1. OPERATION ON MATRICES

Aim: To perform basic operations on matrices.
S**oftware used:** MATLAB.
Program:

```
>> x=[1 2 1 ;2 1 2;1 2 1]
>>y=[1 1 1 ;2 2 2;3 3 3]
>>z1=x+y
>>z2=x-y
>>z3=x*y
>>z4=x.*y
>>z5=x./y
>>z6=x(:,2)
>>x=[x(1,:);x(2,:);[1 1 1];x(3,:)]
>>x(3,:)=[1 2 1]
>> x(2,:)=[]
```

Output graphs:

```
x =
    1    2    1
    2    1    2
    1    2    1

y =
    1    1    1
    2    2    2
    3    3    3

z1 =
    2    3    2
    4    3    4
    4    5    4

z2 =

    0    1    0
    0   -1    0
   -2   -1   -2
```

```
z3 =
     8     8     8
    10    10    10
     8     8     8

z4 =
     1     2     1
     4     2     4
     3     6     3

z5 =
    1.0000    2.0000    1.0000
    1.0000    0.5000    1.0000
    0.3333    0.6667    0.3333

z6 =
     2
     1
     2

x =
     1     2     1
     2     1     2
     1     1     1
     1     2     1

x =
     1     2     1
     2     1     2
     1     2     1
     1     2     1

x =
     1     2     1
     1     2     1
     1     2     1
```

Result:
Hence the matrix operation is done using MATLAB.

Conclusion & Discussions:
The various types of matrix operations are done.

2(a). TYPES OF SIGNALS

Aim: To identify and plot graph of various continuous time signals
S**oftware used:** MATLAB.
Theory:

- Unit impulse function gives value 1 only at origin, otherwise 0.
- Unit Step function gives value 1 for all values of t greater than 0, otherwise 0.
- Unit ramp function gives a straight line passing through origin with slope 1.
- Unit parabolic function is a signal whose magnitude increases with the square of time. It can be obtained by integrating unit ramp.
- Rectangular pulse has value 1 for t value -0.5 to 0.5.
- Sinc function is sine of t divided by t.
- Signum function is 1 for all positive t and -1 for all negative values of t.
- A sinusoidal function is a function that is like a sine function in the sense that the function can be produced by shifting, stretching or compressing the sine function.
- Saw tooth wave is a function or waveform that repeatedly ramps upwards (usually linearly) and then sharply drops.

Program:

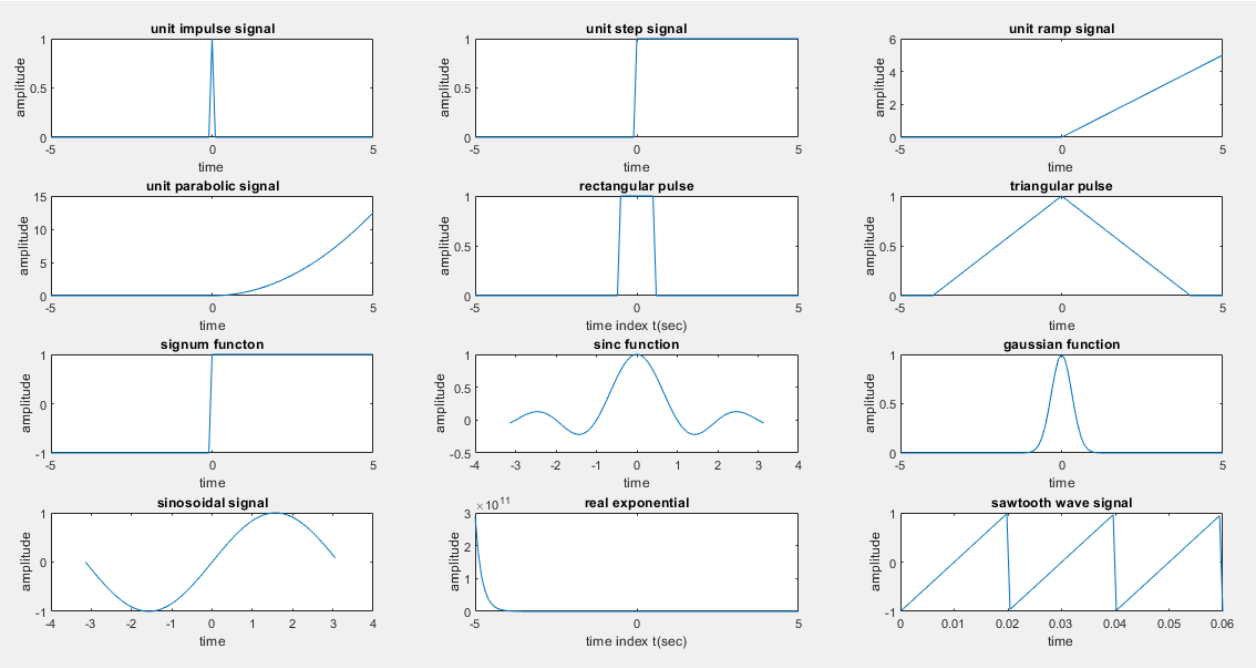
```
%1. unit impulse
t=-5:0.1:5;
x1=1.*(t==0);
x2=0.*(t~=0);
x=x1+x2;
subplot(4,3,1);
plot(t,x);
xlabel('time ');
ylabel('amplitude');
title('unit impulse signal');

% 2.unit step
t=-5:0.1:5;
x1=1.*(t>=0);
x2=0.*(t<0);
x=x1+x2;
subplot(4,3,2);
plot(t,x);
xlabel('time');
ylabel('amplitude');
title('unit step signal');
```

```
%3. unitramp
t=-5:0.1:5;
x1=t.*(t>=0);
x2=0.*(t<0);
x=x1+x2;
subplot(4,3,3);
plot(t,x);
xlabel('time ');
ylabel('amplitude');
title('unit ramp signal');

%4. unitparabolic
t=-5:0.1:5;
x1=t.^2/2.*(t>=0);
x2=0.*(t<0);
x=x1+x2;
subplot(4,3,4);
plot(t,x);
xlabel('time');
ylabel('amplitude');
title('unit parabolic signal');

%5. rectangular pulse
t=-5:0.1:5;
x1=1.*(abs(t)<=0.5);
x2=0.*(abs(t)>0.5);
x=x1+x2;
subplot(4,3,5);
plot(t,x);
xlabel('time index t(sec)');
ylabel('amplitude');
title('rectangular pulse');

%6. triangular pulse
t=-5:0.1:5;
x1=(1-abs(t)/4).*(abs(t)<=4);
x2=0.*(abs(t)>4);
x=x1+x2;
subplot(4,3,6);
plot(t,x);
xlabel('time');
ylabel('amplitude');
title('triangular pulse');

%7. signum func
```

```
t=-5:0.1:5;
x1=1.*(t>=0);
x2=-1.*(t<0);
x=x1+x2;
subplot(4,3,7);
plot(t,x);
xlabel('time');
ylabel('amplitude');
title('signum functon');

%8) sinc func
t=-pi:0.01:pi;
x=sinc(t)
subplot(4,3,8);
plot(t,x);
xlabel('time');
ylabel('amplitude');
title('sinc function');

%9) gaussian func
t=-5:0.1:5;
x=exp(-5*t.^2);
subplot(4,3,9);
plot(t,x);
xlabel('time');
ylabel('amplitude');
title('gaussian function');

%10) sinusoidal signal
t=-pi:0.1:pi;
x=sin(t);
subplot(4,3,10);
plot(t,x);
xlabel('time');
ylabel('amplitude');
title('sinosoidal signal');

%11) real exponential
t=-5:0.1:5;
x=4*exp(-5*t);
subplot(4,3,11);
plot(t,x);
xlabel('time index t(sec)');
ylabel('amplitude');
title('real exponential');
```

```
%12.sawtooth wave
f=50;
T=1/f;
t=0:3*T/100:3*T;
x=sawtooth(2*pi*f*t);
subplot(4,3,12);
plot(t,x);
xlabel('time ');
ylabel('amplitude');
title('sawtooth wave signal');
```

Output graphs:



Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The continuous time signals of various signals are plotted.

2(b). TYPES OF SIGNALS

Aim: To identify and plot graph of various discrete time signals.
S**oftware used:** MATLAB.
Theory:

- Unit impulse function gives value 1 only at origin, otherwise 0.
- Unit Step function gives value 1 for all values of t greater than 0, otherwise 0.
- Unit ramp function gives a straight line passing through origin with slope 1.
- Unit parabolic function is a signal whose magnitude increases with the square of time. It can be obtained by integrating unit ramp.
- Rectangular pulse has value 1 for t value -0.5 to 0.5.
- Sinc function is sine of t divided by t.
- Signum function is 1 for all positive t and -1 for all negative values of t.
- A sinusoidal function is a function that is like a sine function in the sense that the function can be produced by shifting, stretching or compressing the sine function.
- Saw tooth wave is a function or waveform that repeatedly ramps upwards (usually linearly) and then sharply drops.

Program:

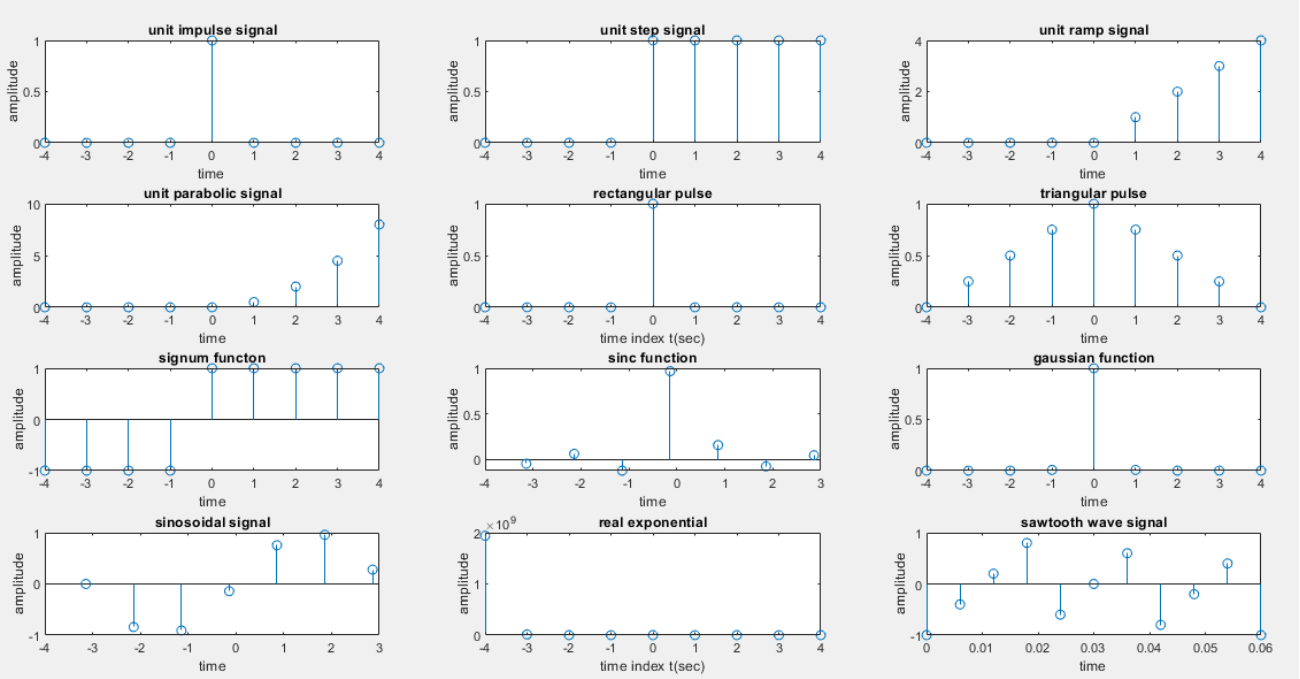
```
%1. unit impulse
t=-4:1:4;
x1=1.*(t==0);
x2=0.*(t~=0);
x=x1+x2;
subplot(4,3,1);
stem(t,x);
xlabel('time ');
ylabel('amplitude');
title('unit impulse signal');

% 2.unit step
t=-4:1:4;
x1=1.*(t>=0);
x2=0.*(t<0);
x=x1+x2;
subplot(4,3,2);
stem(t,x);
xlabel('time');
ylabel('amplitude');
title('unit step signal');
```

```
%3. unitramp
t=-4:1:4;
x1=t.*(t>=0);
x2=0.*(t<0);
x=x1+x2;
subplot(4,3,3);
stem(t,x);
xlabel('time ');
ylabel('amplitude');
title('unit ramp signal');

%4. unitparabolic
t=-4:1:4;
x1=t.^2/2.*(t>=0);
x2=0.*(t<0);
x=x1+x2;
subplot(4,3,4);
stem(t,x);
xlabel('time');
ylabel('amplitude');
title('unit parabolic signal');

%5. rectangular pulse
t=-4:1:4;
x1=1.*(abs(t)<=0.5);
x2=0.*(abs(t)>0.5);
x=x1+x2;
subplot(4,3,5);
stem(t,x);
xlabel('time index t(sec)');
ylabel('amplitude');
title('rectangular pulse');

%6. triangular pulse
t=-4:1:4;
x1=(1-abs(t)/4).*(abs(t)<=4);
x2=0.*(abs(t)>4);
x=x1+x2;
subplot(4,3,6);
stem(t,x);
xlabel('time');
ylabel('amplitude');
title('triangular pulse');
```

```
%7. signum func
t=-4:1:4;
x1=1.*(t>=0);
x2=-1.*(t<0);
x=x1+x2;
subplot(4,3,7);
stem(t,x);
xlabel('time');
ylabel('amplitude');
title('signum functon');

%8) sinc func
t=-pi:1:pi;
x=sinc(t)
subplot(4,3,8);
stem(t,x);
xlabel('time');
ylabel('amplitude');
title('sinc function');

%9) gaussian func
t=-4:1:4;
x=exp(-5*t.^2);
subplot(4,3,9);
stem(t,x);
xlabel('time');
ylabel('amplitude');
title('gaussian function');

%10) sinusoidal signal
t=-pi:1:pi;
x=sin(t);
subplot(4,3,10);
stem(t,x);
xlabel('time');
ylabel('amplitude');
title('sinosoidal signal');
 %11) real exponential
t=-4:1:4;
x=4*exp(-5*t);
subplot(4,3,11);
stem(t,x);
xlabel('time index t(sec)');
ylabel('amplitude');
title('real exponential');
```

```
 %12.sawtooth wave
f=50;
T=1/f;
t=0:3*T/10:3*T;
x=sawtooth(2*pi*f*t);
subplot(4,3,12);
stem(t,x);
xlabel('time ');
ylabel('amplitude');
title('sawtooth wave signal');
```

Output graphs:



Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The discrete time signals of various signals are plotted.

3(a). OPERATIONS OF SIGNALS

Aim: To identify and plot graph of various operations of signals on continuous time signals, to calculate power and energy.

S**oftware used:** MATLAB.

Theory:

- Time shifting: This kind of signal operation results in a positive or negative "shift" of the signal along its time axis. This means that the time-shifting operation results in the change of just the positioning of the signal without affecting its amplitude or span.
- Time scaling: If a constant is multiplied to the time axis then it is known as Time scaling.
- Amplitude scaling: Multiplication of a constant with the amplitude of the signal causes amplitude scaling.
- Time reversal: This causes the original signal to flip along its *y*-axis. That is, it results in the reflection of the signal along its vertical axis of reference.
- Power: Power is the rate (energy amount per time period) at which work is done or energy converted. The scientific unit of power is the watt (W), which is equal to one joule (energy amount) per second (time period).
- Energy: the capacity for doing work.

Program:

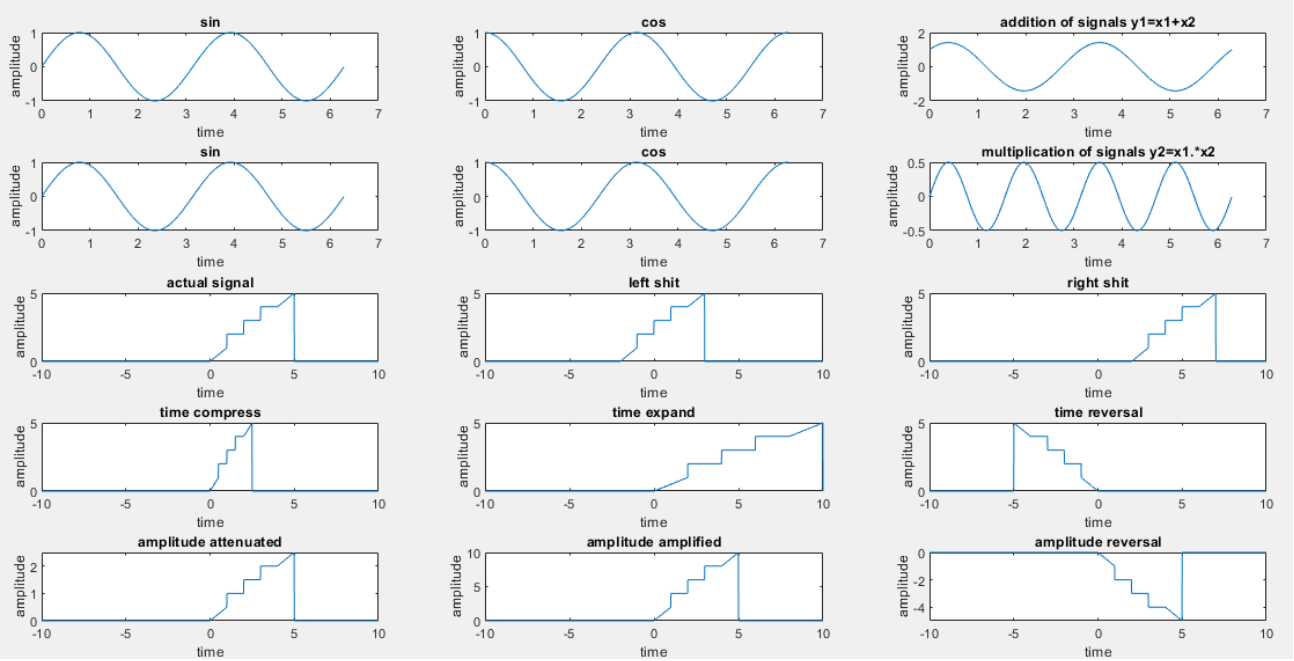
```
%sin
t=0:0.01:2*pi;
x1=sin(2*t);
subplot(5,3,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('sin')
%cos
t=0:0.01:2*pi;
x2=cos(2*t);
subplot(5,3,2)
plot(t,x2);
xlabel('time');
ylabel('amplitude');
title('cos')
% 1.Addition
t=0:0.01:2*pi;
x1=sin(2*t);
x2=cos(2*t);
y1=x1+x2;
subplot(5,3,3)
```

```
plot(t,y1)
xlabel('time')
ylabel('amplitude')
title('addition of signals y1=x1+x2')
%sin
t=0:0.01:2*pi;
x1=sin(2*t);
subplot(5,3,4);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('sin')
%cos
t=0:0.01:2*pi;
x2=cos(2*t);
subplot(5,3,5)
plot(t,x2);
xlabel('time')
ylabel('amplitude')
title('cos')
% 2.mulitplication
t=0:0.01:2*pi;
x1=sin(2*t);
x2=cos(2*t);
y2=x1.*x2;
subplot(5,3,6)
plot(t,y2)
xlabel('time')
ylabel('amplitude')
title('multiplication of signals y2=x1.*x2')
t=-10:0.01:10;
% 1.actual signal
y1=y(t);
subplot(5,3,7)
plot(t,y1)
xlabel('time')
ylabel('amplitude')
title('actual signal')
% 2.time left shift
y2=y(t+2);
subplot(5,3,8)
plot(t,y2)
xlabel('time')
ylabel('amplitude')
title('left shit')
```

```
 % 3. time right shift
y3=y(t-2);
subplot(5,3,9)
plot(t,y3)
xlabel('time')
ylabel('amplitude')
title('right shit')
% 4.time scaling (compress)
y4=y(2*t);
subplot(5,3,10)
plot(t,y4)
xlabel('time')
ylabel('amplitude')
title('time compress')
% 5.time scaling(expand)
y5=y(0.5*t);
subplot(5,3,11)
plot(t,y5)
xlabel('time')
ylabel('amplitude')
title('time expand')
% 6.time reversal
y6=y(-t);
subplot(5,3,12)
plot(t,y6)
xlabel('time')
ylabel('amplitude')
title('time reversal')
% 7.amplitude scaling (attenuated)
y7=0.5*y(t);
subplot(5,3,13)
plot(t,y7)
xlabel('time')
ylabel('amplitude')
title('amplitude attenuated')
% 8.amplitude scaling(amplified)
y8=2*y(t);
subplot(5,3,14)
plot(t,y8)
xlabel('time')
ylabel('amplitude')
title('amplitude amplified')
% 9.amplitude reversal
y9=-y(t);
subplot(5,3,15)
```

```
plot(t,y9)
xlabel('time')
ylabel('amplitude')
title('amplitude reversal')
function x=y(t)
x=t.*(t>=0&t<1)+2.*(t>=1&t<2)+3.*(t>=2&t<3)+4.*(t>=3&t<4)+t.*(t>=4&t<5);
end
%power and energy:
tmin=-5;
tmax=5;
dt=0.01;
t=tmin:dt:tmax;
x=10*sin(2*pi*t);
xsq=x.^2;
E=trapz(t,xsq);
P=E/(tmax-tmin);
disp(['energy:E is',num2str(E),'joules']);
disp(['power:P is',num2str(P),'watts']);
```

Output graphs:




```
energy:E is500joules
power:P is50watts
```

Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The continuous time signals of various operations are plotted, power and energy calculated.

3(b). OPERATIONS OF SIGNALS

Aim: To identify and plot graph of various operations of signals on discrete time signals, to calculate power and energy.

S**oftware used:** MATLAB.

Theory:

- Time shifting: This kind of signal operation results in a positive or negative "shift" of the signal along its time axis. This means that the time-shifting operation results in the change of just the positioning of the signal without affecting its amplitude or span.
- Time scaling: If a constant is multiplied to the time axis then it is known as Time scaling.
- Amplitude scaling: Multiplication of a constant with the amplitude of the signal causes amplitude scaling.
- Time reversal: This causes the original signal to flip along its *y*-axis. That is, it results in the reflection of the signal along its vertical axis of reference.
- Power : Power is the rate (energy amount per time period) at which work is done or energy converted. The scientific unit of power is the watt (W), which is equal to one joule (energy amount) per second (time period).
- Energy: the capacity for doing work.

Program:

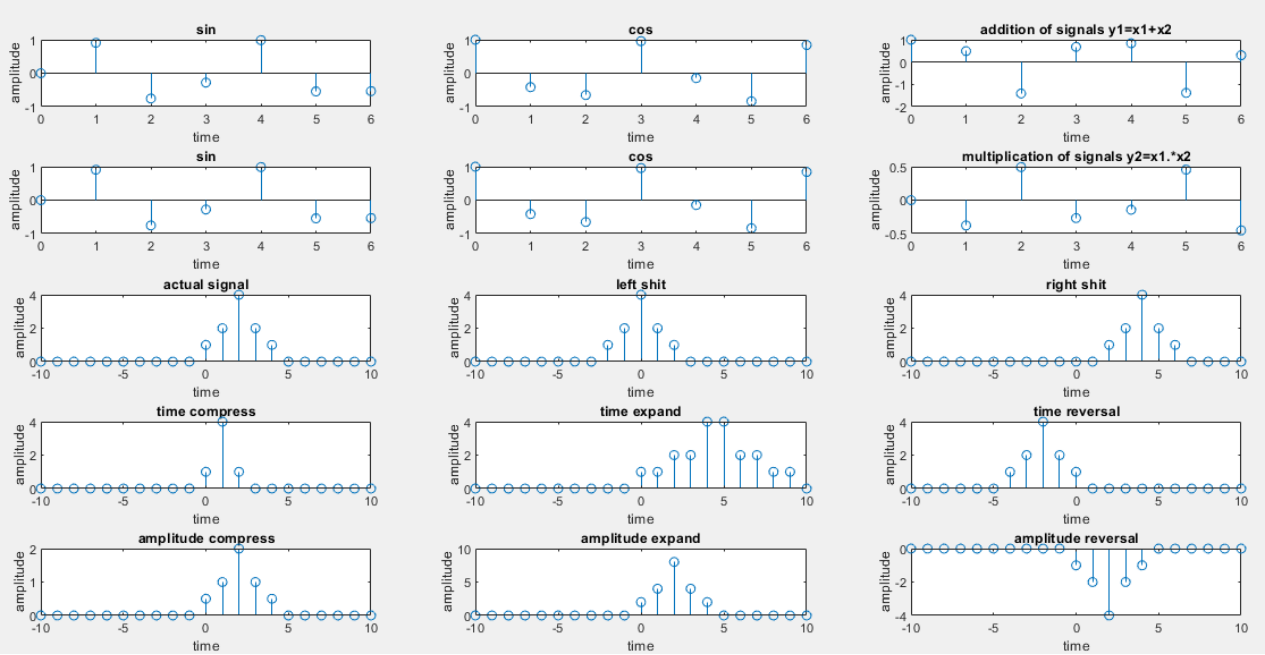
```
%sin
n=0:1:2*pi;
x1=sin(2*n);
subplot(5,3,1);
stem(n,x1);
xlabel('time');
ylabel('amplitude');
title('sin')
%cos
n=0:1:2*pi;
x2=cos(2*n);
subplot(5,3,2)
stem(n,x2);
xlabel('time');
ylabel('amplitude');
title('cos')
% 1.Addition
n=0:1:2*pi;
x1=sin(2*n);
x2=cos(2*n);
y1=x1+x2;
subplot(5,3,3)
```

```
stem(n,y1)
xlabel('time')
ylabel('amplitude')
title('addition of signals y1=x1+x2')
%sin
n=0:1:2*pi;
x1=sin(2*n);
subplot(5,3,4);
stem(n,x1);
xlabel('time');
ylabel('amplitude');
title('sin')
%cos
n=0:1:2*pi;
x2=cos(2*n);
subplot(5,3,5)
stem(n,x2);
xlabel('time')
ylabel('amplitude')
title('cos')
% 2.mulitplication
n=0:1:2*pi;
x1=sin(2*n);
x2=cos(2*n);
y2=x1.*x2;
subplot(5,3,6)
stem(n,y2)
xlabel('time')
ylabel('amplitude')
title('multiplication of signals y2=x1.*x2')
n=-10:1:10;
% 1.actual signal
y1=y(n);
subplot(5,3,7)
stem(n,y1)
xlabel('time')
ylabel('amplitude')
title('actual signal')
% 2.time left shift
y2=y(n+2);
subplot(5,3,8)
stem(n,y2)
xlabel('time')
ylabel('amplitude')
title('left shit')
```

```
% 3. time right shift
y3=y(n-2);
subplot(5,3,9)
stem(n,y3)
xlabel('time')
ylabel('amplitude')
title('right shit')
% 4.time scaling (compress)
y4=y(2*n);
subplot(5,3,10)
stem(n,y4)
xlabel('time')
ylabel('amplitude')
title('time compress')
% 5.time scaling(expand)
y5=y(0.5*n);
subplot(5,3,11)
stem(n,y5)
xlabel('time')
ylabel('amplitude')
title('time expand')
% 6.time reversal
y6=y(-n);
subplot(5,3,12)
stem(n,y6)
xlabel('time')
ylabel('amplitude')
title('time reversal')
% 7.amplitude scaling (compress)
y7=0.5*y(n);
subplot(5,3,13)
stem(n,y7)
xlabel('time')
ylabel('amplitude')
title('amplitude compress')
% 8.amplitude scaling(expand)
y8=2*y(n);
subplot(5,3,14)
stem(n,y8)
xlabel('time')
ylabel('amplitude')
title('amplitude expand')
% 9.amplitude reversal
y9=-y(n);
subplot(5,3,15)
```

```
stem(n,y9)
xlabel('time')
ylabel('amplitude')
title('amplitude reversal')
function x=y(n)
x=1.*(n>=0&n<1)+2.*(n>=1&n<2)+4.*(n>=2&n<3)+2.*(n>=3&n<4)+1.*(n>=4&n<5);
end
%power and energy
nmin=-8;
nmax=8;
dn=0.01;
n=nmin:dn:nmax;
y=10*sin(2*pi*n);
xsq=y.^2;
E=trapz(n,xsq);
P=E/(nmax-nmin);
disp(['energy:E is',num2str(E),'joules']);
disp(['power:P is',num2str(P),'watts']);
```

Output graphs:




```
energy:E is500joules
power:P is50watts
```

Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The discrete time signals of various operations are plotted, power and energy are calculated.

4(a). ODD, EVEN, REAL, IMAGINARY SIGNALS

Aim: To identify and plot graph of odd, even, real and imaginary parts of continuous time signals.
S**oftware used:** MATLAB.
Theory:

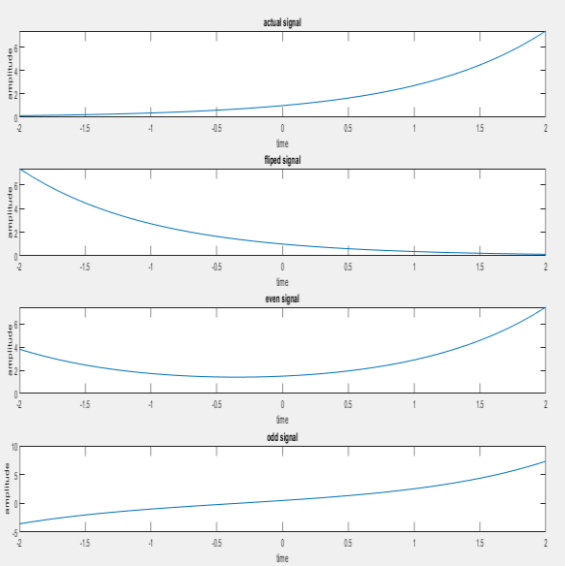
- **Even signal:** A signal is even if x(t) = x(-t).
- **Odd signal:** A signal is odd if x(t) = -x(-t).

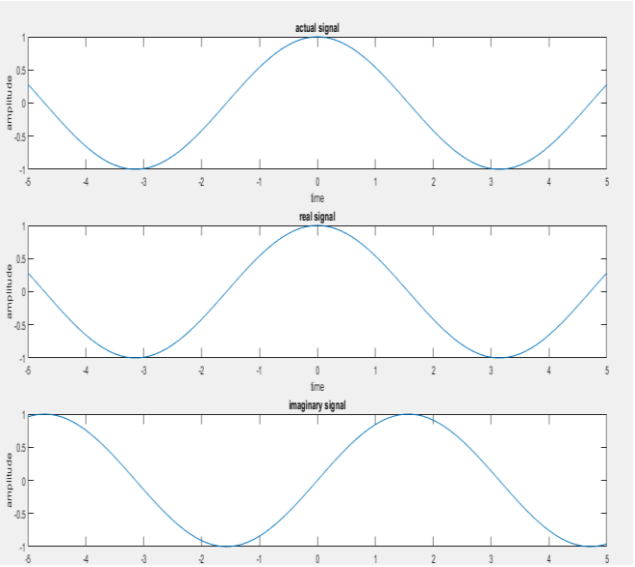
Program:

```
% Even and Odd
t=-2:0.1:2;
x1=exp(t);
x2=exp(-t);
xe=x1+x2./2;
xo=x1-x2./2;
subplot(4,1,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('actual signal');
subplot(4,1,2);
plot(t,x2);
xlabel('time');
ylabel('amplitude');
title('fliped signal');
subplot(4,1,3);
plot(t,xe);
xlabel('time');
ylabel('amplitude');
title('even signal');
subplot(4,1,4);
plot(t,xo);
xlabel('time');
ylabel('amplitude');
title('odd signal');

if(x1==x2)
   disp('signal is even');
else if(x1==-x2)
   disp('signal is odd');
else
   disp('signal is niether even nor odd');
end
end
```

```
% Real and imaginary
t=-5:0.1:5;
x1=exp(1i.*t);
x2=real(x1);
x3=imag(x1);
subplot(3,1,1);
plot(t,x1);
xlabel('time');
ylabel('amplitude');
title('actual signal');
subplot(3,1,2);
plot(t,x2);
xlabel('time');
ylabel('amplitude');
title('real signal');
subplot(3,1,3);
plot(t,x3);
xlabel('time');
ylabel('amplitude');
title('imaginary signal');
```

Output graphs:





Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The continuous time signals of odd, even, real, imaginary parts are plotted.

4(b). ODD, EVEN, REAL, IMAGINARY SIGNALS

Aim: To identify and plot graph of various operations of signals on discrete time signals.
S**oftware used:** MATLAB.
Theory:

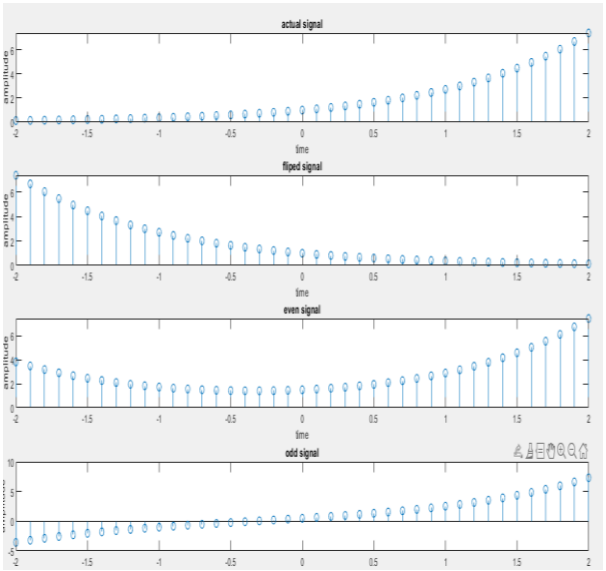
- **Even signal:** A signal is even if x(t) = x(-t).
- **Odd signal:** A signal is odd if x(t) = -x(-t).

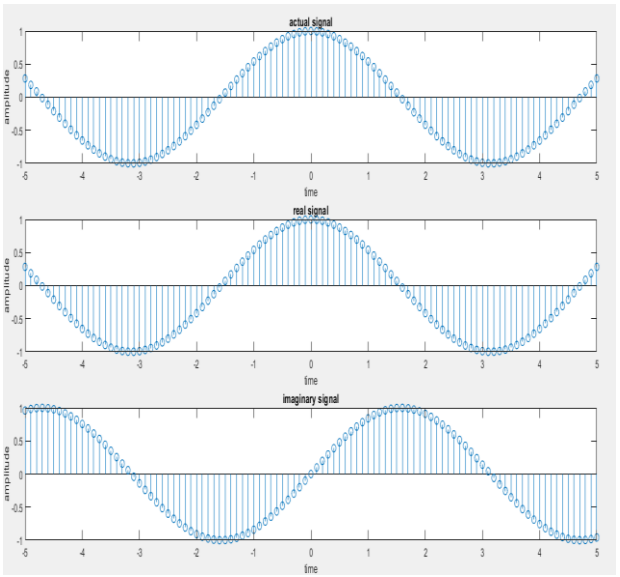
Program:

```
n=-2:0.1:2;
x1=exp(n);
x2=exp(-n);
xe=x1+x2./2;
xo=x1-x2./2;
subplot(4,1,1);
stem(n,x1);
xlabel('time');
ylabel('amplitude');
title('actual signal');
subplot(4,1,2);
stem(n,x2);
xlabel('time');
ylabel('amplitude');
title('fliped signal');
subplot(4,1,3);
stem(n,xe);
xlabel('time');
ylabel('amplitude');
title('even signal');
subplot(4,1,4);
stem(n,xo);
xlabel('time');
ylabel('amplitude');
title('odd signal');
if(x1==x2)
   disp('signal is even');
else if(x1==-x2)
   disp('signal is odd');
else
   disp('signal is niether even nor odd');
end
end
```

```
%Real and imaginary
n=-5:0.1:5;
x1=exp(1i.*n);
x2=real(x1);
x3=imag(x1);
subplot(3,1,1);
stem(n,x1);
xlabel('time');
ylabel('amplitude');
title('actual signal');
subplot(3,1,2);
stem(n,x2);
xlabel('time');
ylabel('amplitude');
title('real signal');
subplot(3,1,3);
stem(n,x3);
xlabel('time');
ylabel('amplitude');
title('imaginary signal');
```

Output graphs:





Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The discrete time signals of odd, even signals and real, imaginary parts are plotted.

5(a). CONVOLUTION THEOREM ON SIGNALS

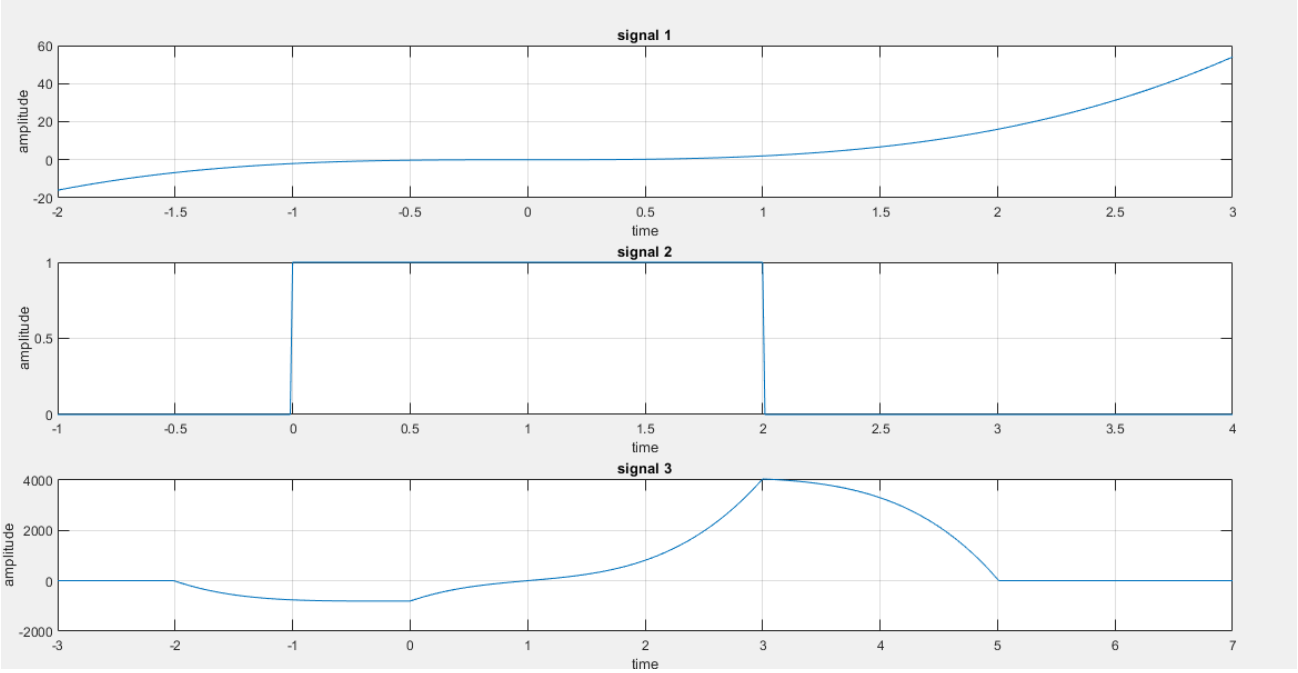
Aim: To plot graphically the convolution theorem on continuous time signals.
S**oftware used:** MATLAB.
Theory:
The convolution theorem for z transforms states that for any (real or) complex causal signals x and y, convolution in the time domain is multiplication in the z domain, i.e.,

$$Zz\{x*y\} = X(z)Y(z)$$

Program:

```
t1=-2:0.01:3;
x=2*(t1.^3);
t2=-1:0.01:4;
h=1.*(t2>=0&t2<=2)+0.*(t2<0&t2>2);
y=conv(x,h);
a=min(t1)+min(t2);
b=max(t1)+max(t2);

t3=a:0.01:b;

subplot(3,1,1);
plot(t1,x);
xlabel('time');
ylabel('amplitude');
title ('signal 1');
grid;

subplot(3,1,2);
plot(t2,h);
xlabel('time');
ylabel('amplitude');
title ('signal 2');
grid;

subplot(3,1,3);
plot(t3,y);
xlabel('time');
ylabel('amplitude');
title ('signal 3');
grid;
```

Output graphs:



Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The Convolution Theorem on Continuous Time Signals has been plotted.

5(b). CONVOLUTION THEOREM ON SEQUENCES

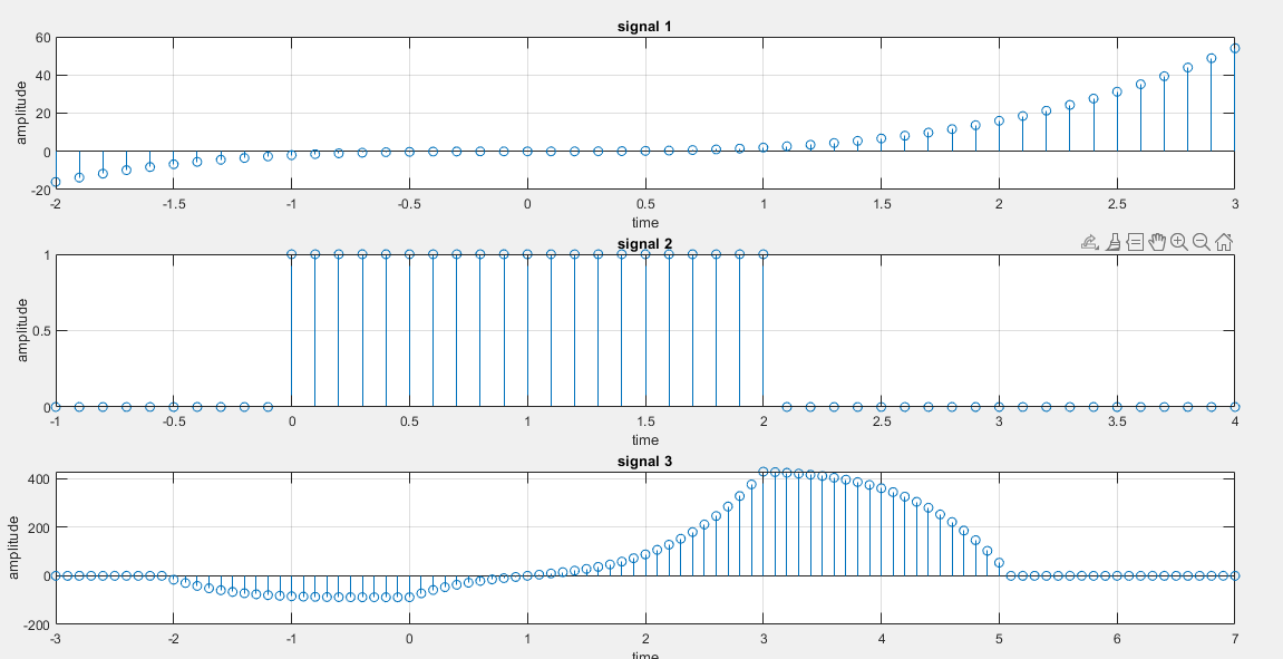
Aim: To plot graphically the convolution theorem on discrete time signals.
S**oftware used:** MATLAB.
Theory:
The convolution theorem for z transforms states that for any (real or) complex causal signals x and y, convolution in the time domain is multiplication in the z domain, i.e.,

$$Zz\{x*y\} = X(z)Y(z)$$

Program:

```
n1=-2:0.1:3;
x=2*(n1.^3);
n2=-1:0.1:4;
h=1.*(n2>=0&n2<=2)+0.*(n2<0&n2>2);
y=conv(x,h);
a=min(n1)+min(n2);
b=max(n1)+max(n2);

n3=a:0.1:b;

subplot(3,1,1);
stem(n1,x);
xlabel('time');
ylabel('amplitude');
title ('signal 1');
grid;

subplot(3,1,2);
stem(n2,h);
xlabel('time');
ylabel('amplitude');
title ('signal 2');
grid;

subplot(3,1,3);
stem(n3,y);
xlabel('time');
ylabel('amplitude');
title ('signal 3');
grid;
```

Output graphs:



Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
The Convolution Theorem on Discrete Time Signals has been plotted.

6.GIBB'S PHENOMENON

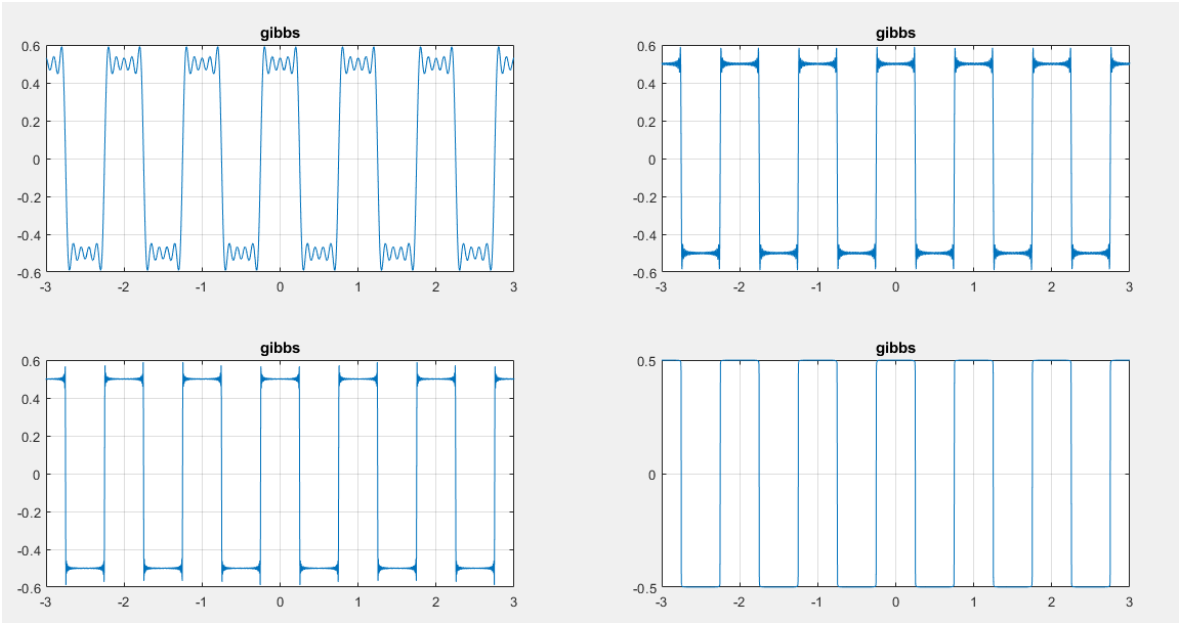
Aim: To plot Gibb's Phenomenon.
S**oftware used:** MATLAB.
Theory:
To describe a signal with a discontinuity in the time domain requires infinite frequency content. In practice, it is not possible to sample infinite frequency content. The truncation of frequency content causes a time domain ringing artifact on the signal, which is called the "Gibbs phenomenon".

Program:

```
t=-3:0.003:3;
w0=2*pi;
C0=zeros(1,length(t));
x=C0;
N=input('No. of components');
for n=1:N
   Cn=(1/(n*pi))*sin((n*pi)/2);
   Cnn=Cn;
   x=x+(Cn)*exp(1j*n*w0*t)+(Cnn)*exp(-1j*n*w0*t);
end
subplot(2,2,1);
plot(t,x);
title('gibbs');
grid;
t=-3:0.003:3;
w0=2*pi;
C0=zeros(1,length(t));
x=C0;
N=input('No. of components');
for n=1:N
   Cn=(1/(n*pi))*sin((n*pi)/2);
   Cnn=Cn;
   x=x+(Cn)*exp(1j*n*w0*t)+(Cnn)*exp(-1j*n*w0*t);
end
subplot(2,2,2);
plot(t,x);
title('gibbs');
grid;
t=-3:0.003:3;
w0=2*pi;
C0=zeros(1,length(t));
x=C0;
N=input('No. of components');
for n=1:N
```

```
    Cn=(1/(n*pi))*sin((n*pi)/2);
    Cnn=Cn;
    x=x+(Cn)*exp(1j*n*w0*t)+(Cnn)*exp(-1j*n*w0*t);
end
subplot(2,2,3);
plot(t,x);
title('gibbs');
grid;
t=-3:0.003:3;
w0=2*pi;
C0=zeros(1,length(t));
x=C0;
N=input('No. of components');
for n=1:N
    Cn=(1/(n*pi))*sin((n*pi)/2);
    Cnn=Cn;
    x=x+(Cn)*exp(1j*n*w0*t)+(Cnn)*exp(-1j*n*w0*t);
end
subplot(2,2,4);
plot(t,x);
title('gibbs');
grid;
```

Output graphs:



Result:

Hence the signals are plotted using MATLAB.

Conclusion & Discussions:

The Gibb's Phenomenon has been plotted.

7(a). AUTO CORRELATION

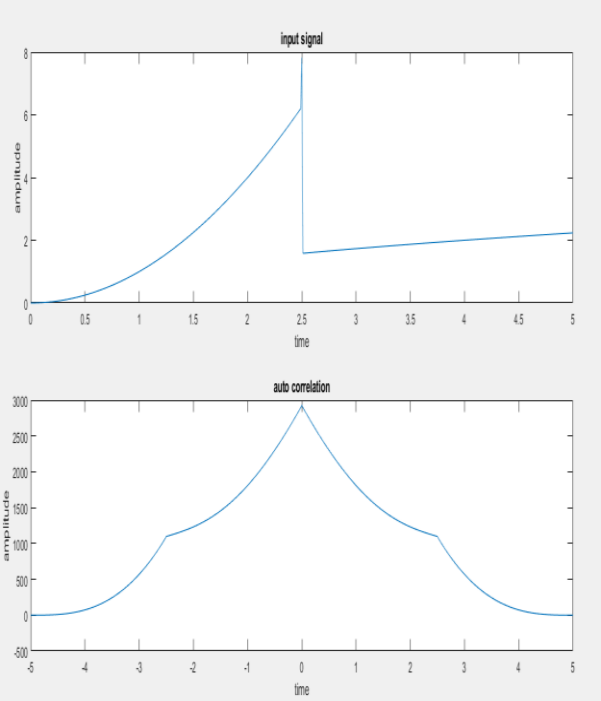
Aim: To identify and plot graph of auto-correlation.
S**oftware used:** MATLAB.
Theory:
Autocorrelation function of a signal is defined w.r.t the signal itself.This means that the signal is being compared (for similarity) with a time shift. Autocorrelation, also known as serial correlation, is the cross-correlation of a signal with itself. Informally, it is the similarity between observations as a function of the time lag between them. the principal property of the autocorrelation is that it is symmetric with respect to the time lag zero, and therefore it is normally represented in the positive time lag axis.

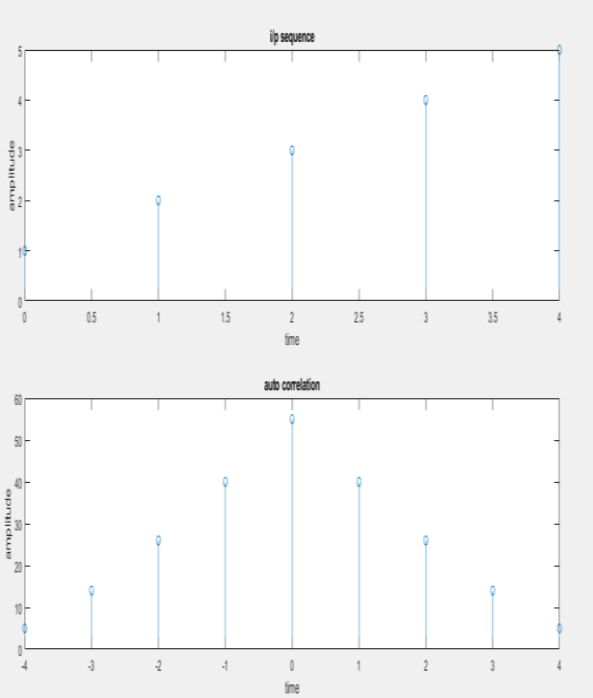
Program:

```
t1=0:0.01:5;
x=(t1.^2).*(t1>=0&t1<=2.5)+(t1.^0.5).*(t1>=2.5&t1<=5);
t2=-fliplr(t1);
y=xcorr(x,x);
a=min(t1)+min(t2);
b=max(t1)+max(t2);
t3=a:0.01:b;

subplot(2,1,1);
plot(t1,x);
xlabel('time');
ylabel('amplitude');
title('input signal');

subplot(2,1,2);
plot(t3,y);
xlabel('time');
ylabel('amplitude');
title('auto correlation');

t1=0:1:4;
x=[1 2 3 4 5];
t2=-fliplr(t1);
y=xcorr(x,x);
a=min(t1)+min(t2);
b=max(t1)+max(t2);
t3=a:1:b;

subplot(2,1,1);
stem(t1,x);
xlabel('time');
```

```
ylabel('amplitude');
title('i/p sequence');

subplot(2,1,2);
stem(t3,y);
xlabel('time');
ylabel('amplitude');
title('auto correlation');
```

Output graphs:





Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
Hence the auto-correlation is proved.

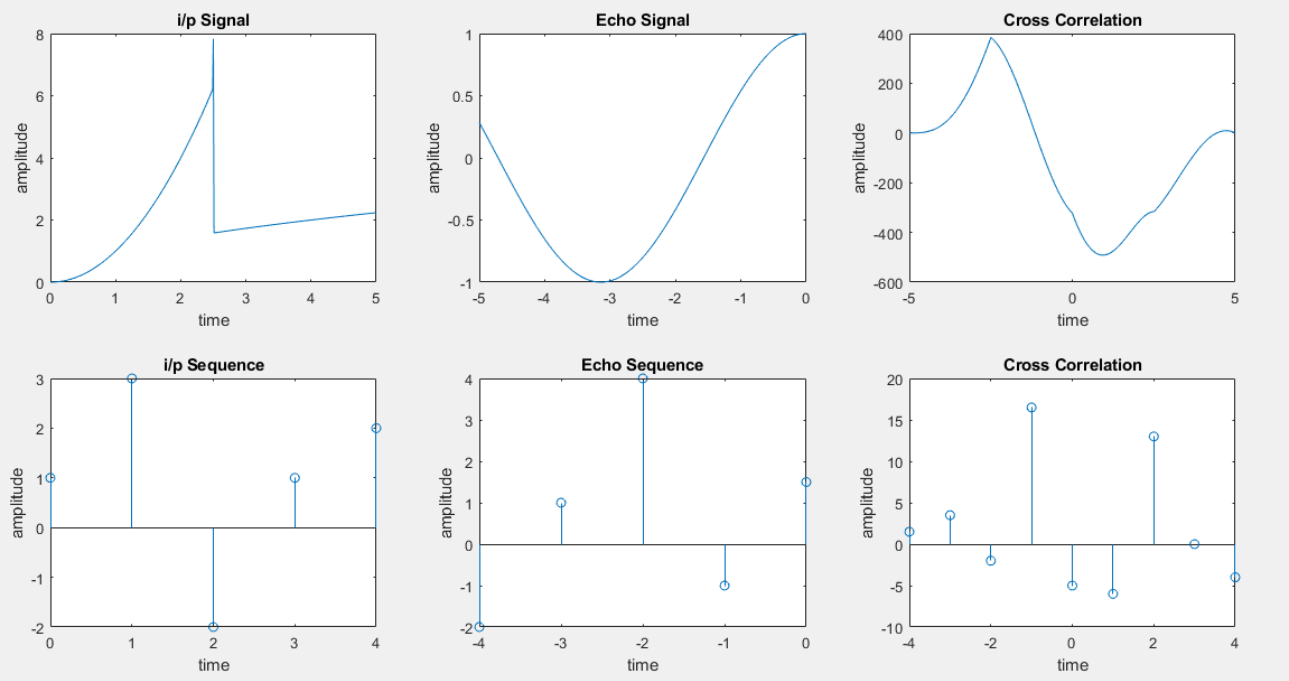
7(b). CROSS CORRELATION

Aim: To identify and plot graph of cross-correlation.
S**oftware used:** MATLAB.
Theory:
Crosscorrelation function of a signal is correlation of two independent signals with a time shift(generally time advancement) in of the two independent signals. Cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. cross correlation must be symmetric with respect to the time lag zero.
Program:

```
t1=0:0.01:5;
x= (t1.^2).*(t1>=0&t1<=2.5)+(t1.^0.5).*(t1>=2.5&t1<=5);
t2= -fliplr(t1);
y= cos(t2);
h=xcorr(x,y);
a=min(t1)+min(t2);
b=max(t1)+max(t2);
t3=a:0.01:b;
subplot(2,3,1);
plot(t1,x);
xlabel('time');
ylabel('amplitude');
title('i/p Signal');
subplot(2,3,2);
plot(t2,y);
xlabel('time');
ylabel('amplitude');
title('Echo Signal');
subplot(2,3,3);
plot(t3,h);
xlabel('time');
ylabel('amplitude');
title('Cross Correlation');

t1=0:1:4;
x=[1 3 -2 1 2];
t2= -fliplr(t1);
y=[-2 1 4 -1 1.5];
h=xcorr(x,y);
a=min(t1)+min(t2);
b=max(t1)+max(t2);
t3=a:1:b;
subplot(2,3,4);
```

```
stem(t1,x);
xlabel('time');
ylabel('amplitude');
title('i/p Sequence');
subplot(2,3,5);
stem(t2,y);
xlabel('time');
ylabel('amplitude');
title('Echo Sequence');
subplot(2,3,6);
stem(t3,h);
xlabel('time');
ylabel('amplitude');
title('Cross Correlation');
```

Output graphs:



Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
Hence the cross-correlation is proved.

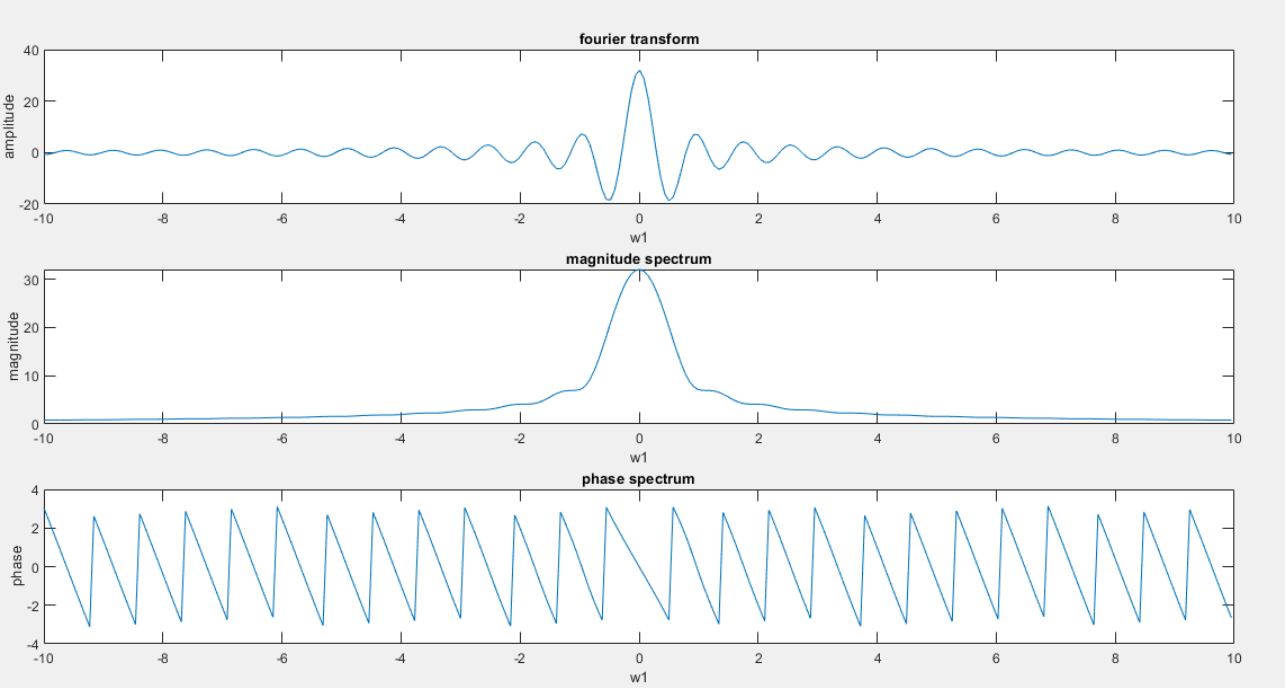
8. FOURIER TRANSFORM

Aim: To plot graph of Fourier transform, magnitude spectrum and phase spectrum
S**oftware used:** MATLAB.
Theory:
The Fourier transform is a mathematical function that takes a time-based pattern as input and determines the overall cycle offset rotation speed and strength for every possible cycle in the given pattern. The Fourier transform is applied to waveforms which are basically a function of time, space or some other variable. The Fourier transform decomposes a waveform into a sinusoid and thus provides another way to represent a waveform. Magnitude spectrum is a graph plotted between magnitude and frequency Phase spectrum is a graph plotted between phase angle and frequency.

Program:

```
syms t w;
x=int(t*exp(-1*1i*w*t),t,[0,8]);
w1=-10:0.07:10;
x1=subs(x,w,w1);
xmag=abs(x1);
xphas=angle(x1);

subplot(3,1,1);
plot(w1,x1);
xlabel('w1');
ylabel('amplitude');
title('fourier transform');

subplot(3,1,2);
plot(w1,xmag);
xlabel('w1');
ylabel('magnitude');
title('magnitude spectrum');

subplot(3,1,3);
plot(w1,xphas);
xlabel('w1');
ylabel('phase');
title('phase spectrum');
```

Output graphs:



Result:
Hence the signals are plotted using MATLAB.

Conclusion & Discussions:
Hence Fourier transform of given signal is plotted.

9. VERIFICATION OF LINEARITY AND TIME INVARIANCE

Aim: To verify linearity and time invariance properties of a given continuous or discrete system.
S**oftware used:** MATLAB.
Theory: Linear systems are systems whose outputs for a linear combination of inputs are the same as a linear combination of individual responses to those inputs. Time-invariant systems are systems where the output does not depend on *when* an input was applied.

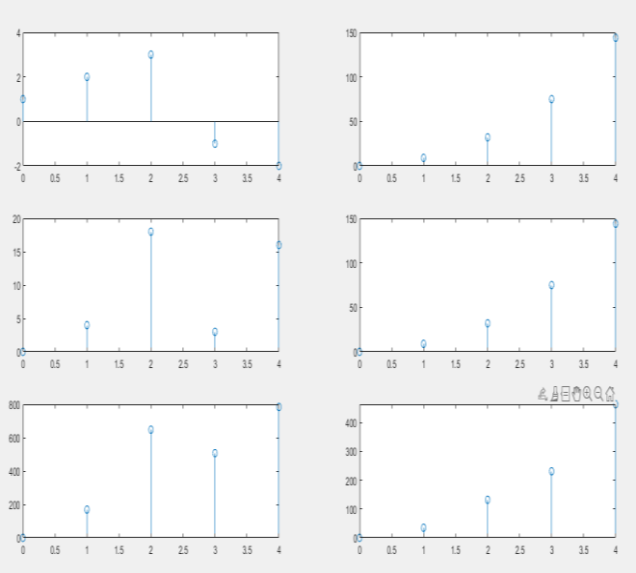
Program:

```
%Linearity:-
N=5;
x1=[1 2 3 -1 -2];
x2=[2 3 4 5 6];
a1=2;
a2=3;
n=0:N-1;
x=a1*x1+a2*x2;
y01=n.*(x.^2);
y1=n.*(x1.^2);
x2=n.*(x2.^2);
y02=a1*y1+a2*x2;
subplot(3,2,1);
stem(n,x1);
subplot(3,2,2);
stem(n,x2);
subplot(3,2,3);
stem(n,y1);
subplot(3,2,4);
stem(n,x2);
subplot(3,2,5);
stem(n,y01);
subplot(3,2,6);
stem(n,y02);
 if(y01==y02)
   disp('linear');
else
   disp('non linear');
end

%Time variance:-
x=[1 2 3 4 5];
n=0:length(x)-1;
d=2;
```

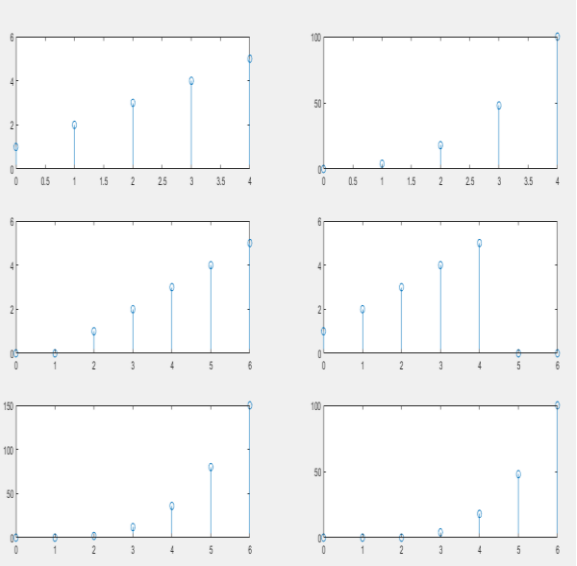
```
y=n.*(x.^2);
xd=[zeros(1,d),x];
nxd=0:length(xd)-1;
yd=nxd.*(xd.^2);
nyd=0:length(yd)-1;
xp=[x,zeros(1,d)];
yp=[zeros(1,d),y];
subplot(3,2,1);
stem(n,x);
subplot(3,2,2);
stem(n,y);
subplot(3,2,3);
stem(nxd,xd);
subplot(3,2,4);
stem(nyd,xp);
subplot(3,2,5);
stem(nxd,yd);
subplot(3,2,6);
stem(nyd,yp);
if(yd==yp)
   disp('Time invariant');
else
   disp('Time variant');
end
```

Output Graphs:

>> linearty: non linear



>> timevariance: time variant



Result:

Hence the signals are plotted using MATLAB.

Conclusion & Discussions:

Hence LTI of given system is plotted.

10. COMPUTATION OF RESPONSES OF GIVEN LTI SYSTEM

Aim: To compute unit sample, unit step and sinusoidal responses of the given LTI system and verifying its physical reliability and stability properties.

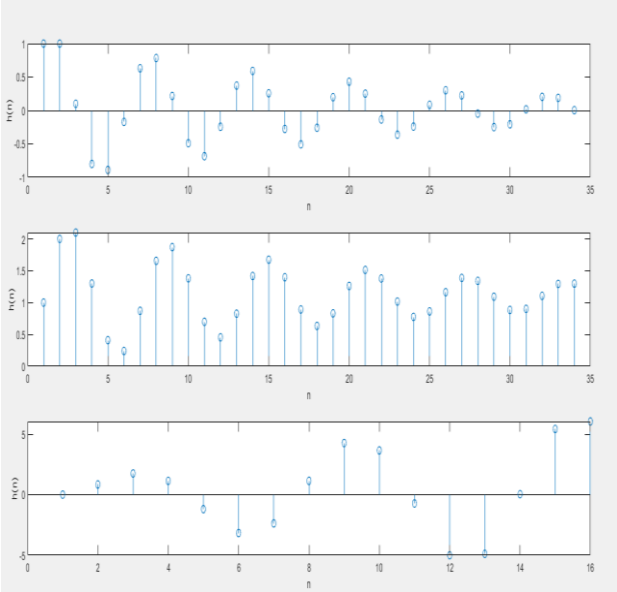
S**oftware used:** MATLAB.

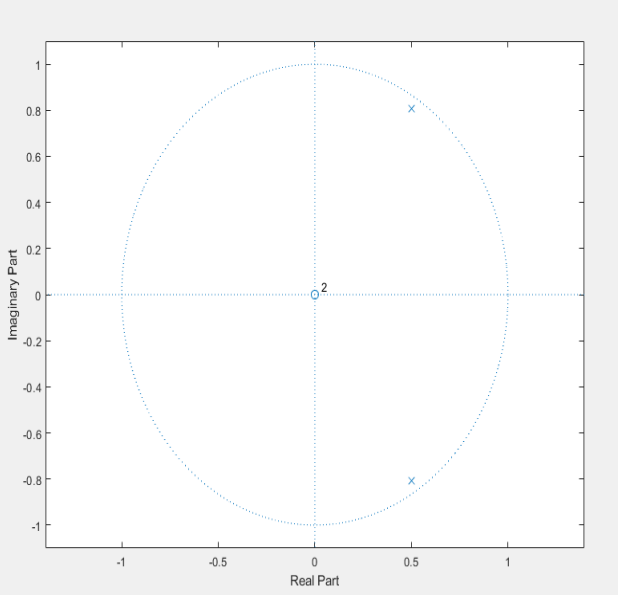
Theory: Linear systems are systems whose outputs for a linear combination of inputs are the same as a linear combination of individual responses to those inputs. Time-invariant systems are systems where the output does not depend on when an input was applied.

Program:

```
clear all;
b=[1];
a=[1 -1 0.9];
n=0:3:100;
x1=1.*(n==0);
h1=filter(b,a,x1);
subplot(3,1,1);
stem(h1);
xlabel('n');
ylabel('h(n)');

x2=1.*(n>=0)+0.*(n<0);
h2=filter(b,a,x2);
subplot(3,1,2);
stem(h2);
xlabel('n');
ylabel('h(n)');
n=0:1:5*pi;
x=sin(n);
h3=filter(b,a,x);
subplot(3,1,3);
stem(h3);
xlabel('n');
ylabel('h(n)');
figure;
zplane(b,a);
```

Output graphs:





Result:

Hence the signals are plotted using MATLAB.

Conclusion & Discussions:

Hence LTI of given system is plotted.

11. SAMPLING THEOREM VERIFICATION

Aim: To verify sampling theorem

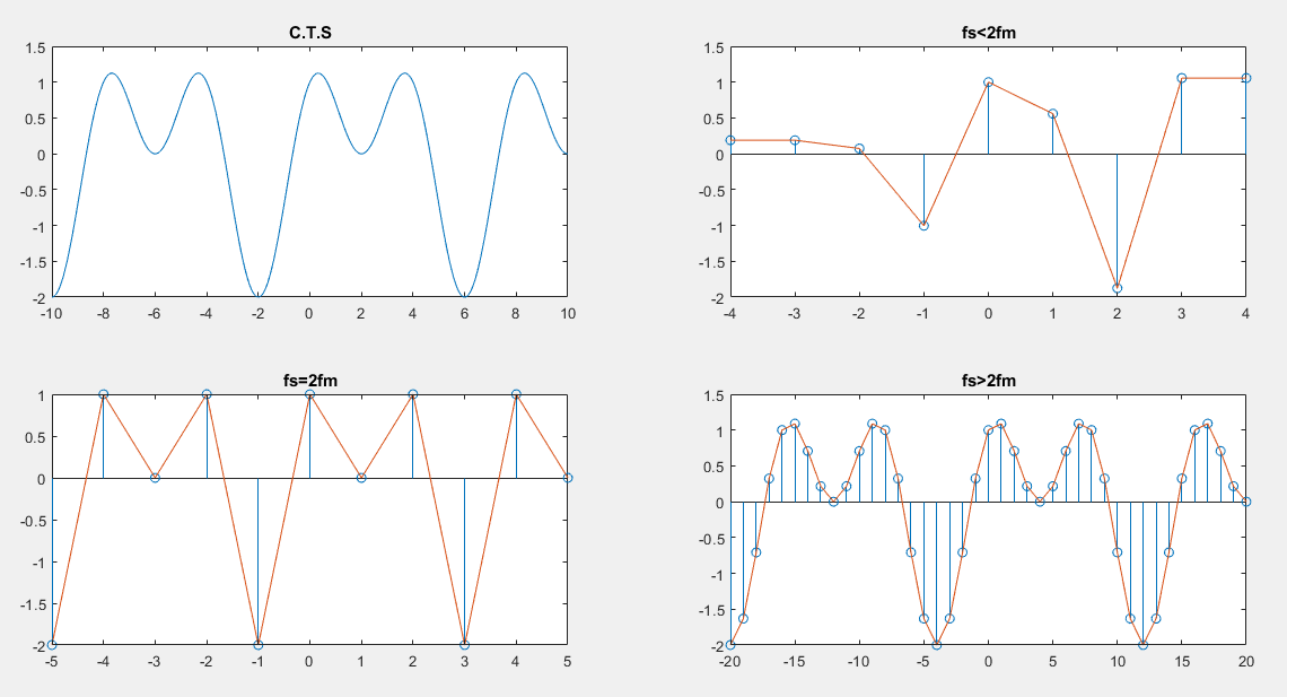
S**oftware used:** MATLAB.

Theory: A continuous time signal can be represented in its samples and can be recovered back when sampling frequency is greater than or equal to the twice the highest frequency signal, i. e. **(fs≥2fm)**

Program:

```
t=-10:0.01:10;
T=4; fm=1/T;
x=sin(pi*fm*t)+cos(2*pi*fm*t);
subplot(2,2,1);
plot(t,x);
title('C.T.S');
fs1=1.4*fm; fs2=2*fm; fs3=8*fm;
n1=-4:1:4;
x1=sin(pi*fm*n1/fs1)+cos(2*pi*fm*n1/fs1);
subplot(2,2,2);
stem(n1,x1);
hold on;
subplot(2,2,2)
plot(n1,x1);
title('fs<2fm');
n2=-5:1:5;
x2=sin(pi*fm*n2/fs2)+cos(2*pi*fm*n2/fs2);
subplot(2,2,3);
stem(n2,x2);
hold on;
subplot(2,2,3)
plot(n2,x2);
title('fs=2fm');
n3=-20:1:20;
x3=sin(pi*fm*n3/fs3)+cos(2*pi*fm*n3/fs3);
subplot(2,2,4);
stem(n3,x3);
hold on;
subplot(2,2,4)
plot(n3,x3);
title('fs>2fm');
```

Output graph:



Result:

Hence the signals are plotted using MATLAB.

Conclusion & Discussions:

Hence sampling theorem is verified.

12. REMOVAL OF NOISE BY AUTO & CROSS CORRELATION

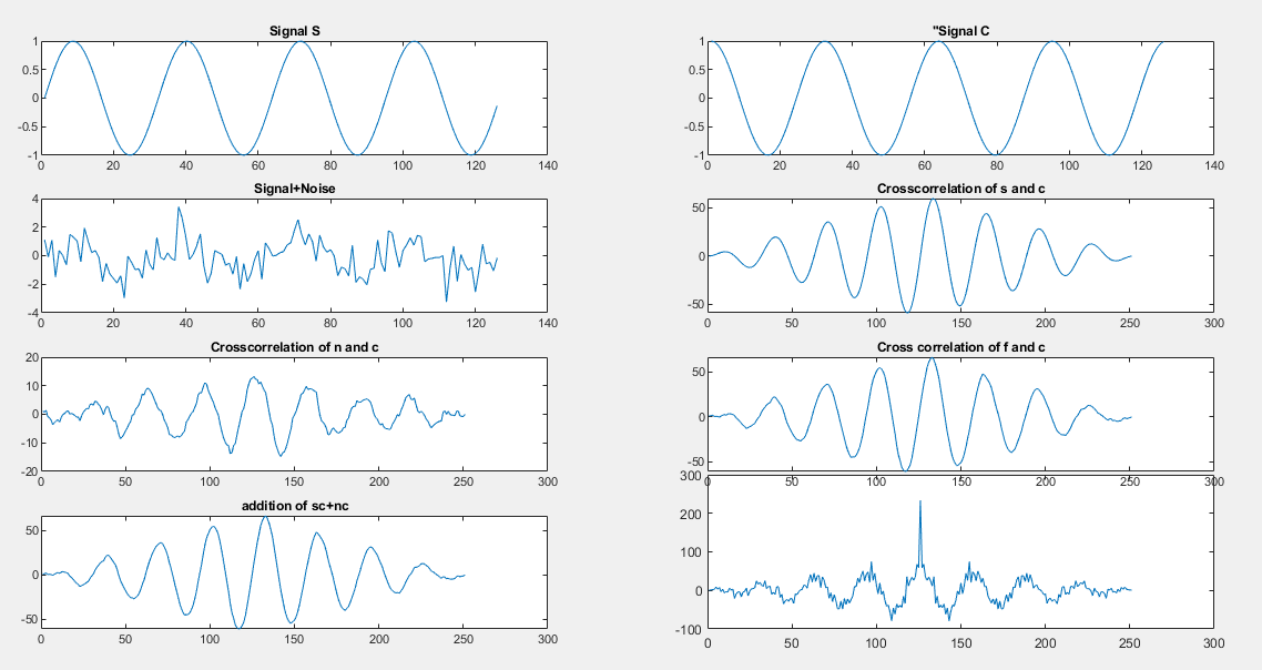
Aim: To verify removal of noise by auto correlation and cross correlation.
Software used: MATLAB.
Theory: Generally, clean speech signal and noise will consider uncorrelated, therefore, if x(m,n) and v(m,n) are considered uncorrelated, then the auto & cross correlation of the noisy speech signal can be written as: **ryy(m,k) = rxx(m,k) + rvv(m,k) 0 ≤ m ≤ M−10 ≤ k ≤ N−1.**

Program:

```
clc;
t=0:0.2:8*pi;
s=cos(t);
subplot(3,2,1);
plot(s);
title('Input Signal "s"');
n=randn([1 126]);
f=s+n;
subplot(3,2,2);
plot(f);
title('Noisy Signal f=s+n');
as=xcorr(s,s);
subplot(3,2,3);
plot(as);
title('Auto corr of i/p (as)');
an=xcorr(n,n);
subplot(3,2,4);
plot(an);
title('Auto corr of noise (an)');
cff=xcorr(f,f);
subplot(3,2,5);
plot(cff);
title('Auto corr of f');
hh=as+an;
subplot(3,2,6);
plot(hh);
title('Addition of as+an');
%Extraction of signal
s=sin(t);
subplot(4,2,1);
plot(s);
title('Signal S');
c=cos(t);
subplot(4,2,2);
plot(c);
```

```
title('"Signal C');
n=randn([1 126]);
f=s+n;
subplot(4,2,3);
plot(f);
title('Signal+Noise');
asc=xcorr(s,c);
subplot(4,2,4);
plot(asc);
title('Crosscorrelation of s and c');
anc=xcorr(n,c);
subplot(4,2,5);
plot(anc);
title('Crosscorrelation of n and c');
cfc=xcorr(f,c);
subplot(4,2,6);
plot(cfc);
title('Cross correlation of f and c');
hh=asc+anc;
subplot(4,2,7);
plot(hh);
title('addition of sc+nc');
```

Output graph:



Result:

Hence the signals are plotted using MATLAB.

Conclusion & Discussions:

Hence noise is removed by auto & cross correlation. The signal is extracted.