LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

L.B REDDY NAGAR, MYLAVARAM-521230

SIGNAL MODELING & ANALYSIS (SMA) LAB MANUAL

FOR

B. Tech III SEM (ECE)

COURSE CODE: 20ECS1

DEPARTMENT OF ECE

2021-2022

LIST OF EXPERIMENTS

EXPERIMENT-1

Plot the continuous and discrete signals from the given function f*(t)*

AIM: To Plot the continuous and discrete signals from the given function f*(t)=sin (2π10t + π/6)*

Software Used:MATLAB R2016a

Theory:

Continuous Signal: A signal is considered to be a continuous time signal if it is defined over a continuum of the independent variable

Eg: $x(t)=A\cos(\omega t+\theta)$

Discrete Signal: A discrete-time signal is a sequence of values that correspond to particular instantsin time

Eg: $x(t)=A\cos(\omega n+\theta)$

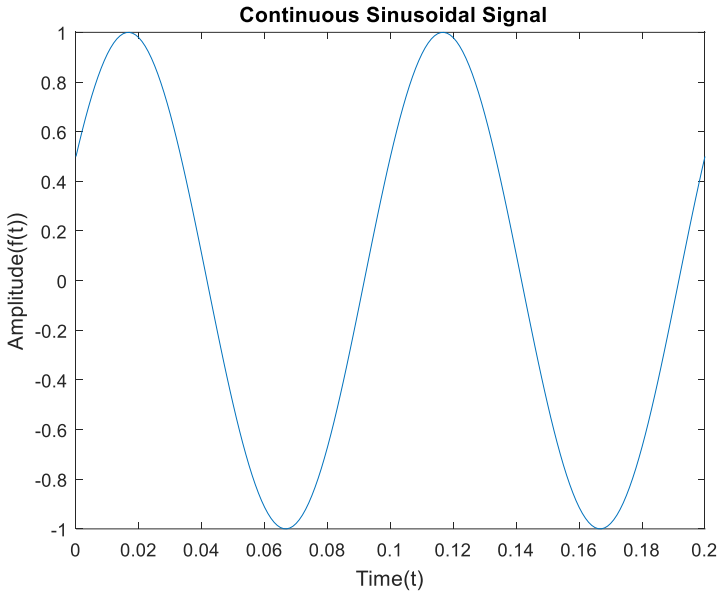
Commands used:

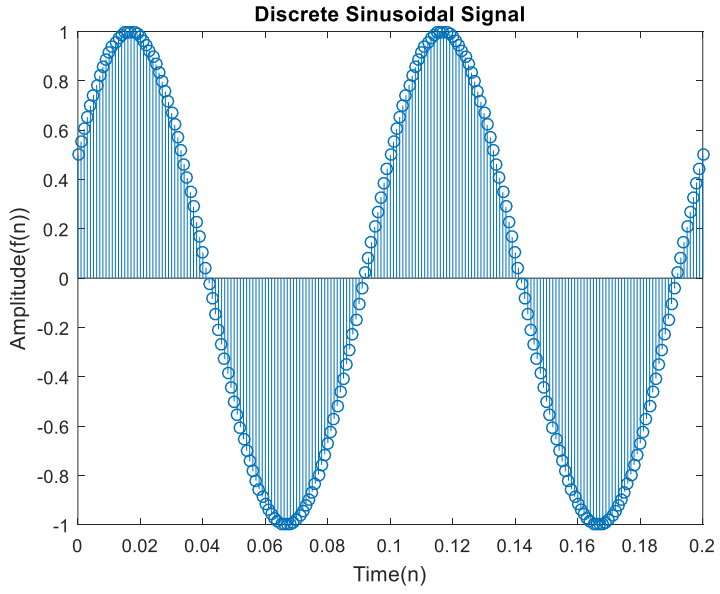
clc - clears all the text from the Command Window, resulting in a clear screen
clear all - Remove items from workspace, freeing up system memory
close all - closes all figures
figure - creates a new figure window using default property values
plot - plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

Program:

```
clc;
clear all;
close all;
%Continuous Signal Generation using trigonometric functions
t=0:0.001:.2;
f=sin(2*pi*10*t+pi/6);
figure(1)
plot(t,f)
xlabel('Time(t)')
ylabel('Amplitude(f(t))')
title(' Continuous Sinusoidal Signal')

%Discrete Signal Generation using trigonometric functions
n=0:0.001:.2;
f=sin(2*pi*10*n+pi/6);
figure(2)
stem(n,f)
xlabel('Time(n)')
ylabel('Amplitude(f(n))')
title('Discrete Sinusoidal Signal')
```

Outputs:





Result:

Hence continuous and discrete signals from the given function f*(t)=sin (2π10t + π/ 6)*is plotted.

Experiment: 2A

Generate function f(t) by performing product operation on two given functions h(t) and g(t).

AIM: To Generate function f(t) by performing product operation on two given functions h(t) and g(t).

Where $h(t) = sin\ (2\pi 10t + \pi/6)$ and $g(t) = e^{-10t}$ and show the signals in different figure windows

Software Used: MATLAB R2016a

Theory:

Multiplication of two signals is nothing but multiplication of their corresponding amplitudes. Mathematically, this can be given as:

$y(t) = x_1(t) \times x_2(t)$ … for continuous-time signals $x_1(t)$ and $x_2(t)$

and

$y[n] = x_1[n] \times x_2[n]$ … for discrete-time signals $x_1[n]$ and $x_2[n]$

Commands used:

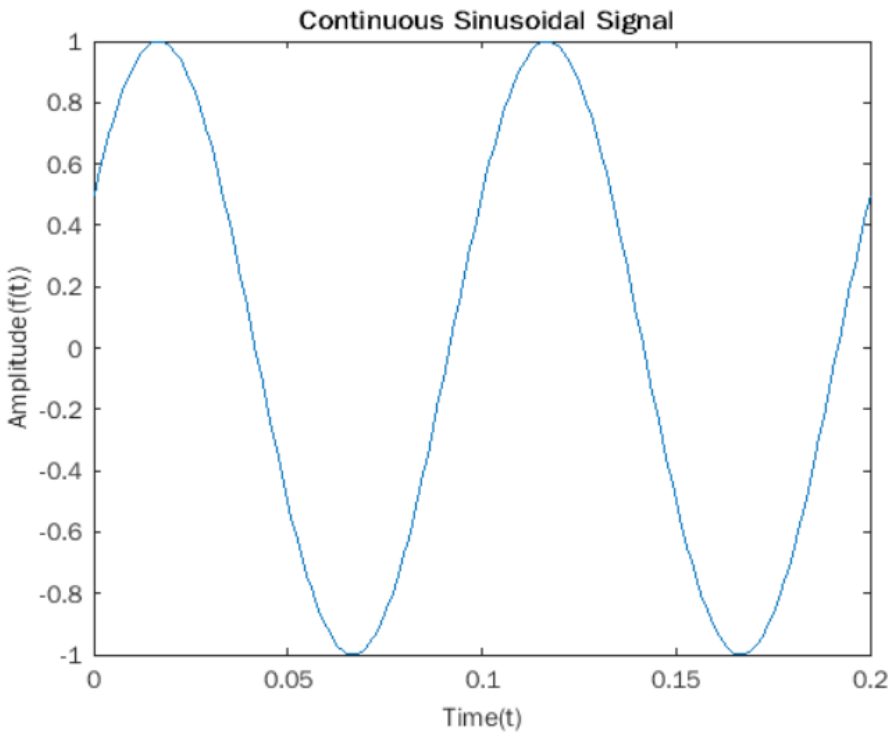
- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- plot - plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X and gives a continuous signal
- stem - stem(X,Y) plots the data sequence, Y, at values specified by X and gives a discrete signal
- figure - creates a new figure window using default property values

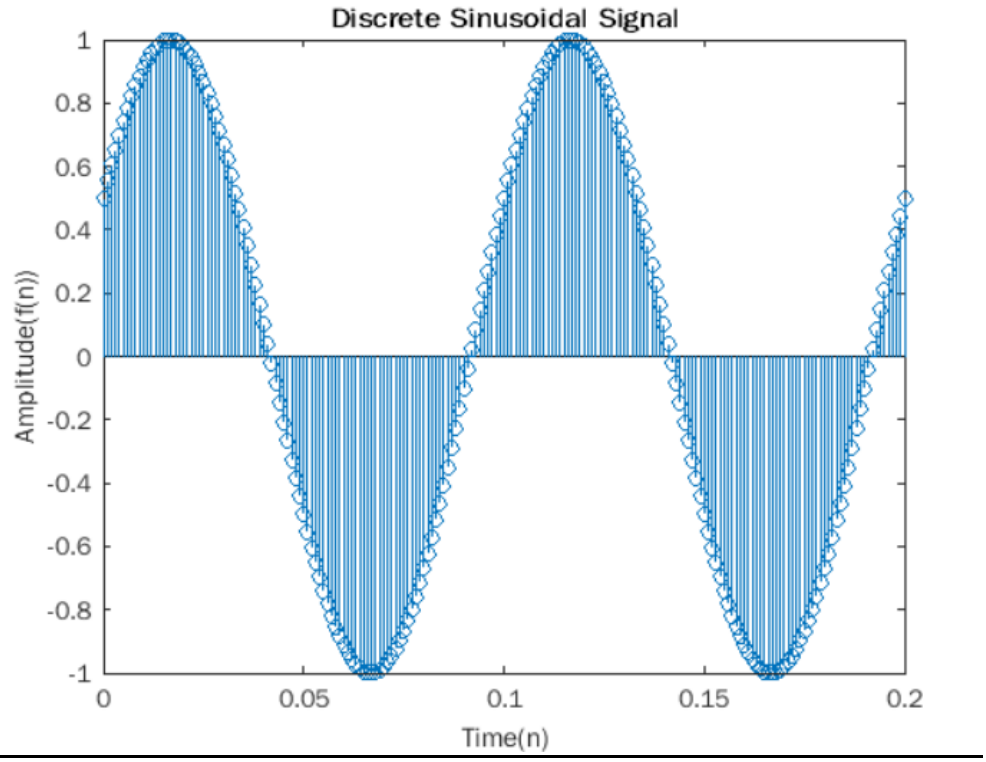
Program:

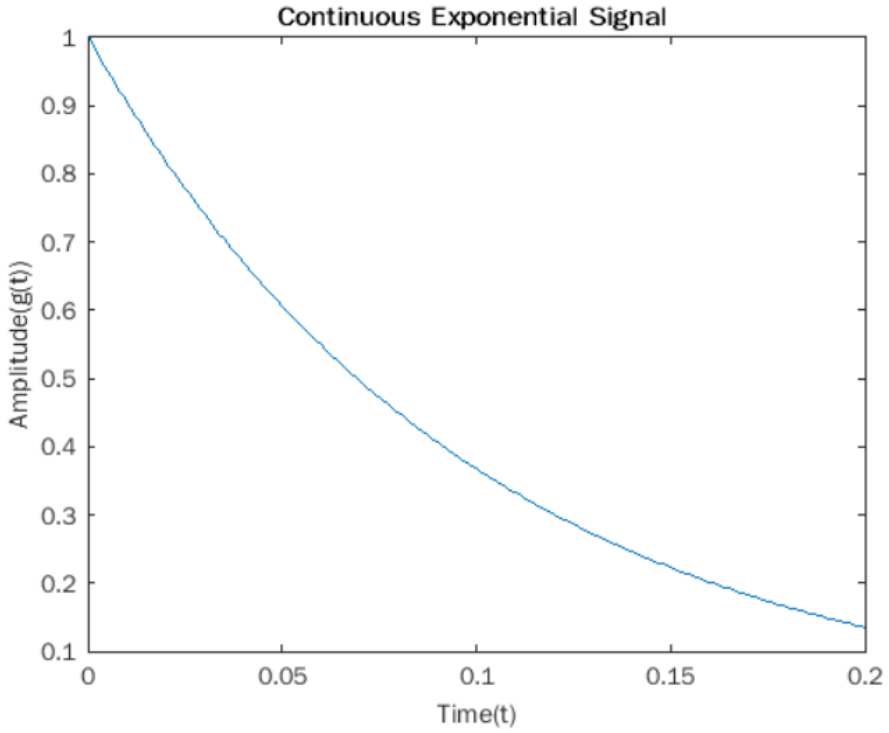
```
clc;
clear all;
close all;
% Continuous Sinusoidal Signal
t=0:0.001:.2;
h=sin(2*pi*10*t+pi/6);

figure(1)
plot(t,h)
xlabel('Time(t)')
ylabel('Amplitude(f(t))')
title(' Continuous Sinusoidal Signal')

% Discrete Sinusoidal Signal
n=0:0.001:.2;
```

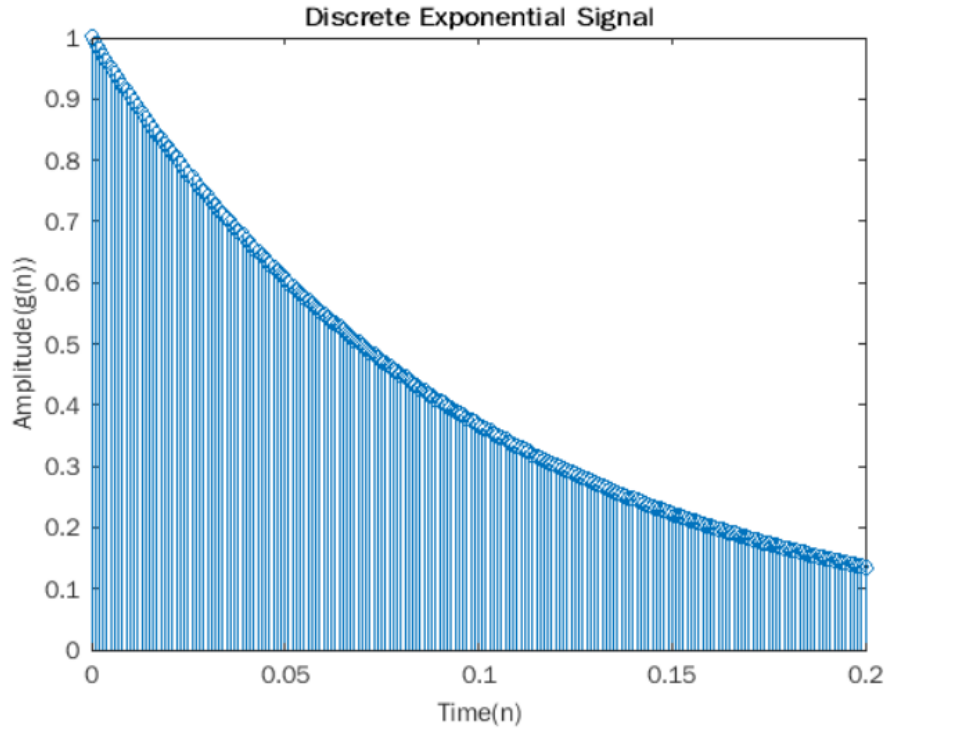
```
h=sin(2*pi*10*n+pi/6);

figure(2)
stem(n,h)
xlabel('Time(n)')
ylabel('Amplitude(f(n))')
title('Discrete Sinusoidal Signal')

%Continuous Exponential Signal
t=0:0.001:.2;
g=exp(-10*t);

figure(3)
plot(t,g)
xlabel('Time(t)')
ylabel('Amplitude(g(t))')
title('Continuous Exponential Signal ')

%Discrete Exponential Signal
n=0:0.001:.2;
g=exp(-10*n);

figure(4)
stem(n,g)
xlabel('Time(n)')
ylabel('Amplitude(g(n))')
title('Discrete Exponential Signal')

%Continuous Product Signal
t=0:0.001:.2;
f=g.*h;
figure(5)
plot(t,f)
xlabel('Time(t)')
ylabel('Amplitude(f(t))')
title('Continuous Product Signal')

%Discrete Product Signal
n=0:0.001:.2;
f=g.*h;
figure(6)
stem(n,f)
xlabel('Time(n)')
ylabel('Amplitude(f(n))')
title('Discrete Product Signal')
```

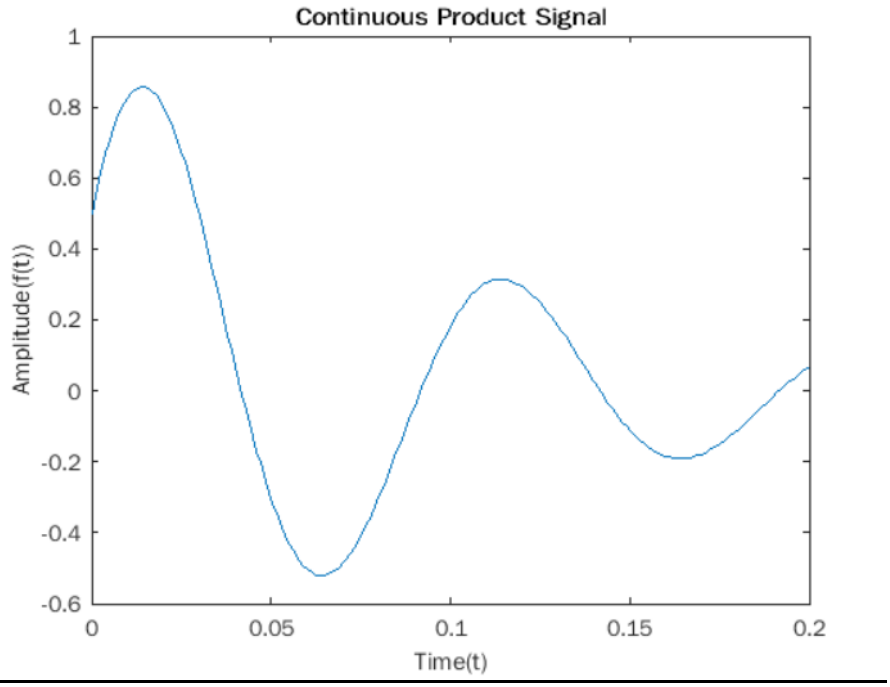
Outputs:

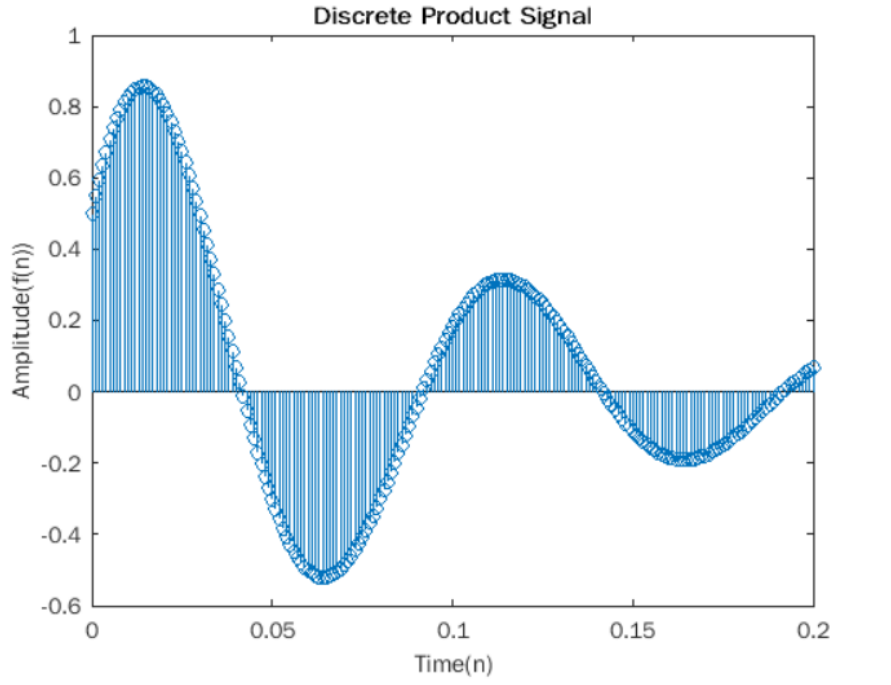

Continuous Sinusoidal Signal
Amplitude(f(t))
Time(t)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.05
0.1
0.15
0.2


Discrete Sinusoidal Signal
Amplitude(f(n))
Time(n)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.05
0.1
0.15
0.2


Continuous Exponential Signal
Amplitude(g(t))
Time(t)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.05
0.1
0.15
0.2


Discrete Exponential Signal
Amplitude(g(n))
Time(n)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.05
0.1
0.15
0.2





Result:

Hence function f(t) by performing product operation on two given functions h(t) and g(t) is generated.

Experiment :2B

Generate function f(t) by performing product operation on two given functions h(t) and g(t).

AIM: Generate function f(t) by performing product operation on two given functions h(t) and g(t).

$h(t) = sin\ (2\pi 10t + \pi/6\ )$ and $g(t) = e^{-10t}$ by using subplot to show the signals in single figure window

Software Used: MATLAB R2016a

Theory:

Multiplication of two signals is nothing but multiplication of their corresponding amplitudes. Mathematically, this can be given as:

$y(t) = x_1(t) \times x_2(t)$ … for continuous-time signals $x_1(t)$ and $x_2(t)$

and

$y[n] = x_1[n] \times x_2[n]$ … for discrete-time signals $x_1[n]$ and $x_2[n]$

Commands used:

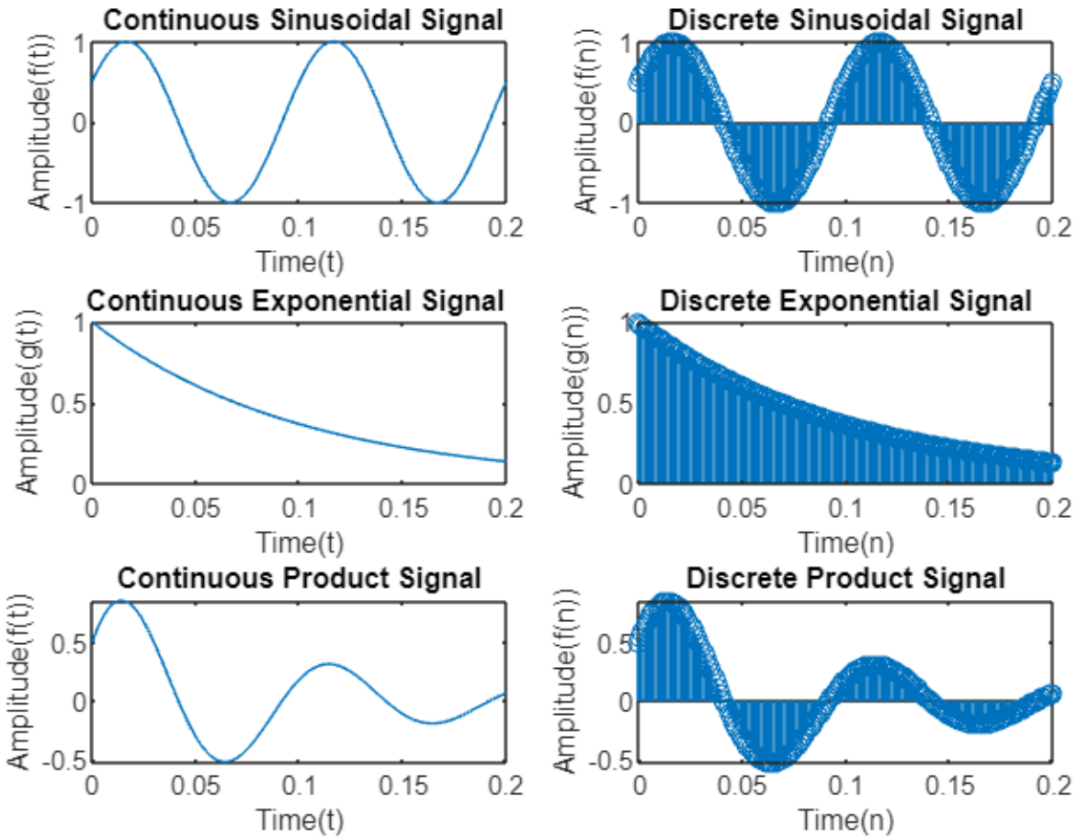
- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- plot - plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X and gives a continuous signal
- stem - stem(X,Y) plots the data sequence, Y, at values specified by X and gives a discrete signal
- figure - creates a new figure window using default property values
- subplot - subplot(m,n,p) divides the current figure into an m-by-n grid and creates axes in the position specified by p

Program:

```
clc;
clear all;
close all;
% Continuous Sinusoidal Signal
t=0:0.001:.2;
h=sin(2*pi*10*t+pi/6);

subplot(3,2,1)
plot(t,h)
xlabel('Time(t)')
ylabel('Amplitude(f(t))')
title(' Continuous Sinusoidal Signal')

% Discrete Sinusoidal Signal
n=0:0.001:.2;
h=sin(2*pi*10*n+pi/6);
```

```
subplot(3,2,2)
stem(n,h)
xlabel('Time(n)')
ylabel('Amplitude(f(n))')
title('Discrete Sinusoidal Signal')

%Continuous Exponential Signal
t=0:0.001:.2;
g=exp(-10*t);

subplot(3,2,3)
plot(t,g)
xlabel('Time(t)')
ylabel('Amplitude(g(t))')
title('Continuous Exponential Signal ')

%Discrete Exponential Signal
n=0:0.001:.2;
g=exp(-10*n);

subplot(3,2,4)
stem(n,g)
xlabel('Time(n)')
ylabel('Amplitude(g(n))')
title('Discrete Exponential Signal')

%Continuous Product Signal
t=0:0.001:.2;
f=g.*h;
subplot(3,2,5)
plot(t,f)
xlabel('Time(t)')
ylabel('Amplitude(f(t))')
title('Continuous Product Signal')

%Discrete Product Signal
n=0:0.001:.2;
f=g.*h;
subplot(3,2,6)
stem(n,f)
xlabel('Time(n)')
ylabel('Amplitude(f(n))')
title('Discrete Product Signal')
```

Outputs:



Result:

Hence function f(t) by performing product operation on two given functions h(t) and g(t) in signal figure window is generated

Experiment:3A

Plot the signal with constant beta value using the $h_k(t)$

Aim: To plot the signal with constant beta value using the $h_k(t) = e^{-\beta t} sin\ (2\pi 10t + \pi/6)$ over a time over $0 \leq t \leq 0.2$.

Software Used: MATLAB R2016a

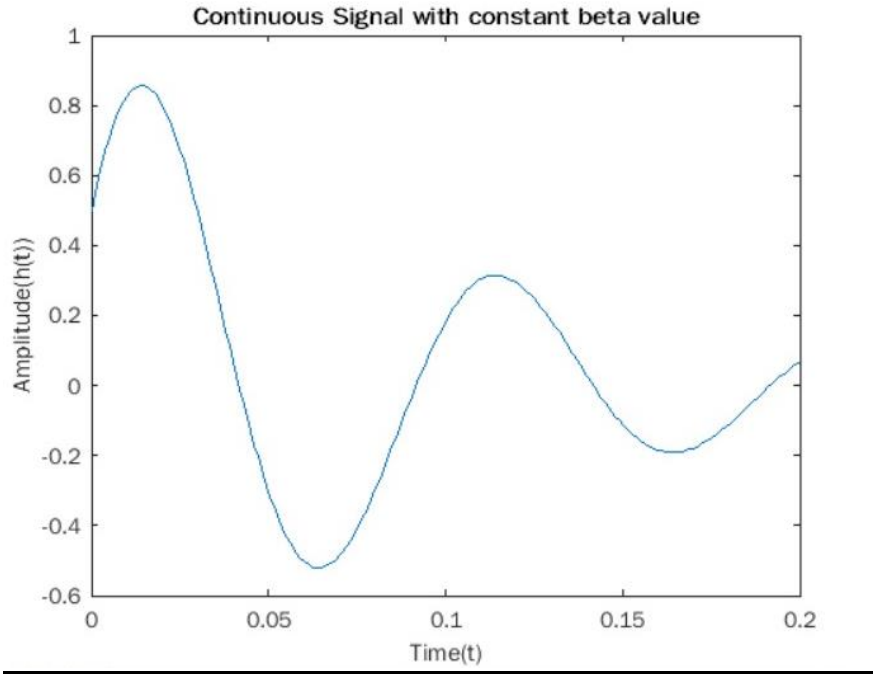
Commands used:

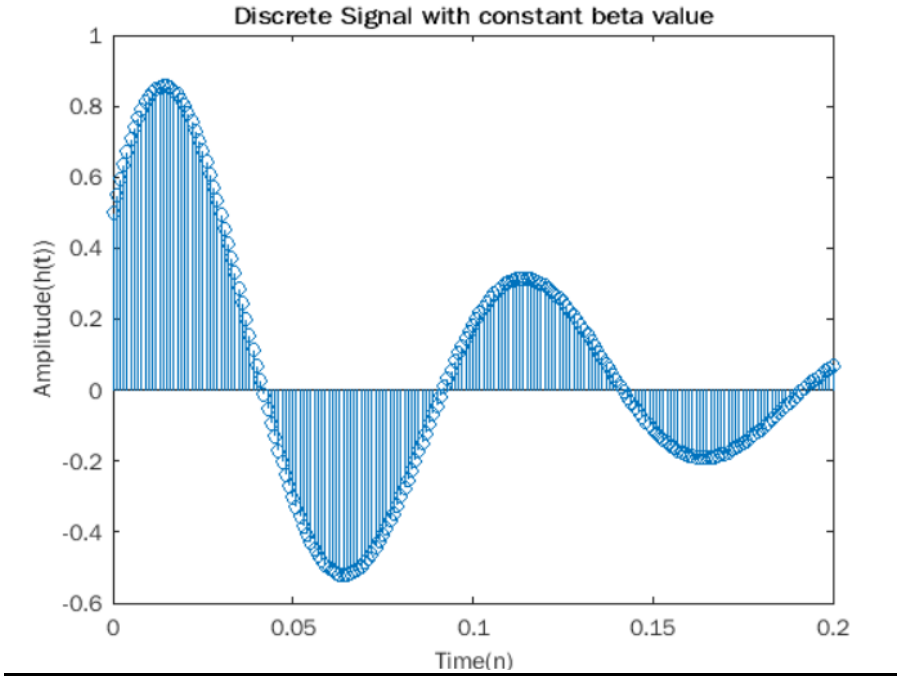
- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- plot - plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X and gives a continuous signal
- stem - stem(X,Y) plots the data sequence, Y, at values specified by X and gives a discrete signal
- figure - creates a new figure window using default property values

Program:

```
clc;
clear all;
close all;

% Continuous Signal with constant Beta(b) value
t=0:0.001:0.2;
b=10;
h=(exp(-b*t)).*sin(2*pi*b*t + pi/6);
figure (1)
plot(t,h)
xlabel('Time(t)')
ylabel('Amplitude(h(t))')
title('Continuous Signal with constant beta value')


% Discrete Signal with constant Beta(b) value
figure (2)
n=0:0.001:0.2;
h=(exp(-b*n)).*sin(2*pi*b*n + pi/6);
figure(2)
stem(t,h)
xlabel('Time(n)')
ylabel('Amplitude(h(t))')
title('Discrete Signal with constant beta value')
```

Outputs:





Result:

Hence the signal with constant beta value using the $h_k(t)$is plotted

Experiment-3B
Plot the signal with variable beta value using the $h_k(t)$

Aim: To plot the signal with variable beta value using the $h_k(t) = e^{-\beta t} sin\ (2\pi 10t + \pi/6)$over a time over $0 \leq t \leq 0.2$.

Software Used: MATLAB R2016a

Commands used:

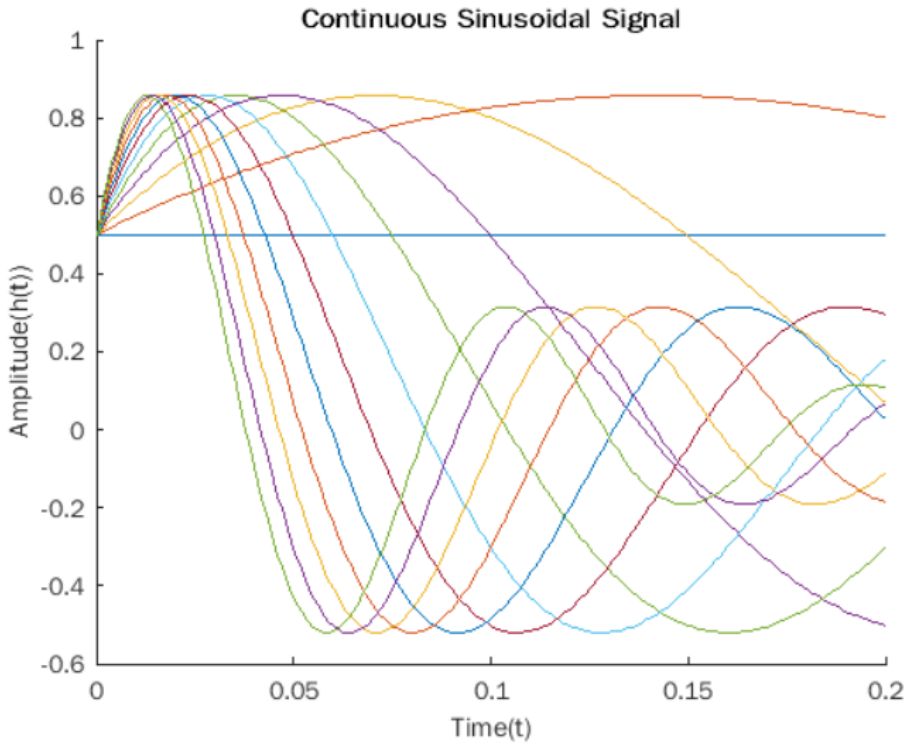
- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- plot - plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X and gives a continuous signal
- stem - stem(X,Y) plots the data sequence, Y, at values specified by X and gives a discrete signal
- figure - creates a new figure window using default property values
- for - for loop is used to repeat task for specified number of times
- hold on - hold on retains plots in the current axes so that new plots added to the axes do not delete existing plots.
- hold off - hold off sets the hold state to off so that new plots added to the axes clear existing plots and reset all axes properties.

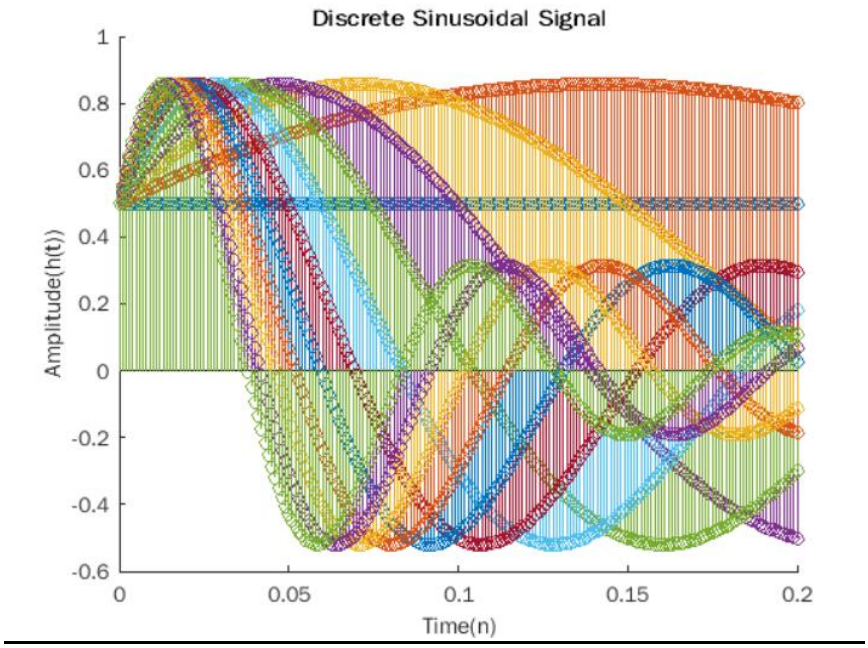
Program:

```
clc;
clear all;
close all;
% Continuous Signal with variable Beta(b) value
figure(1)
t=0:0.001:0.2;
for b=0:11
h=(exp(-b*t)).*sin(2*pi*b*t + pi/6);
hold on
plot(t,h)
end
hold off
xlabel('Time(t)')
ylabel('Amplitude(h(t))')
title('Continuous Sinusoidal Signal')


% Discrete Signal with variable Beta(b) value
figure(2)
n=0:0.001:0.2;
for b=0:11
h=(exp(-b*n)).*sin(2*pi*b*n + pi/6);
hold on
stem(n,h)
end
hold off
xlabel('Time(n)')
```

```
ylabel('Amplitude(h(t))')
title('Discrete Sinusoidal Signal')
```

Outputs:





Result: Hence the signal with variable beta value using the $h_k(t)$is plotted.

Experiment-4

Solving linear equations using inverse method

Aim: Solving linear equations using inverse method

$$1x + 2y + 3z = 10$$

$$4x + 5y + 6z = 20$$

$$7x + 2y + 9z = 9$$

Software Used: MATLAB R2016a

Theory:

Steps involved in finding inverse

1. Converting linear equations in to Matrix.

```
A=[1 2 3;4 5 6;7 2 9];
X=[x;y;z];
B=[10;20;9]
```

2. Assigning matrix values to variables

```
a1=A(1,1); b1=A(1,1); c1=A(1,3);
d1=A(2,1); e1=A(2,2); f1=A(2,3);
h1=A(1,1); i1=A(3,2); j1=A(3,3);
```

3. Finding Determinant
4. Finding cofactor matrix
5. Finding Adjoint of the matrix
6. Inverse of the matrix
7. $X=M^{-1} * B$

Commands used:

- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- inv - inv(X) computes the inverse of square matrix X.
- display – To show information about variable or result of expression

Program:

```
%solving linear equations using inverse methods
clc; clear all; close all;
```

```
A=[1 2 3;4 5 6;7 2 9];B=[10;20;9];
a = A(1,1);b = A(1,2);c = A(1,3);
d = A(2,1);e = A(2,2);f = A(2,3);
g = A(3,1);h = A(3,2);i = A(3,3);
deta = a*(e*i-f*h);detb = b*(d*i-f*g);detc = c*(d*h-e*g);
detA = deta - detb + detc;
if(detA == 0)
    fprintf('Matrix is not invertible.\n');
end
    %calculate cofactor
    a2 = e*i-h*f;
    b2 = -(d*i-g*f);
    c2 = d*h-g*e;
    d2 = -(b*i-h*c);
    e2 = a*i-g*c;
    f2 = -(a*h-g*b);
    g2 = b*f-e*c;
    h2 = -(a*f-d*c);
    i2 = a*e-d*b;
    %cofactor matrix
    Cofactor_Matrix = [a2 b2 c2; d2 e2 f2; g2 h2 i2];
    %calculate transpose of cofactor by column operator
    %Ajoint_matrix(:,1) = Cofactor_Matrix(1,:);
    %Ajoint_matrix(:,2) = Cofactor_Matrix(2,:);
    %Ajoint_matrix(:,3) = Cofactor_Matrix(3,:);
    %calculate transpose of cofactor by Cofactor_Matrix' symbol
    Ajoint_matrix=Cofactor_Matrix';
    %calculate inverse
    InvA=Ajoint_matrix/detA;
    display(InvA)
    %Finding x,y,Z values
```

X=InvA*B

Output:

InvA =

-0.9167	0.3333	0.0833
-0.1667	0.3333	-0.1667
0.7500	-0.3333	0.0833

X =

-1.7500
3.5000
1.5833

Result:

Hence linear equations using inverse method are solved

Experiment 5

Solving linear equations using Cramer's methods

Aim: To Solve linear equations using Cramer's methods

$$x+ y +z = 11$$

$$2x -6y -z = 0$$

$$3x + 4y+ 2z = 0.$$

Software Used: MATLAB R2016a

Theory:

Steps involved in finding Solution using Cramer's Method

1. Converting linear equations in to Matrix.

```
A=[a b c ;d e f; g h i];
X=[x; y; z];
B= [b1 b2 b3]
```

2. Assigning matrix values to variables

```
a=A(1,1); b=A(1,2); c=A(1,3);
d=A(2,1); e=A(2,2); f=A(2,3);
g=A(3,1); h=A(3,2); i=A(3,3);
b11=B(1,1);
b21=B(2,1);
b31=B(3,1);
```

3. Finding Determinant A
4. Finding Determinant detla_Ax
5. Finding Determinant detla_Ay
6. Finding Determinant detla_Az
7. x= Determinant detla_Ax/ Determinant A;
8. y= Determinant detla_Ay/ Determinant A;
9. z= Determinant detla_Az/ Determinant A;

Commands used:

- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- det - det(A) returns the determinant of square matrix A

Program:

```
%solving linear equations using Cramer's methods
clc; clear all; close all;
A=[1 1 1;2 -6 -1;3 4 2];
B=[11; 0; 0];
  a = A(1,1);b = A(1,2);c = A(1,3);
  d = A(2,1);e = A(2,2);f = A(2,3);
  g = A(3,1);h = A(3,2);i = A(3,3);

    b11=B(1,1);

    b21=B(2,1);

    b31=B(3,1);

    detla_Ax = [b11 b c; b21 e f; b31 hi];

    detla_Ay = [a b11 c; d b21 f; g b31 i];

    detla_Az = [a b b11; de b21; gh b31];

    x = det(detla_Ax)/det(A)

    y = det(detla_Ay)/det(A)

    z = det(detla_Az)/det(A)

    display(x)
    display(y)
    display(z)
```

Output:

```
x =

   -8


y =

-7


z =

26
```

Result:
Hence linear equations using Cramer's methods are sloved.

Experiment-6

:Compute Eigen values and Eigen vectors

Aim: Compute Eigen values and Eigen vectors of given matrix

.

Software used: MATLAB R2016a

Theory:

Eigenvalues The number λ is an eigenvalue of A if and only if $A - \lambda I$ is singular:

$$\det(A - \lambda I) = 0. \tag{3}$$

This "characteristic equation" $\det(A - \lambda I) = 0$ involves only λ, not $\boldsymbol{x}$. When A is n by n, the equation has degree n. Then A has n eigenvalues and each λ leads to $\boldsymbol{x}$:

For each λ solve $(A - \lambda I)\boldsymbol{x} = \boldsymbol{0}$ or $A\boldsymbol{x} = \lambda\boldsymbol{x}$ to find an eigenvector $\boldsymbol{x}$.

Properties of Eigen Values

(a)The sum of the n eigenvalues equals the sum of the n diagonal entries.

(b)The product of the n eigenvalues equals the determinant.

Commands used:

- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- eig - eig(A) returns a column vector containing the eigenvalues of square matrix A.
- trace - trace(A) calculates the sum of the diagonal elements of matrix A.
- det - det(A) returns the determinant of square matrix A
- sum - sum(A) returns the sum of the elements of A along the first array dimension whose size does not equal 1.
- prod - prod(A) returns the product of the array elements of matrix

Program:

```
%Finding Eigen Values and Eigen Vectors using commands
%verify the properties of Eigen Values

clc; clear all; close all;
A=[1 2 3;4 0 0;5 0 6]

%Eigen Values
EValues=eig(A);
Lamba1=EValues(1,1)
Lamba2=EValues(2,1)
```

```
Lamba3=EValues(3,1)

%Properties of Eigen Values
% 1.Sum of Eigen Values = Trace of A
S_EValues=sum(EValues);
T_A=trace(A)

%2.Product of Eigen Values =determinant of A
P=prod(EValues);
detA=det(A)
%%%%%%%%%%       A*X=LAMBA*X

[V,D]=eig(A);
Eigenvector_1=V(:,1)
Eigenvector_2=V(:,2)
Eigenvector_3=V(:,3)

d_D=diag(D);

L1=A*Eigenvector_1
R1=D(1,1)*Eigenvector_1


L2=A*Eigenvector_2
R2=D(2,2)*Eigenvector_2

L3=A*Eigenvector_3
R3=D(3,3)*V(:,3)
```

Output:

```
A =

     1     2     3
     4     0     0
     5     0     6


Lamba1 =

    8.3477


Lamba2 =

   -3.1646
```

```
Lamba3 =

    1.8170


T_A =

     7


detA =

  -48.0000


Eigenvector_1 =

   -0.4165
   -0.1996
   -0.8870


Eigenvector_2 =

   -0.5877
    0.7428
    0.3206


Eigenvector_3 =

   -0.3707
   -0.8162
    0.4432


L1 =

   -3.4765
   -1.6659
   -7.4042
```

```
R1 =

   -3.4765
   -1.6659
   -7.4042


L2 =

    1.8599
   -2.3508
   -1.0147


R2 =

    1.8599
   -2.3508
   -1.0147


L3 =

   -0.6736
   -1.4830
    0.8052


R3 =

   -0.6736
   -1.4830
    0.8052
```

Result: Hence Eigen values and Eigen vectors using matrices are computed

EXPERIMENT-7

LU matrix factorization

Aim: Compute LU matrix factorization A

Software used:MATLAB R2016a

Theory:

% Crout LU Decomposition

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 8 & 14 \\ 2 & 6 & 13 \end{bmatrix} = LU$$

$$\text{where } L = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \quad U = \begin{bmatrix} 1 & U_{12} & U_{13} \\ 0 & 1 & U_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying out LU and setting the answer equal to A gives

$$\begin{bmatrix} L_{11} & L_{11}U_{12} & L_{11}U_{13} \\ L_{21} & L_{21}U_{12} + L_{22} & L_{21}U_{13} + L_{22}U_{23} \\ L_{31} & L_{31}U_{12} + L_{32} & L_{31}U_{13} + L_{32}U_{23} + L_{33} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 8 & 14 \\ 2 & 6 & 13 \end{bmatrix}$$

Commands used:

- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- for - for loop is used to repeat task for specified number of times
- size - size(A) returns a row vector whose elements are the lengths of the corresponding dimensions of A.

Program Code:

```
clc;
clear all;
close all;
A=[1,2,4;3,8,14;2,6,13]
[R,C]=size(A);
for i = 1:R
   L(i,1)=A(i,1);
   U(i,i)=1;
end
for j = 2:R
   U(1,j)=A(1,j)/L(1,1);
end
for i = 2:R
for j = 2:i
      L(i,j) = A(i,j)-L(i,1:j-1)*U(1:j-1,j);
end
for j = i-1:R
      U(i,j) = (A(i,j)-L(i,1:i-1)*U(1:i-1,j))/L(i,i);
end
end
```

Output:

```
A =

    1    2    4
    3    8   14
    2    6   13
```

Result:

Hence LU matrix factorization is computed.

Experiment-8

Basic Operations on Signals

Aim: To perform various Basic operations on Signals using Matlab

Software used: MATLAB R2016a

Theory:

- **Addition:**

Addition of two signals is addition of their corresponding amplitudes.

- **Subtraction:**

Subtraction of two signals is subtraction of their corresponding amplitudes.

- **Multiplication:**

Multiplication of two signals is multiplication of corresponding amplitudes of signals.

- **Shifting a Signal:**

If the time shifting operation is applied on a signal x(t), then the signal is shifted to left or right without changing its characteristics represented with $x(t \pm t0)$, where t0 is shift (delay or advance).

- **Reversing a Signal**:

If the given signal is x(t), then its time reversal form is represented with x(-t)

Commands Used:

- subplot - subplot(m,n,p) divides the current figure into an m-by-n grid and creates axes in the position specified by p
- stem - stem(X,Y) plots the data sequence, Y, at values specified by X and gives a discrete signal
- min - **min**(X,Y) returns the smallest element taken from X or Y
- max - **max**(X,Y) returns the largest element taken from X or Y
- axis - **axis** Control axis scaling and appearance.
- flipr - **fliplr()** function can be used to perform reversing or folding a signal.

Program Codes:

Addition :

```
x=[1 2 3 4];
subplot(3,1,1);
stem(x);
title('X');
y=[1 1 1 1];
subplot(3,1,2);
stem(y);
title('Y');
z=x+y;
subplot(3,1,3);
stem(z);
title('Z=X+Y');
```

Subtraction :

```
n1=-2:1;
x=[1 2 3 4];
subplot(3,1,1);
stem(n1,x);
title('X') ;
axis([-4 4 -5 5]);
n2=0:3;
y=[1 1 1 1];
subplot(3,1,2);
stem(n2,y);
title('Y');
axis([-4 4 -5 5]);
n3 =min (min(n1) ,min( n2 ) ) : max ( max ( n1 ) , max ( n2 ) ); % finding the
duration of output signal
s1 =zeros(1,length (n3) );
s2 =s1;
s1 (find ( ( n3>=min( n1 ) ) & ( n3 <=max ( n1 ) )==1 ) )=x;
% signal x with the duration of output signal 'sub'
s2 (find ( ( n3>=min ( n2 ) ) & ( n3 <=max ( n2 ))==1) )=y;
% signal y with the duration of output signal 'sub'
```

```
sub=s1 - s2; % subtraction
subplot(3,1,3)
stem(n3,sub)
title('Z=X-Y');
axis([-4 4 -5 5]);
```

Multiplication :

```
n1=-2:1;
x=[1 2 3 4];
subplot(3,1,1);
stem(n1,x);
title('X') ;
axis([-4 4 -5 5]);
n2=0:3;
y=[1 1 1 1];
subplot(3,1,2);
stem(n2,y);
title('Y');
axis([-4 4 -5 5]);
n3 =min (min(n1) ,min( n2 ) ) : max ( max ( n1 ) , max ( n2 ) ); % finding the
duration of output signal (out)
s1 =zeros(1,length (n3) );
s2 =s1;
s1 (find ( ( n3>=min( n1 ) ) & ( n3 <=max ( n1 ) )==1 ) )=x;
% signal x with the duration of output signal 'mul'
s2 (find ( ( n3>=min ( n2 ) ) & ( n3 <=max ( n2 ))==1) )=y;
% signal y with the duration of output signal 'mul'
mul=s1 .* s2; % multiplication
subplot(3,1,3)
stem(n3,mul)
title('Z=X*Y');
axis([-4 4 -5 5]);
```

Shifting a Signal :

```
n1=input('Enter the amount to be delayed');
n2=input('Enter the amount to be advanced');
n=-2:2;
```

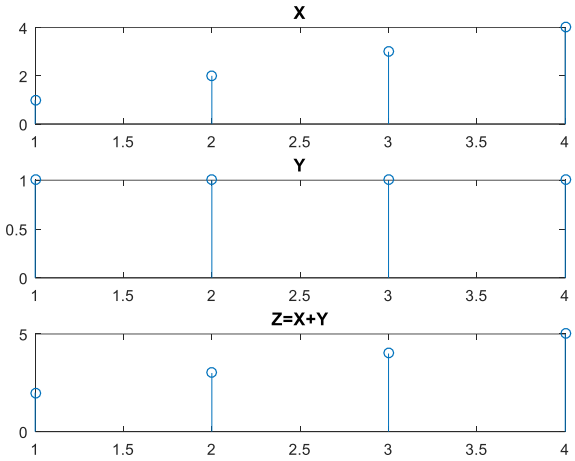
```
x=[-2 3 0 1 5];
subplot(3,1,1);
stem(n,x);
title('Signal x(n)');
m=n+n1;
y=x;
subplot(3,1,2);
stem(m,y);
title('Delayed signal x(n-n1)');
t=n-n2;
z=x;
subplot(3,1,3);
stem(t,z);
title('Advanced signal x(n+n2)');
```

Reversing a Signal :

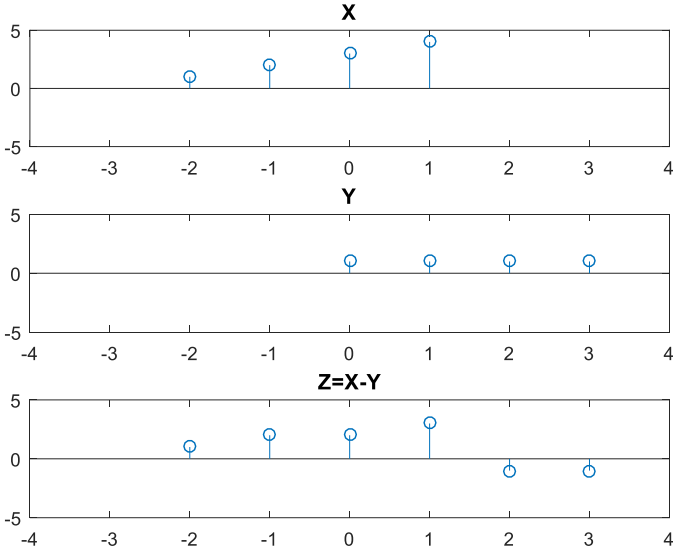
```
n=-1:2;
x=[3 -1 0 -4];
subplot(2,1,1)
stem(n,x);
axis([-3 3 -5 5]);
title('Signal x(n)');
c=fliplr(x);
y=fliplr(-n);
subplot(2,1,2);
stem(y,c);
axis([-3 3 -5 5]);
title('Reversed Signal x(-n)') ;
```

Outputs:

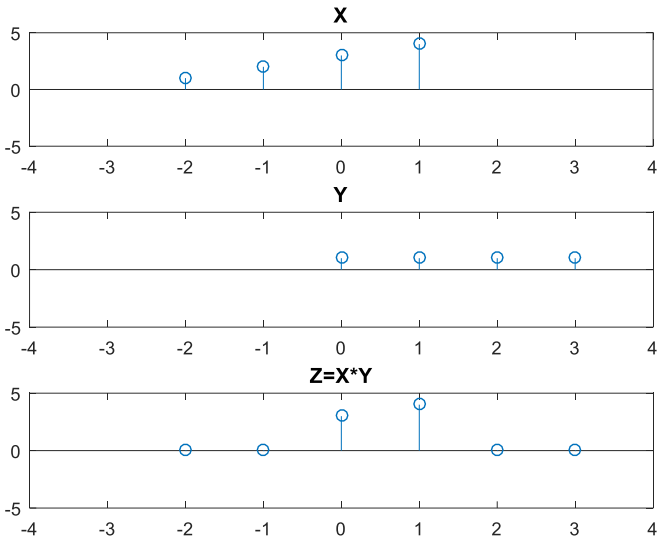
Signal Addition



Signal Subtraction



Signal Multiplication

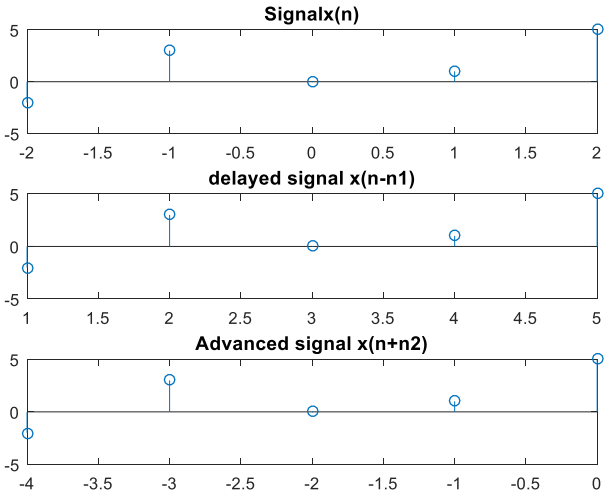


Signal Shifting

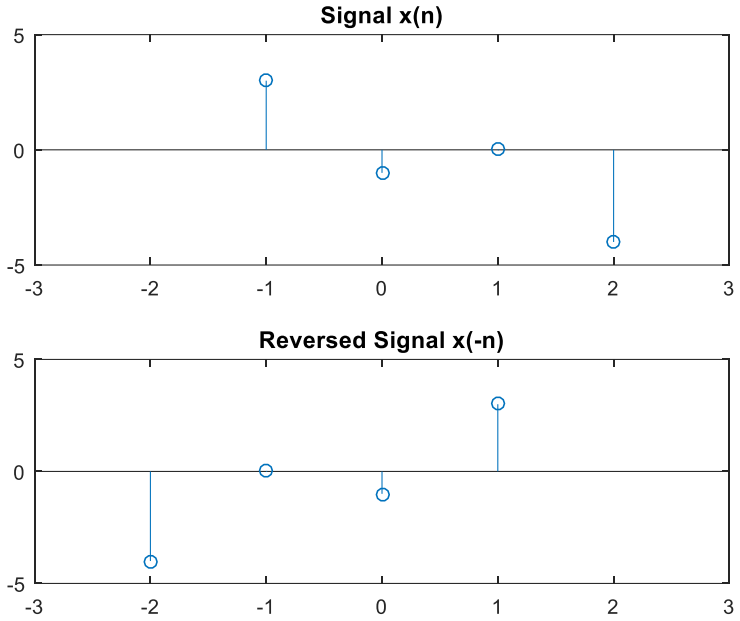
Eg:

Enter amount to be delayed 3

Enter amount to be advanced 2



Signal Reversal



Result:

Hence Basic operations on signals are performed using MATLAB.

Experiment-9

Convolution of Signals

Aim: To perform Convolution of Two Signals without conv function

Software used: MATLAB R2016a

Theory:

Convolution is an operation, which is used in almost all signal processing applications to analyze signals and systems in both the time and frequency domain. Convolution is a special operation, which includes four different operations, namely

- Folding,
- Shifting,
- Multiplication and
- Integration in the case of continues time signals,
  Summation in the case of discrete time signals.

Convolution in continuous time domain can be defined as

$$x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(r)x_2(t-r)dr = \int_{-\infty}^{\infty} x_2(r)\, x_1(t-r)dr$$

Where, $x_1(t)$ and $x_2(t)$ are two continuous time signals

Convolution in discrete time domain can be defined as

$$x_1(n) * x_2(n) = \sum_{m=-\infty}^{\infty} x_1(m)\, x_2(n-m)$$

Where, $x_1(n)$ and $x_2(n)$ are two discrete time signals

Procedure to compute the convoluted signal through the graphical method:

- First draw the graphical representation of given signals x1(τ) and x2(τ).
- Draw the folding form of x2(τ), i.e. x2(-τ).
- Draw the shifting form of x2(-τ) by t units, i.e., x2(t-τ).
- Take the product of x1(τ) and x2(t-τ).
- Integrate 'x1(τ)x2(t-τ)' with respect to τ over wide range of time.

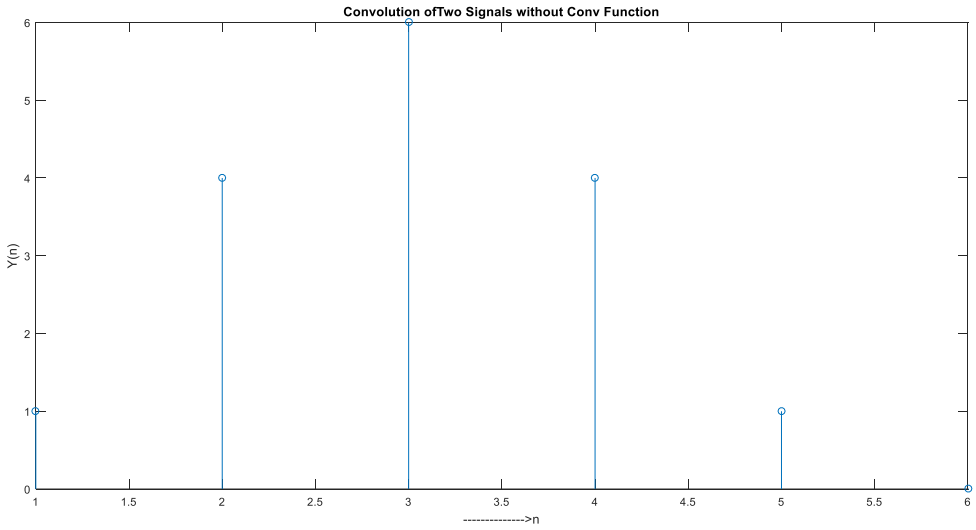
Commands used:

- input : x = input(prompt) displays the text in prompt and waits for the user to input a valueand press the **Return** key. The user can enter expressions, like pi/4 or rand(3), and can use variables in the workspace.
- length :L = length(X) returns the length of the largest array dimension in X.
- for :for loop is used to repeat specified number of times.
- if-else :if *expression*, *statements*, end evaluates an expression, and executes a group of statements when the expression is true

Program:

```
close all;clearall;
x=input('Enter x:    ');
h=input('Enter h:    ');

m=length(x);
n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)];
fori=1:n+m-1
    Y(i)=0;
for j=1:m
if(i-j+1>0)
            Y(i)=Y(i)+X(j)*H(i-j+1);
else
end
end
end
stem(Y);
ylabel('Y[n]');
xlabel('----->n');
title('Convolution of Two Signals without conv function');
```

Output:



Result:

Hence Convolution of Two Signals without conv function is performed.

Experiment-10

Transformation of signals from time to frequency domains

Aim: Write a MATLAB program to transform a time domain signal representation to frequency domain representation

Software Used: MATLAB R2016a

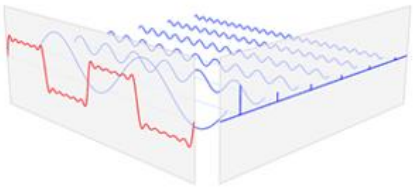
Commands used:

- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- plot - plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X and gives a continuous signal
- figure - creates a new figure window using default property values
- stem - stem(X,Y) plots the data sequence, Y, at values specified by X and gives a discrete signal

Theory:

A given function or signal can be converted between the time and frequency domains with a pair of mathematical operators called transforms. The Fourier transform converts the function's time-domain representation, to the function's frequency-domain representation. It converts a time function into a sum or integral of sine waves of different frequencies, each of which represents a frequency component.The "spectrum" of frequency components is the frequency-domain representation of the signal. The inverse Fourier transform converts the frequency-domain function back to the time-domain function. The component frequencies, spread across the frequency spectrum, are represented as peaks in the frequency domain.

Time Signal Frequency Components



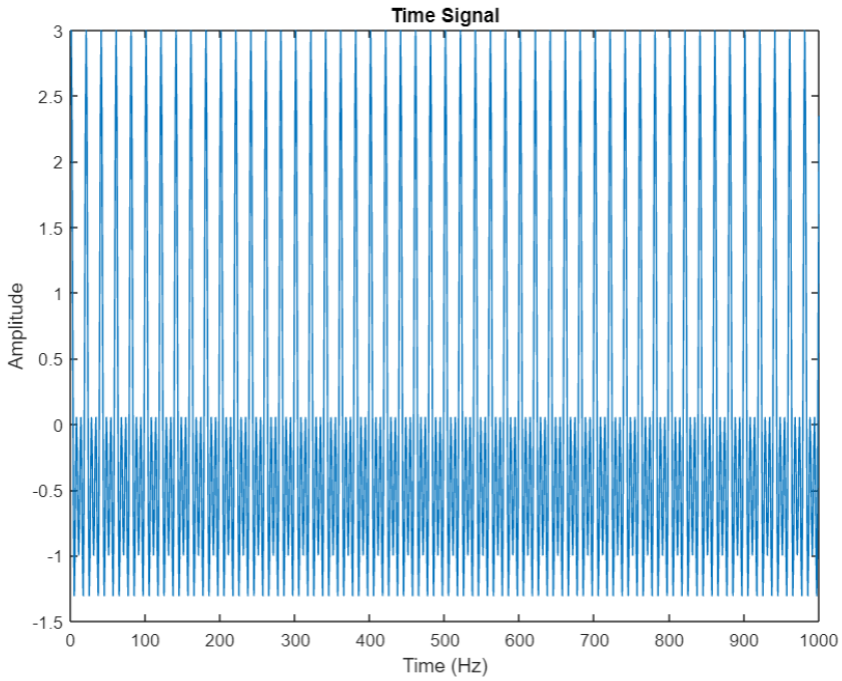
3D Representation

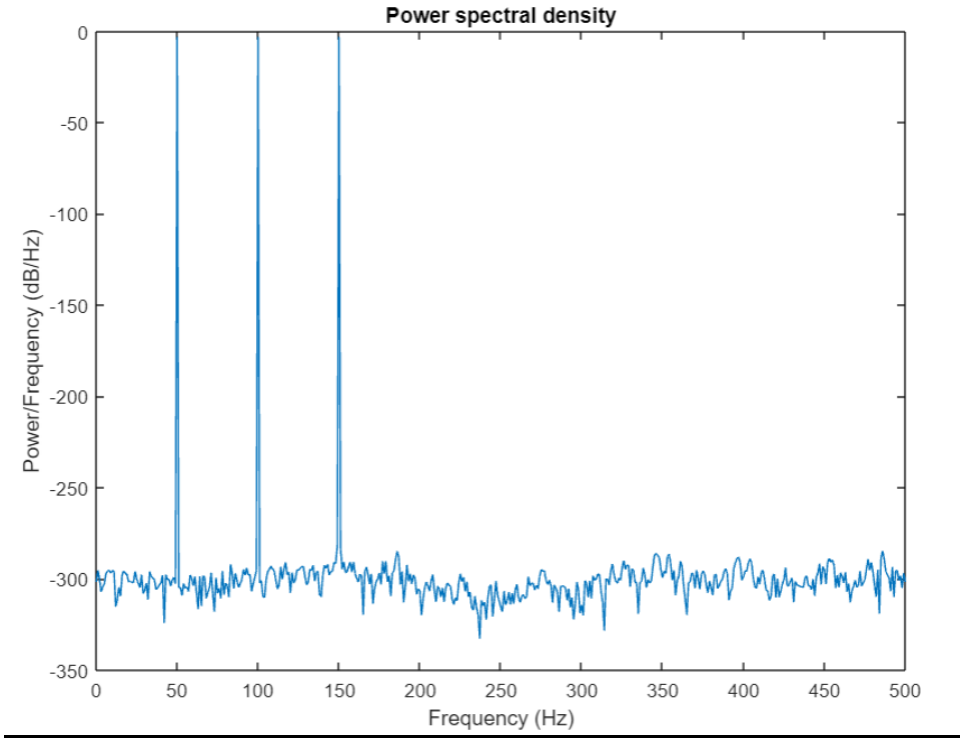
Program:

```
clc;
clear all;
close all;
Fs = 1000;
t = 0:1/Fs:1-1/Fs;
x = cos(2*pi*50*t)+cos(2*pi*100*t)+cos(2*pi*150*t);
figure(1)
plot(x)
title('Time Signal')
xlabel('Time (Hz)')
ylabel('Amplitude')
 N = length(x);
xdft = fft(x);
xdft = xdft(1:N/2+1);
psdx = (1/(Fs*N)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);
freq = 0:Fs/length(x):Fs/2;
figure(2)
```

```
plot(freq,10*log10(psdx))
title('Power spectral density')
xlabel('Frequency (Hz)')
ylabel('Power/Frequency (dB/Hz)')
```

Outputs:





Result:

The signal representation from time domain signal to frequency domain is transformed

Experiment-11

Compute & plot the Fourier coefficient for the periodic signals

Aim: To compute & plot the fourier coefficient for the periodic signals

Software Used: MATLAB R2016a

Commands used:

- clc - clears all the text from the Command Window, resulting in a clear screen
- clear all - Remove items from workspace, freeing up system memory
- close all - closes all figures
- sstem - stem(X,Y) plots the data sequence, Y, at values specified by X and gives a discrete signal
- subplot - subplot(m,n,p) divides the current figure into an m-by-n grid and creates axes in the position specified by p

Theory:

The Fourier series shows that f(t) can be described as

$$f(t) = a_v + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)$$

where

a_v, a_n, and b_n are known as the **Fourier coefficients**

given by

$$a_v = \frac{1}{T}\int_{t_0}^{t_0+T} f(t)dt,$$

$$a_k = \frac{2}{T}\int_{t_0}^{t_0+T} f(t)\cos(k\omega_0 t)dt,$$

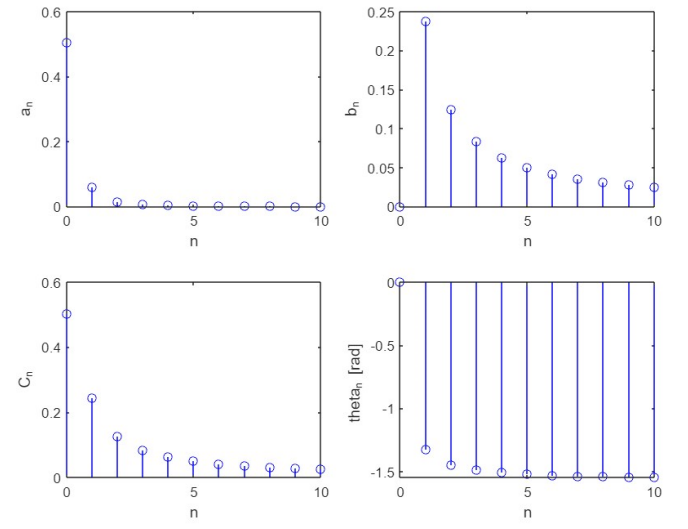
$$b_k = \frac{2}{T}\int_{t_0}^{t_0+T} f(t)\sin(k\omega_0 t)dt,$$

Program:

```
clc;clear all;close all;
 a_n0 = 0.504;
 b_n0 = 0;
 C_n0 = a_n0 ;
 theta_n0 = 0;

for i = 1:10
     a_n(i) = 0.504*2./(1+16*i.^2);
     b_n(i)=0.504*8*i./(1+16* i.^2 ) ;
```

```
    C_n(i)=sqrt(a_n(i).^2 + b_n(i).^2);
    theta_n(i) = atan2 (-b_n(i) ,a_n(i));
end

n= 0:10;
a_n=[a_n0 a_n];
subplot (2,2,1);stem (n,a_n,'b');
ylabel('a_n'); xlabel ('n');

b_n=[b_n0 b_n];
subplot (2,2,2);
stem (n,b_n,'b');
ylabel('b_n');
xlabel('n');

C_n=[C_n0 C_n];
subplot(2,2,3) ;
stem (n,C_n,'b');
ylabel('C_n' );
xlabel( ' n ');

theta_n=[theta_n0,theta_n];
subplot(2,2,4 ) ;
stem ( n , theta_n , ' b ');
ylabel ('theta_n [rad] ');
 xlabel ( ' n ');
```

Outputs:



Result:

hence the Fourier coefficient for a periodic signals is computed& plotted