Digital Signal Processing
Practical File

Session: 2022-23

Name:	Kartavya Gupta
Course:	B.Sc.(Hons) Electronics
Semester:	V
Roll No.	1620016

INDEX

Experiment no. 1

Aim:

Program for generating some basic signals (Unit Sequence, Impulse Sequence, Ramp Sequence, Exponential & Real Sinusoidal Signal.)

Input (a):

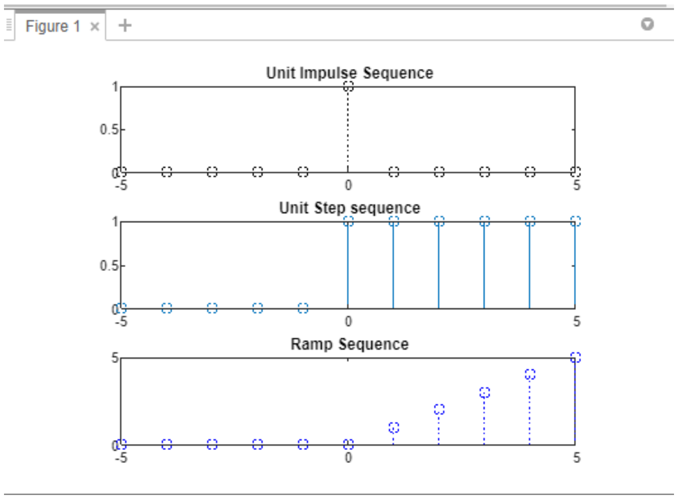
```
%Program for generating some basic signals (Unit Sequence, Impulse Sequence, Ramp Sequence)
function[x,n]=basicsignal(n0,n1,n2)
n=[n1:n2];
%Impulse Sequence
x1=[n-n0==0];
subplot(3,1,1);
stem(n,x1,':o k')
title("Unit Impulse Sequence")
%Unit sequence
x2=[n-n0>=0];
subplot(3,1,2);
stem(n,x2)
title('Unit Step sequence')
%Ramp sequence
R=n.*x2;
subplot(3,1,3);
stem(n,R,':o b')
title('Ramp Sequence')
end
```

Input (b):

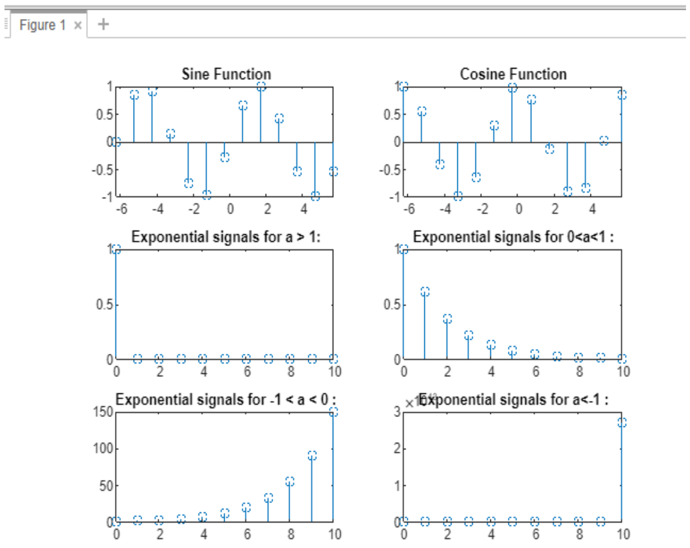
```
%Program to generate basic signals like exponential and real sinusoidal
%signals
%Real Sinusoidal Signals
n1=-2*pi;
n2=2*pi;
n=[n1:n2];
x=sin(n);
subplot(3,2,1)
stem(n,x)
title('Sine Function')
y=cos(n);
subplot(3,2,2)
stem(n,y)
title('Cosine Function')
%Exponential Signals
n1=0;
n2=10;
n=[n1:n2]
a=10;
x1=exp(-a.*n);
subplot(3,2,3)
stem(n,x1)
```

```
title('Exponential signals for a > 1:')
a=0.5;
x2=exp(-a.*n);
subplot(3,2,4)
stem(n,x2)
title('Exponential signals for 0<a<1 :')
a=-0.5;
x3=exp(-a.*n)
subplot(3,2,5)
stem(n,x3)
title('Exponential signals for -1 < a < 0 :')
a=-10;
x4=exp(-a.*n);
subplot(3,2,6)
stem(n,x4)
title('Exponential signals for a<-1 :')
```

Output (a):



Output (b):



Experiment no. 2

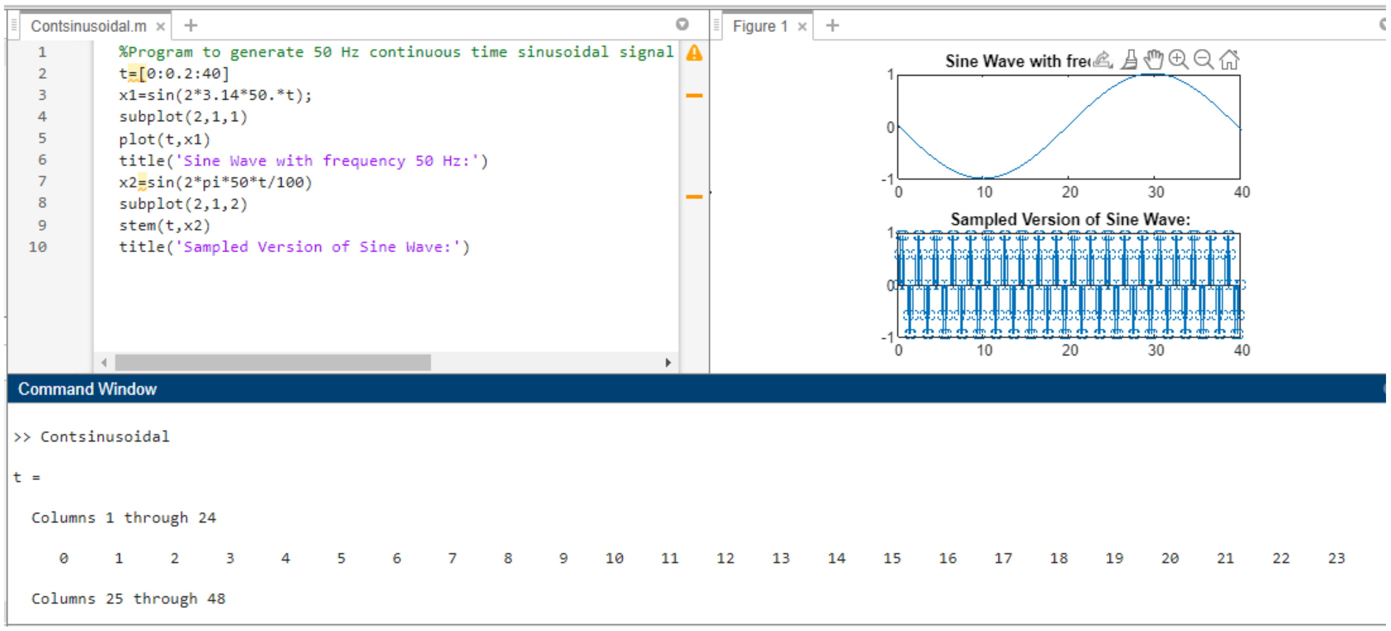
Aim:

Write a program to generate 50 Hz continuous time sinusoidal signal and its sampled version and plot the signal.

Input:

```
%Program to generate 50 Hz continuous time sinusoidal signal and its sampled version and
plot %the signal.
t=[0:0.2:40]
x1=sin(2*3.14*50.*t);
subplot(2,1,1)
plot(t,x1)
title('Sine Wave with frequency 50 Hz:')
x2=sin(2*pi*50*t/100)
subplot(2,1,2)
stem(t,x2)
title('Sampled Version of Sine Wave:')
```

Output:



```
%Program to generate 50 Hz continuous time sinusoidal signal
t=[0:0.2:40]
x1=sin(2*3.14*50.*t);
subplot(2,1,1)
plot(t,x1)
title('Sine Wave with frequency 50 Hz:')
x2=sin(2*pi*50*t/100)
subplot(2,1,2)
stem(t,x2)
title('Sampled Version of Sine Wave:')
```

```
>> Contsinusoidal

t =

  Columns 1 through 24

     0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20    21    22    23

  Columns 25 through 48
```

Experiment no. 3

Aim:

Program for Time Shifting, and reversal of signals.

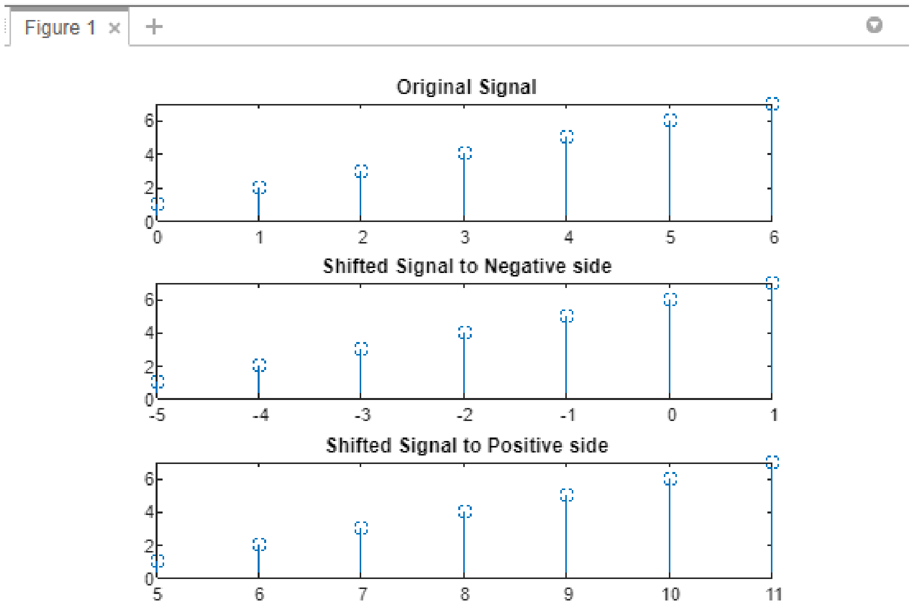
Input (a):

```
%Program to shift the signal by a given factor
function[y,n]=SignalShift(x,m,k)
%y(n)=x(n-k)
k=[0:6];
x=[1,2,3,4,5,6,7];
subplot(3,1,1)
stem(k,x)
title('Original Signal')
m=5;
n=k-m;
y=x;
subplot(3,1,2)
stem(n,y)
title('Shifted Signal to Negative side')
m=-5;
n=k-m;
y=x;
subplot(3,1,3)
stem(n,y)
title('Shifted Signal to Positive side')
end
```

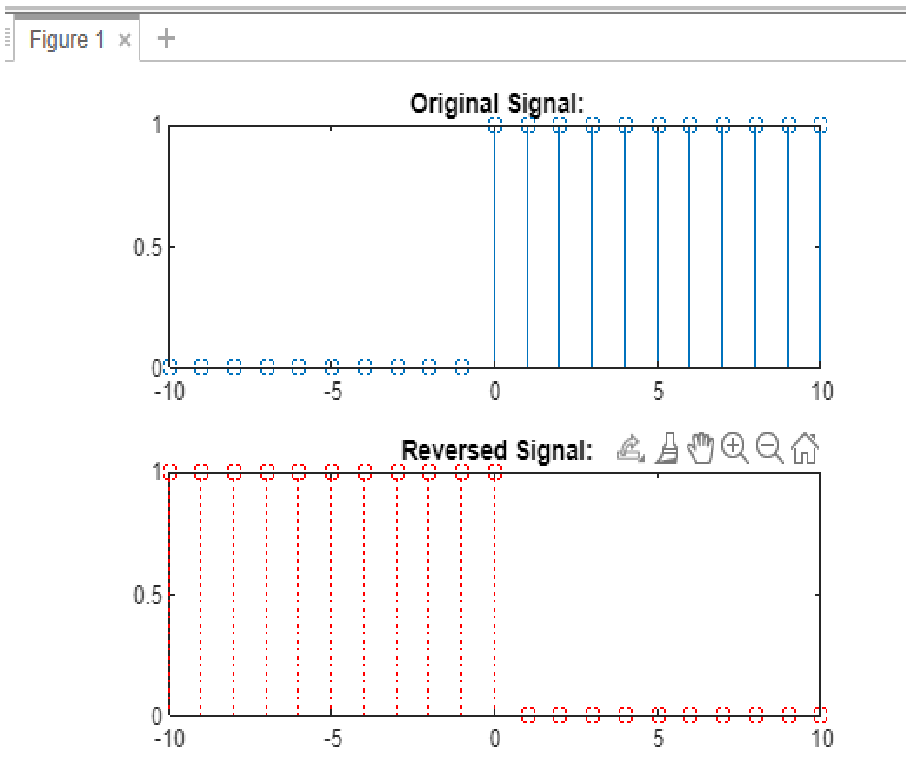
Input (b):

```
%Program for finding time reversal of a signal
n1=-10;
n2=10;
n0=0;
N=[n1:n2];
% Reversal function needs only X(output) & N(Range)
x=unitstep(n0,n1,n2);
subplot(2,1,1)
stem(N,x)
title('Original Signal:')
X2=fliplr(x);
N2=-fliplr(N);
subplot(2,1,2)
stem(N2,X2,':o r')
title('Reversed Signal:')
```

Output (a):



Output (b):



Experiment no. 4

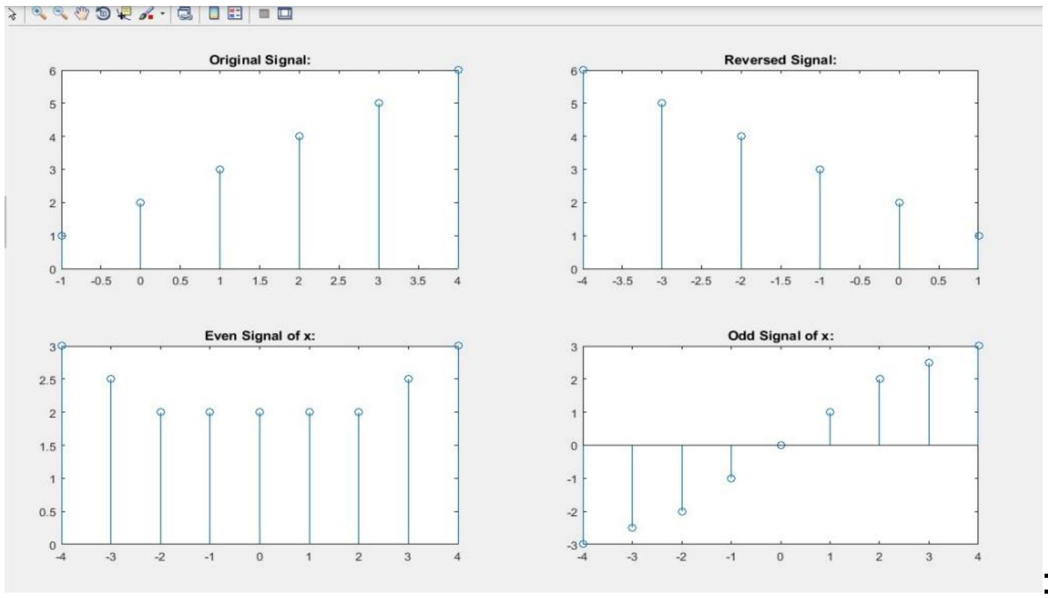
Aim:

Generating Even and Odd Signals

Input:

```
%PROGRAM TO FIND THE ODD AND EVEN COMPONENT OF A SIGNAL
x=[1,2,3,4,5,6]
n=[-1:4];
subplot(2,2,1)
stem(n,x)
title('Original Signal:')
n1=-fliplr(n)
x1=fliplr(x)
subplot(2,2,2)
stem(n1,x1)
title('Reversed Signal:')
xe=0.5.*SignalAdd(x,x1,n,n1)
rf=[min(min(n1),min(n)):max(max(n1),max(n))]
subplot(2,2,3)
stem(rf,xe)
title('Even Signal of x:')
xo=0.5.*SignalAdd(x,-x1,n,n1)
subplot(2,2,4)
stem(rf,xo)
title('Odd Signal of x:')
```

Output:



Experiment no. 5

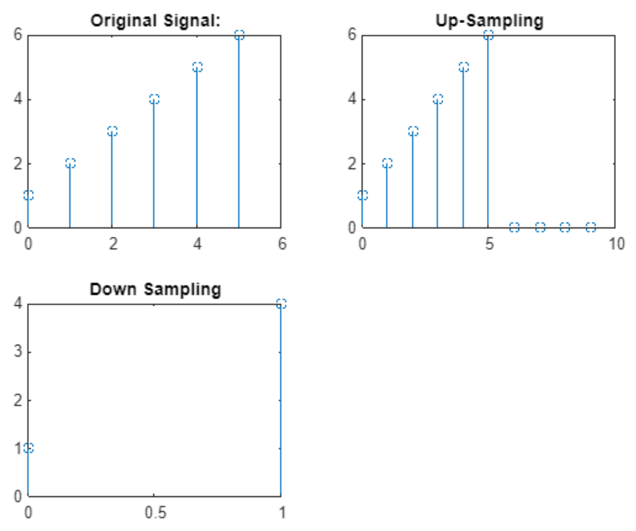
Aim:

Program to do Time Scaling (Up Sampling and Down Sampling)

Input:

```
%Program to do Time Scaling (Up Sampling and Down Sampling)
x=[1,2,3,4,5,6]
n=[0:5]
subplot(2,2,1)
stem(n,x)
title('Original Signal:')
nxf=9;
nxl=0
nx=[nxl:nxf];
y2=zeros(1,10)
y2(find((nx>=min(n))&(nx<=max(n))==1))=x
subplot(2,2,2)
stem(nx,y2)
title('Up-Sampling')
new=mod(n,3)
new=(new==0)
y3=x(find(new==1))
n1=0:length(y3)-1;
subplot(2,2,3)
stem(n1,y3)
title('Down Sampling')
```

Output:



Experiment no. 6

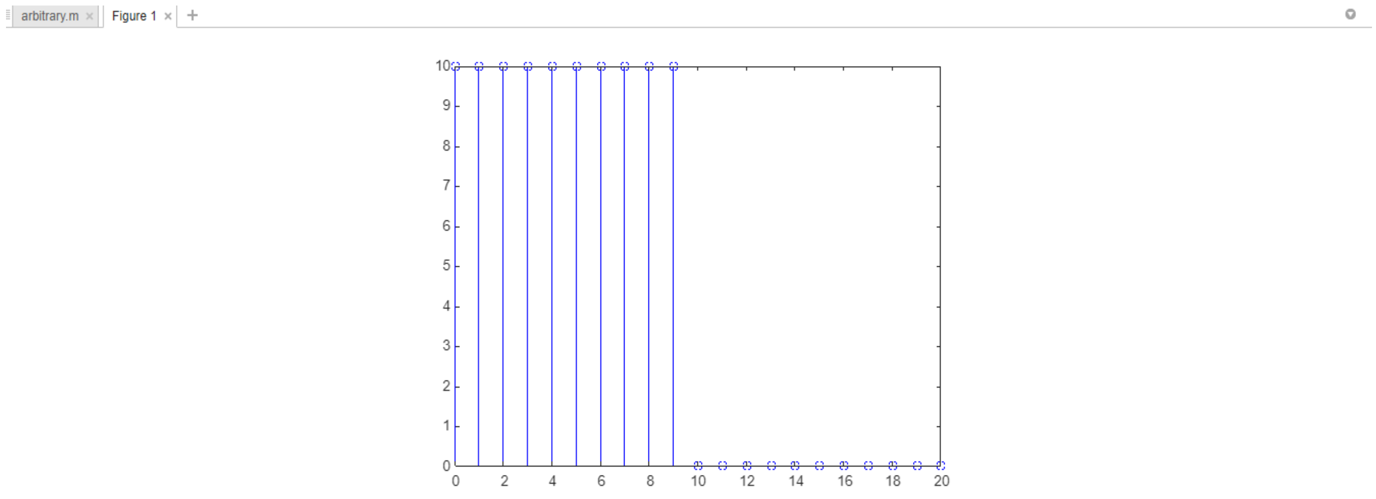
Aim:

Generating and plot the discrete time sequences in a given interval.

Input:

```
%Program to plot the function % x(n)= 10*[u(n)-u(n-10)] 0<=n>=20
function[y,n]=arbitrary(n1,n2)
n=[n1:n2]
y=10*[unitstep(0,n1,n2) - unitstep(10,n1,n2)]
stem(n,y,'b')
end
```

Output:



:

Experiment no. 7

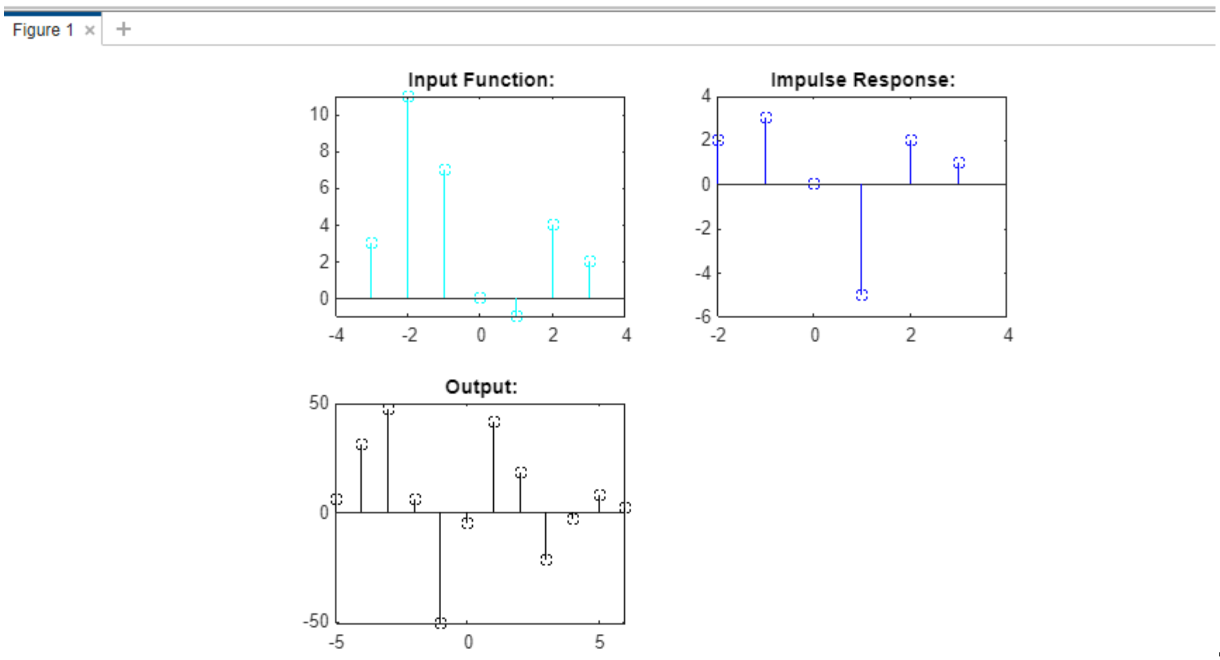
Aim:

Program for Convolution of Discrete Time Sequence.

Input:

```
%Program To find the convolution of the given x and h
x=[3,11,7,0,-1,4,2]
rx=[-3:3]
subplot(2,2,1)
stem(rx,x,'c')
title('Input Function:')
rh=[-2:3]
h=[2,3,0,-5,2,1]
subplot(2,2,2)
stem(rh,h,'b')
title('Impulse Response:')
nzl=rx(1)+rh(1)
nzr=rx(length(x))+ rh(length(h));
nz=nzl:nzr;
z=conv(x,h);
subplot(2,2,3)
stem(nz,z,'k')
title('Output:')
```

Output:



:

Experiment no. 8

Aim:

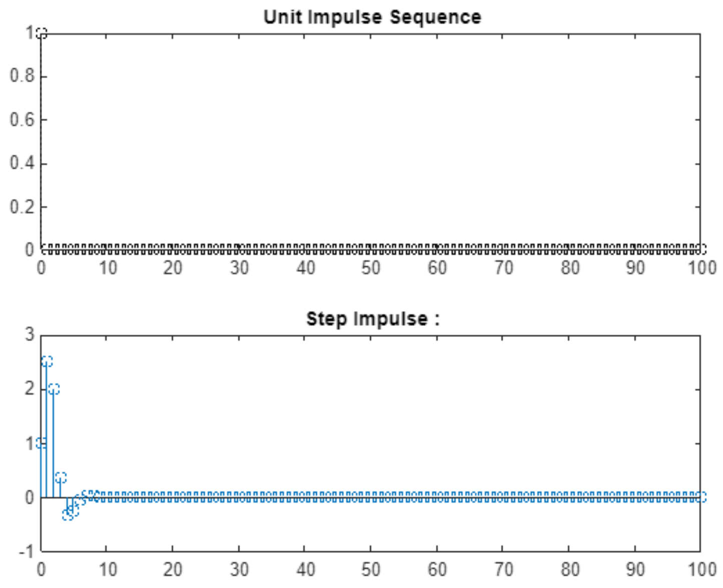
To find the solution of the given Difference Equation.

Input:

```
%y(n)-0.5y(n-1)+0.25y(n-2)=x(n)+2x(n-1)+x(n-3)
% n=[0:100];
% Compute and Plot the impulse response of a system over range 0<n<100
% Determine the Stability of the system from this impulse response.
b=[1,2,1];
a=[1,-0.5,0.25];
n=[0:100];
h=impz(b,a,101);
subplot(2,1,1);
stem(n,h)
title('Impulse Response:')
x=unitimpulse(0,0,100);
s=filter(b,a,x);
subplot(2,1,2)
stem(n,s);
title('Step Impulse :')
z=sum(abs(h));
fprintf('\n The Sum of Impulse Response:%f',z)
```

Output:

```
>>>soldiffeq
 The Sum of Impulse Response:6.571429
```



Experiment no. 9

Aim:

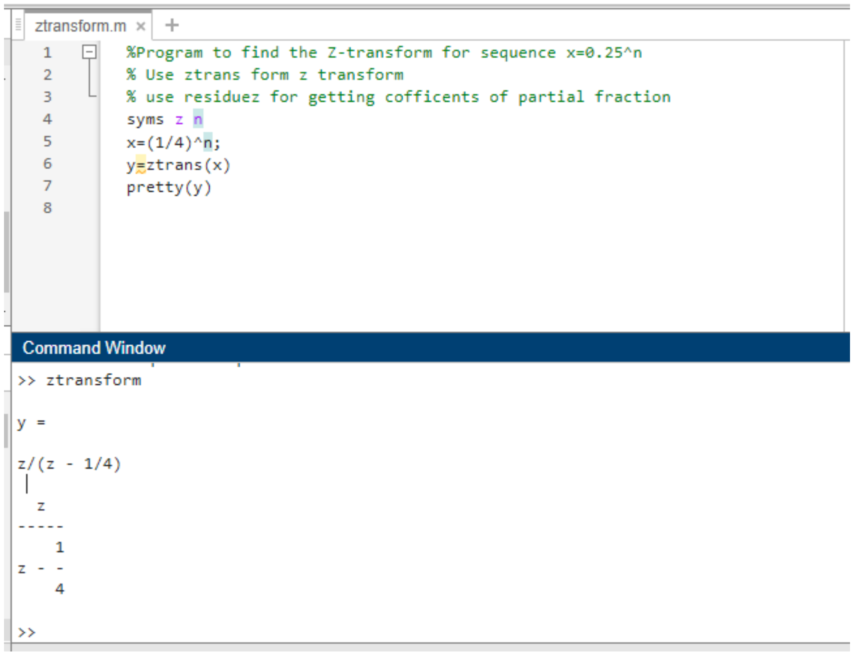
To find the Z Transform of the given Signal.

Input:

```
%Program to find the Z-transform for sequence x=0.25^n
% Use ztrans form z transform
% use residuez for getting coefficents of partial fraction
syms z n
x=(1/4)^n;
y=ztrans(x)
pretty(y)
```

Output:

```
ztransform

y =

z/(z - 1/4)

  z
-----
    1
z - -
    4
```



ztransform.m

```
%Program to find the Z-transform for sequence x=0.25^n
% Use ztrans form z transform
% use residuez for getting cofficents of partial fraction
syms z n
x=(1/4)^n;
y=ztrans(x)
pretty(y)
```

Command Window

```
>> ztransform

y =

z/(z - 1/4)

  z
-----
    1
z - -
    4

>>
```

Experiment no. 10

Aim:

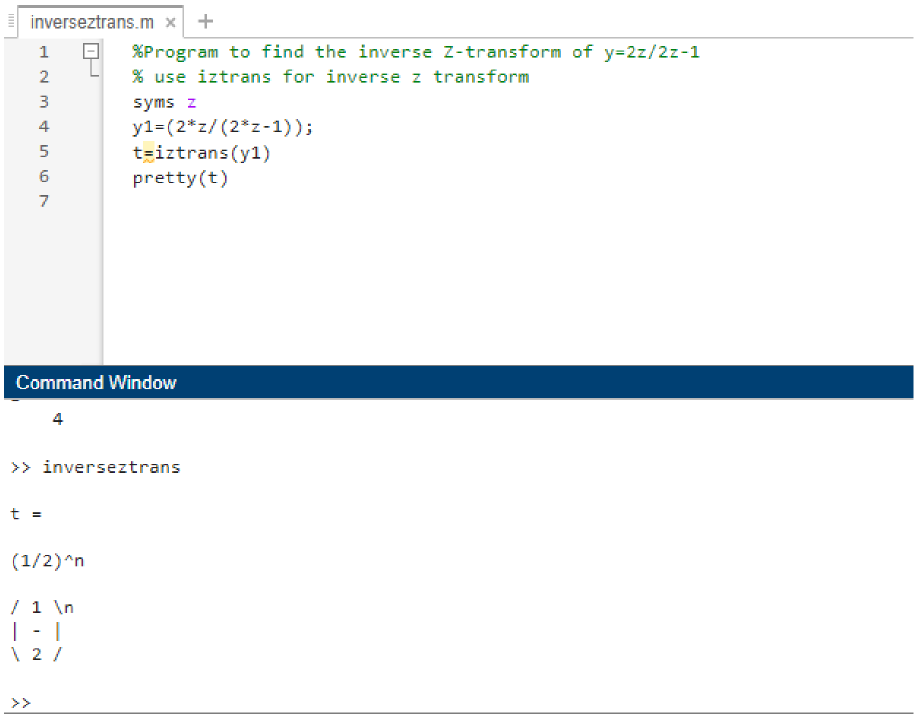
To find the inverse Z Transform of the given Signal.

Input:

```
%Program to find the Z-transform for sequence x=0.25^n
% Use ztrans form z transform
% use residuez for getting coefficents of partial fraction
syms z n
x=(1/4)^n;
y=ztrans(x)
pretty(y)
```

Output:

```
>> inverseztrans

t =

(1/2)^n

/ 1 \n
| - |
\ 2 /
```



inverseztrans.m

```
%Program to find the inverse Z-transform of y=2z/2z-1
% use iztrans for inverse z transform
syms z
y1=(2*z/(2*z-1));
t=iztrans(y1)
pretty(t)
```

Command Window

```
    4

>> inverseztrans

t =

(1/2)^n

/ 1 \n
| - |
\ 2 /

>>
```

Experiment no. 11

Aim:

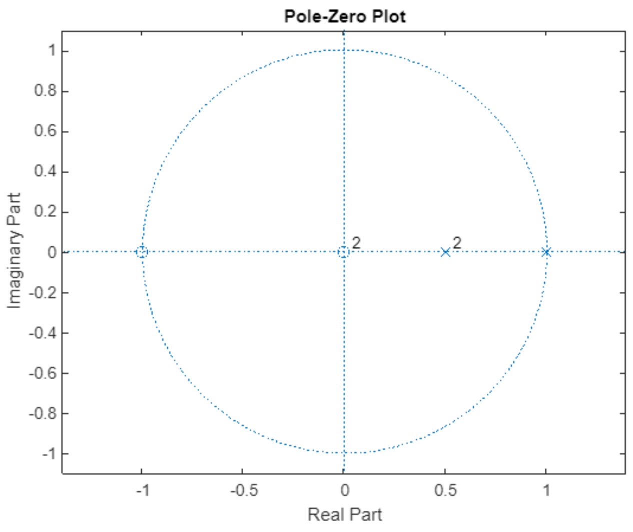
Program to Obtain the partial fraction and plot zero pole diagram.

Input:

```
%  Program to plot the pole zero diagram
%  using residuez for getting coefficents of partial fraction.
b=[4,4];
a=[4,-8,5,-1];
[R,p,c]=residuez(b,a)
% plot the Zplane
zplane(b,a)
```

Output:

```
>>>Polezero
   R =

      8.0000 + 0.0000i
     -4.0000 - 0.0000i
     -3.0000 + 0.0000i

   p =

      1.0000 + 0.0000i
      0.5000 + 0.0000i
      0.5000 - 0.0000i

   c =

        []
```



Experiment no. 12

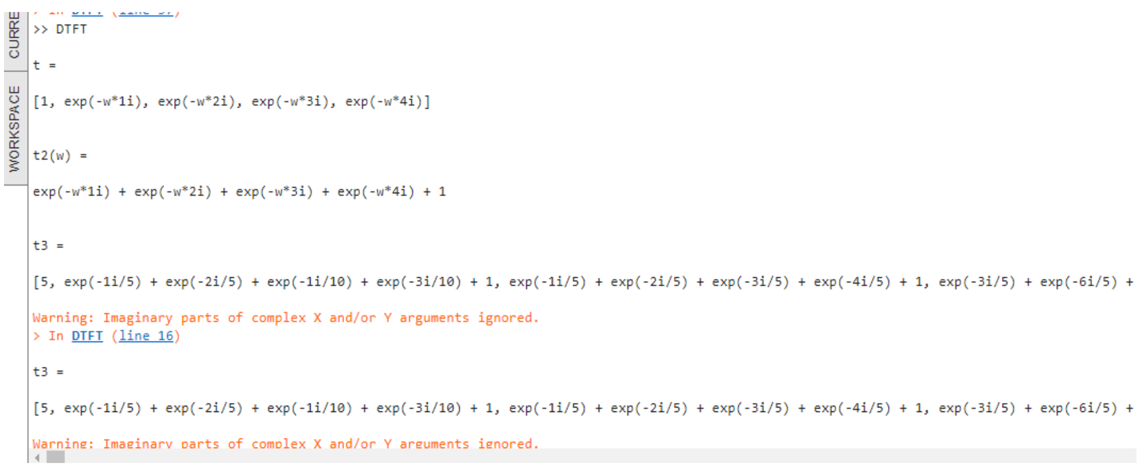
Aim:

To find the Discrete Time Fourier Transform of a given sequence.

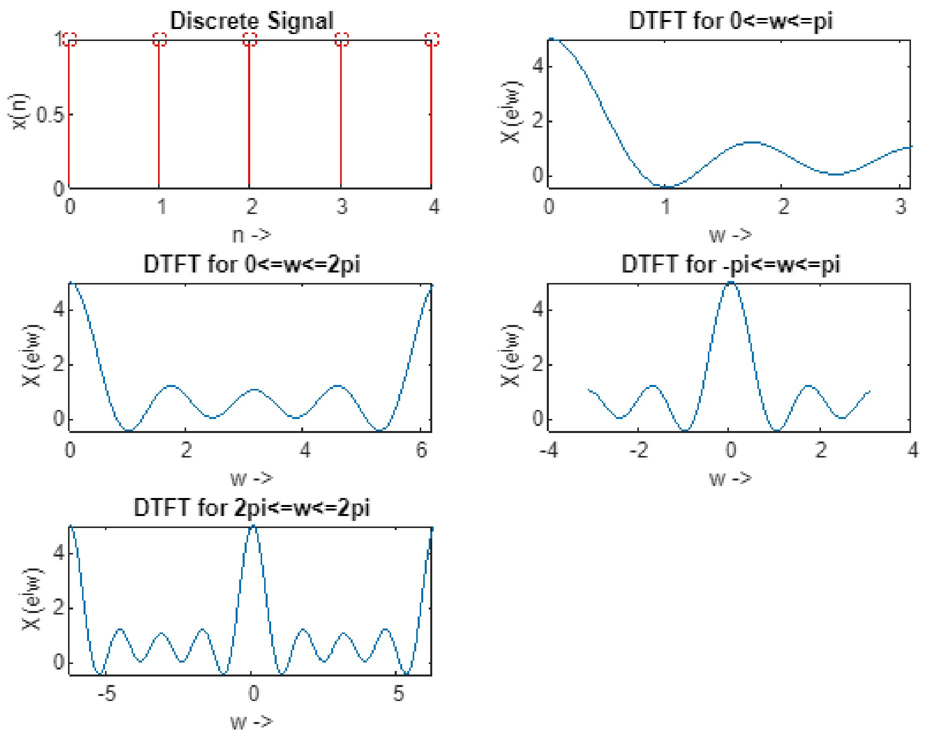
Input:

```
% Program to find the DTFT of a sequence
syms w
x=[1,1,1,1,1];
n=[0,1,2,3,4];
subplot(3,2,1)
stem(n,x,'r')
xlabel(' n -> ')
ylabel(' x(n) ')
title('Discrete Signal')
t=x.*exp(-i*n*w)
t2(w)=sum(t)
%t2(0)
w1=[0:0.1:pi];
t3=t2(w1)
subplot(3,2,2)
plot(w1,t3)
xlabel(' w -> ')
ylabel(' X(e^jw) ')
title('DTFT for 0<=w<=pi')
w2=[0:0.1:2*pi];
t3=t2(w2)
subplot(3,2,3)
plot(w2,t3)
xlabel(' w -> ')
ylabel(' X(e^jw) ')
title('DTFT for 0<=w<=2pi')
w3=[-pi:0.1:pi];
t3=t2(w3)
subplot(3,2,4)
plot(w3,t3)
xlabel(' w -> ')
ylabel(' X(e^jw) ')
title('DTFT for -pi<=w<=pi')
w4=[-2*pi:0.1:2*pi];
t3=t2(w4)
subplot(3,2,5)
plot(w4,t3)
xlabel(' w -> ')
ylabel(' X(e^jw) ')
title('DTFT for 2pi<=w<=2pi')
```

Output:



```
>> DTFT

t =

[1, exp(-w*1i), exp(-w*2i), exp(-w*3i), exp(-w*4i)]


t2(w) =

exp(-w*1i) + exp(-w*2i) + exp(-w*3i) + exp(-w*4i) + 1


t3 =

[5, exp(-1i/5) + exp(-2i/5) + exp(-1i/10) + exp(-3i/10) + 1, exp(-1i/5) + exp(-2i/5) + exp(-3i/5) + exp(-4i/5) + 1, exp(-3i/5) + exp(-6i/5) +

Warning: Imaginary parts of complex X and/or Y arguments ignored.
> In DTFT (line 16)

t3 =

[5, exp(-1i/5) + exp(-2i/5) + exp(-1i/10) + exp(-3i/10) + 1, exp(-1i/5) + exp(-2i/5) + exp(-3i/5) + exp(-4i/5) + 1, exp(-3i/5) + exp(-6i/5) +

Warning: Imaginary parts of complex X and/or Y arguments ignored.
```



Experiment no. 13

Aim:

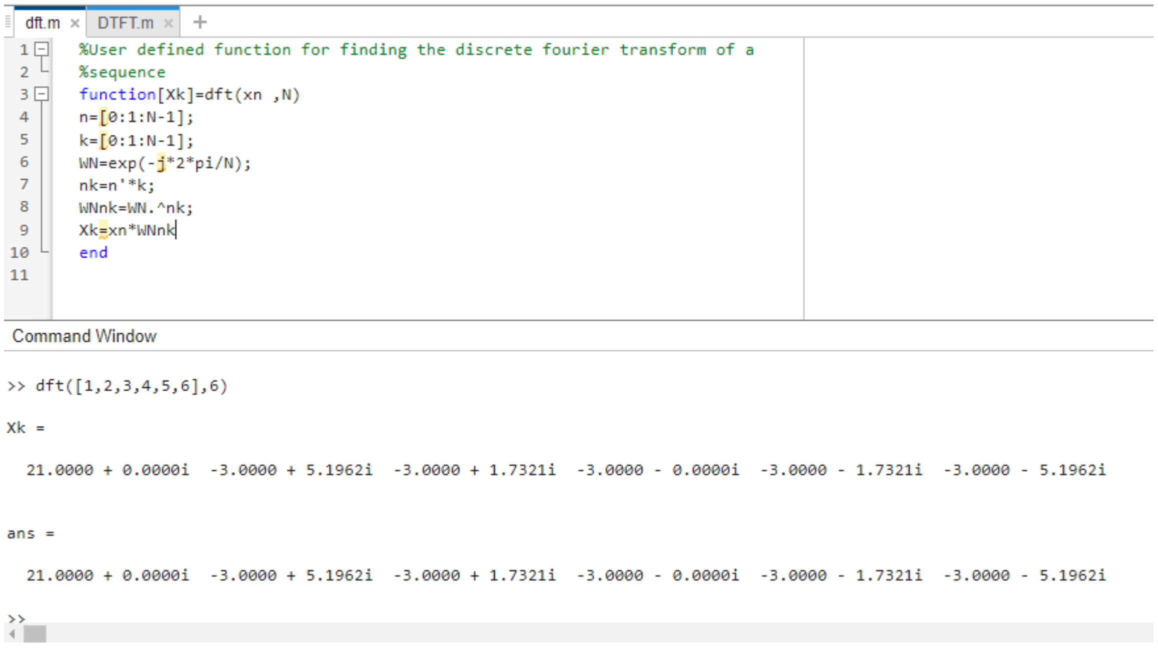
To find the Discrete Fourier Transform of the given sequence.

Input:

```
%User defined function for finding the discrete fourier transform of a
%sequence
function[Xk]=dft(xn ,N)
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
Xk=xn*WNnk
end
```

Output:

```
>> dft([1,2,3,4,5,6],6)

Xk =

  21.0000 + 0.0000i    -3.0000 + 5.1962i    -3.0000 + 1.7321i    -3.0000 - 0.0000i
-3.0000 - 1.7321i      -3.0000-5.1962i

ans =

  21.0000 + 0.0000i    -3.0000 + 5.1962i    -3.0000 + 1.7321i    -3.0000 - 0.0000i
-3.0000 - 1.7321i    -3.0000 - 5.1962i
```



```
%User defined function for finding the discrete fourier transform of a
%sequence
function[Xk]=dft(xn ,N)
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
Xk=xn*WNnk
end
```

Command Window

```
>> dft([1,2,3,4,5,6],6)

Xk =

  21.0000 + 0.0000i  -3.0000 + 5.1962i  -3.0000 + 1.7321i  -3.0000 - 0.0000i  -3.0000 - 1.7321i  -3.0000 - 5.1962i


ans =

  21.0000 + 0.0000i  -3.0000 + 5.1962i  -3.0000 + 1.7321i  -3.0000 - 0.0000i  -3.0000 - 1.7321i  -3.0000 - 5.1962i

>>
```

Experiment no. 14

Aim:

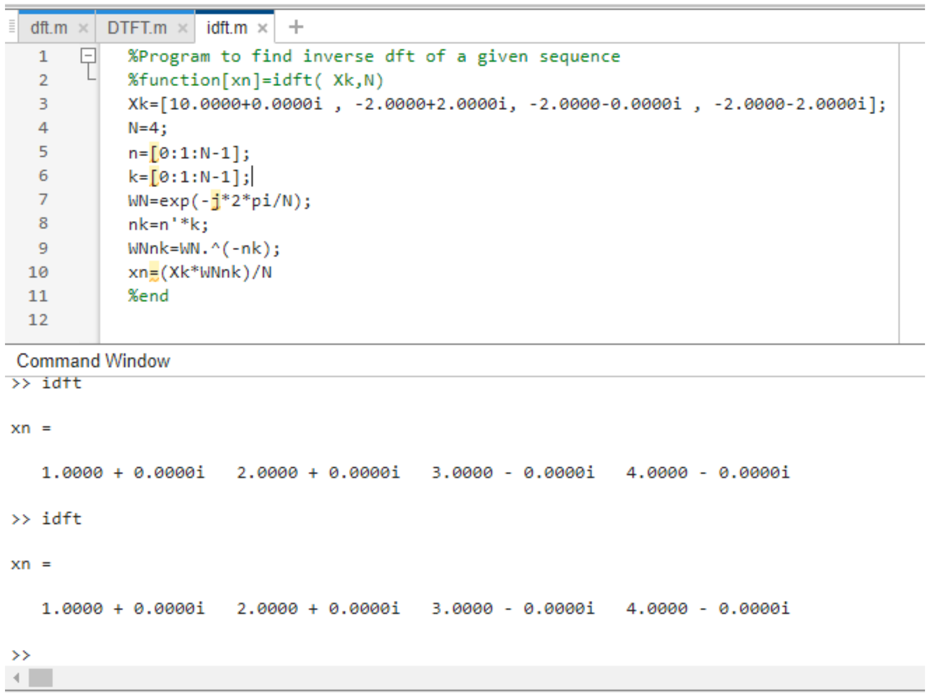
To find the Inverse Discrete Fourier Transform of the given sequence.

Input:

```
%Program to find inverse dft of a given sequence
%function[xn]=idft( Xk,N)
Xk=[10.0000+0.0000i , -2.0000+2.0000i, -2.0000-0.0000i , -2.0000-2.0000i];
N=4;
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^(-nk);
xn=(Xk*WNnk)/N
%end
```

Output:

```
>>>idft

   xn =

      1.0000 + 0.0000i     2.0000 + 0.0000i     3.0000 - 0.0000i     4.0000 - 0.0000i
```



```
%Program to find inverse dft of a given sequence
%function[xn]=idft( Xk,N)
Xk=[10.0000+0.0000i , -2.0000+2.0000i, -2.0000-0.0000i , -2.0000-2.0000i];
N=4;
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^(-nk);
xn=(Xk*WNnk)/N
%end
```

```
>> idft

xn =

   1.0000 + 0.0000i   2.0000 + 0.0000i   3.0000 - 0.0000i   4.0000 - 0.0000i

>> idft

xn =

   1.0000 + 0.0000i   2.0000 + 0.0000i   3.0000 - 0.0000i   4.0000 - 0.0000i

>>
```

Experiment no. 15

Aim:

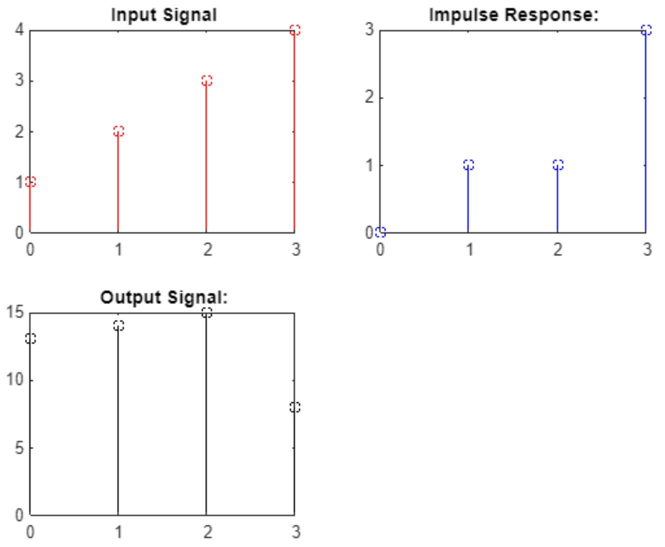
To find the Circular Convolution of given sequence

Input:

```
% Program for finding out circular Convolution of given x[n] and h[n].
n=[0:3];
x=[1,2,3,4];
subplot(2,2,1)
stem(n,x,'r')
title('Input Signal')
h=[0,1,1,3];
subplot(2,2,2)
stem(n,h,'b')
title('Impulse Response:')
y=cconv(x,h,4)
subplot(2,2,3)
stem(n,y,'k')
title('Output Signal:')
```

Output:

```
>> CircConv

y =

    13    14    15     8
```



Experiment no. 16

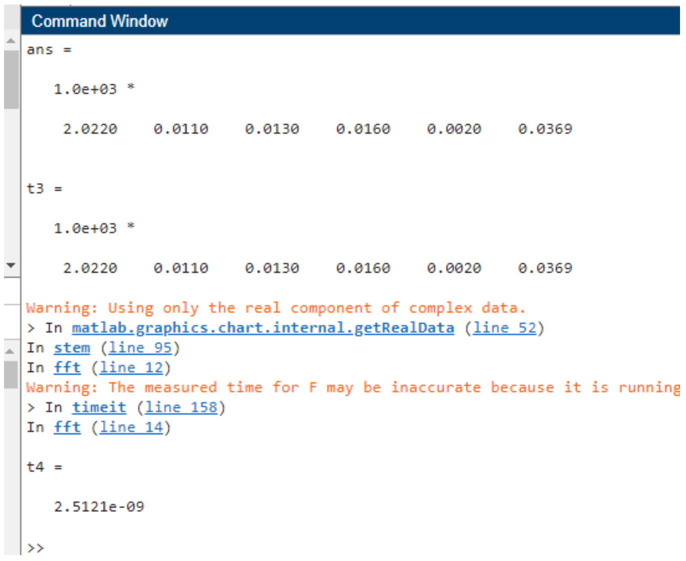
Aim:

Implementation of Fast Fourier Transform and Comparison with Discrete Fourier Transform.

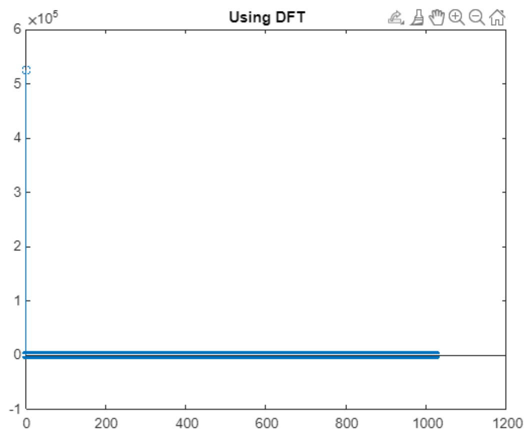
Input:

```
%Program for implementing fast fourier transform (fft)
x=[0:1:1023]
n=[0:1:1023]
N=1024;
clock
%etime(t1,t2)
t3=clock
WN=exp(-j*2*pi/N);
nk=n'*n;
WNnk=WN.^nk;
Xk=x*WNnk;
stem(n,Xk)
title('Using DFT')
t4=timeit(@() Xk)
```

Output:



```
ans =

   1.0e+03 *

    2.0220    0.0110    0.0130    0.0160    0.0020    0.0369


t3 =

   1.0e+03 *

    2.0220    0.0110    0.0130    0.0160    0.0020    0.0369

Warning: Using only the real component of complex data.
> In matlab.graphics.chart.internal.getRealData (line 52)
In stem (line 95)
In fft (line 12)
Warning: The measured time for F may be inaccurate because it is running
> In timeit (line 158)
In fft (line 14)

t4 =

   2.5121e-09

>>
```


Using DFT
×10^5