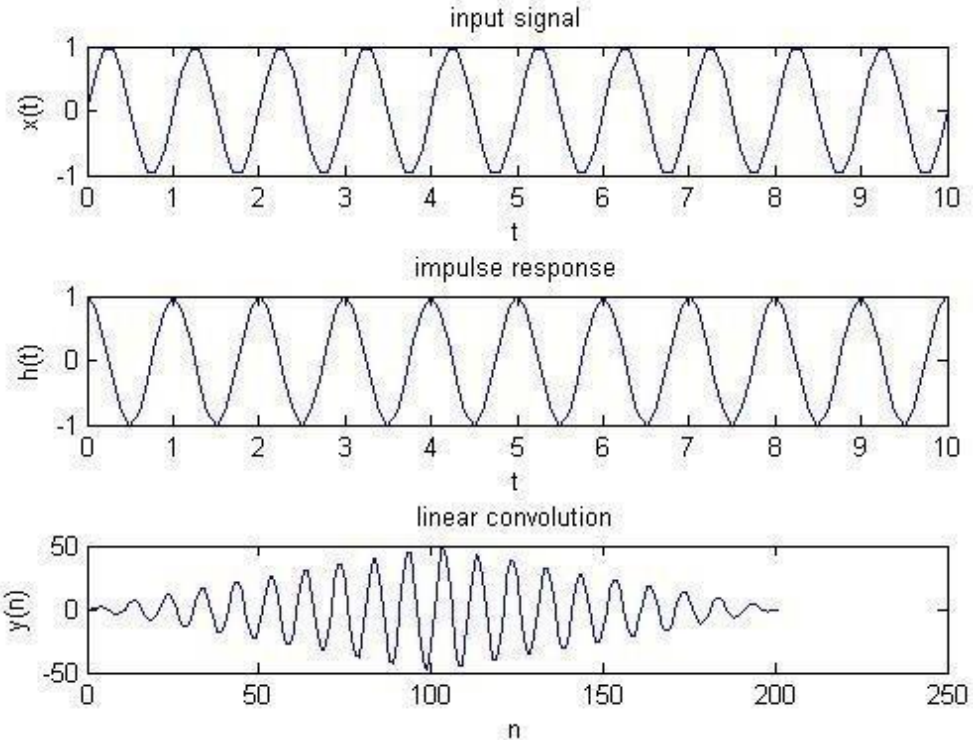
MATLAB PROGRAM:

```
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 signal')
subplot(3,1,2);
stem(h);
xlabel('n');
ylabel('h(n)');
title('impulse response')
subplot(3,1,3);
stem(y);
xlabel('n');
ylabel('y(n)');
title('linear convolution')
disp('The resultant signal is');
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(t,x1);
xlabel('t');
ylabel('x(t)');
title('input signal')
subplot(3,1,2);
plot(t,h1);
xlabel('t');
ylabel('h(t)');
title('impulse response')
subplot(3,1,3);
plot(y1);
xlabel('n');
ylabel('y(n)');
title('linear convolution');
```

OUTPUT:




input signal
x(t)
t
impulse response
h(t)
t
linear convolution
y(n)
n

Program 9: Autocorrelation and Cross correlation between Signals and Sequences.

Aim: Matlab programs to perform Autocorrelation and Cross correlation between Signals and Sequences.

Software Required: MATLAB

Theory:

Cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. This is also known as a *sliding dot product* or *sliding inner-product*. It is commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology. The cross-correlation is similar in nature to the convolution of two functions. In an autocorrelation, which is the cross-correlation of a signal with itself, there will always be a peak at a lag of zero, and its size will be the signal energy.

Autocorrelation, also known as serial correlation, is the correlation of a signal with adelayed copy of itself as a function of delay. Informally, it is the similarity between observations as a function of the time lag between them. The analysis of autocorrelation is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise, or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies. It is often used in signal processing for analyzing functions or series of values, such as time domain signals.

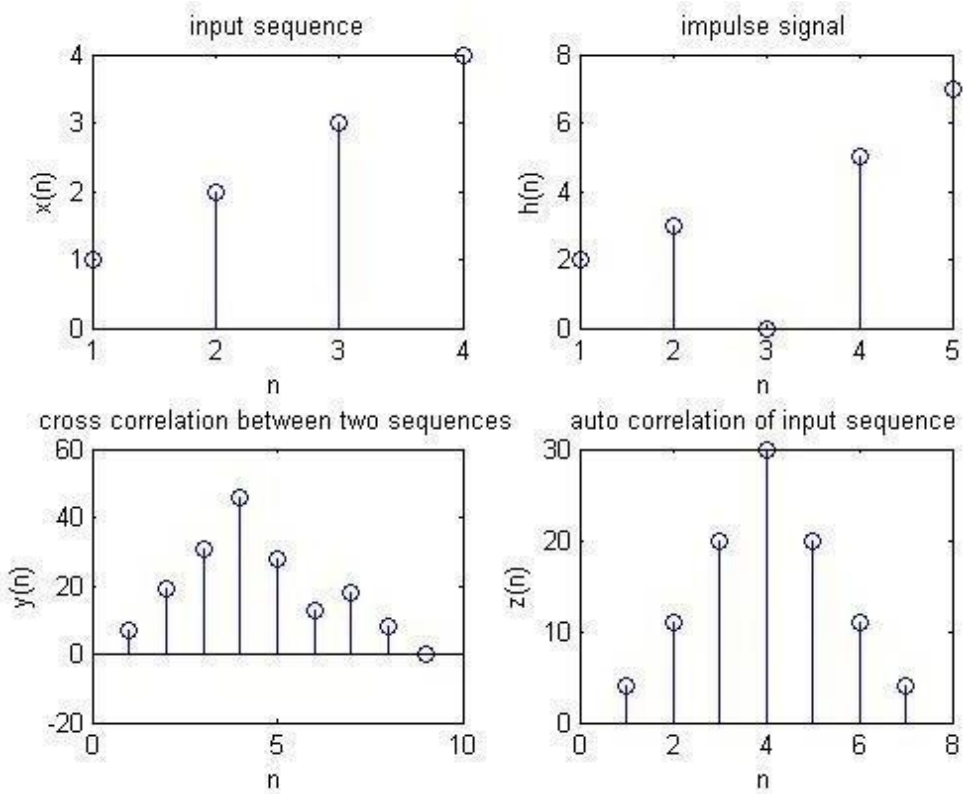
MATLAB PROGRAM:

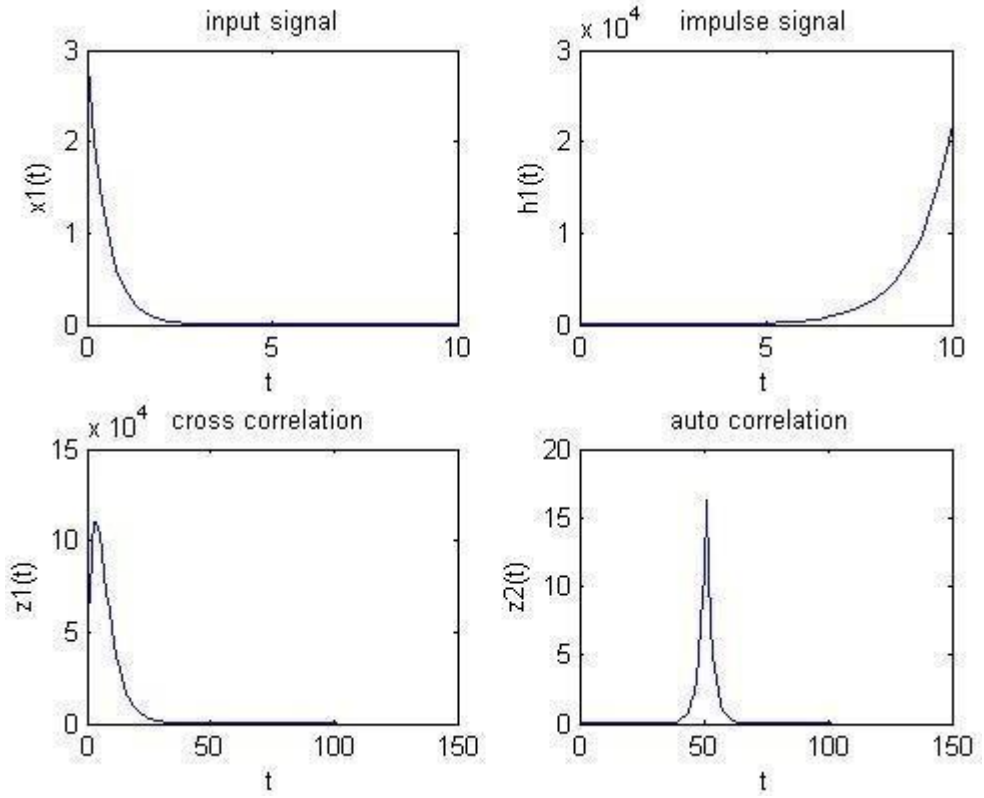
```
clc;
close all;
clear all;
% two input sequences
x=input('enter input sequence');
h=input('enter the impulse suquence');
subplot(2,2,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input sequence');
subplot(2,2,2);
stem(h);
xlabel('n');
```

```
ylabel('h(n)');
title('impulse signal');

% cross correlation between two sequences
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 :




input signal
x1(t)
t
x 10^4
impulse signal
h1(t)
t
x 10^4
cross correlation
z1(t)
t
auto correlation
z2(t)
t

Program 10: Sampling Theorem Verification.

Aim: Matlab program to verify Sampling Theorem.

Software Required: MATLAB

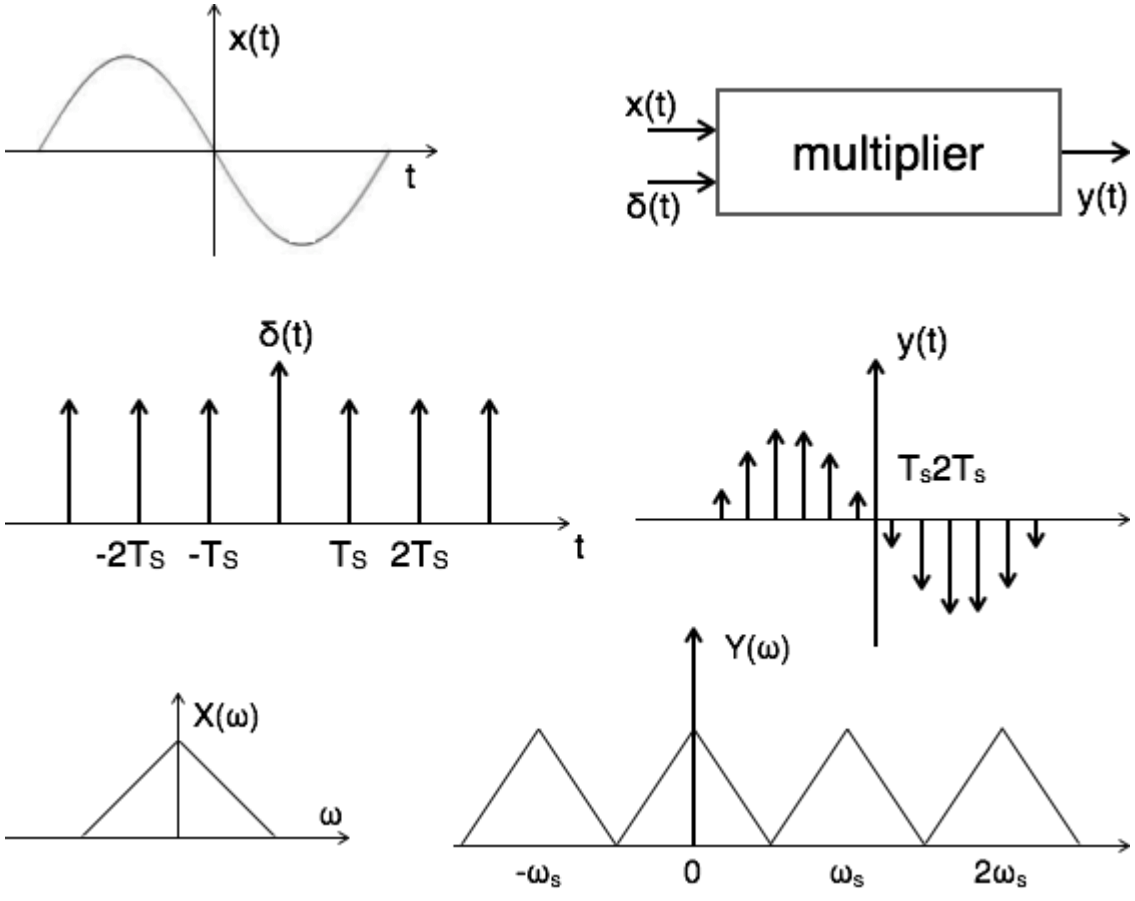
Theory:

A continuous time signal can be represented in its samples and can be recovered back when sampling frequency f_s is greater than or equal to the twice the highest frequency component of message signal. i. e.

$$fs \geq 2fm.$$

Proof: Consider a continuous time signal x(t). The spectrum of x(t) is a band limited to f_m Hz i.e. the spectrum of x(t) is zero for $|\omega| > \omega_m$.

Sampling of input signal x(t) can be obtained by multiplying x(t) with an impulse train $\delta(t)$ of period T_s. The output of multiplier is a discrete signal called sampled signal which is represented with y(t) in the following diagrams:



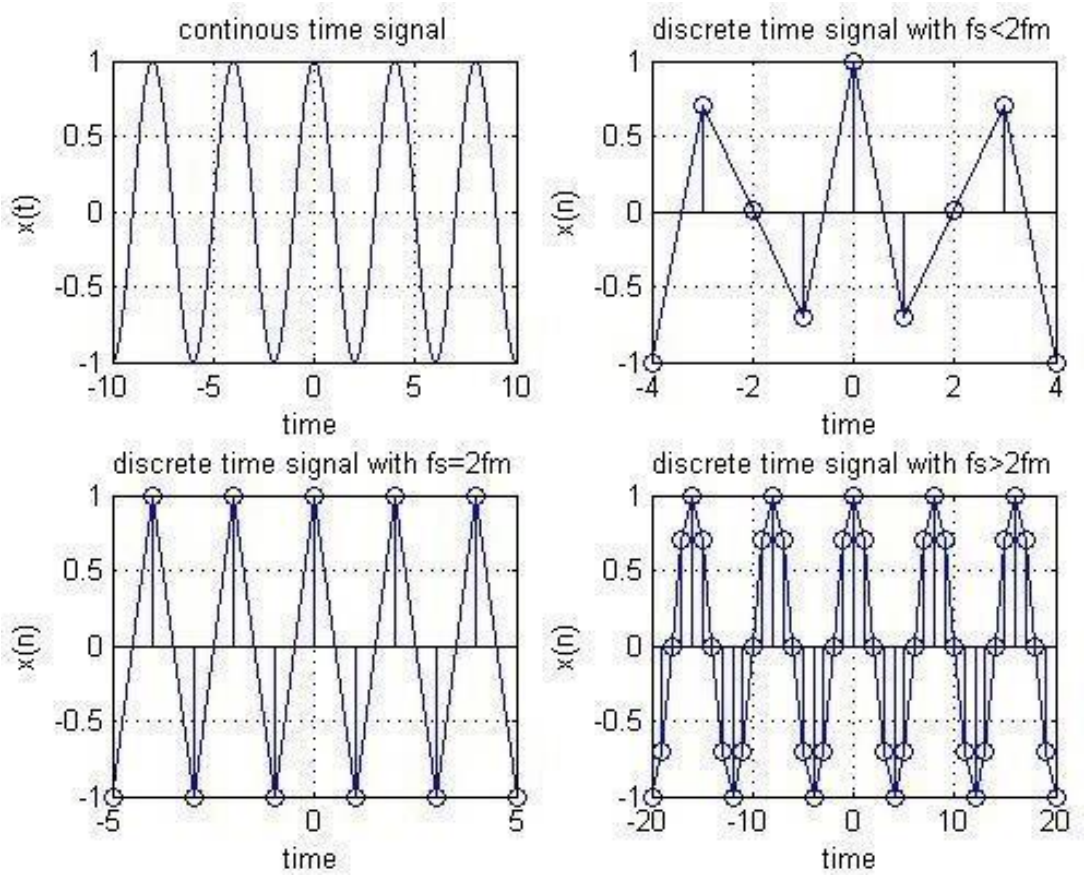
Here, you can observe that the sampled signal takes the period of impulse.

MATLAB 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:



REFERENCES

1. B.P Lathi, "Linear Systems and Signals", Oxford University, 2nd edition ,2008.
2. Micheal J Roberts, "Fundamentals of Signals and Systems", Tata Mc Graw-Hill, 4th edition, 2009.
3. Alan V. Oppenheim, Alan S. Willsky with S Hamid Nawab, "Signals and Systems", Pearson edition, 2003.
4. Rudra Pratap ," Getting Started with MATLAB: A Quick Introduction for Scientists and Engineers", Oxford University Press, 5th edition, 2016.
5. Luis F. Chaparro, Aydin Akan "Signals and Systems using MATLAB", Springer, 2nd edition ,2016.
6. Alex Palamides, Anastasia Veloni "Signals and Systems Laboratory with MATLAB", Springer, 1st edition, 2018.