EXPERIMENT NO.:1

Study of types of signals Deterministic and Stochastic (Continuous)

Contents **Page No.**

1. Objective:

Study of types of signals Deterministic and Stochastic (Continuous).

2. Expected outcomes of experiment:

1 Understanding continuous and stochastic (continuous) signal.

2 Plotting both of them in MATLAB.

3.Theory:

Definition of Signal:

In communication systems, signal processingand Electrical Engineering, a signal is defined as a function that "conveys information about the behavior or attributes of some phenomenon".

The *IEEE Transactions on Signal Processing* states that the term "signal" includes audio, video, speech, image, communication, geophysical, sonar, radar, medical and musical signals.

In a *communication system*, a *transmitter* encodes a *message* to a signal, which is carried to a *receiver* by the communications *channel*.

Continuous-time Signal:

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. It is denoted by *x(t)*. Figure 1(a) shows continuous-time signal.

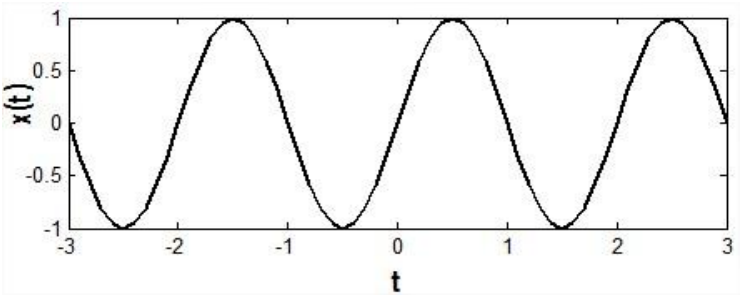


Fig.1.1 Continuous-time signal

Discrete-time Signal:

Signals that can be defined at discrete instant of time is called discrete time signal. Basically discrete time signals can be obtained by sampling a continuous-time signal. It is denoted as *x(n)*.Figure 1(b) shows discrete-time signal.

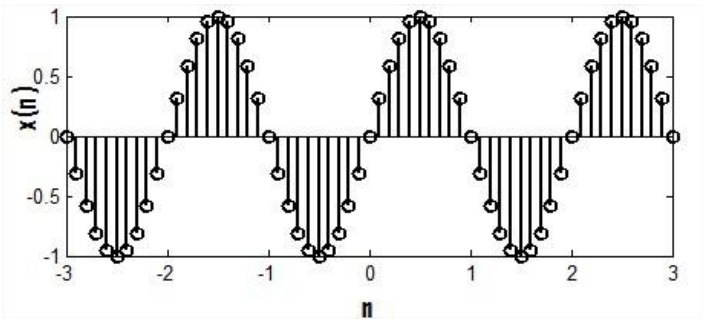


Fig.1.2 Discrete-time signal

Deterministic and Stochastic Signals

A random signal cannot be described by any mathematical function, where as a deterministic signal is one that can be described mathematically. A common example of random signal is noise

Further Continuous signals can be classified into two groups:

Deterministic Signals:

A signal is said to be deterministic if there is no uncertainty with respect to its value at any instant of time. Or, signals which can be defined exactly by a mathematical formula are known as deterministic signals.

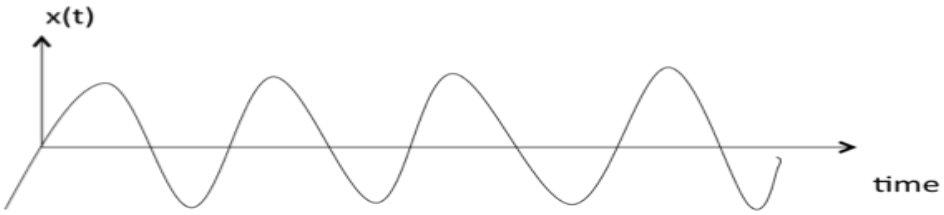


Figure 1.3 Deterministic Signal

- **Stochastic or Non-deterministic Signals:** A signal is said to be non-deterministic if there is uncertainty with respect to its value at some instant of time. Non-deterministic signals are random in nature hence they are called random signals.

Random signals cannot be described by a mathematical equation. They are modeled in probabilistic terms.

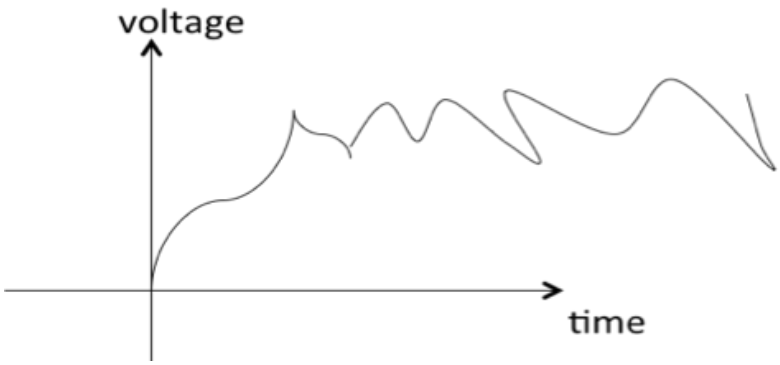


Figure 1.4 Stochastic Signal

4.Equipments Required:

A PC installed with MATLAB software.

5 Procedure:

- Open Matlab in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.
- Run the script.
- Desired result will be displayed on the command window.

6.Coding:

Deterministic Signals

(a) Square Wave

Square wave of amplitude A, fundamental frequency wo (in each rad/sec) and duty cycle rho (rho is the function of each period for which the signal is positive). Following is the complete set of commands:-

```
A=1;
wo=10*pi;
rho=0.5;
```

```
t=0:0.001:1;
sq=A*square(wo*t+rho);%generate square waveform %
plot(t,sq)
inv_sq=-sq;  % generate inverse square waveform%
plot(t,inv_sq)
amp_sq=1-sq;  %generate amplitude shift waveform%
plot(t,amp_sq)
hold on
```

(b) Triangular Wave

Triangular wave of amplitude A, fundamental frequency Wo (measured in rad/sec) and width W. Let the period of triangular wave be T, with the first maximum value occurring at t=WT.

```
A=1;
Wo=10*pi;
W=0.5;
t=0:0.001:1;
tri=A*sawtooth(Wo*t+W);%generate triangular waveform%
plot(t,tri)
abs_tri=abs(tri);   % generate modulus of triangular waveform  %
plot(t, abs_tri)
half_tri=( tri+ abs_tri)/2; % generate half waveform of triangular waveform%
plot(t,half_tri)
```

c) Decaying exponential

```
B=5;
a=6;
t=0:0.001:1;
X=B*exp(-a*t); %generate decaying exponential waveform%
plot(t,X)
```

d) Growing exponential

```
B=5;
a=6;
```

```
t=0:0.001:1;
X=B*exp(a*t);% generate growing exponential waveform%
plot(t,X)
```

e) Sinusoidal signal

```
A=4;
Wo=20*pi;
phi=pi/6;
t=0:0.001:1;
Cosine=A*cos(Wo*t+phi); % generate sinusoidal signals%
plot(t,Cosine)
```

Random Signal:

```
sig_length = 200;
hold on
sig = rand(1,sig_length);
plot(1:sig_length,sig)
axis tight
title('Signal')
```

7. Results and Discussions:

Deterministic Signals:

(a) Square Wave: Following square waveform is obtained by running program in Matlab of which amplitude is varying from -1 to 1.

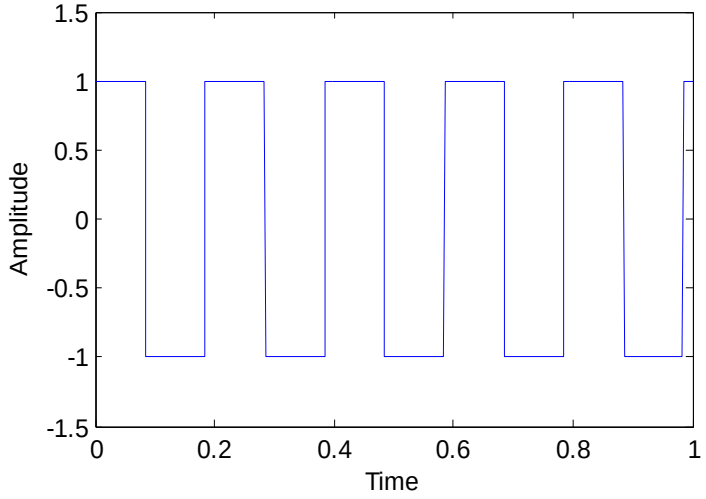


Fig 1.5(a) square waveform

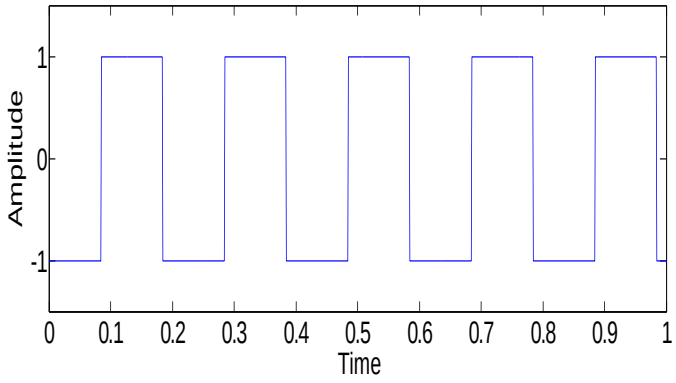


Fig 1.5(b) inverse square waveform of Fig 1.5(a)

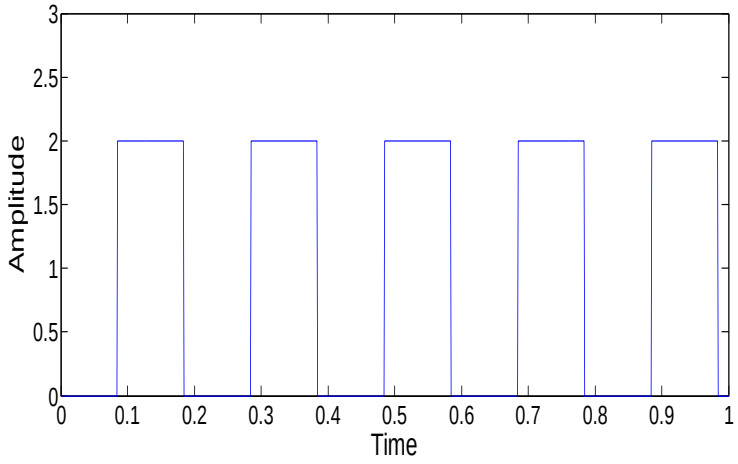


Fig 1.5(c) amplitude shift Square wave

(b) Triangular Wave:

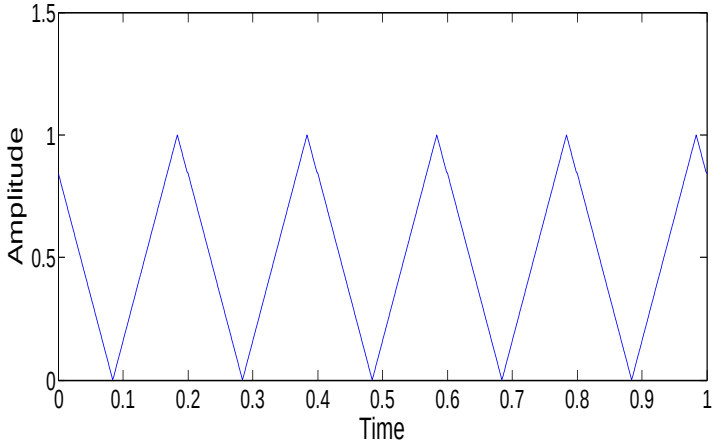


Fig 1.6 (a) Triangular waveform

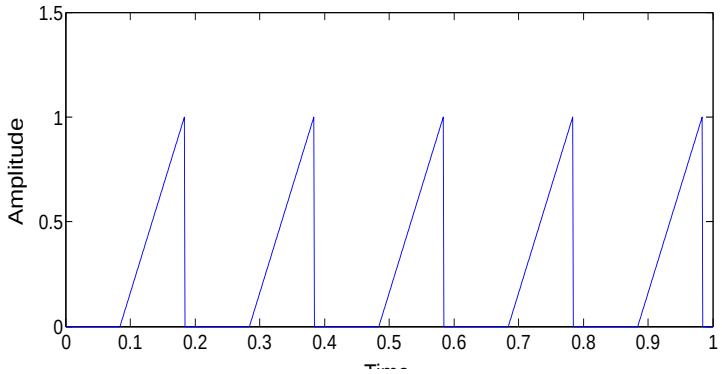


Fig 1.6(b)Half waveform of Triangular waveform of Fig 1.6 (a)

(c) Decaying Exponential:

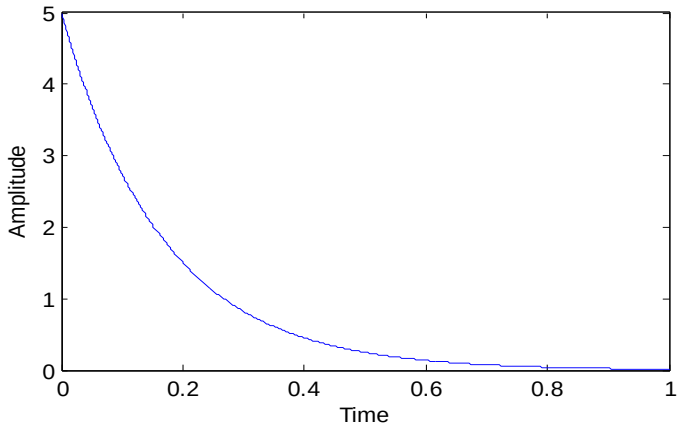


Fig 1.6 Decaying Exponential

(d) Growing Exponential:

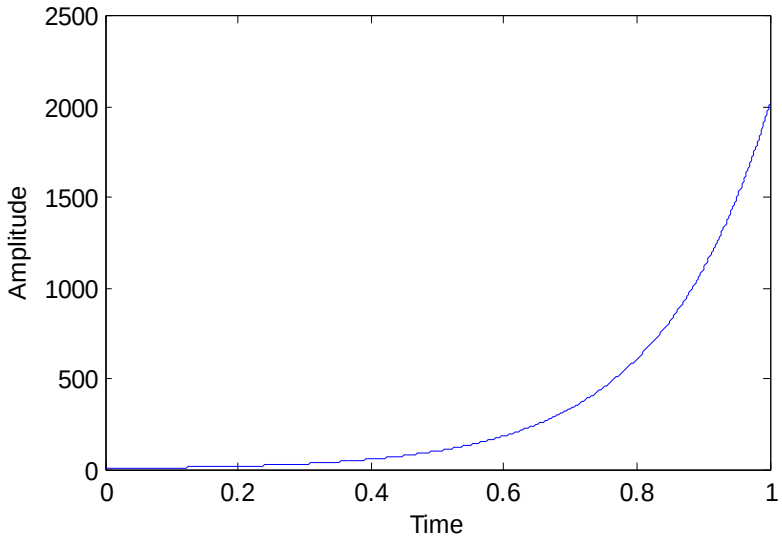


Fig 1.7 Growing Exponential

(e) Sinusoidal Signal:

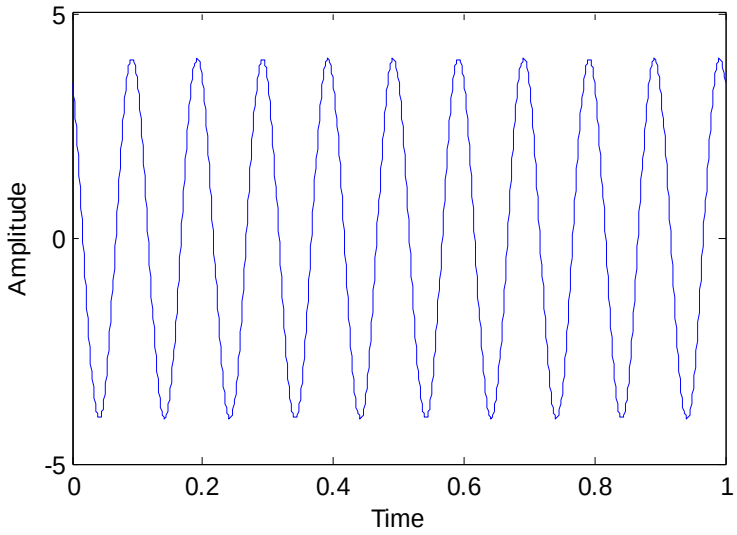


Fig 1.8Sinusoidal Signal

Stochastic Signal:

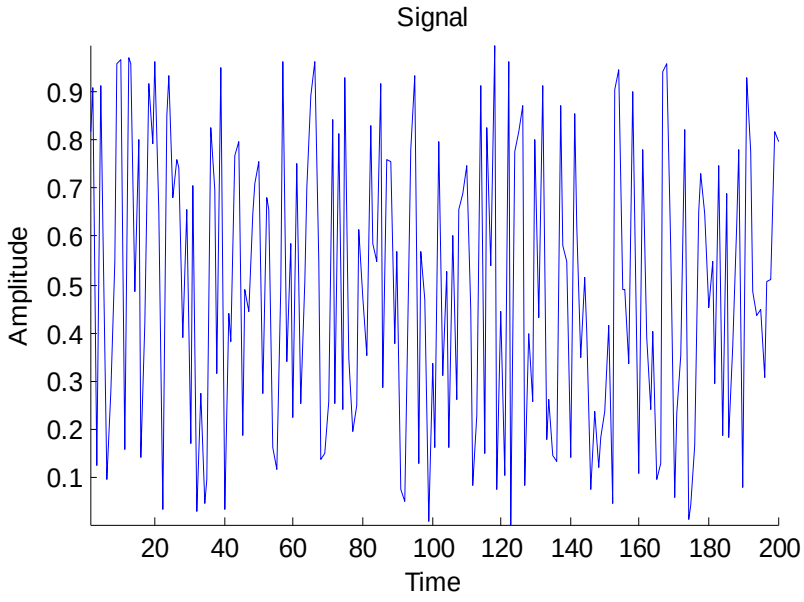


Fig 1.9Stochastic Signal

8. Assignment:

1. Write a MATLAB program to generate growing exponential and a sinusoidal signal.
2. Generate various types of Random signals using MATLAB.

9. Conclusion

In this experiment, different types of Deterministic signals i.e., square wave, triangular, decaying exponential, growing exponential, sinusoidal and Stochastic (Continuous)signals are discussed and plotted in MATLAB.

EXPERIMENT NO.:02

Study of time properties of signals

Contents **Page No.**

1.Objective:

Study of time properties of signals

2. Expected outcomes of experiments:

1. Understanding time properties of signals.
2. Carrying out different transformation on signals such as multiplication, shifting, scaling, addition, subtraction, etc in MATLAB.

3.Theory:

Properties of signals

There are some important properties of signal such as time reversal, amplitude-scaling, time-scaling and time-shifting.

Time Reversal:

Until now, we have assumed our independent variable representing the signal to be positive. Why should this be the case? Can't it be negative?

It can be negative. In fact, one can make it negative just by multiplying it by -1. 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. As a result, the operation is aptly known as the time reversal or time reflection of thesignal.

For example, let's consider our input signal to be $x[n]$, shown in figure(a). The effect of substituting $–n$ in the place of n results in the signal $y[n]$ as shown in figure (b).

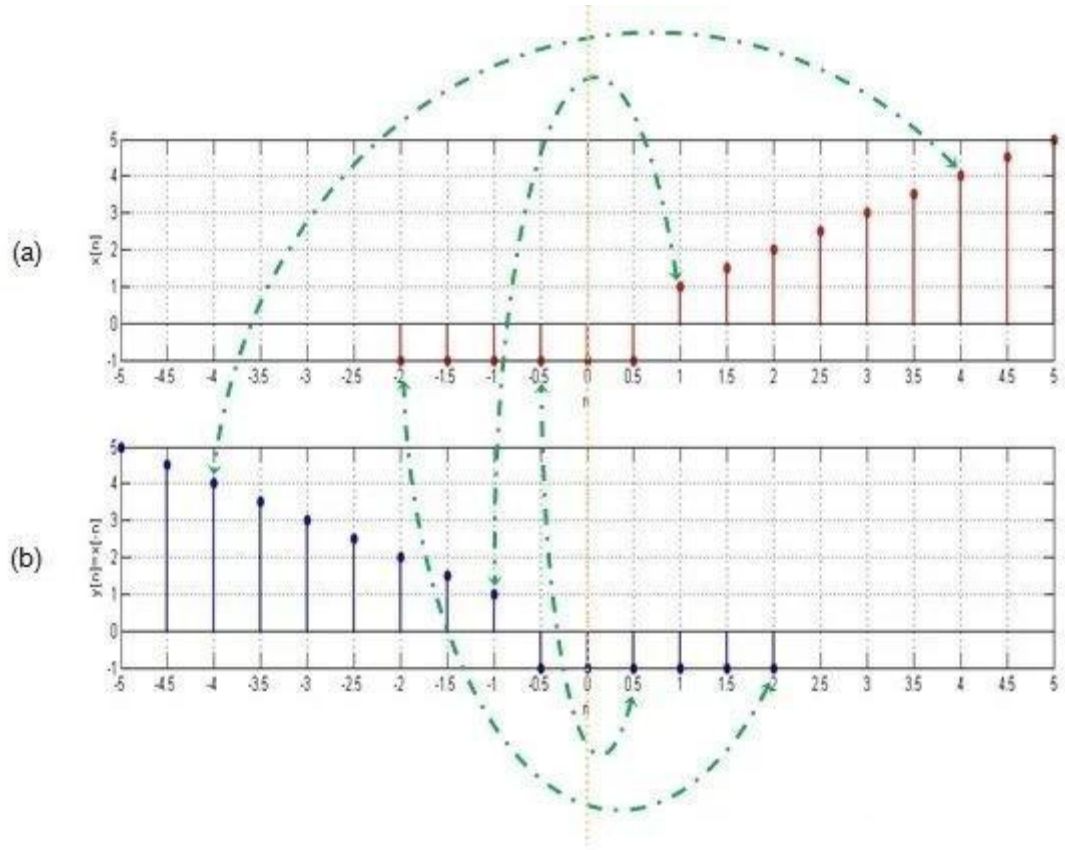


Fig 2.1 Time Reversal

Here you can observe that the value of $x[n]$ at the time instant n = -2 is -1. This is equal to the value of $y[n]$ at n = 2. Likewise, $x[-0.5] = y[0.5] = -1$, $x[1] = y[-1] = 1$, and $x[4] = y[-4] = 4$ (as indicated by the green dotted-dashed-line arrows).

This indicates that the graph of $y[n]$ is nothing but the original signal $x[n]$ reflected along the vertical axis (shown as a dotted orange line in the figure).This applies to both continuous- and discrete-time signals.

Amplitude scaling:

Consider a signal x(t) which is multiplying by a constant 'A' and this can be indicated by a notation x(t) → Ax(t). For any arbitrary 't', this multiplies the signal value x(t) by a constant 'A'. Thus, x(t) → Ax(t) multiplies x(t) at every value of 't' by a constant 'A'. This is called

Amplitude-scaling. If the amplitude-scaling factor is negative, then it flips the signal with the t-axis as the rotation axis of the flip. If the scaling factor is -1, then only the signal will be flip. This is shown in the figures given below:

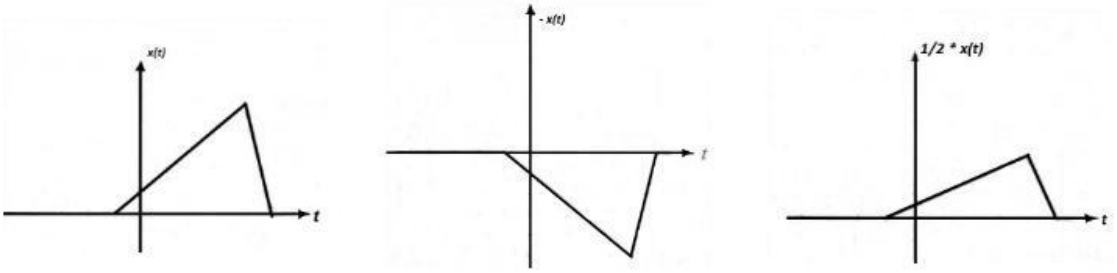


Fig 2.2 Amplitude scaling

Time scaling:

Time scaling compresses or dilates a signal by multiplying the time variable by some quantity. If that quantity is greater than one, the signal becomes narrower and the operation is called compression. If that quantity is less than one, the signal becomes wider and the operation is called dilation. Figures below show the signal x(t), compression of signal and dilation of signal respectively.

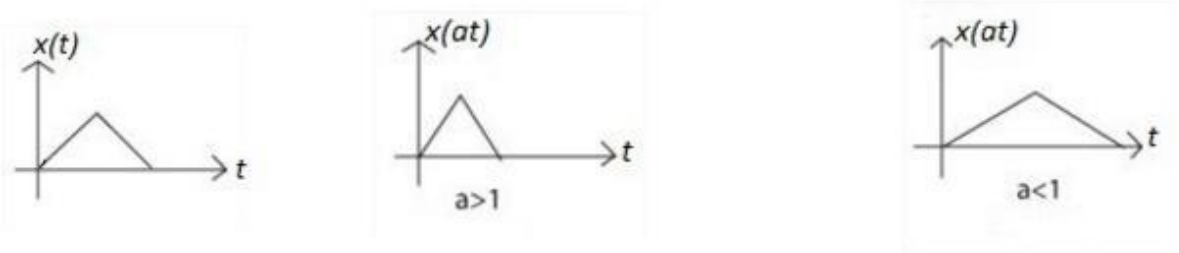


Fig 2.3Time scaling

Time Shifting:

In signals and system amplitude scaling, time shifting and time scaling are some important properties. If a continuous time signal is defined as *x(t) = s(t - t1)*. Then we can say that *x(t)* is the time shifted version of*s(t)*.

Consider a simple signal *s(t)* for *0 < t < 1*

Now shifting the function by time *t1 = 2* sec.

x(t) = s (t-2)=t-2 *for* *0 < (t - 2) <1=t-2* *for 2 < t <3*

Which is simply signal *s(t)* with its origin delayed by *2* sec. Now
if we shift the signal by *t1 = -1* sec.

$$\text{then } x(t) = s(t+1) = t+1 \quad for \quad 0 < (t+1)$$
$$= t+1 \quad for \quad -1 < t < 0.$$

Which is simply *s(t)* with its origin shifted to the left or advance in time by *1* seconds. This time-shifting property of signal is shown in the Figures given below.

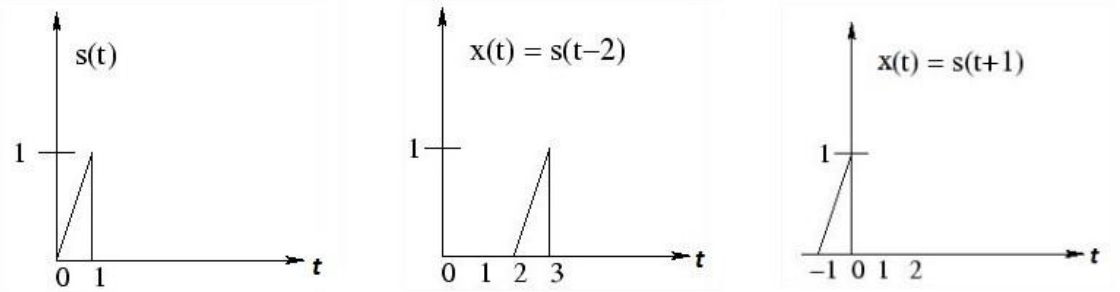


Fig 2.4 Time Shifting:

Time-Delayed Signals:

Suppose that we want to move this signal right by three units (i.e., we want a new signal whose amplitudes are the same but are shifted right three times).

This means that we desire our output signal $y[n]$ to span from $n = 0$to $n = 6$. Such a signal is shown as Figure 1(b) and can be mathematically written as $y[n] = x[n-3]$.

This kind of signal is referred to as time-delayed because we have made the signal arrive three units late.

Time-Advanced Signals:

On the other hand, let's say that we want the same signal to arrive early. Consider a case where we want our output signal to be advanced by, say, two units. This objective can be accomplished by shifting the signal to the left by two time units, i.e., $y[n] = x[n+2]$.

The corresponding input and output signals are shown in Figure (a) and (b), respectively. Our output signal has the same values as the original signal but spans from n = -5 to n = 1 instead of n = -3 to n = 3. The signal shown in Figure 2(b) is aptly referred to as a time-advanced signal.

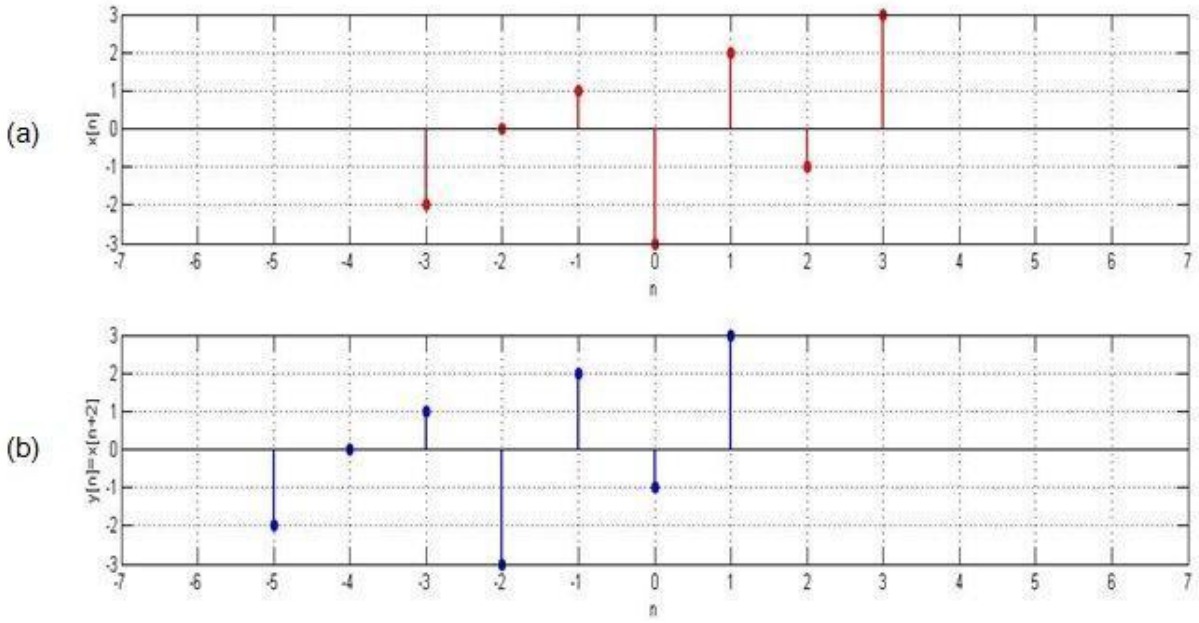


Fig 2.6 Original signal and its time-advanced version

For both of the above examples, note that the time-shifting operation performed over the signals affects not the amplitudes themselves but rather the amplitudes with respect to the time axis. We have used discrete-time signals in these examples, but the same applies to continuous-time signals.

4.EquipmentsRequired:

A PC installed with MATLAB software

5 Procedure:

- Open Matlab in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.
- Run the script.

- Desired result will be displayed on the command window.

6 Coding:

Time Shifting of Signals:

MATLAB can be used to perform shifting of signals. A signal can be delayed as well as advanced. The basic idea is to add the shift value to indices and thereby plotting the signal.

The following is a program to delay or advance a signal x(n). The shift value is decided at the runtime.

```
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)');
```

Time reversing of signal:

The inbuilt function fliplr() function can be used to perform reversing or folding a signal.

Syntax:

fliplr(a) : if a is row vector it returns a vector with the same size of a but with reversed order. if a is column vector it flips the elements one column to the other.

Basic idea :

1. flip theelements
2. flip the indices with a ‘ – ‘sign. 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)') ;
```

Time shifting and time scaling programming:

```
t = -10 : 0.001 : 10;
x =@(t) 0.*(t<-2) + (-4-(2.*t)).*(t>=-2 & t<0) + (-4+(3.*t)).*(t>=0 & t<4) + (16-
(2.*t)).*(t>=4 & t<8) + 2.*(t>=8);
hold on plot(t, x(t),'r')
plot(t, x(t+1).*3, 'c')
plot(t, x(4*t)./2, '--m')
plot(t, -2*x((t-1)/2) , ':b') grid on
hold off
```

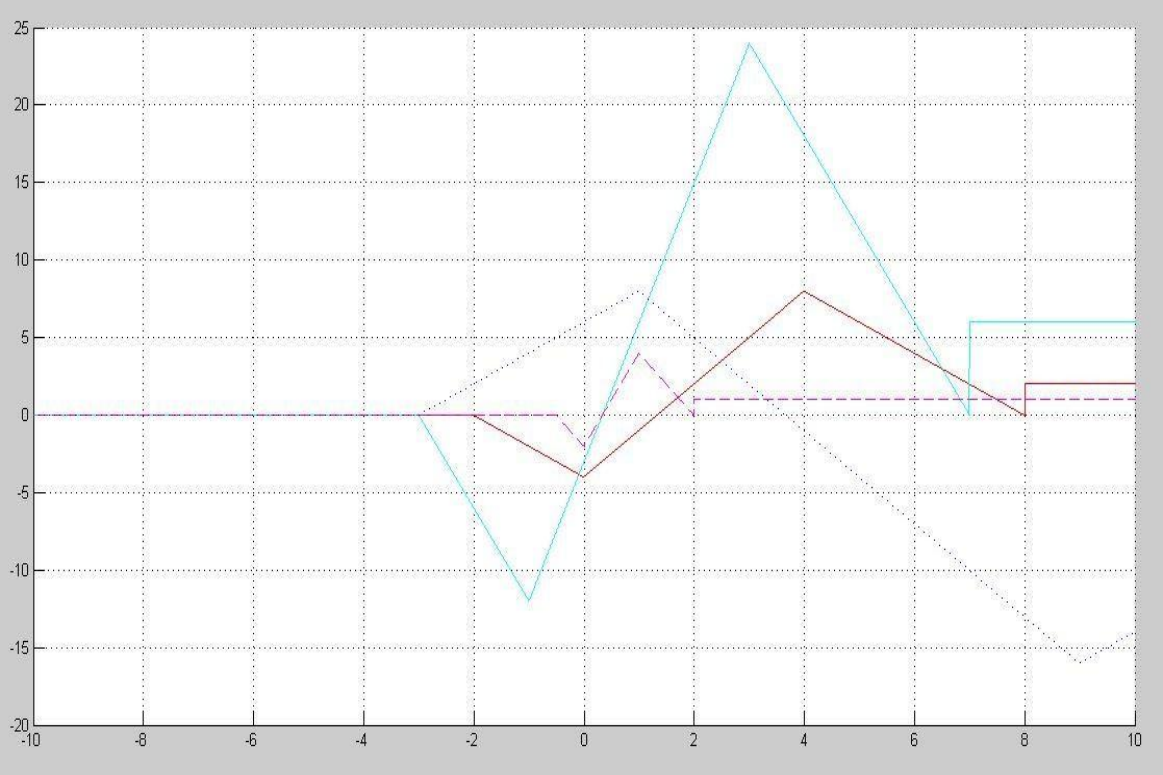


Fig 2.7 Time shifting and time scaling

7. Results and Discussion: Time shifting ofsignal:

In this program, we have given delay of 2 units and advance of 3 units, therefore getting the graph of Original signal x(n), Delayed signal x(n-2) and Advanced signal x(n+2) in the following way:

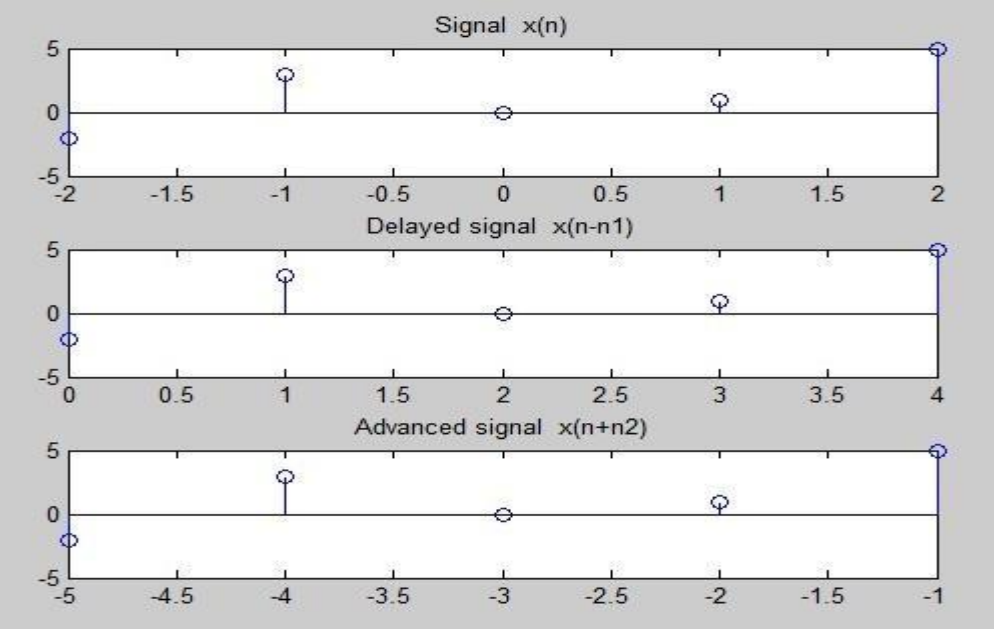


Fig 2.8 Time shifting ofsignal

Time Reversing of signal:

Using fliplr command, the following Reversed signal x(-n) has been obtained from the Original signal x(n)

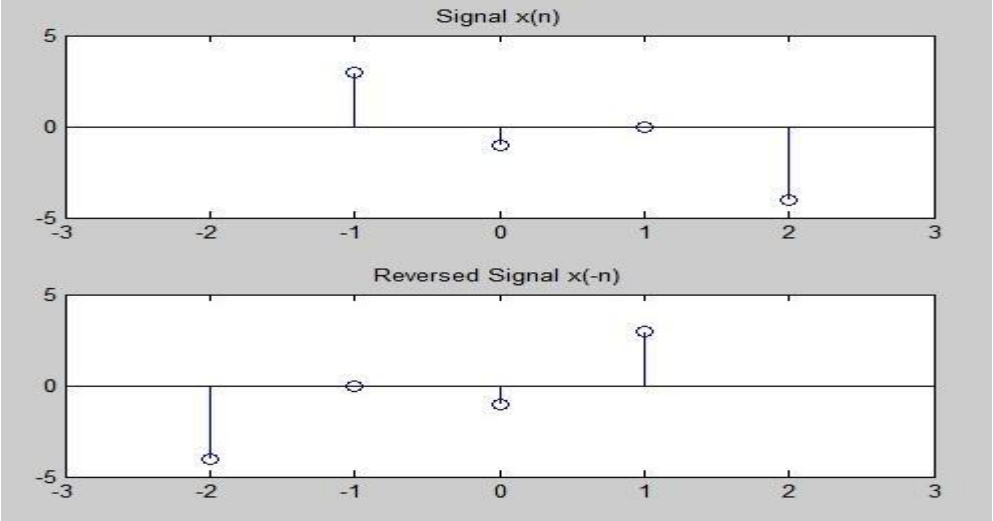


Fig .2.9 Time Reversing of signal

8. Assignments:

1.1 Add two signal X & Y where X= [0 1 6 5] and Y will be user defined.

1.2 Multiply signal X with signal Y given that; X= [5 1 0 6] and Y= [-1 0 0.5 3]

1.3 Compute the convolution of X= [5 4 0] and Y= [2 8 6].

1.4 Amplitude scale the convolved signal obtained in by 3.

9. Conclusion:

This experiment introduces different types of operation of signals i.e. addition, subtraction, multiplication, shifting, reversal, convolution of two signals using MATLAB.It helped students in applying such operation in real time scenario on signals.

EXPERIMENT NO.:03

Study of frequency properties of signal

Contents **Page No.**

1 Objective:

Study of frequency properties of signal

2 Expected Outcomes of The Experiment:

1. Understanding various aspects of Fourier transform.
2. Understanding Discrete Fourier Transform (DFT) which is suitable for MATLAB implementation being discrete both in time and frequency
3. Computing DFT using MATLAB commands `fft` and `ifft`

3 Theory:

Given a length N vector x representing one period of an N periodic signal x[n], the command x = fft (x) /N produces a length N vector X containing the DFT coefficients.

Similarly, given DFT coefficients in vector X, the command

x = ifft(x)*N

(a) Find the DFT coefficient of signal:

x[n]= 1 + sin{(nπ)/12 + (3π)/8}

Program:

```
x= ones(1,24) + sin ( ( (0:23)*pi) /12 + (3*pi)/8)
x=fft(x)/24
```

Solution:

x[n]= 1 + sin{(nπ)/12 + (3π)/8}

```
x =

  Columns 1 through 12

   1.9239   1.9914   1.9914   1.9239   1.7934   1.6088   1.3827   1.1305   0.8695   0.6173   0.3912   0.2066

  Columns 13 through 24

   0.0761   0.0086   0.0086   0.0761   0.2066   0.3912   0.6173   0.8695   1.1305   1.3827   1.6088   1.7934


x =

  Columns 1 through 6
```

1.0000 0.4619 - 0.1913i 0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 - 0.0000i

Columns 7 through 12

0.0000 - 0.0000i 0.0000 - 0.0000i 0.0000 + 0.0000i -0.0000 - 0.0000i -0.0000 - 0.0000i -0.0000 - 0.0000i

Columns 13 through 18

0 -0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 + 0.0000i 0.0000 - 0.0000i 0.0000 + 0.0000i

Columns 19 through 24

0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 - 0.0000i -0.0000 - 0.0000i 0.0000 - 0.0000i 0.4619 + 0.1913i

(b) Reconstruction of signal from DFT coefficients:

X recon = ifft(x,24)

Program:

```
x= ones(1,24) + sin ( ( (0:23)*pi) /12 + (3*pi)/8)
x=fft(x)/24
recon = ifft(x,24)
```

Solution:

X recon = ifft(x,24)

x =

Columns 1 through 12

1.9239 1.9914 1.9914 1.9239 1.7934 1.6088 1.3827 1.1305 0.8695 0.6173 0.3912 0.2066

Columns 13 through 24

0.0761 0.0086 0.0086 0.0761 0.2066 0.3912 0.6173 0.8695 1.1305 1.3827 1.6088 1.7934

x =

Columns 1 through 6

1.0000 0.4619 - 0.1913i 0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 - 0.0000i

Columns 7 through 12

0.0000 - 0.0000i 0.0000 - 0.0000i 0.0000 + 0.0000i -0.0000 - 0.0000i -0.0000 - 0.0000i -0.0000 - 0.0000i

Columns 13 through 18

0 -0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 + 0.0000i 0.0000 - 0.0000i 0.0000 + 0.0000i

Columns 19 through 24

0.0000 + 0.0000i -0.0000 + 0.0000i -0.0000 - 0.0000i -0.0000 - 0.0000i 0.0000 - 0.0000i 0.4619 + 0.1913i

recon =

Columns 1 through 12

0.0802 0.0830 0.0830 0.0802 0.0747 0.0670 0.0576 0.0471 0.0362 0.0257 0.0163 0.0086

Columns 13 through 24

0.0032 0.0004 0.0004 0.0032 0.0086 0.0163 0.0257 0.0362 0.0471 0.0576 0.0670 0.0747

(c) Partial sum evaluation tp x[n]:

```
k=0:24
n=-24:25
B(1)= (25/50)
B(2:25)= 2*sin((k*pi*(25/25))/(50*sin(k*pi/50)))
B(26)= sin((k*pi*(25/50))/(50*sin(25*pi/50)))
xhat(1,:)=B(1)*cos(n*0*pi/25)
for k=2:26
xhat(k,:)=xhat(k-1,:)+B(k)*cos(n*(k-1)*pi/25)
```

```
end
```

(d) Compute the time bandwidth product using DFT:

```
N=24;
x=[0.0802 0.0830 0.0830 0.0802 0.0747 0.0670 0.0576 0.0471 0.0362 0.0257 0.0163
0.00860 .0032 0.0004 0.0004 0.0032 0.0086 0.0163 0.0257 0.0362 0.0471 0.0576
0.0670 0.0747];
m=(N-max(size(x)))/2;
xc=[zeros(1,m),x,zeros(1,m)];
n=[-(N-1)/2:(N-1)/2];
n2=n'*n;
td=sqrt((xc.*xc)*n2'/(xc*xc'));
X=fftshift(fft(xc)/N);
bw=sqrt(real((X.*conj(X))*n2'/(X*X')));
tbp=td.*bw
```

Solution:

```
tbp =

  Columns 1 through 6

      0 +14.3537i       0 +13.1055i       0 +11.8574i       0 +10.6092i       0 + 9.3611i
0 + 8.1129i

  Columns 7 through 12

      0 + 6.8648i       0 + 5.6166i       0 + 4.3685i       0 + 3.1204i       0 + 1.8722i
0 + 0.6241i

  Columns 13 through 18

      0 + 0.6241i       0 + 1.8722i       0 + 3.1204i       0 + 4.3685i       0 + 5.6166i
0 + 6.8648i

  Columns 19 through 24

      0 + 8.1129i       0 + 9.3611i       0 +10.6092i       0 +11.8574i       0 +13.1055i
0 +14.3537i
```

4 Equipment Required:

A PC installed with MATLAB software (preferably Matlab R 2017a)

5 Procedure:

- Open Matlab in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.
- Run the script.
- Desired result will be displayed on the command window.

6 Coding:

6.1 Properties of Fourier Transform:

(1) Linearity: Let x(t) be an input signal and y(t) be an output signal then according to linearity property-

$$\mathcal{F}[ax(t) + by(t)] = a\mathcal{F}[x(t)] + b\mathcal{F}[y(t)]$$

MATLAB Coding:

```
n=0:1:40;
a=2;
b=-3;
x1=(cos(2*pi*0.4*n)).*(cos(2*pi*0.4*(n-1)));
x2=(sin(2*pi*0.4*n)).*(sin(2*pi*0.4*(n-1)));
x=a*x1+b*x2;
num=[2.2403 2.4908 2.2403];
den=[1 -0.4 0.75];
ic=[0 0];
y1=filter(num,den,x1,ic);
y2=filter(num,den,x2,ic);
y=filter(num,den,x,ic);
yt=a*y1+b*y2;
d=y - yt;
subplot(3,2,1)
stem(n,y);
xlabel('Discrete Time Index n');
ylabel('Amplitude');
title('Response of input multiplication & scaling');
subplot(3,2,2)
stem(n,yt);
xlabel('Discrete Time Index n');
ylabel('Amplitude');
title('Response of output multiplication & scaling');
```

```
subplot(3,2,3)
stem(n,d);
xlabel('Time Index n');
ylabel('Amplitude');
title('Difference Signal');
```

(2) Time Shifting: We assume that $F[x(t)]=x(jw)$ and $F[y(t)]=y(jw)$

Then $F[x(t\pm t_0)]=x(jw)\exp(\pm jwt_0)$

MATLAB Coding:

```
x=input('Enter the input sequence');
n=input('Enter the delay integer');
x1=length(x);
xn=x1+n;
fori=1:xn;
if(i<=n)
xn0(i)=0;
else
xn0(i)=x(i-n);
end;
end;
subplot(2,1,1);
stem(x);
title('Input sequence');
xlabel('time index');
ylabel('amplitude');
subplot(2,1,2);
stem(xn0);
title('delayed sequence');
xlabel('time index');
ylabel('amplitude');
figure;
w=0:pi/xn:pi*(xn-1)/xn;
X=fft(x,length(w));
Xn0=fft(xn0,length(w));
s=exp(-j*w*n).*X;
subplot(2,1,1);
stem(abs(Xn0));
title('DFT of the delayed sequence');
xlabel('time index');
ylabel('amplitude');
subplot(2,1,2);
```

```
stem(abs(s));
title('DFT of the original sequence*(e^-j*w*n)');
xlabel('time index');
ylabel('amplitude');
```

3) Frequency Shifting: We assume thatF[x(t)]=x(jw)

MATLAB Coding:

```
N=input('how many point dft do you want?');
x1=input('enter the seq');
n2=length(x1);
c=zeros(N);
x1=[x1 zeros(1,N-n2)];
for k=1:N
for n=1:N
w=exp((-2*pi*i*(k-1)*(n-1))/N);
x(n)=w;
end
c(k,:)=x;
end
disp('dft is ');
r=c*x1';
a1=input('enter the amount of shift in frequency domain');
for n=1:N
w=exp((2*pi*i*(n-1)*(a1))/N);
x2(n)=w;
end
r1=x2.*x1;
subplot(221);
stem(abs(r));
gridon;
title('orginaldft magnitude plot');
subplot(222);
stem(angle(r));
gridon;
title('orginaldft angle');
for k=1:N
for n=1:N
w=exp((-2*pi*i*(k-1)*(n-1))/N);
x(n)=w;
end
c(k,:)=x;
```

```
end
disp('dft is');
r2=c*r1';
subplot(223);
stem(abs(r2));
gridon;
title('shifted dft magnitude');
subplot(224);
stem(angle(r2));
gridon;
title('shifeddft angle');
```

4) Convolution Theorem for Fourier Transform:

The **convolution theorem** states that under suitable conditions the Fourier transform of a convolution is the pointwise product of Fourier transforms. In other words, convolution in one domain (e.g., time domain) equals point-wise multiplication in the other domain (e.g., frequency domain). Versions of the convolution theorem are true for various Fourier-related transforms. Let f and g be two functions with convolution f*g . (Note that the asterisk denotes convolution in this context, and not multiplication. The tensor product symbol is sometimes used instead.)

Let F denote the Fourier transform operator, so F{f} and F{g} are the Fourier transforms of f and g , respectively. Then

F{f*g} = F{f}.F{g}

where . denotes point-wise multiplication.

It also works the other way around:

F{f.g}=F{f}.F{g}

By applying the inverse Fourier transform F^{-1}, we can write:

$f*g=F^{-1}\{F\{f\}.F\{g\}\}$

f.g=F{F{f}*F{g}}

5) Symmetry property of Fourier transform:

One of the discrete-time Fourier transform properties is that if a sequence is conjugate symmetric, $x[n] = x^*[-n]$, then the Fourier transform is real.

For example:

x=[1 2 3 2 1]

The sequence above appears to be symmetric, but the output of fft(x) is complex:

fft(x)

thenans=9.000 -2.1180-1.5388i 0.1180+0.3633i 0.1180-0.3633i -2.1180+1.5388i

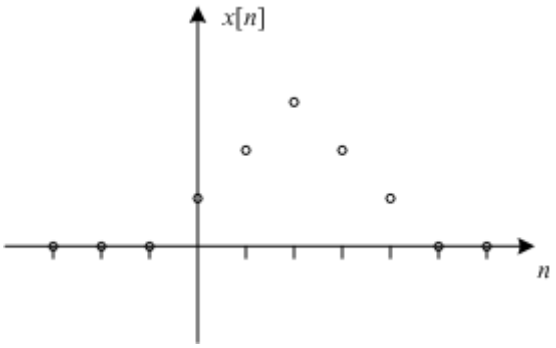


Fig. 3.1 Symmetry property of fourier transform

For example, the conjugate symmetry property for the discrete Fourier transform looks like this: if $x[n] = x^*[((-n))_N]$, then the DFT is real. Then x is circularly shifted as:

xs = circshift(x,[0 -2])

xs=3 2 1 1 2

Now take the DFT of xs:

fft(xs)

ans=9.000 2.6180 0.3820 2.6180

Now the output is real, as expected.

7 Results:

a) **Linearity:**

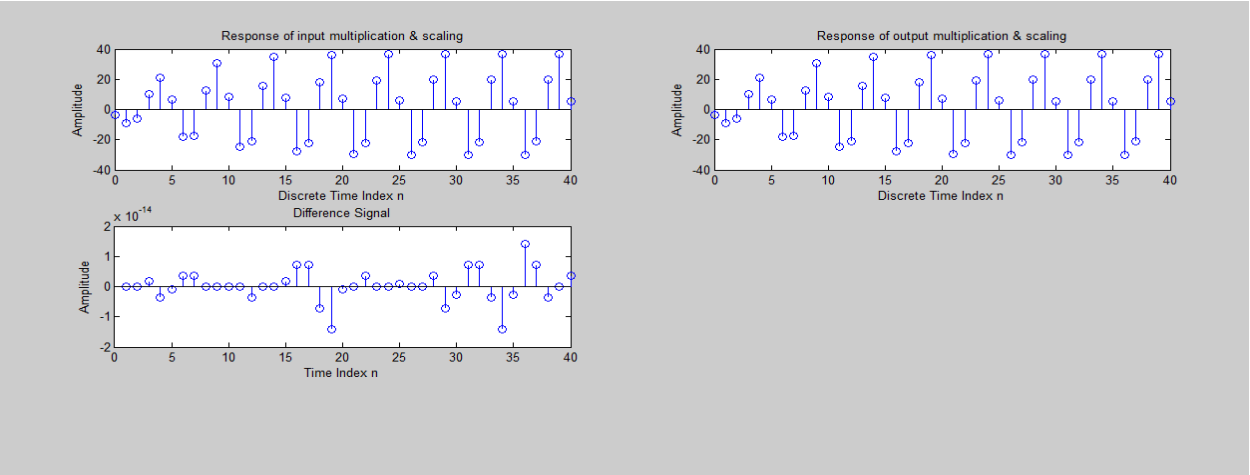


Fig. 3.2 Linearity Property of Fourier Transform

b) Time Shifting:

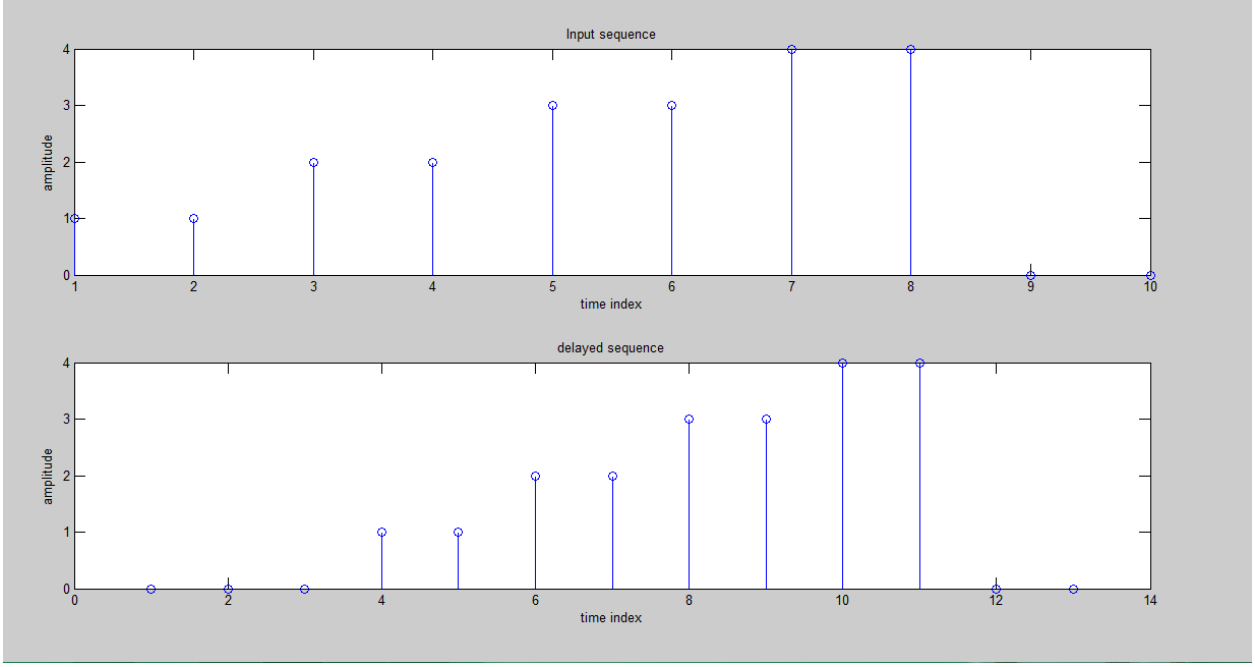


Fig. 3.3 Time Shifting Property of Fourier Transform

c) Frequency Shifting:

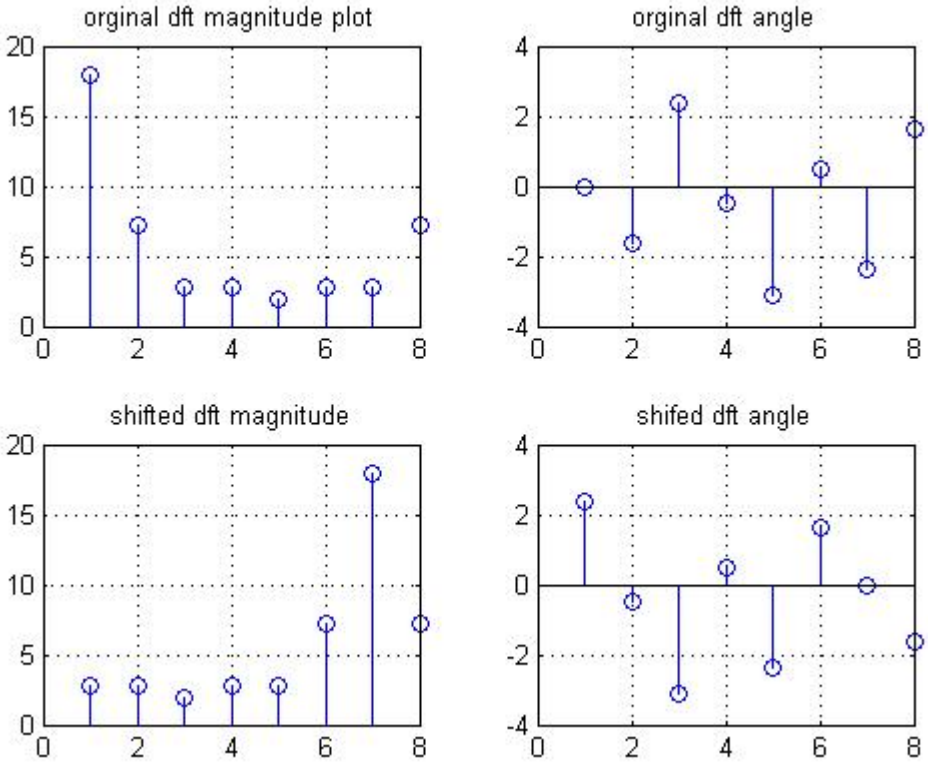


Fig. 3.4Frequency Shifting Property of Fourier Transform

8 ASSIGNMENT:

1. Write a MATLAB programof convolution theorem using Fourier transform.
2. Write a MATLAB programexhibiting symmetry property of Fourier transform.

9 Conclusion:

In this experiment, the motive was to introduce Fourier transform and its properties on various signals through FFT algorithm using MATLAB.

Experiment No. 4

Study of stochastic properties of signal

CONTENTS

1. Objective:

Study ofstochastic properties of signal

2. Expected Outcomes of Experiment:

1. Understanding the concept of a stochastic process.
2. Understanding randomness/ stochasticity, random experiment, random variable, and statistical averages like mean and variance.
3. Understanding correlation between two random signals and its measurement by correlation coefficient, auto-correlation and cross-correlation.
4. Understanding covariance between two random signals and its measurement by covariance coefficient, auto-covariance and cross-covariance.

3. Theory:

3.1 Deterministic Signal:

Each value of these signals is fixed and can be determined by a mathematical expression, rule, or table. Because of this, future values of any deterministic signal can be calculated from past values.

3.1.1 Periodic Signal:

A signal is a periodic signal if it completes a pattern within a measurable time frame, called a period and repeats that pattern over identical subsequent periods.

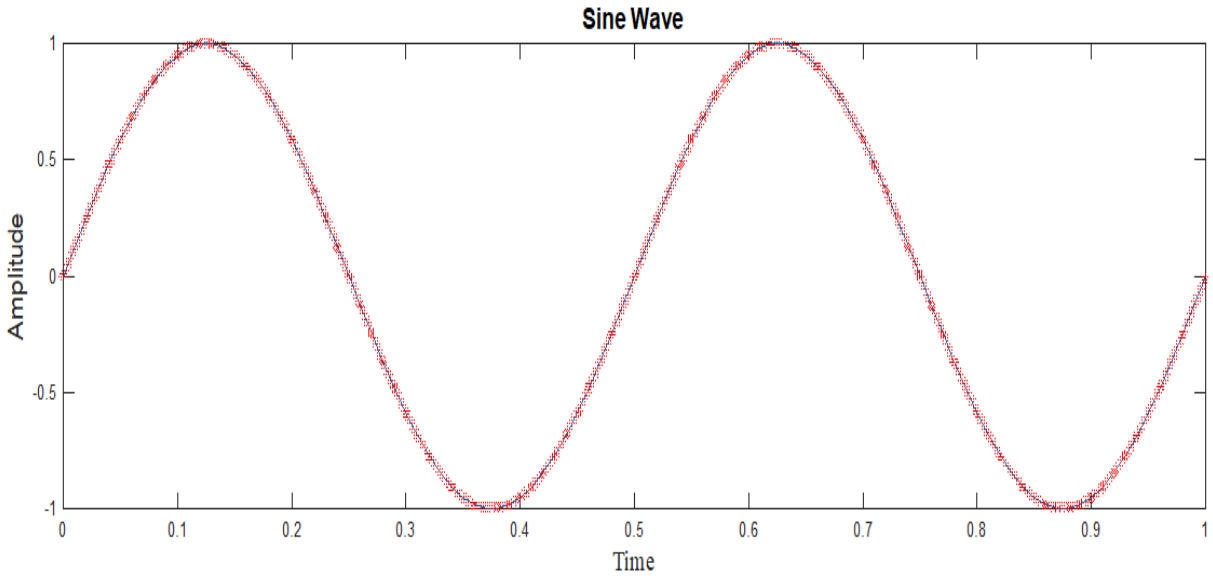


Fig. 4.1 A Periodic Signal

3.1.2 Aperiodic Signals:

A signal that does not repeats its pattern over a period is called aperiodic signal or non-periodic.

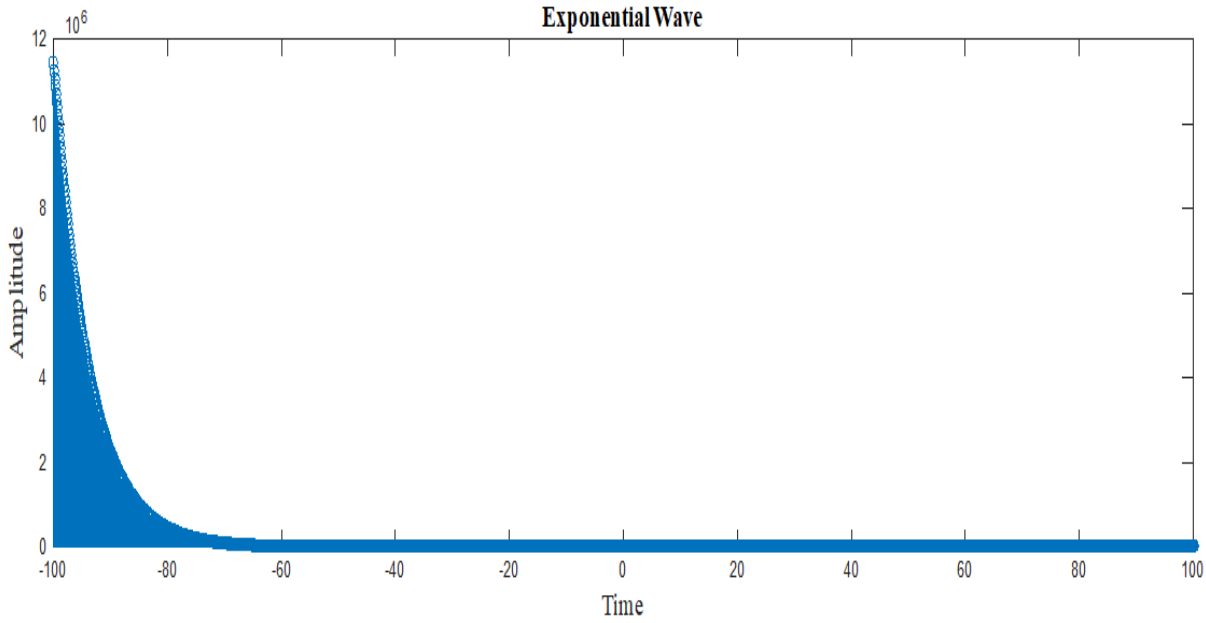


Fig.4. 2 An Aperiodic Signal

3.2 Stochastic Signal:

Stochastic/Random signals are those whose present value is neither depends of past nor on future values. Random signals cannot be characterized by a simple, well-defined mathematical equation. Hence their future values cannot be predicted. We must use probability and statistics to analyse their behaviour.

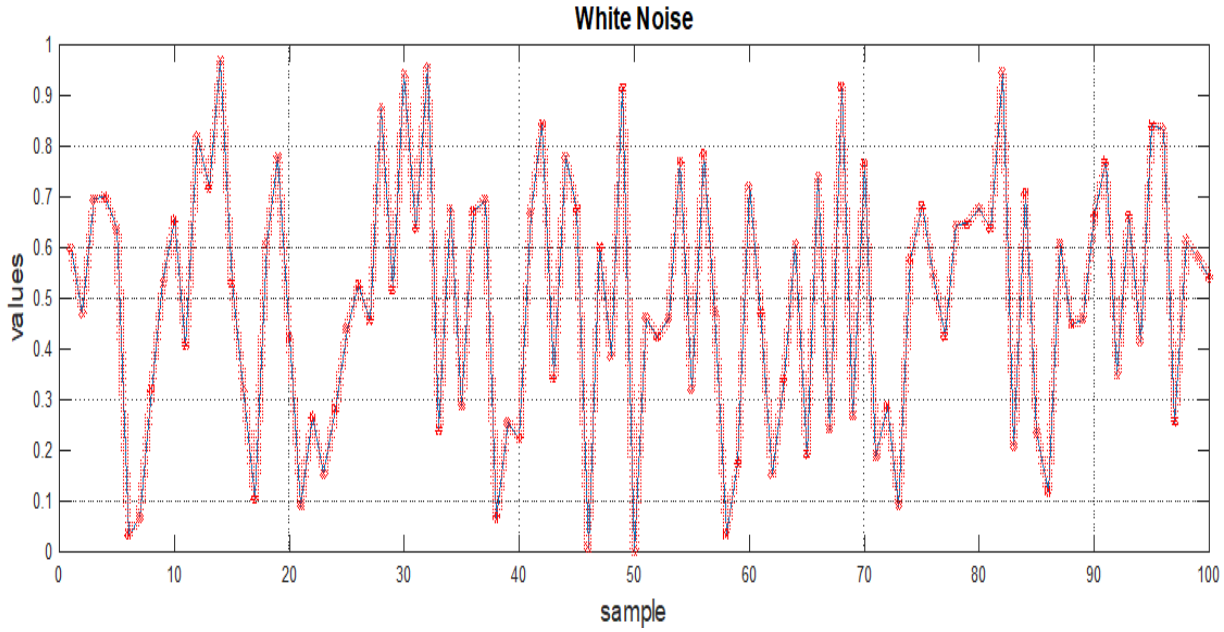


Fig. 4.3 Plot of a Stochastic Signal

3.2.1 Stationary Signals:

Statistical properties of the signal or process unchanged over time.

3.2.2 Nonstationary Signals:

Statistical properties of the signal/process change over time.

3.3 Statistical Properties:

3.3.1 Mean (μ):

A measure of the centre or location of a probability distribution and indicate the average value of a random variable.

$$\mu = \frac{\sum_{i=0}^{n} Xi}{n}$$

MATLAB code- *mean(A, dim)*

An *mean(A, dim)* returns the mean along dimension dim.

For example, if A is a matrix, then mean *(A,2)* is a column vector containing the mean of each row.

3.3.2 Variance (σ^2):

A measure of spread or dispersion of a distribution about its mean value.

$$\sigma^2 = \frac{\sum_{i=0}^{n}(Xi - \mu)^2}{n-1}$$

MATLAB code- *var(A)*

*var(A)*returns the variance of the elements of A along the first array dimension whose size does not equal 1.

3.3.3 Expectation (E):

A statistical average of a function of a random variable with respect to its probability density function.

$$E = \mu$$

3.4 Stochastic Properties:

3.4.1 Correlation:

A measure of the affine relationship between two random variables, denoted by $\boldsymbol{r_{xy}}$.

$$r_{(x,y)} = \frac{E([X - E(X)][Y - E(Y)])}{\sigma_y \sigma_x} = \frac{COV(X,Y)}{\sigma_y \sigma_x}$$

Where, $r_{(x,y)}$ = correlation of the variables *x* and *y*

cov(*x*, *y*) = covariance of the variables *x* and *y*

σ_x = sample standard deviation of the random variable *x*
σ_y = sample standard deviation of the random variable *y*

3.4.1.1 Auto-Correlation:

Correlation of a stochastic signal with itself as a function of lag is termed as autocorrelation function (ACF).

MATLAB code- *autocorr(y, numLags)*

Autocorr(y, numsLags) plots the ACF, where numLags indicates the number of lags in the sample ACF.

3.4.1.2 Cross-Correlation:

Correlation between the samples of two stochastic signal as a function of lag is termed as cross-correlation sequence.

MATLAB code- *crosscorr(x, y, numLags)*

Crosscorr(x, y, numLags) plots the sample cross correlation (XCF) between the two univariate, stochastic time series y1 and y2 with confidence bounds where numLags
indicates the number of lags in the sample XCF.

3.4.2 Covariance:

The property of a signal of retaining its form when the variables are linearly transformed.

$$COV(X,Y) = E([X - E(X)][Y - E(Y)])$$

x = the independent variable
y = the dependent variable
n = number of data points in the sample
$\bar{x}$ = the mean of the independent variable *x*
$\bar{y}$ = the mean of the dependent variable *y*

3.4.2.1 Auto-Covariance:

Covariance between the samples of a stochastic signal as a function of lag is termed the auto-covariance sequence.

MATLAB code- *cov(A)*
cov(A) returns the covariance.

Input-

```
A = [5 0 3 7; 1 -5 7 3; 4 9 8 10];
C = cov(A)
```

Output-

```
C =
    4.3333    8.8333   -3.0000    5.6667
    8.8333   50.3333    6.5000   24.1667
   -3.0000    6.5000    7.0000    1.0000
    5.6667   24.1667    1.0000   12.3333
```

Since the number of columns of A is 4, the result is a 4-by-4 matrix

3.4.2.2 Cross-Covariance:

Covariance between the samples of two stochastic signals as a function of lag is termed the cross-covariance sequence.

MATLAB code- x*cov(A, B)*

```
c = xcov(x, y)
```

c = xcov(x, y) returns the cross-covariance of two discrete-time sequences, x and y. Cross-covariance measures the similarity between x and shifted (lagged) copies of y as a function of the lag. If x and y have different lengths, the function appends zeros at the end of the shorter vector so it has the same length as the other.

4. Equipment Required:

A PC installed with MATLAB software.

5. Procedure:

- Open MATLAB in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.
- Run the script.

- Desired result will be displayed on the command window

6. CODING:

6.1 Autocorrelation of continuous Sine wave

```
clc;

t=0:.01:2;
f=2;
a=1;
X=a*sin(2*pi*f*t);

subplot(2,1,1)
plot(t,X);                  %plotting Sine wave
title('Sine Wave');
xlabel('Time')
ylabel('Amplitude')
grid on;

subplot(2,1,2)
autocorr(X,[100]);          %plotting Autocorrelation of Sine wave

grid on;
```

6.2 Cross-Correlation between two stochastic signals:

```
clc;

X=rand(1,100)               %Generating an array(100) of Random Numbers i.e.
X
subplot(3,1,1)
plot(X)                     % Plotting X matrix
title('White Noise 1');
xlabel('Time')
ylabel('Amplitude')

Y=rand(1,100)               %Generating an array(100) of Random Numbers i.e.
Y
subplot(3,1,2)
plot(Y)                     % Plotting X matrix
title('White Noise 2');
```

```
xlabel('Time')
ylabel('Amplitude')

subplot(3,1,3)
crosscorr(X,Y,50)          %Cross Correlation between X & Y signal
```

6.3 Auto-Covariance of discrete Sine wave:

```
clc;

A=1;
Omega=2*pi/12;             % Angular frequency
phi=0;
n=-10:10;
y=A*cos(Omega*n);


subplot(2,1,1)
stem(n,y)                  % Plotting discrete Sine wave
title('SineWave');
xlabel('Time')
ylabel('Amplitude')
grid on;

subplot(2,1,2)
[cov_ww,lags] = xcov(y,30,'coeff');
stem(lags,cov_ww)          % Plotting Auto covariance of Discrete Sine Signal
title('Auto covariance of Sine wave');
xlabel('Time Lag')
ylabel('Coeff of Auto-cov')
grid on;
```

6.4 Cross-Covariance of Sine and Cosine wave:

```
clc;

t=0:.01:1;
f=2;
a=2;
X=a*sin(2*pi*f*t);

subplot(3,1,1)
plot(t,X);                 %Plotting Sine wave
title('Sine Wave');
xlabel('Time')
```

```
ylabel('Amplitude')

t=0:.01:1;
f=2;
a=2;
Y=a*cos(2*pi*f*t);
subplot(3,1,2)
plot(t,Y);                    %Plotting Cosine wave
title('Cosine Wave');
xlabel('Time')
ylabel('Amplitude')
subplot(3,1,3)
[cov_ww,lags] = xcov(X,Y,'coeff');
stem(lags,cov_ww)          % Plotting Cross covariance of Sine & Cosine
signals
title('Cross covariance of Sine & Cosine');
xlabel('Time Lag')
ylabel('Coeff of Cross-Cov')
grid on;
```

7 Results:

7.1 Autocorrelation of sine wave (spanning time 0 to 2 sec):

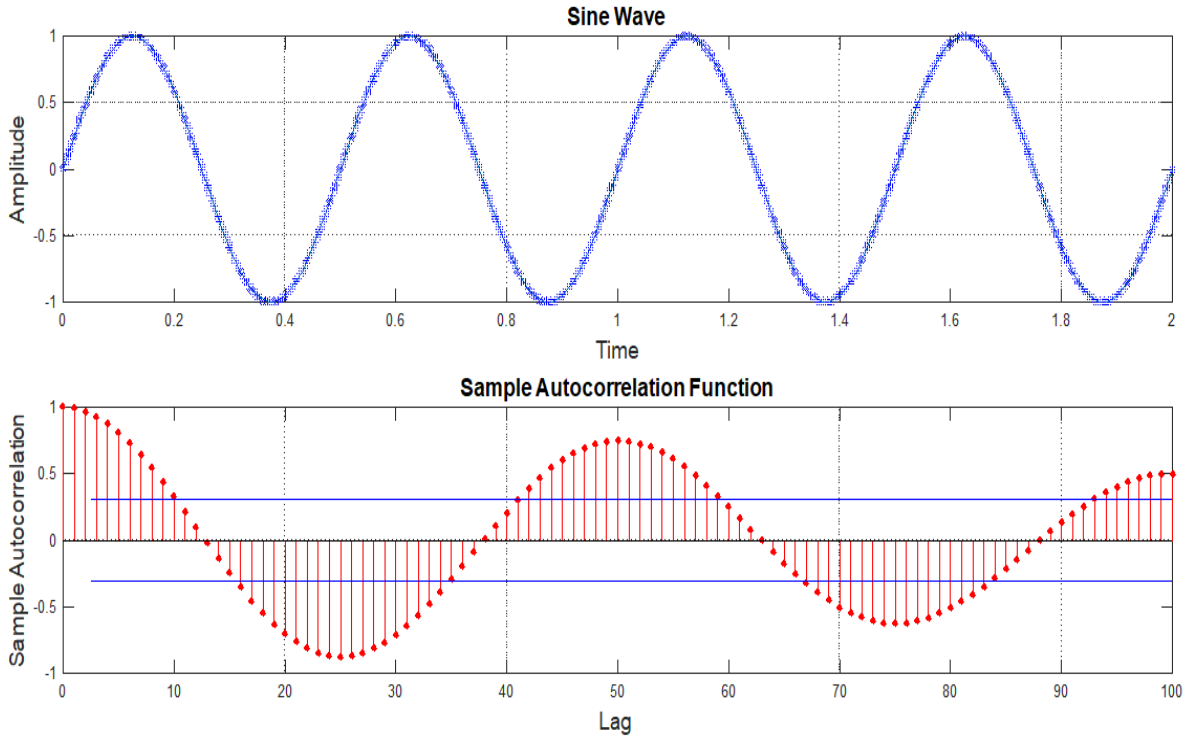


Fig. 4.4 Auto-correlation of sine wave

The correlogram shows the peak correlations at lags 0, 25, 50, 75 & 100.
Zero correlation at lags 13, 38, 63 & 88. Each lag is 0.02 sec.

7.2 Cross-correlation between two stochastic signals:

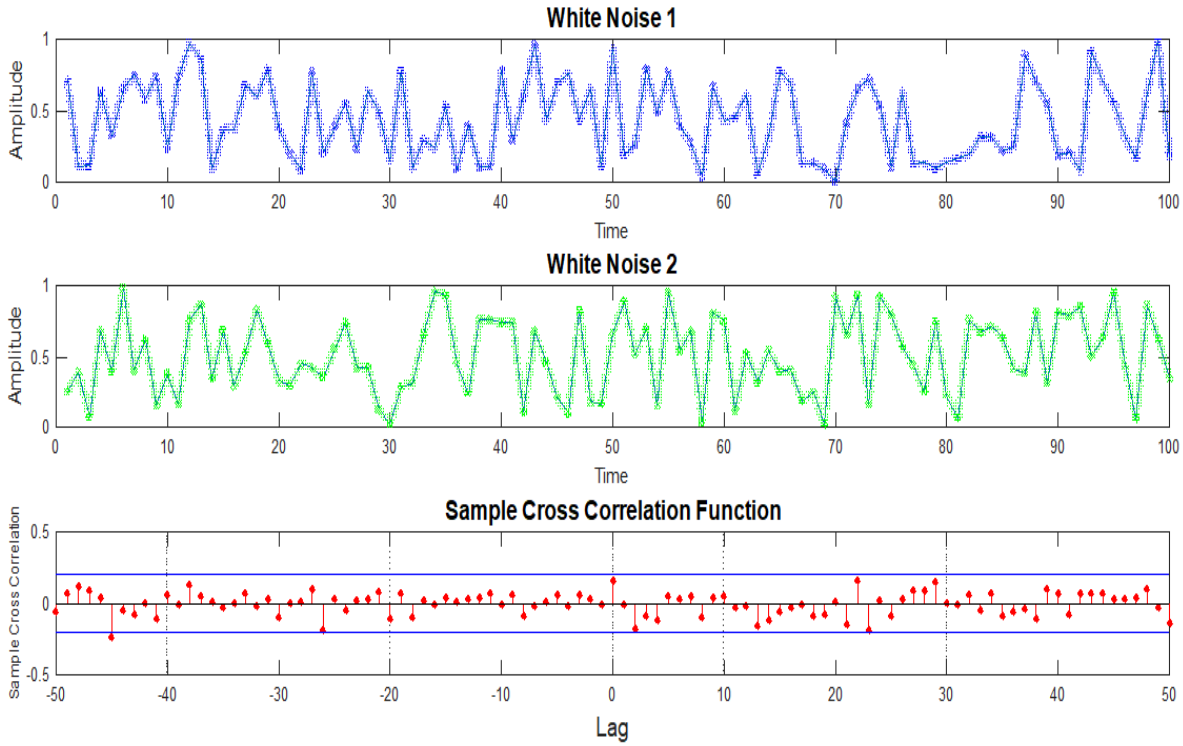


Fig.4. 5 Cross-correlation of two random signals

As the coefficient of cross-correlation in fig. 5 is less than 0.2 for all lags excluding lags -45, -26, 0, 2, 22, 23. It concludes that, two white noise are barely related to each other. Each lag is 2 sec.

7.3 Auto-Covariance of discrete Sine wave (spanning time -10 to 10 sec):

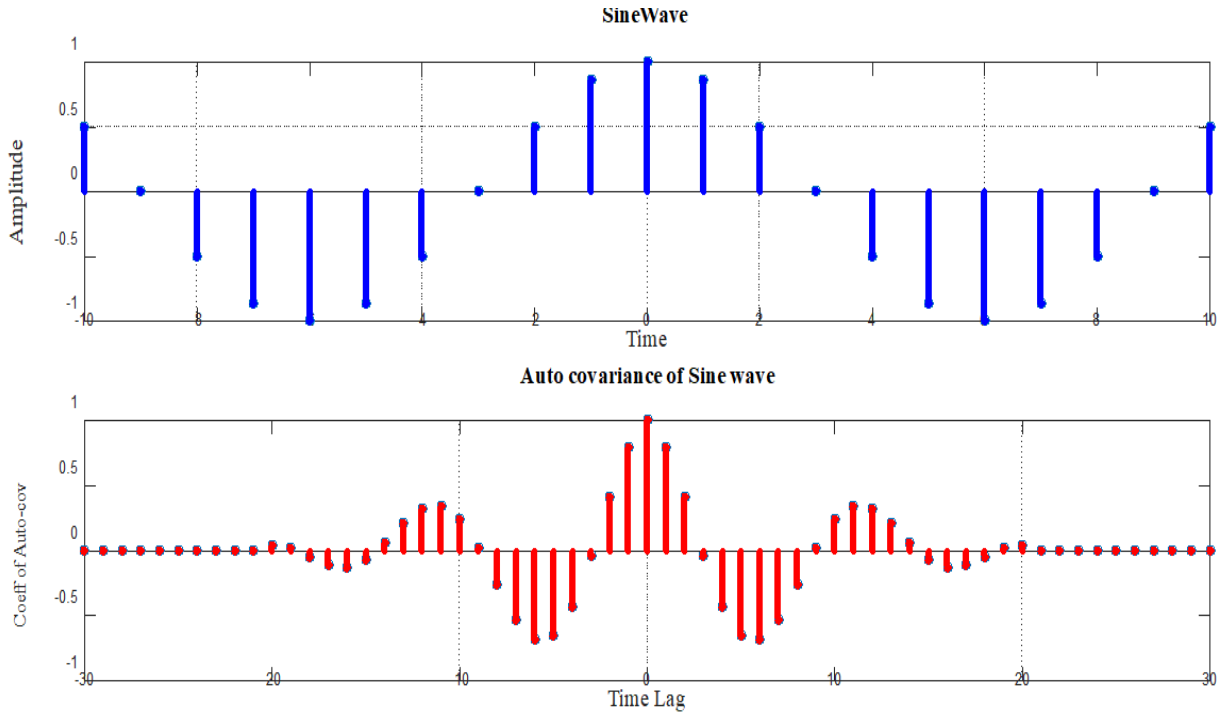


Fig. 4.6 Auto-covariance of discrete sine wave

The peaks of Auto-Covariance of discrete Sine wave in fig. 6 are at lag 0, -6, -12, 6, 12. The zero coefficient of auto-covariance are at lag -3, 3, -9, 9, before -20 & after 20. Each lag is 1 sec.

7.4 Cross-Covariance between Sine and Cosine wave (spanning time 0 to 1 sec):

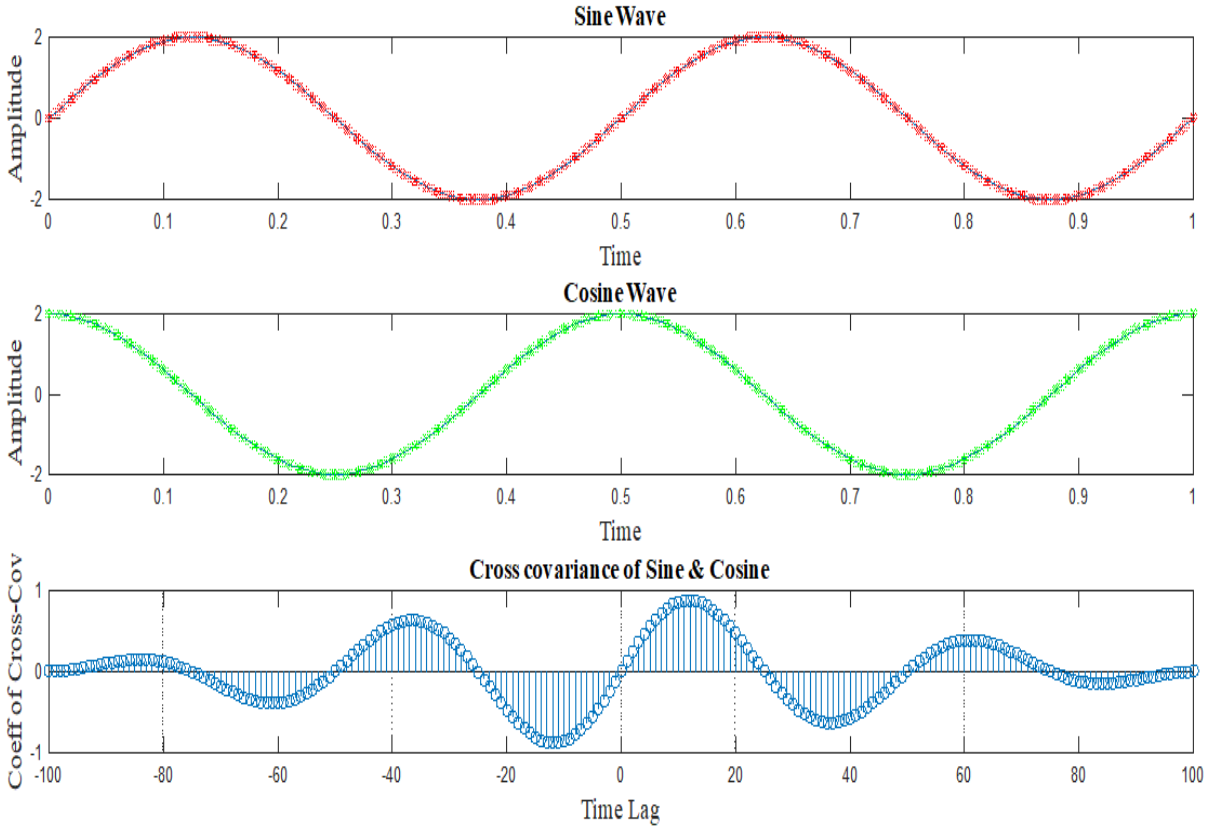


Fig. 4.7 Cross-covariance of sine and cosine wave

Peaks of coefficient of cross-covariance in fig. 4.7 are at lag -12, -13, 12, 13, -37, -38, 37, 38 while zero coefficient are at lag -100, -75, -50, -25, 0, 25, 50 & 100. Each time lag is 0.01sec.

8 Assignments:

1. Plot auto-correlation of a *stochastic signal* of time 50 sec. Find the peaks and zeroes in the coefficient of auto-correlation plot.
2. Plot cross-correlation between continuous Sine and Cosine wave (Amplitude = 2 & frequency = 2 Hz) signal. Find the time lags where both signals are maximally correlated.
3. Plot auto-covariance of a stochastic signal of time 100 sec. Comment on the auto-covariance of the random signal.
4. Plot cross-covariance between discrete exponential signal and sine signals choosing parameter of your own. Find the time lags for peaks of the coefficient of cross-covariance function.

9. Conclusion:

In this experiment, the motive was to introduce stochastic properties of different kinds of signals such as Continuous time signal, Gaussian white noise and discrete time signal and simulate them in MATLAB.

EXPERIMENT NO: 5

Study of Discrete Signals through MATLAB

CONTENTS

1. Objective

Study of Discrete Signals through MATLAB

2. Expected Outcomes of Experiment

1. Acquiring and plotting various discrete signals like Discrete Square Wave, Discrete Exponential Wave, Discrete Sinusoidal Signal, Unit Step Sequences etc. using Matlab.
2. Understanding and implementing various transformations on Discrete signals.
3. Visualizing Impulse signals in discrete domain and their properties.
4. Visualizing odd and even symmetries in signal.

3. Theory

Any time varying physical phenomenon that can convey information is called signal. Some examples of signals are human voice, electrocardiogram, sign language, videos etc. There are several classifications of signals such as Continuous time signal, discrete time signal and digital signal, random signals and non-random signals.

Signals that can be defined at discrete instant of time is called "discrete time signal". Basically discrete time signals can be obtained by sampling a continuous-time signal as shown in Fig.1. x(n) is discrete signal

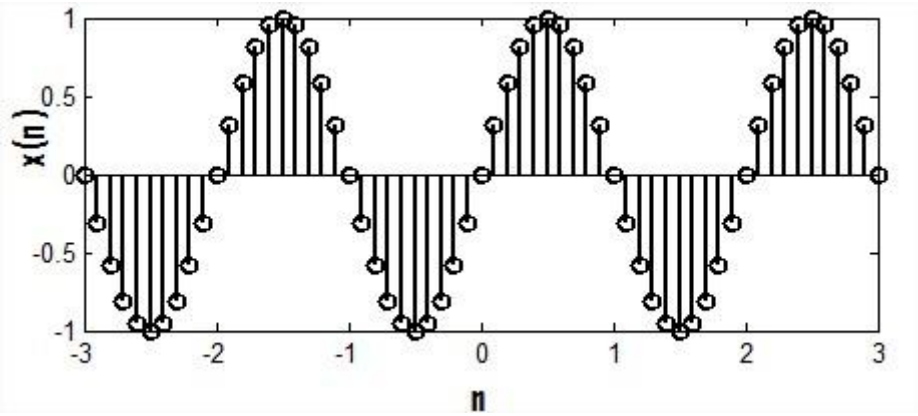


Fig5.1. Discrete signal

It can also be defined in other ways as "**A discrete signal or discrete-time signal is a time series consisting of a sequence of quantities".**

Unlike a continuous-time signal, a discrete-time signal is not a function of a continuous

argument. However, it may have been obtained by sampling from a continuous-time signal. When a discrete-time signal is obtained by sampling a sequence at uniformly spaced times, it has an associated sampling rate.

To distinguish between continuous-time and discrete-time signals we use symbol‘t’ to denote the continuous variable and **‘n’** to denote the discrete-time variable.

A discrete-time signal x[n] may represent a phenomenon for which the independent variable is inherently discrete. A discrete-time signal x[n] may represent successive samples of an underlying phenomenon for which the independent variable is continuous. For example, the processing of speech on a digital computer requires the use of a discrete time sequence representing the values of the continuous-time speech signal at discrete points of time.

Discrete-time signals may have several origins, but can usually be classified into one of two groups:

- By acquiring values of an analog signal at constant or variable rate. This process is called sampling.
- By observing an inherently discrete-time process, such as the weekly peak value of a particular economic indicator.

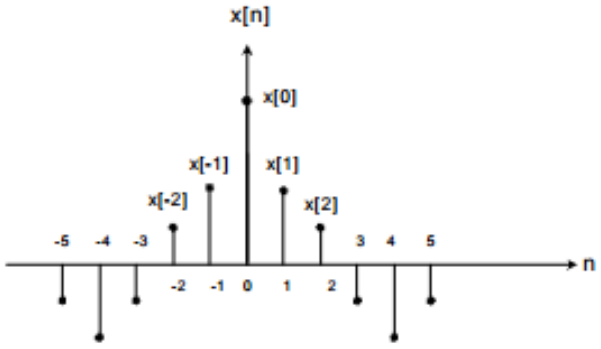


Fig.5.2 Graphical representation of discrete signal.

3.1. Signaloperations

Signal operations play important role in the analysis of signals and systems. A number of important operations are performed on the signals and it is necessary to understand these operations in detail. All these operations are discussed below.

3.1.1. Operations Performed on Discrete Signals

Three important signal operations that can be performed on the signals are *shifting, scaling and reversal (folding)*. Since time is taken as independent variable therefore these operations are also known as *time-shifting, time-scaling and time-reversal* respectively. All these operations are discussed below.

a) Time Shifting: Consider a discrete time signal x[n], then the time shifted version of the signal x[n] represented as y[n] is defined as

$$y[n] = x[n+n_0]$$

Now, n_0 can be negative or positive. If n_0 is negative, signal is shifted backwards and this is called "Time-Delayed Shifting". If n_0 is positive, signal is shifted forward and this is called "Time- Advanced Shifting". Example of time-shifting is shown in Fig.5.3.

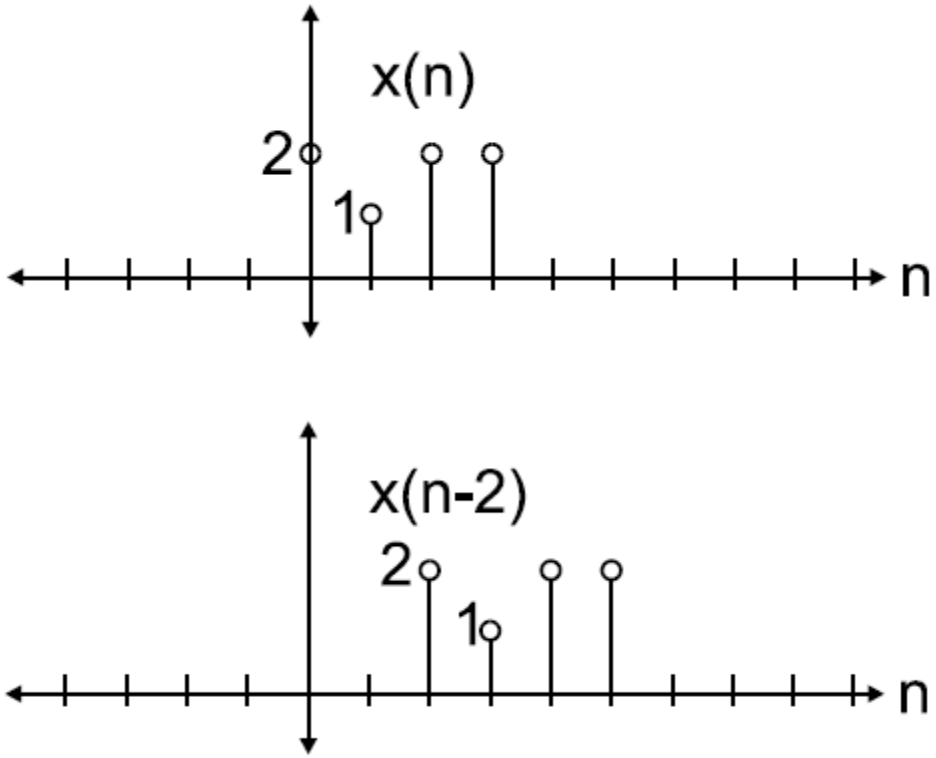


Fig. 5.3 Signal x[n] and its time-delayed version x [n-2]

b) Time Scaling:Consider a discrete time signal x[n],then "*time-scaled*" version of the signal x[n] can be represented as y[n] and is defined as ,

$$y[n]=x[an]$$

If 'a' is greater than 0 and is an integer, then the number of samples in signal y[n] reduces from that of x[n] and this is called " Time-Compression/Decimation".

If 'a' is greater than 0 and is a positive non integer, then the number of samples in

signal y[n] increases from that of x[n] and this is called "Time-Expansion/Interpolation".

For example, let signal x[n] = {7, 9, 3, 2, 5, 4, 8}

Then signal, y[n] =x [2n] can be represented as y[n] = {9, 2, 4} and y[n] represents the Time-Compressed signal. ↑

Similarly, let x[n] = {7,-2, 5},

Then signal y[n] =x [n/2] canbe represented i.e. y[n] = {7, 0,-2, 0, 5}.

Signal y[n] represents the Time-Expanded signal.↑

Note: Student must try to draw signal x[n] and y[n] graphically in matlab on their own and record in their workbook and arrow represents n=0 i.e. origin

c) Time Reversal: Consider a discrete signal x[n] , then the time reversal version y[n] of signal x[n] is defined as,

$$y[n] = x[-n]$$

This is being shown in Fig.5.5

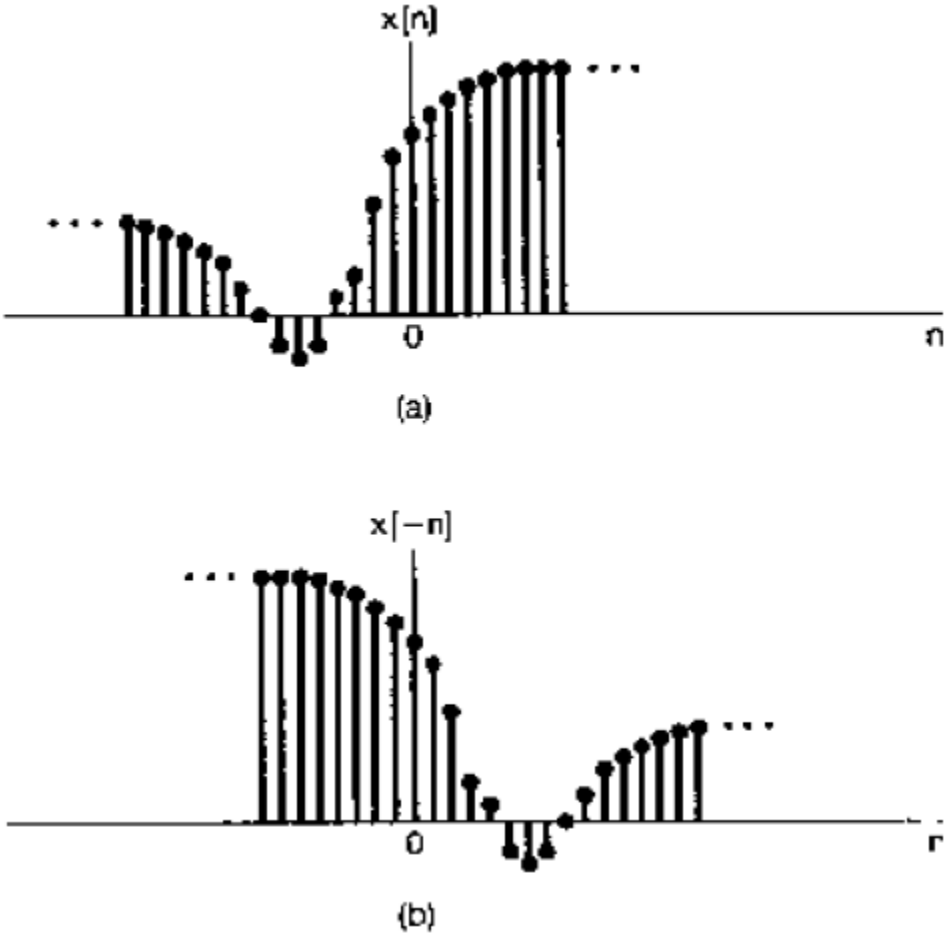


Fig. 5.4 Discrete Signal x[n] and its time-reversal version y[n] =x[-n] about n=0

d) **AMPLITUDE SCALING**: Amplitude scaling of a discrete-time signal can be represented by,

$$y[n] = ax[n], \text{ where 'a' is constant}$$

If a > 1, it is amplification and if a < 1 it is attenuation. For example, as shown in 8fig.7 below

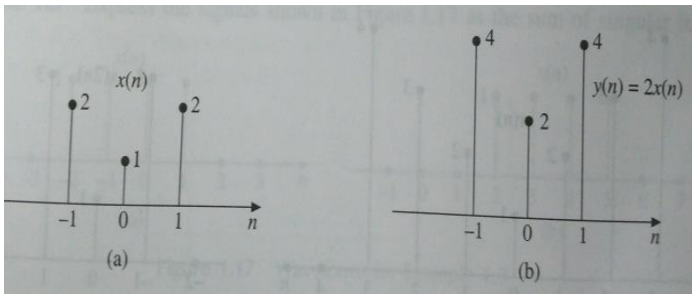


Fig. 5.5 Signal x(n) and its amplitude scaled version y(n)

e) **SIGNAL ADDITION & SUBTRACTION:** In discrete-time domain, the sum of two signals $x_1[n]$ and $x_2[n]$ can be obtained by adding the corresponding sample values

Similarly the subtraction of $x_2[n]$ from $x_1[n]$ can be obtained by subtracting each sample of $x_2[n]$ from the corresponding sample of $x_1[n]$.

Let $x_1[n] = \{1, 2, 3, 1, 5\}$ and $x_2[n] = \{2, 3, 4, 1, -2\}$. Then

$x_1[n] + x_2[n] = \{3, 5, 7, 2, 3\}$

$x_1[n] - x_2[n] = \{-1, -1, -1, 0, 7\}$

f) Signal multiplication: The multiplication of two discrete time sequences is done by multiplying their values at sampling instants as shown below.

If x1[n] = {1, 2, 3, 1, 5} and $x_2[n] = \{2, 3, 4, 1, -2\}$. Then

$x_1[n] . x_2[n] = \{2, 6, 12, 1, -10\}$.

3.2EVEN AND ODD SIGNALS

Any discrete signal x[n] is an even signal if,

$$\mathbf{x[-n] = x[n]}$$

For example, let a signal x[n] is defined as $x[n] = \begin{cases} \frac{1}{2}, & n < 0 \\ 0, & n = 0 \\ \frac{1}{2}, & n > 0 \end{cases}$

Signal x[n] is an even signal as it holds the definition of even signal. Above signal can be graphically represented as:

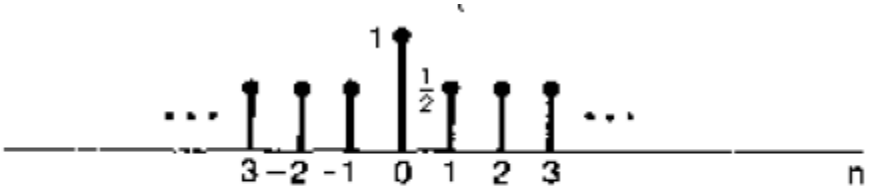


Fig. 5.6 Even Signal

Similarly, any discrete signal x[n] is an odd signal if,

$$x[-n] = -x[n]$$

For example, let a signal x[n] is defined as

$$x[n]= \begin{cases} \frac{-1}{2}\ ,\ n<0 \\ 0\ ,\ n=0 \\ \frac{1}{2}\ \ ,n>0 \end{cases}$$

Signal x[n] is an even signal as it holds the definition of even signal. Above signal can be graphically represented as below:

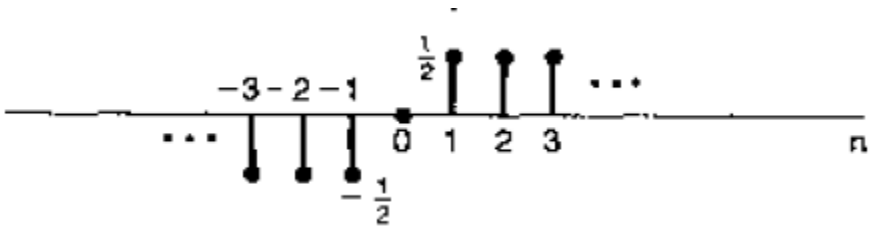


Fig 5.7 Odd Signal

It is important to note that any signal x[n] can be broken into two parts which are even and odd respectively as follows:

$$x_{even}[n]= \frac{x[n]+x[-n]}{2}$$

$$x_{odd}[n]=\frac{x[n]-x[-n]}{2}$$

3.3 CONVOLUTION

In discrete time, convolution of two signals involves summing the product of the two signals, where one of the signals is “flipped and shifted”.

In other words, convolution is also defined as an operation between the input signal to a system, and its impulse response, resulting in the output signal.

Mathematically convolution is given by,

$$\mathbf{y[n] = x[n] * z[n] = \sum_{-\infty}^{+\infty} x[k]z[n-k]}$$

where, x[n] and z[n] are the two discrete signals to be convolved.

3.3.1 PROPERTIES OF CONVOLUTION

- Commutative property: x[n] * h[n] = h[n] * x[n]
- Associative property: (x[n] * h_1[n]) * h_2[n] = x[n] * (h_1[n] * h_2[n])
- Distributive property: x[n] * (h_1[n]+h_2[n]) = x[n] * h_1[n]+ x[n]* h_2[n])
- Shifting property: if x[n] * h[n] = y[n] , then x[n-k] * h[n-m] = y[n-k-m]
- Convolution with an impulse: x[n] * δ[n] = x[n]

3.4 DISCRETE IMPULSE FUNCTION

The discrete- time unit impulse function δ(n), also called unit sample sequence, is defined as:

$$\delta(n) = \begin{cases} 1 \; for n = 0 \\ 0 \; for n \neq 0 \end{cases}$$

This means that unit sample sequence is a signal that is zero everywhere, except at n=0, where its value is unity. It is the most widely used elementary signal used for analysis of signal and systems

3.4.1 PROPERTIES OF DISCRETE IMPULSE FUNCTION

a) δ(n) = u(n) - u(n-1) where u(n) is discrete unite step function and u(n-1) is its time shifted version.

b) $\delta(n-k) = \begin{cases} 1 \; for n = k \\ 0 \; for n \neq k \end{cases}$.

c) δ(n) is an even function.

d) δ(an) = δ(n).

e) x(n)= $\sum_{k=-\infty}^{k=+\infty} x(k)\delta(n-k)$.

f) x(n)*δ(n-n_1) = x(n-n_1).

g) x(n) .δ(n-n_1) = x(n_1) .δ(n-n_1) .

3.5 Z-TRANSFORM

The bilateral or two sided Z- transform of a discrete signal x[n] is defined as:

$$\mathbf{X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}}$$

Where ‘z’ is a complex variable.

The one sided or unilateral Z-transform is defined as:

$$\mathbf{X(z)= \sum_{n=0}^{\infty} x[n]z^{-n}}$$

3.5.1 CONVOLUTION PROPERTY OF Z-TRANSFORM

The convolution property of z-transform states that the Z-transform of the convolution of two signals is equal to the multiplication of their z-transform, i.e.

If $x_1(n) \leftrightarrow X_1(z)$ (means $X_1(z)$ is z-transform of $x_1(n)$) with ROC=R_1 and $x_2(n) \leftrightarrow X_2(z)$ then,

$$\mathbf{x_1(n)* x_2(n) \leftrightarrow X_1(z). X_2(z)}$$

4. Equipment Required:

A PC installed with MATLAB software (preferably Matlab R 2017a)

5. Procedure

- Open Matlab in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.
- Run the script.
- Desired result will be displayed on the command window.

6 CODING

(a) Discrete square wave:

```
A=1;
Omega=pi/4;
rho=0.5;
n=-10:10;
x=A*square(Omega*n+rho);
stem(n,x)
```

(b) Discrete exponential wave:

```
B=1;
r=0.85;
n=-10:10;
x=B*power(r,n);
stem(n,x)
```

Note: 0<r<1 for Decaying exponential and r>1 for Growing exponential

(c) Discrete sinusoidal signal:

```
A=1;

Omega=2*pi/12; % this is angular frequency

phi=0;

n=-10:10;

y=A*cos(Omega*n);

stem(n,y)
```

(d) Unit Step Sequence

```
No=1;

n1=-10;

n2=10;

n= [n1:n2];

x= [(n-no)>=0];

stem(n,x)
```

(e) Even and Odd components of a sequence y(n)=u(n)-u(n-10).

```
n= -15:1:15
y1=[zeros(1,15),ones(1,10),zeros(1,6)]
y2=fliplr(y1)
ye=0.5*(y1+y2)
yo=0.5*(y1-y2)
subplot(2,2,1)
stem(n,y1)
xlabel('time ')
ylabel('amplitude')
title('y(n)')
subplot(2,2,2)
stem(n,y2)
xlabel('time ')
ylabel('amplitude')
title('y(-n)')
subplot(2,2,3)
stem(n,ye)
xlabel('time ')
ylabel('amplitude')
```

```
title('ye(n)')
subplot(2,2,4)
stem(n,yo)
xlabel('time ')
ylabel('amplitude')
title('yo(n)')
```

(f) Multiplication of discrete-time signals.

```
% x1(n)= 6*a^n    ……signal 1
n=0:0.1:5
a=2
x1=6*(a.^n)
subplot(3,1,1)
stem(n,x1)
title('x1(n)')
% x2(n)=2*cos(wn)……signal 2
f=1.2
x2=2*cos(2*pi*f*n)
subplot(3,1,2)
stem(n,x2)
title('x2(n)')
%multiplication of two sequences
y=x1.*x2 ……..multiplication of two signals
subplot(3,1,3)
stem(n,y)
xlabel('time n')
ylabel('amplitude')
title('y(n)')
```

(g) **Convolution of two sequences**

```
x1=[1 2 0 1]
x2=[2 2 1 1]
y=conv(x1,x2)
disp('the convolution output is')
disp(y)
subplot(3,1,1)
stem(x1)
xlabel('Discrete time')
ylabel('amplitude')
title('first input sequence')
subplot(3,1,2)
stem(x2)
xlabel('Discrete time')
ylabel('amplitude')
title('second input sequence')
```

```
subplot(3,1,3)
stem(y)
xlabel('Discrete time')
ylabel('amplitude')
title('convolution output')
```

7. RESULTS AND DISCUSSION:

a) **Discrete square wave:**

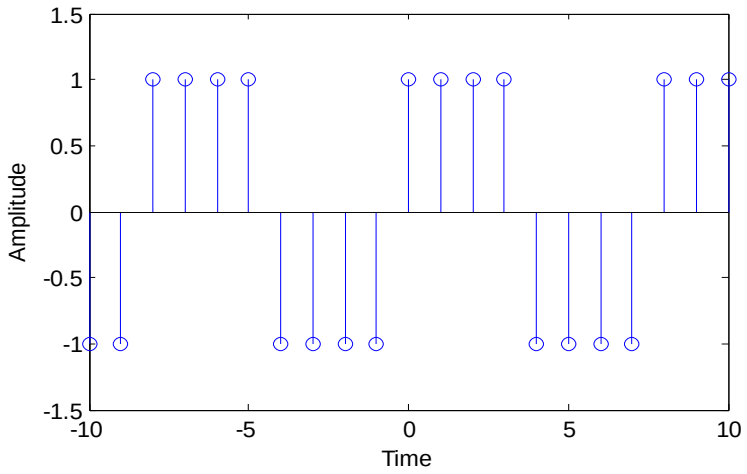


Fig 5.8Discrete square wave

b) **Discrete exponential wave:**

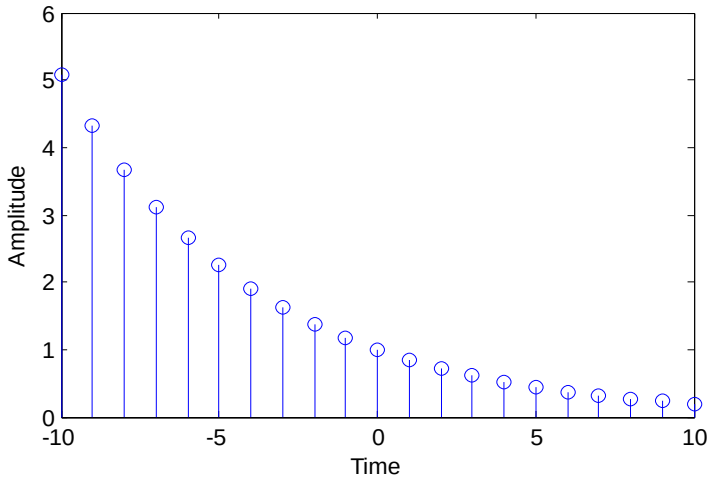


Fig 5.9Discrete exponential wave

c) **Discrete sinusoidal signal:**

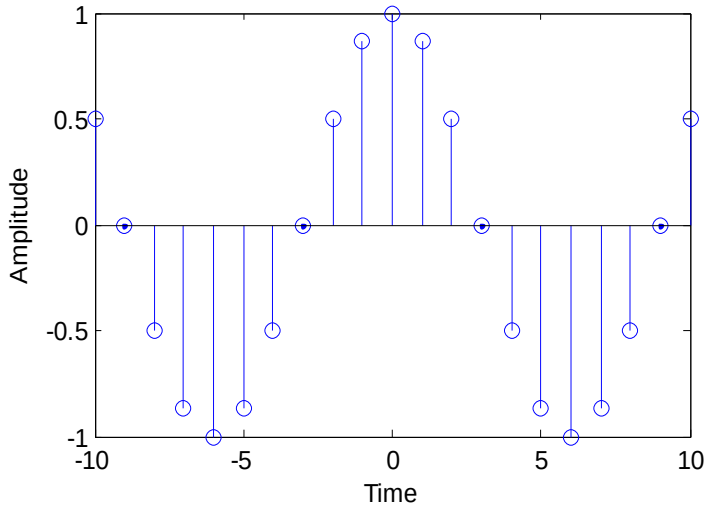


Fig 5.10 Discrete sinusoidal signal

d) Unit Step sequence:

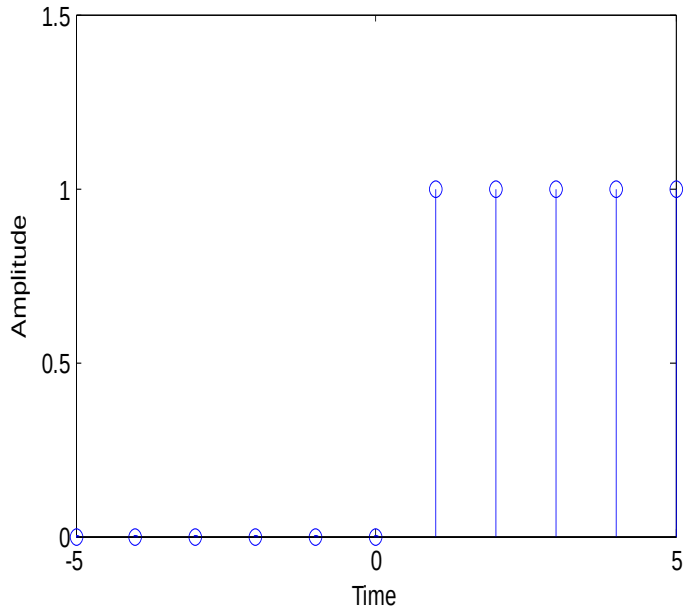


Fig 5.11 Unit Step sequence

e) Even and Odd components of a sequence y(n)=u(n)-u(n-10)

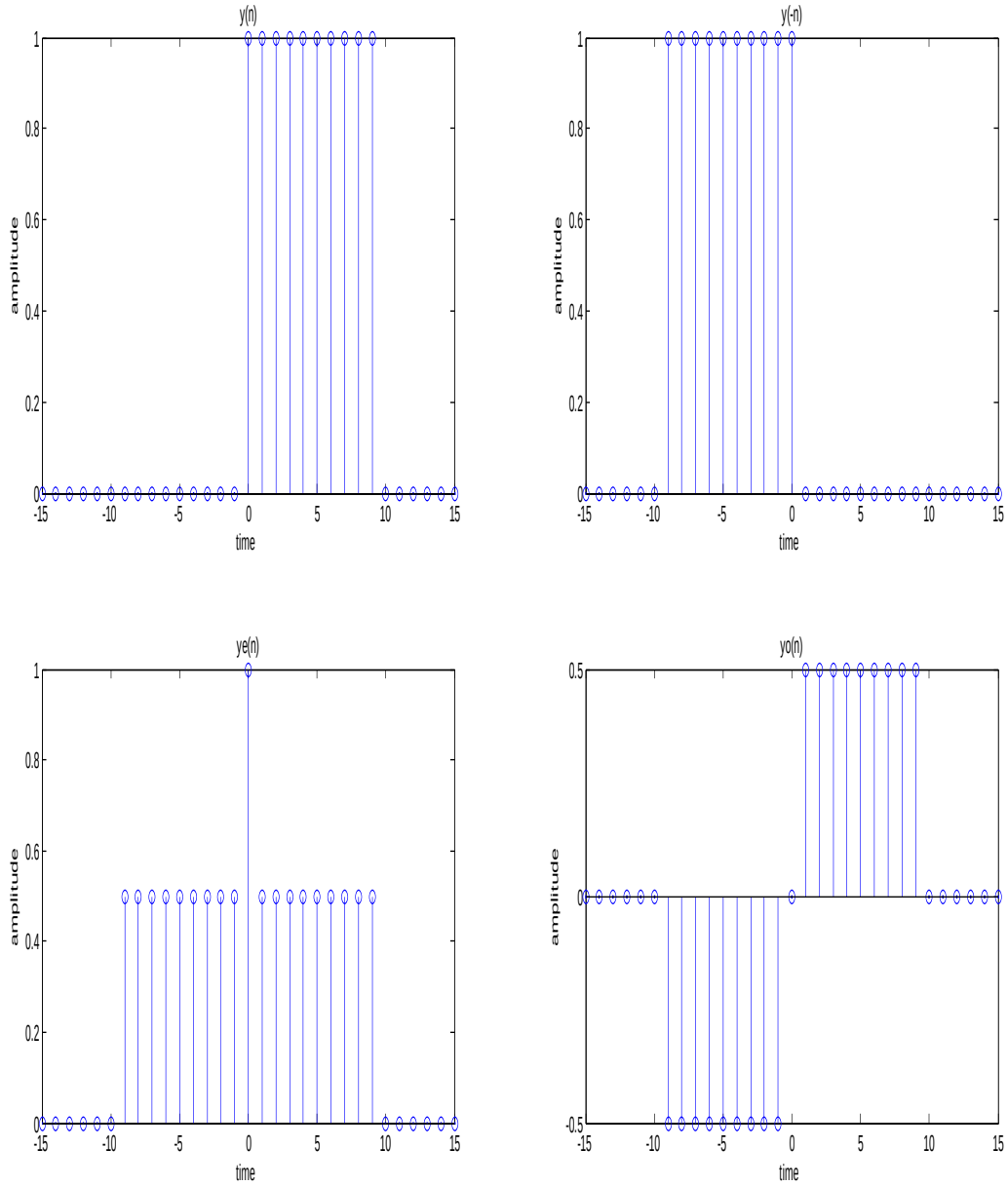


Fig 5.12 Even and Odd components

f) Multiplication of two discrete signals

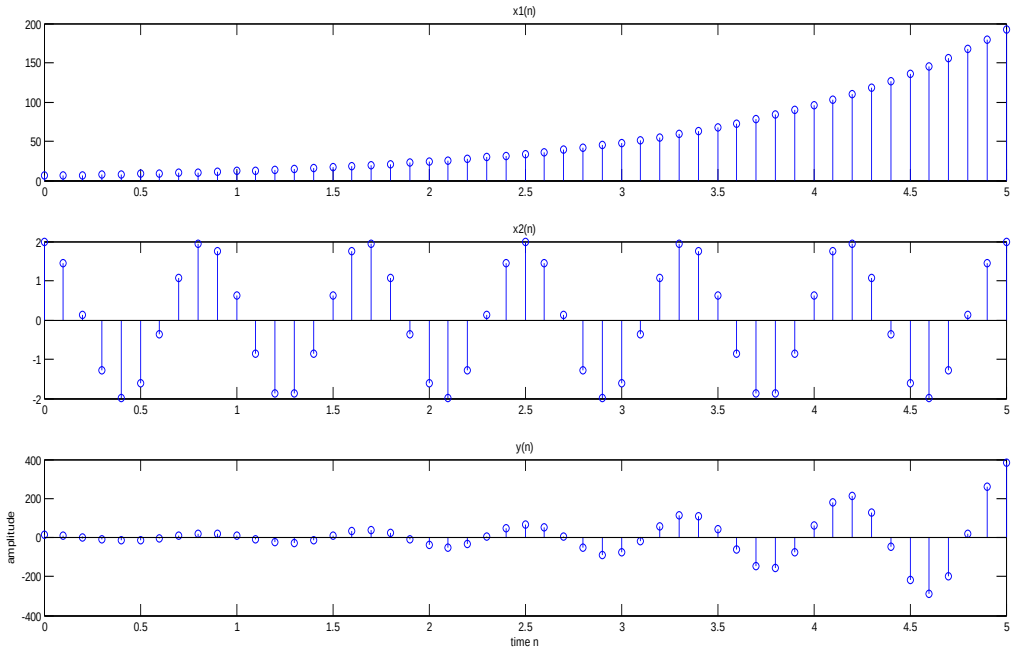


Fig 5.13 Multiplication of two discrete signals

g) Convolution of two sequences

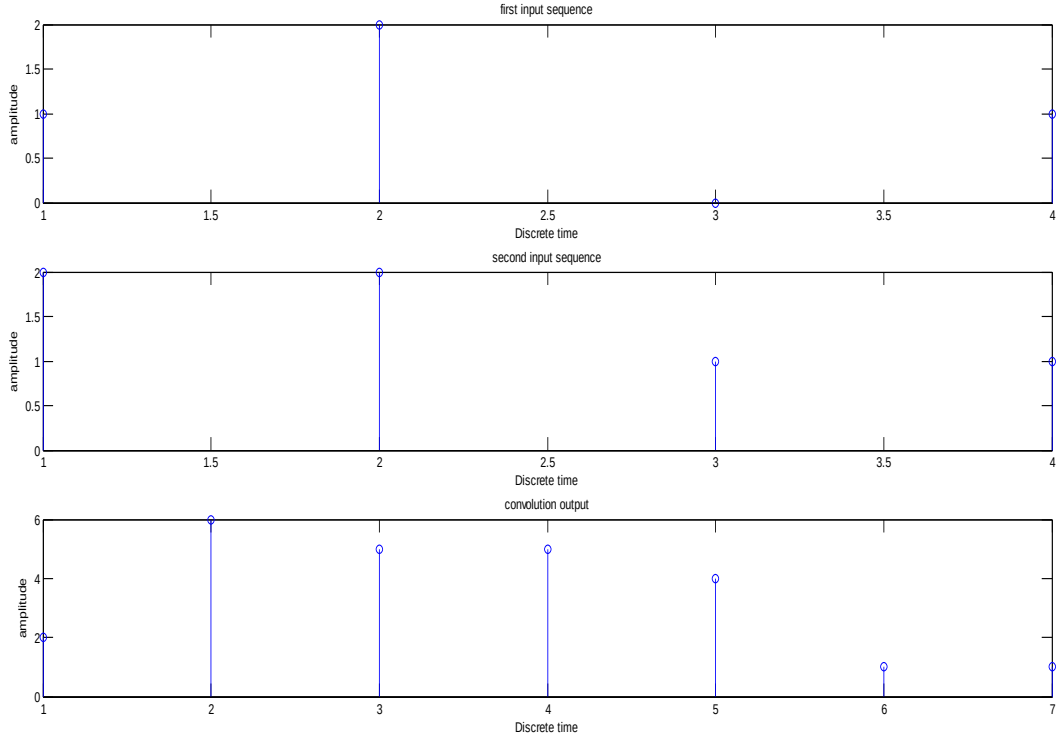


Fig 5.14 Convolution of two sequences

The convolution output is [2 6 5 5 4 1 1]

8. Assignments

1. Write Matlabcode to generate “discrete triangular wave”. Apply the following properties on the generated waveform:
    a. Amplitude Scaling.
    b. Time Scaling.
    c. Time Shifting.
2. If we shift a discrete-time impulse function by 0.6 then
    a. Discrete time impulse function will be scaled by a value of 0.6.
    b. Discrete time impulse function will not shift for any float value.
    c. Discrete time impulse function will shift with 0.6 value in time axis.

    Also cite the reason for the chosen answer.
3. Generate the signal x[n] = u[n+3]+5u[n-15]+4u[n+10] and record the waveforms in your record book.
4. Write a program in Matlab to find z-transform of signal x[n]=cos(w_0n).

5. Write a program in Matlab to generate a unit ramp sequence, unit parabolic sequence and a unit impulse sequence. Assume data accordingly.
6. Write a program in Matlab to ADD and SUBTRACT the signals used in "multiplication of two discrete signals".

9. Conclusion

In this experiment, various discrete signals are obtained and their properties are studied using MATLAB. After performing this experiment students will be able to tackle different signals in discrete time domain. At the end, students should try to learn as much as they can from this experiment and should try to find innovative ways of doing whatever is left in discrete systems (due to time constraint) on their own which would enhance their knowledge about use of MATLAB in discrete time domain.

Experiment No: 6

Basic Properties of Linear systems

1 Objective:

Basic Properties of Linear systems

2 Expected Outcomes of Experiment:

1. Understanding various properties of a system like linearity, shift invariance, stability, causality, memoryless, invertibilty.
2. Implementing and visualizing signal addition, signal subtraction, signal multiplication.

3 Theory:

A system is any physical device, process or computer algorithm that transforms input signals into output signals.In this experiment we shall look at following properties of system: Linearity, time Invariant, Stability, Causality, Memory, etc.

Systems are classified into the following categories:

- Linear and Non-linear Systems
- Time Variant and Time Invariant Systems
- Liner Time variant and Liner Time invariant systems
- Static and Dynamic Systems
- Causal and Non-causal Systems
- Invertible and Non-Invertible Systems
- Stable and Unstable Systems

(a) Linear and Non-linear Systems

A system is said to be linear when it satisfies superposition and homogenate principles. Consider two systems with inputs as $x_1(t)$, $x_2(t)$, and outputs as $y_1(t)$, $y_2(t)$ respectively. Then, according to the superposition and homogenate principles,

$$T [a_1 x_1(t) + a_2 x_2(t)] = a_1 T[x_1(t)] + a_2 T[x_2(t)]$$

$$\therefore T [a_1 x_1(t) + a_2 x_2(t)] = a_1 y_1(t) + a_2 y_2(t)$$

From the above expression, is clear that response of overall system is equal to response of individual system.

Example:

$(t) = x^2(t)$

Solution:

$y_1(t) = T[x_1(t)] = x_1^2(t)$

$y_2(t) = T[x_2(t)] = x_2^2(t)$

$T[a_1 x_1(t) + a_2 x_2(t)] = [a_1 x_1(t) + a_2 x_2(t)]^2$

Which is not equal to $a_1 y_1(t) + a_2 y_2(t)$. Hence the system is said to be non linear.

(b) Time Variant and Time Invariant Systems

A system is said to be time variant if its input and output characteristics vary with time. Otherwise, the system is considered as time invariant.

The condition for time invariant system is:

$y(n, t) = y(n-t)$

The condition for time variant system is:

$y(n, t) \neq y(n-t)$

Where $y(n, t) = T[x(n-t)]$ = input change

$y(n-t)$ = output change

Example:

$y(n) = x(-n)$

$y(n, t) = T[x(n-t)] = x(-n-t)$

$y(n-t) = x(-(n-t)) = x(-n + t)$

$\therefore\therefore$ $y(n, t) \neq y(n-t)$. Hence, the system is time variant.

(c) Linear Time variant (LTV) and Linear Time Invariant (LTI) Systems

If a system is both linear and time variant, then it is called linear time variant (LTV) system.

If a system is both linear and time Invariant then that system is called linear time invariant (LTI) system.

(d) Static and Dynamic Systems

Static system is memory-less where as dynamic system is a memory system.

Example 1: y(t) = 2 x(t)

For present value t=0, the system output is y(0) = 2x(0). Here, the output is only dependent upon present input. Hence the system is memory less or static.

Example 2: y(t) = 2 x(t) + 3 x(t-3)

For present value t=0, the system output is y(0) = 2x(0) + 3x(-3).

Here x(-3) is past value for the present input for which the system requires memory to get this output. Hence, the system is a dynamic system.

(e) Causal and Non-Causal Systems

A system is said to be causal if its output depends upon present and past inputs, and does not depend upon future input.

For non causal system, the output depends upon future inputs also.

Example 1: y(n) = 2 x(t) + 3 x(t-3)

For present value t=1, the system output is y(1) = 2x(1) + 3x(-2).

Here, the system output only depends upon present and past inputs. Hence, the system is causal.

Example 2: y(n) = 2 x(t) + 3 x(t-3) + 6x(t + 3)

For present value t=1, the system output is y(1) = 2x(1) + 3x(-2) + 6x(4) Here, the system output depends upon future input. Hence the system is non-causal system.

(f) Invertible and Non-Invertible systems

A system is said to invertible if the input of the system appears at the output.

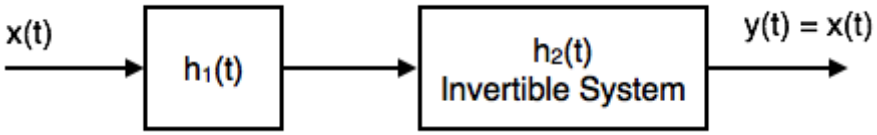


Fig. 6.1 Invertible and Non-Invertible system

Y(S) = X(S) H1(S) H2(S)

= X(S) H1(S) · 1(H1(S))1(H1(S)) *Since H2(S) = 1/(H1(S))*

$\therefore Y(S) = X(S)$

$y(t) = x(t)$

Hence, the system is invertible.

If $y(t) \neq x(t)$, then the system is said to be non-invertible.

(g) Stable and Unstable Systems

The system is said to be stable only when the output is bounded for bounded input. For a bounded input, if the output is unbounded in the system then it is said to be unstable.

Note: For a bounded signal, amplitude is finite.

Example 1: $y(t) = x^2(t)$

Let the input is u(t) (unit step bounded input) then the output y(t) = u2(t) = u(t) = bounded output.

Hence, the system is stable.

Example 2: $y(t) = \int x(t)dt \int x(t)dt$

Let the input is u (t) (unit step bounded input) then the output $y(t) = \int u(t)dt \int u(t)dt$= ramp signal (unbounded because amplitude of ramp is not finite it goes to infinite when $t \rightarrow\rightarrow$ infinite).

Hence, the system is unstable.

4 Equipment Required:

A PC installed with MATLAB software

5 Procedure:

- Open Matlab in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.
- Run the script.
- Desired result will be displayed on the command window.

6 Coding:

(a) MATLAB Coding for checking Linearity Property of System:

```
n=0:40;
a=2;
b=-3;
A=5;
B=6;
x1=cos(2*pi*.1*n);
x2=cos(2*pi*.5*n);
y=A.*(a.*x1+b.*x2)+B;
subplot(3,1,1);
stem(n,y);
yt=A.*(a.*x1+b.*x2)+a.*B+b.*B;
subplot(3,1,2);
stem(n,yt);
d=y-yt;
subplot(3,1,3);
stem(n,d);
axis([0 40 -10 20]);
```

(b) MATLAB Coding for checking causality Property of System: Write a MATLAB program to find whether the given system is causal or not : y(n) = x(n)– 0.9y(n-1)

```
n= -10:1:10
b = [1 0];
a = [1 0.9]
x = [zeros(1,10) 1 zeros(1,10)]
y1 = filter (a, b, x)
subplot (2, 1, 1)
stem (n,y1)
xlabel (' Samples ')
ylabel (' Amplitude ')
```

(c) MATLAB Coding for checking stability Property of System: Write a MATLAB program to find whether the given system is stable or not : y(n) = x(n)– 0.9y(n-1)

```
T = abs (y1);
t = sum (T);
if (t < 1000)
   disp (' Stable ')
else
   disp (' Unstable ')
end
```

7 Result:

(a) Linearity Property of System:

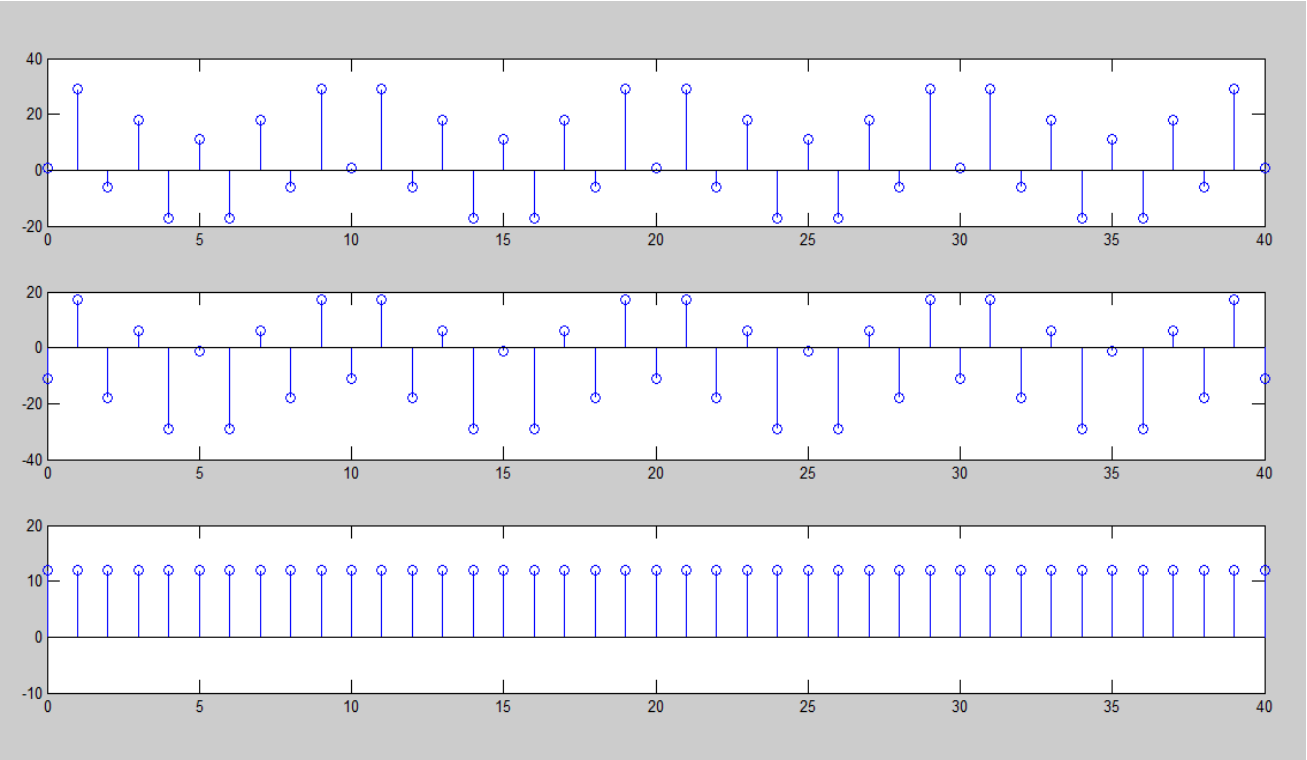


Fig 6.2 Linearity Property of System

(b) Causality Property of System:

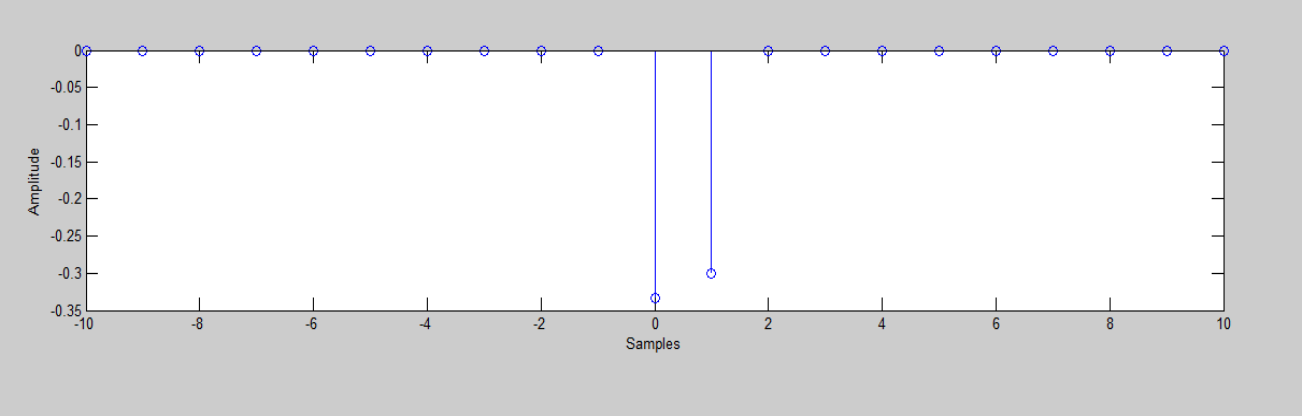


Fig 6.3Causality Property of System

(c) Stability Property of System:

Stable

8 Assignment:

1. Check whether y(t) = tx(t) and y(n) = {x(t)}^2 are linear or non linear.
2. For y(t) = 3x(t+1) + 5 check following properties of the system:
   (a) linearity (b) time invariant (c) causality
   (d) memoryless (e) stability
3. Write a MATLAB program to find whether the given systemis linear, stable and causal
   y(n) = exp x(n)

9 Conclusion:

In this experiment, the motive was to introduce various properties of system like linearity, shift invariance, stability, causality and visualizing signal addition, signal subtraction, signal multiplication.

Experiment No.- 07

Study of Impulse response of Linear system

CONTENTS **PAGE NO.**

1. Objective:

Study of impulse response of linear system.

2. Expected outcomes of Experiment:

1. Understanding impulse response of a linear system.
2. Implementing and visualizing response of linear system to a impulse input.

3. Theory:

3.1 Impulse Function:

It is equal to zero everywhere except for zero and whose integral over the entire real line is equal to one. It was introduced by Paul Dirac. Also known as Dirac delta function.

$$\delta(t) = 0 \quad \text{for} \quad t \neq 0$$

$$\int_{-\infty}^{\infty} \delta(t)\ dt = 1$$

Properties of impulse function:

A. Integral of impulse function:

$$\int_{a}^{b} \delta(t)dt = \begin{cases} 1, & a < 0 < b \\ 0, & \text{otherwise} \end{cases}$$

If the integral includes the origin (where the impulse lies), the integral is one. If it doesn't include the origin, the integral is zero.

B. Sifting/Sampling of impulse function:

$$\int_{a}^{b} \delta(t - T)f(t)dt = \begin{cases} f(T), & a < T < b \\ 0, & \text{otherwise} \end{cases}$$

This is called the "sifting" property because the impulse function δ(t-T) sifts through the function f(t) and pulls out the value f(T).

C. Convolution of impulse function:

$$f(t)\delta(t - T) = f(t - T)$$

Convolution of a function with a shifted impulse yields a shifted version of that function.

3.2 Impulse response:

Impulse response of a dynamic system is its output when given brief input signal i.e. Impulse.

If H(s) be the transfer function of any system then inverse Laplace Transform of H(s) i.e. h(t) is Impulse response.

$$h(t) = Z^{-1}[H(s)]$$

In MATLAB:**impulse(sys)**

impulse calculates the unit impulse response of a dynamic system model. For continuous-time dynamic systems, the impulse response is the response to a Dirac input δ(t). For discrete-time systems, the impulse response is the response to a unit area pulse of length Ts and height 1/Ts, where Ts is the sample time of the system. (This pulse approaches δ(t) as Ts approaches zero.) For state-space models, impulse assumes initial state values are zero.

3.3 Linear System:

A system which obeys the Superposition principle; output is proportional to the input. Such systems comprise of linear devices and govern by linear differential equations.

Superposition principle consists of homogeneity and additive.

Homogeneity:

$$F(ax) = aF(x)$$

Where, **a**is a scalar.

Additivity:

$$F(x_1 + x_2) = F(x_1) + F(x_2)$$

Example- Electrical circuits composed of resistor, inductor & capacitor.

4. **Equipment Required:**

- A pc installed with MATLAB software.

5. **Procedure:**

- Open MATLAB.
- Go to Simulink.
- Set parameter of pulse generator.
- Insert values into state space matrices.
- Run.

6. **Program:**

(A)

```
A=[-0.5572 -0.7814;0.7814  0]
B=[1 0]'
C=[1.9691  6.4493]
D=[0]
sys=ss(A,B,C,D)
impulse(sys)
```

Impulse response is shown in fig. 7.1

(B)

```
A=[-3   2; 5   1]
B=[0 1]'
C=[13]
D=[0]
sys=ss(A,B,C,D)
impulse(sys)
```

Impulse response is shown in fig. 7.2

7. Results:

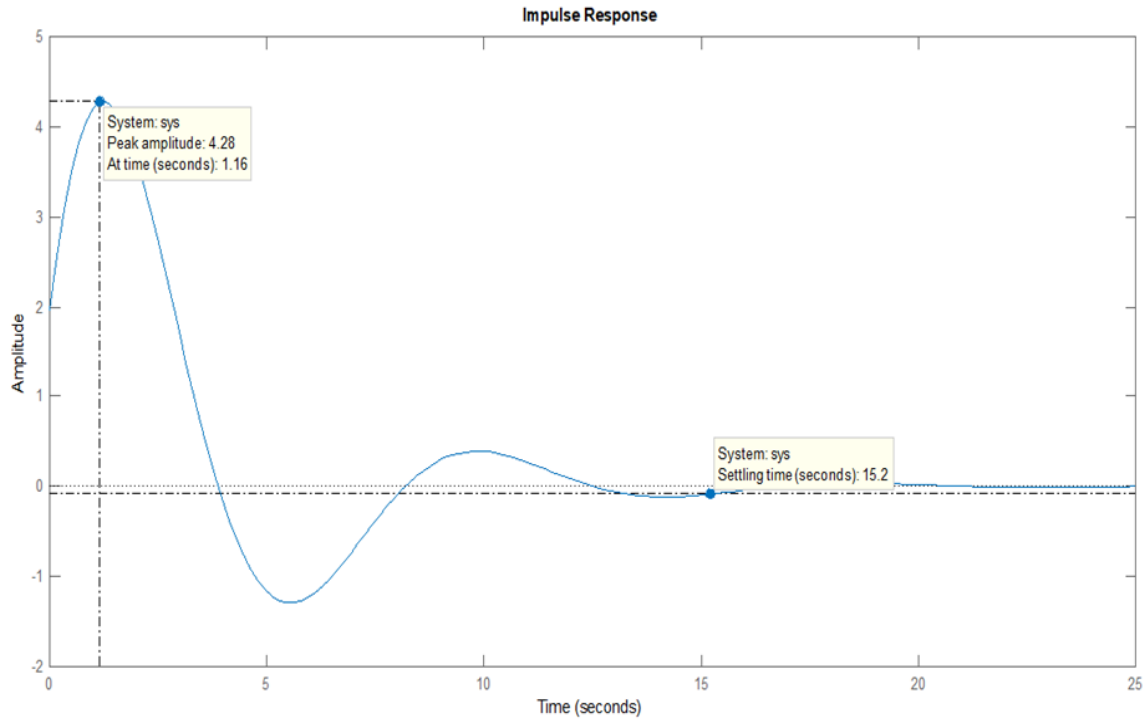


Fig. 7.1 Impulse response

From the fig. 7.1; system is stable.

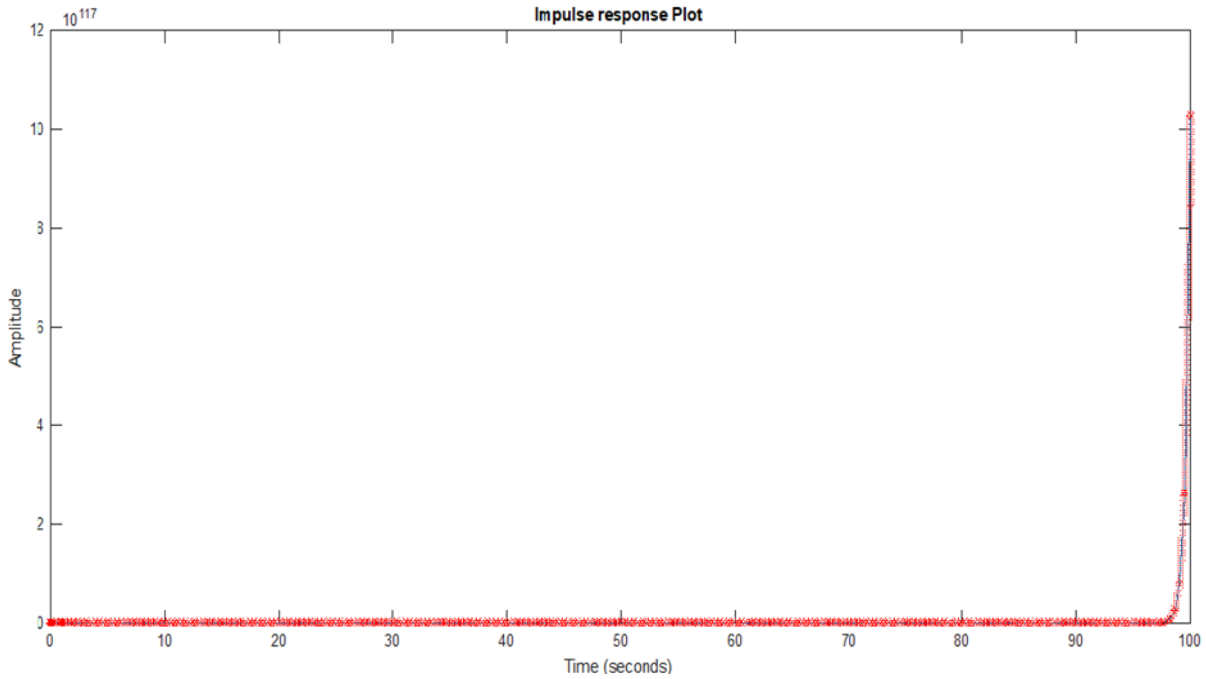


Fig. 7.2 Impulse response

From the impulse response in fig. 7.2; the given system is unstable.

8. Assignment:

1. Find impulse response of the system $\mathbf{Y(t)=e^{-t}sin5t,}$ comment on its stability.Also, find the Peak amplitude and settling time.
2. Find impulse response of the system $\mathbf{Y(t)=e^{2t}cos4t,}$ comment on the stability.Also, find the Peak amplitude and settling time.

9. Conclusion:

In this experiment, impulse response was introduced for a linear system. It helped in examining the stability of any system. Impulse response is meant to evaluate the response of the system for all frequency elements with the same magnitude. The only signal which contains all single-frequency elements with unit magnitude is Impulse (if you take the Laplace transform of impulse, it is 1 which means all frequencies have same contribution). So, by having the impulse response of a system, we have the overall behavior of that system.

At the end, student should try to learn as much as they can from this experiment and should try to find out impulse response of different system.

Experiment No: 8

Analysis of MIMO SYSTEM (2-ports)

Contents Page No.

1) Objective:

Analysis of MIMO SYSTEM (2-ports)

2) Expected Outcomes of Experiment:

1. Studying and understanding various types of systems such as SISO, SIMO, MISO, MIMO.
2. Analyzing a MIMO system using MATLAB.

3) Theory:

The communication links can be classified depending on the number of antennas used to transmit and to receive signal. The different schemes may be available in different scenarios, depending on the application they are used for.

3.1 Single Input Single Output (SISO)

SISO refers to the familiar wireless configuration with a single antenna both at the transmitter and receiver. It is less complex and easier to make. Assume we have an antenna, which transmits a signal S at a frequency *f* [1].

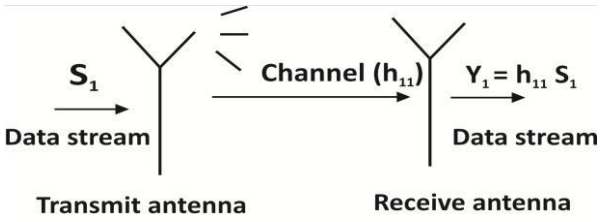


Fig 8.1: Single Input Single Output (SISO)

As the signal propagates through an environment, the signal is faded, which is modeled as a multiplicative coefficient h. The received signal y will be hs. Single Input Single Output (SISO) is illustrated in fig. 1.

The Channel capacity is poor as compare to other Technique but System design is not Complex. Theoretically, the 1Gbps barrier can be achieved using this configuration if we are allowed to use much power and as much BW as possible. Thus channel capacity and system performance is fully dependent on system design and more number of antennae. System Model of SIMO, MISO and MIMO is given below.

3.2 Single Input Multi Output (SIMO)

SIMO refers to single antenna at the transmitter and Multiple at receiver side. Single Input Multi Output (SIMO) system is shown in fig. 8.2.

For two receiving antennas, there will be two received signals y1 and y2 with different fading coefficients h1 and h2. The effect upon the signal s for a given path (from a transmit antenna to a receive antenna) is called a channel.

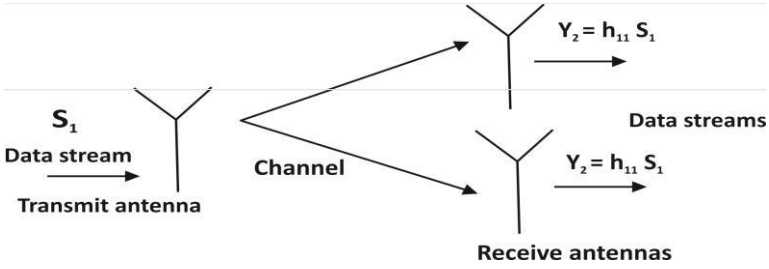


Fig 8.2: Single Input Multi Output (SIMO)

The channel capacity has not increased. The multiple receive antennas can help us get a stronger signal through diversity.

3.3 Multi Input Single Outputs (MISO)

SIMO refers to multiple antennas at the transmitter and a single antenna at receiver side, as in fig. 8.3. Assume 2 transmitting antennas and 1 receive antenna. There will be one received signal. In order to separate s1 and s2 we will need to also transmit, at a different time.

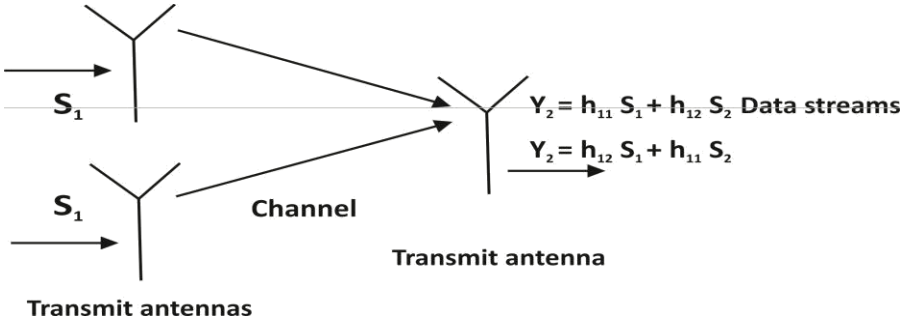


Fig 8.3: Multi Input Single Outputs (MISO)

The channel capacity has not really increased because we still have to transmit two signals at time 2.

3.4 Multi-Input Multi-Output (MIMO)

Multi-Input Multi-Output (MIMO) is the combination of both Single Input Multi Output (SIMO) and Multi Input Single Output (MISO) system. Multiple antennas can be used at the transmitter and receiver, an arrangement called a multiple-input multiple-output (MIMO) system. The proper operation of MIMO systems requires careful design, with the encoded signals received from each transmitting antenna and the decoded signal at the receiver side. The better combination of number of transmitting and receiving antenna for MIMO systems in BPSK modulation technique that satisfy the good signal-to-noise ratio (SNR). Bit error rate (BER) is inversely proportional to the SNR values of the system. The main arguments today, for using multiple antennas when transmitting over a wireless link are: Array gain, Interference suppression, Transmitter localization, Bit rate and Data rate, spatial diversity, Reliability, Complexity.

3.4.1 MIMO Antenna Configuration

MIMO antenna configuration describes that use of multiple transmit and multiple receive antennas for a single user produces higher spectral efficiency and more data rates, as shown in fig. 5. Spatial multiplexing technique, different data streams are transmitted from the different antenna elements, Interference can be reduced easily in the wireless system.

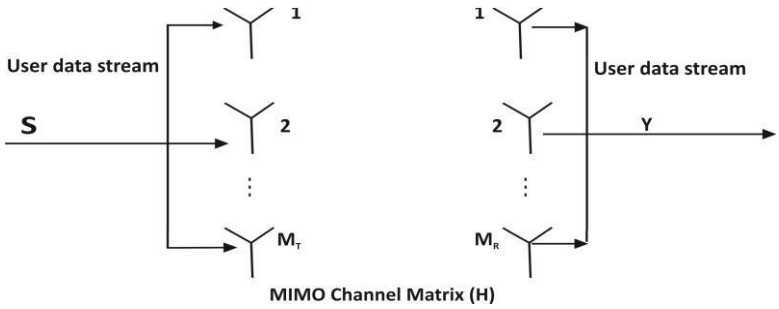


Fig 8.4: MIMO Antenna Configuration

Here we take MT transmit and MR receive antennas with input data stream is S and output data stream is Y.

3.4.2 MIMO System Model

MIMO systems include considerations around signal preconditioning, as well as channel predictions, as illustrated in fig 8.5.

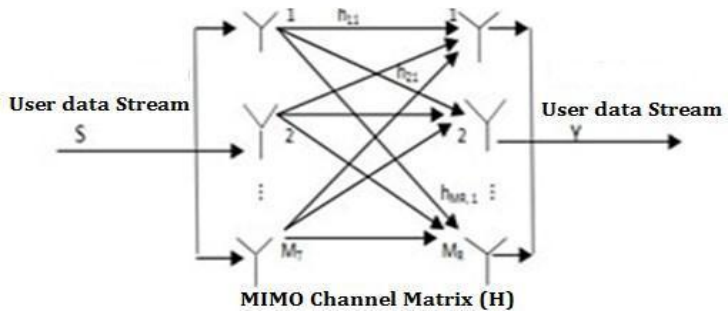
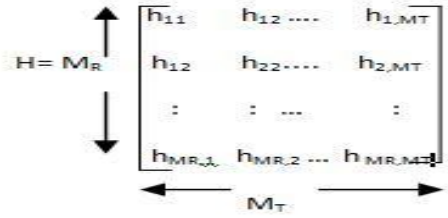


Fig 8.5: MIMO antennas system model

The MIMO channel can be represented using a MR× MT matrix format H is given by,



Where hij is a complex Gaussian random variable that models fading gain between the ith transmit and jth receive antenna.

3.4.3 Frequency Response Properties of MIMO Systems:

Consider again the block diagram of a MIMO feedback system, where the plant has p inputs and q outputs.

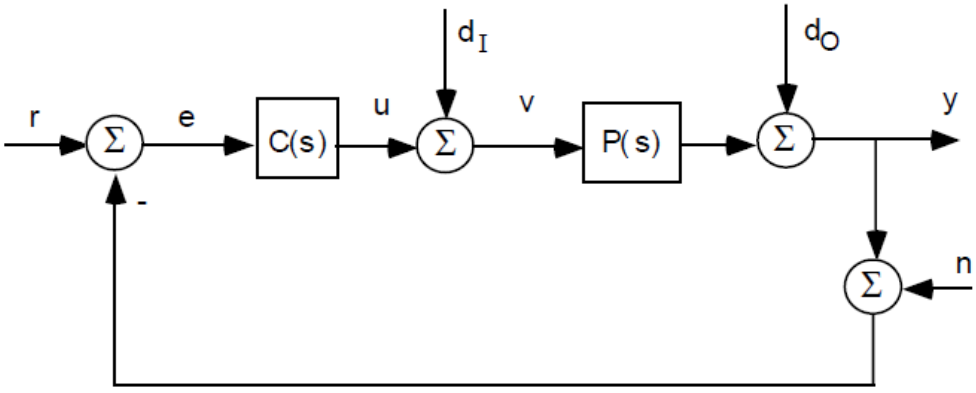


Fig 8.6: Block Diagram of a MIMO feedback system

Note that P(s) is a q× p matrix and C(s) is a p × q matrix.

Recall that the two sets of transfer functions needed to describe the behavior of a MIMO feedback system are given by:

Input and Output Open Loop Transfer functions:

$L_I(s) = C(s)\,P(s)$, $L_O(s) = P(s)\,C(s)$

Input and Output Sensitivity Functions:

$S_I(s) = (I + L_I(s))^{-1}$, $S_O(s) = (I + L_O(s))^{-1}$

Input and Output Complementary Sensitivity Functions:

$T_I(s) = L_I(s)\,(I + L_I(s))^{-1}$, $T_O(s) = L_O(s)\,(I + L_O(s))^{-1}$

Note that

$S_O(s) + T_O(s) = I_{q\times q}$, $S_I(s) + T_I(s) = I_{p\times p}$

4) Equipment Required:

A PC installed with MATLAB software

5) Procedure:

- Open Matlab in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.
- Run the script.
- Desired result will be displayed on the command window.

6) Coding:

a) Calculate power gains for a sample 2-port network:

```
s11 = 0.61*exp(i*165/180*pi);
s21 = 3.72*exp(i*59/180*pi);
s12 = 0.05*exp(i*42/180*pi);
s22 = 0.45*exp(i*(-48/180)*pi);
sparam = [s11 s12; s21 s22];
z0 = 50;
```

```
zs = 10 + i*20;
zl = 30 - i*4;
Gt = powergain(sparam,z0,zs,zl,'Gt') % Calculate the transducer power gain of the network
Ga = powergain(sparam,z0,zs,'Ga') % Calculate the available power gain of the network
Gp = powergain(sparam,z0,zl,'Gp') % Calculate the operating power gain of the network
Gmag = powergain(sparam,'Gmag') % Calculate the maximum available power gain of the network
Gmsg = powergain(sparam,'Gmsg') % Calculate the maximum stable power gain of the network
```

7) Results:

Calculate power gains for a sample 2-port network given:

Transducer power gain of the network: Gt = 7.4654

Available power gain of the network: Ga = 11.4361

Operating power gain of the network: Gp = 16.2358

Maximum available power gain of the network: Gmag = 41.5032

Maximum stable power gain of the network Gmsg = 74.4000

8) Conclusion:

In this experiment, various types of systems like SISO, SIMO, MISO, MIMO are studied using single to multiple antennas. The capacity of a MIMO system (MT, MR) = (3, 3) is approximately three times the capacity of a (1, 1) SISO system. The SISO system will give 1 to 7 bps/Hz and MIMO will give 3 to 48 bps/Hz data.

EXPERIMENT NO.:09

Study of Realization Theorem and Filters.

Contents **Page No.**

1. Objective:

Study of Realization Theorem and Filters

2. Expected Outcomes of Experiment:

1. Studying and understanding different types of filters.
2. Computing and visualizing behavior of frequency response of filters

3. Theory:

3.1 Introduction:

Filters are a basic component of all signal processing and telecommunication system. A filtering a device or process that removes some unwanted components or features from a signal. Filtering is a class of signal processing used to complete or partial suppression of some aspects of the signal. Most often, this means to remove some frequencies and not others in order to suppress the interfacing signals and reduce the background noise.

3.2 Types of Filter:

In signal processing, the filters can be of four types, i.e. Lowpass Filters, Highpass Filter,Bandpass Filters, and Bandstop Filters.

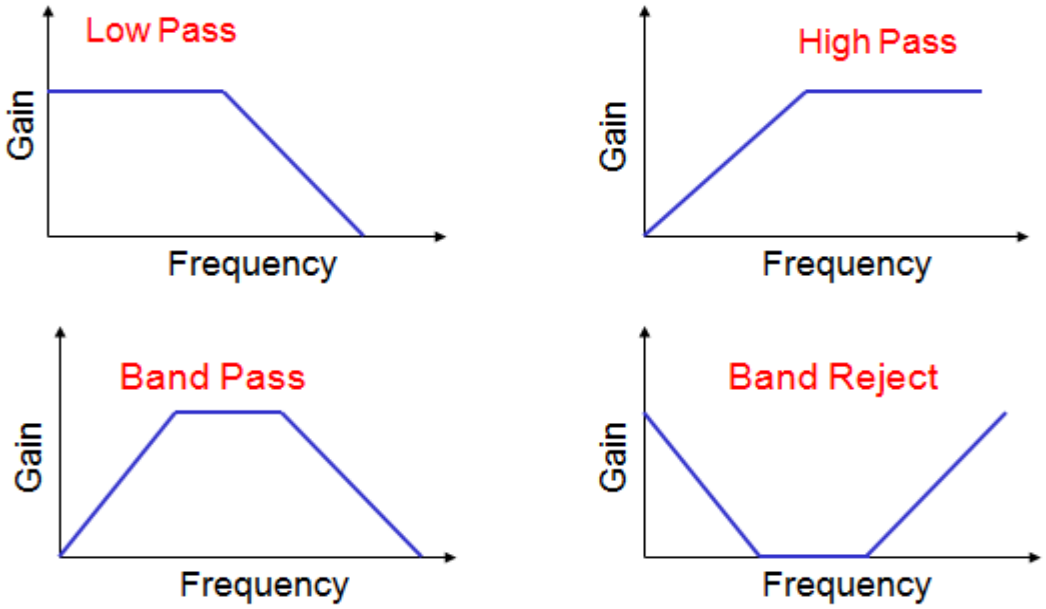
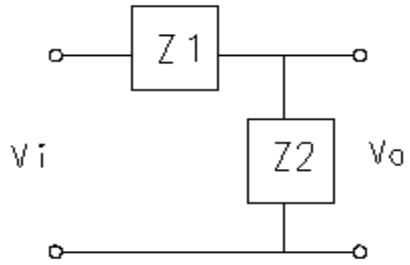


Fig 9.1 Characteristics of filter

3.2.1 Low pass Filter:

To see how complex impedances are used in practice consider the simple case of a voltage divider.



$$\frac{V_o}{V_i} = \frac{Z_2}{Z_1 + Z_2}$$

If Z_1 is a resistor and Z_2 is a capacitor then

$$\frac{V_o}{V_i} = \frac{\frac{1}{sC}}{\frac{1}{sC} + R} = \frac{1}{1 + sCR}$$

Generally we will be interested only in the magnitude of the response:

$$\left|\frac{V_o}{V_i}\right| = \left|\frac{1}{1 + sCR}\right| = \left|\frac{1}{1 + j\omega CR}\right| = \frac{1}{\sqrt{1^2 + (\omega CR)^2}}$$

Recall that the magnitude of a complex number is the square root of the sum of the squares of the real and imaginary parts. There are also phase shifts associated with the transfer function (or gain, V_o/V_i), thought we will generally ignore these.

This is obviously a low pass filter (i.e., low frequency signals are passed and high frequency signals are blocked).If ω<<1/RC then ωCR<<1 and the magnitude of the gain is approximately unity, and the output equals the input. If ω>>1/RC (ωCR>>1) then the gain goes to zero, as does the output. At ω=1/RC, called the break frequency (or cut-off frequency, or 3dB frequency, or half-power frequency, or bandwidth), the magnitude of the gain is 1/sqrt(2)0.71. In this case (and all first order RC circuits) high frequency is defined as ω>>1/RC; the capacitor acts as a short circuit and all the voltage is across the resistance. At low frequencies, ω<<1/RC, the capacitor acts as an open circuit and there is no current (so the voltage across the resistor is near zero).

If Z_1 is an inductor and Z_2 is a resistor another low pass structure results with a break frequency of R/L

3.2.2 High-Pass Circuit

If Z_1 is a capacitor and Z_2 is a resistor we can repeat the calculation:

$$\frac{V_o}{V_i} = \frac{R}{\frac{1}{sC} + R} = \frac{sCR}{1 + sCR}$$

and

$$\left|\frac{V_o}{V_i}\right| = \left|\frac{sCR}{1 + sCR}\right| = \left|\frac{j\omega CR}{1 + j\omega CR}\right| = \frac{\omega CR}{\sqrt{1^2 + (\omega CR)^2}}$$

At high frequencies, $\omega >> 1/RC$, the capacitor acts as a short and the gain is 1 (the signal is passed). At low frequencies, $\omega << 1/RC$, the capacitor is an open and the output is zero (the signal is blocked). This is obviously a high pass structure and you can show that the break frequency is again 1/RC.

If Z_1 is a resistor and Z_2 is an inductor the resulting circuit is high pass with a break frequency of R/L.

3.2.3 Band pass Filters:

For a second-order band-pass filter the transfer function is given by

$$H(s) = \frac{V_o}{V_i} = \frac{H_o \beta s}{s^2 + \beta s + \omega_o^2}$$

where ω_o is the centre frequency, β is the bandwidth and H_o is the maximum amplitude of the filter. These quantities are shown on the diagram below. The quantities in parentheses are in radian frequencies, the other quantities are in Hertz (i.e. $f_o = \omega_o/2\pi$, $B = \beta/2\pi$).

4. Equipment Required:

A PC installed with MATLAB software

5. Procedure:

- Open MATLAB in your PC.
- Open a new script.
- Write the code as mentioned in the next heading in the script.
- Save the script.

- Run the script.
- Desired results will be displayed on the command window.

6 CODING:

6.1To create a simple Low Pass Filter in the mfile, with a Cut-off frequency of 3Hz and sampling frequency of 100Hz.

```
% Low pass Filter
d = fdesign.lowpass('Fp,Fst,Ap,Ast',3,5,0.5,40,100);
Hd = design(d,'equiripple');

fvtool(Hd)
```

6.2To create a simple High Pass Filter in the mfile, with a Cut-off frequency of 0.5 Hz and sampling frequency of 48 Hz.

```
% High pass Filter

d=fdesign.highpass(10,12,80,0.5,48);

Hd = design(d,'equiripple');

fvtool(Hd)
```

6.3 To create a simple Band Pass Filter in the mfile, with a Cut-off frequency of 0.5 Hz and sampling frequency of 48 Hz.

```
% Band pass Filter

d = fdesign.bandpass(10, 12, 14, 16, 80, .5, 60, 48)

Hd = design(d,'equiripple');

fvtool(Hd)
```

7. Results

7.1 Low pass Filter

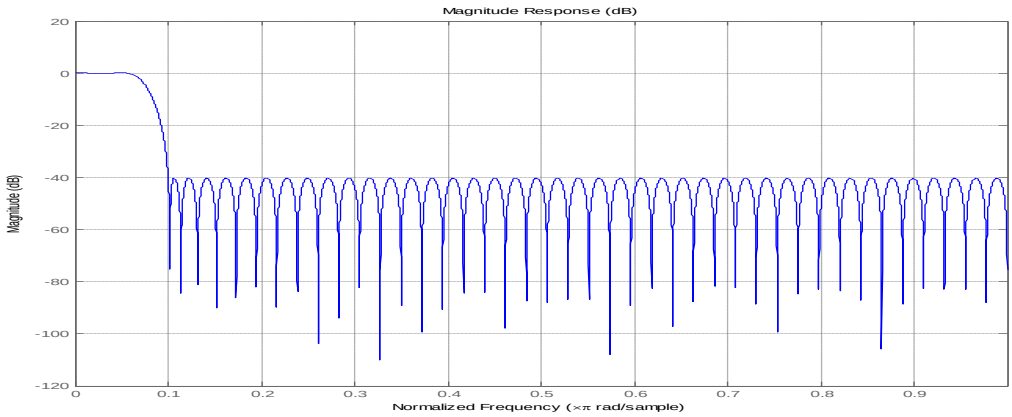


Fig 9.2 Low pass Filter

7.2 High Pass filter

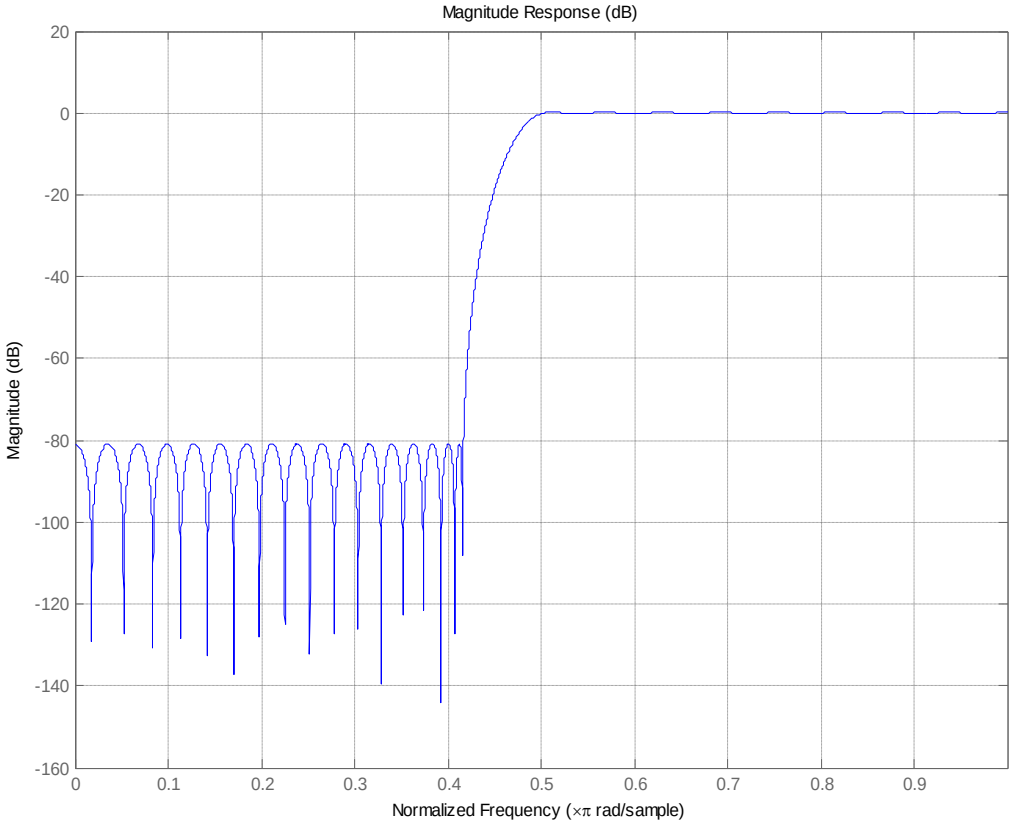


Fig 9.3 High Pass filter

7.3 Band Pass filter

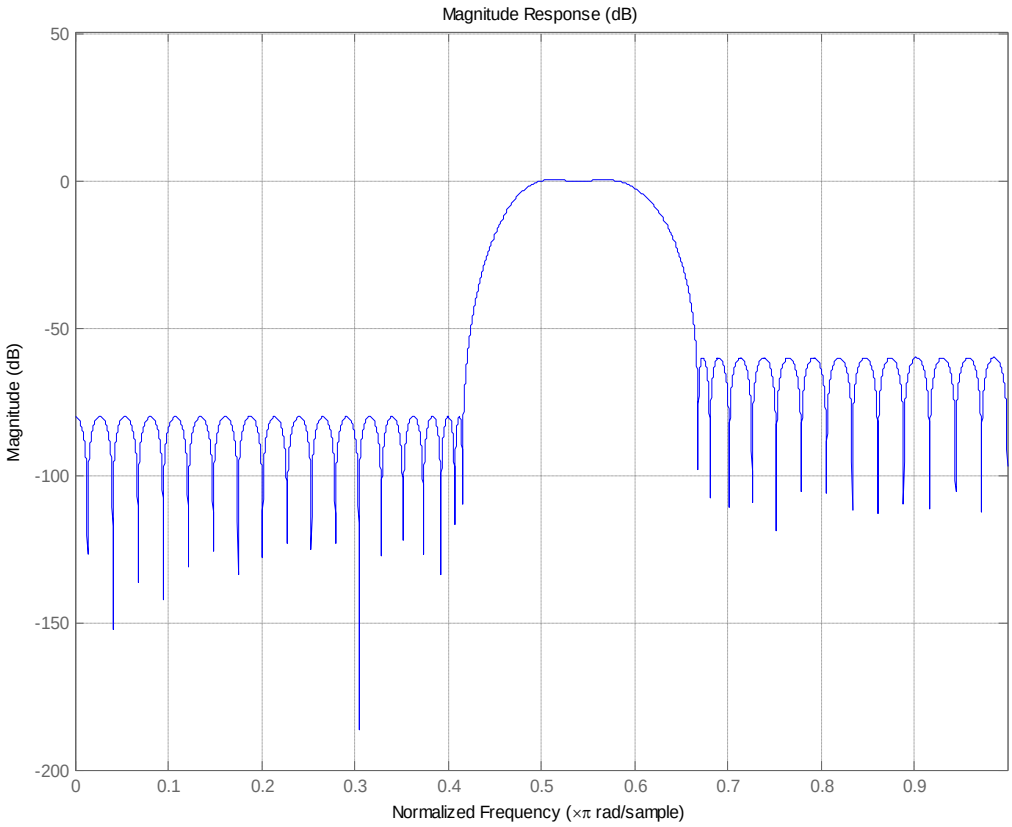


Fig 9.4 Band Pass filter

8. Assignment

1 Create a simple Low Pass Filter using MATLAB which has a Cut-off frequency of 5 Hz and sampling frequency of 10Hz.

2 Create a simple High Pass Filter using MATLAB which has a Cut-off frequency of 1 Hz and sampling frequency of 50 Hz.

9. Conclusion:

In this experiment, the motive was to introduce filters viz. high pass, low pass and their behavior (frequency response) is studied.

EXPERIMENT NO.:10

Simulation of systems using Op-amps/Software tools (MATLAB)

Contents **Page No.**

1. **Objective:**

Simulation of systems using Op-amps/Software tools

2. **Expected Outcomes of Experiment:**

1 Studying and understanding working of an OP-AMP.

2 Simulating the behavior of different types of OP-AMP.

3. **Theory:**

Introduction

Their applications were initially in the area of analog computation and instrumentation. Op amp is very popular because of its versatility.Op-amp circuits work at levels that are quite close to their predicted theoretical performance. The op-amp is treated a building block to study its terminal characteristics and its applications.

Ideal operational amplifier:

• Infinite input impedance.

• Infinite open loop gain A_{OL} for differential signal.

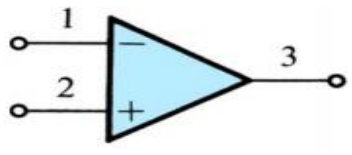
• Zero gain for the common - mode signal.

• Zero output impedance.

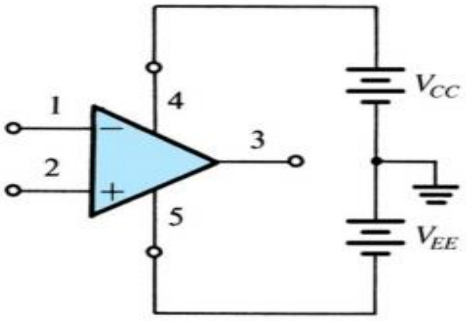
• Infinite bandwidth.

• Infinite bandwidth.

Circuit symbol for op amp

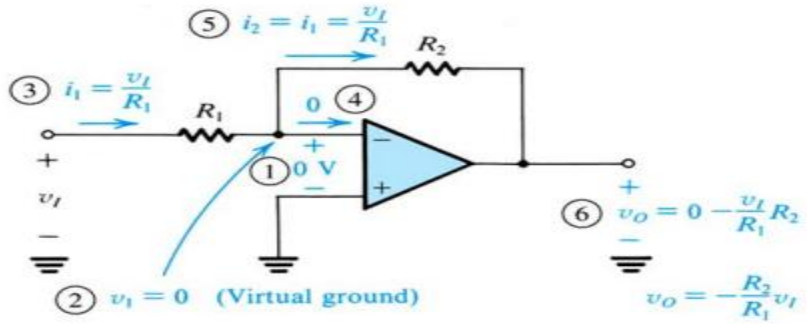


Op amp with dc power supplies

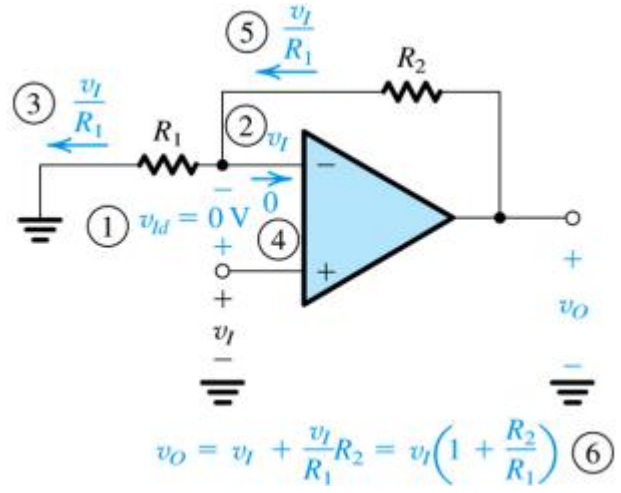


Common Op-Amp Circuits

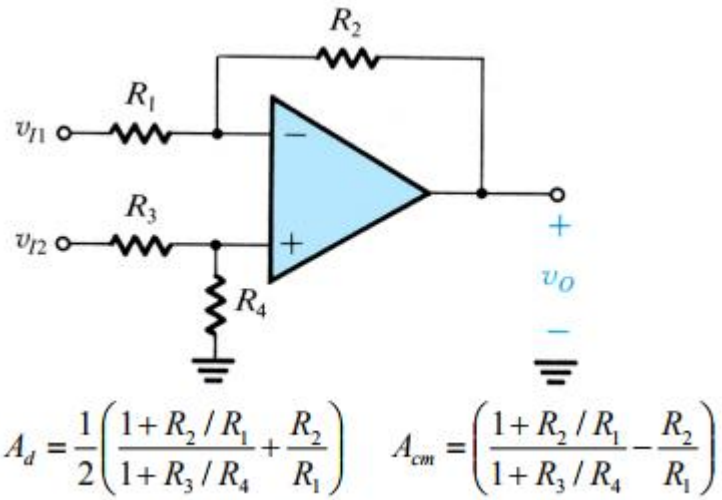
3.1 Inverting Op-Amp



3.2 Noninverting Op-Amp



3.3 Differential Input Op-Amp



4. Equipment's Required:

A PC installed with MATLAB software.

5. Procedure:

5.1 Open MATLAB in your PC.

5.2 Open a new script.

5.3 Write the code as mentioned in the next heading in the script.

5.4 Save the script.

5.5 Run the script.

5.6 Desired result will be displayed on the command window.

6. CODING:

6.1 The MATLAB program that can be used to find the poles, zero and plot the frequency response is as follows:

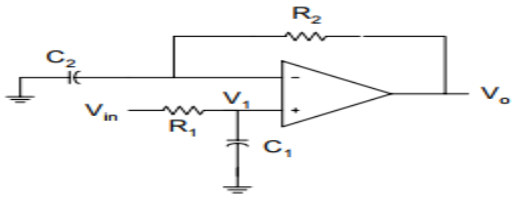


Figure 10.1 Inverter with Finite Open-loop Gain

```
% Poles and zeros, frequency response of above fig.
c1 = 1e-7; c2 = 1e-3; r1 = 10e3; r2 = 10;
```

```
% poles and zeros b1 = c2*r2;
a1 = c1*r1;
 num = [b1 1];
 den = [a1 1];
disp('the zero is') z = roots(num)
disp('the poles are') p = roots(den)
% the frequency response
 w = logspace(-2,6);
 h = freqs(num,den,w);
gain = 20*log10(abs(h));
 f = w/(2*pi);
 phase = angle(h)*180/pi;
subplot(211),semilogx(f,gain,'w');
 xlabel('Frequency, Hz')
 ylabel('Gain, dB')
 axis([1.0e-2,1.0e6,0,22])
 text(2.0e-2,15,'Magnitude Response')
subplot(212),
semilogx(f,phase,'w')
 xlabel('Frequency, Hz')
 ylabel('Phase')
axis([1.0e-2,1.0e6,0,75])
 text(2.0e-2,60,'Phase Response')
```

The magnitude and phase plots are shown in Figure 3

6.2 In Figure 2, R1 = 500 Ω, and R2 = 50 KΩ. Plot the closed-loop gain as the open-loop gain increases from 10^2 to 10^8 .

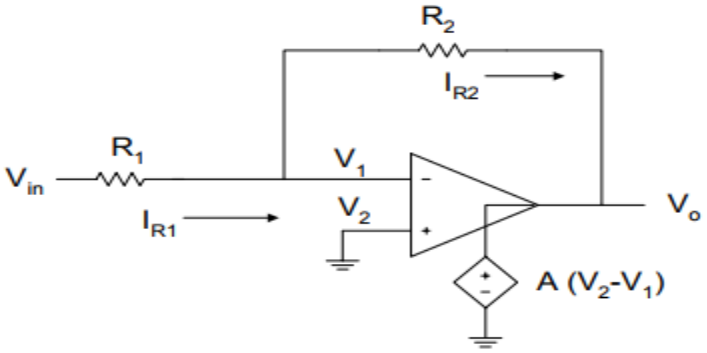


Figure 10.2 Inverter with Finite Open-loop Gain

```
% Effect of finite open-loop gain %
 a = logspace(2,8);
 r1 = 500; r2 = 50e3;
 r21 = r2/r1; g = [];
 n = length(a);
 for
i = 1:n
g(i) = r21/(1+(1+r21)/a(i));
 end
semilogx(a,g,'w')
 xlabel('Open loop gain')
ylabel('Closed loop gain')
```

title('Effect of Finite Open Loop Gain')

axis([1.0e2,1.0e8,40,110])

The characteristics of the closed-loop gain as a function of the open-loop gain is shown in fig.4

7 Results:

7.1 z = -100 ;

p = -1000

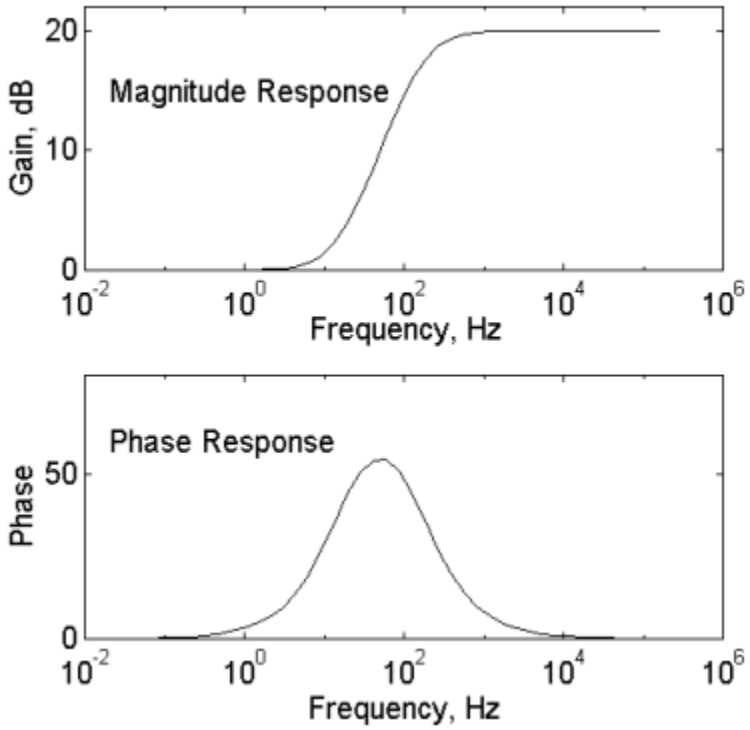


Figure 10.3 Frequency Response

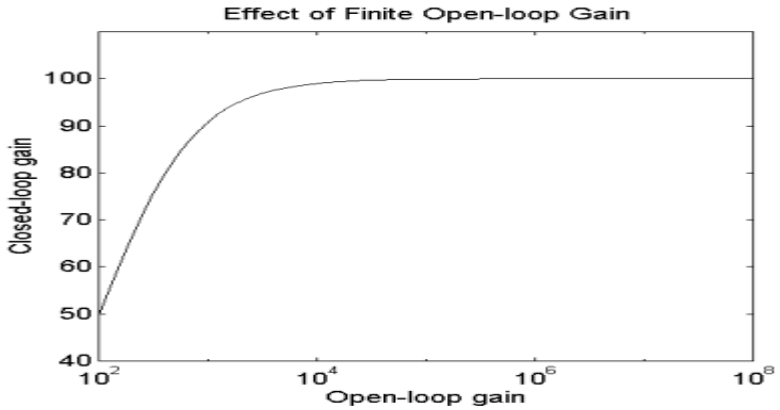


Figure 10.4 Closed-Loop Gain versus Open-Loop Gain

8. Assignments:

1 Design a non-inverting amplifier with an appropriate closed-loop gain of 150 and a minimum input impedance of 100MΩ.

2 Design an inverting amplifier using a 741 op-amp. The voltage gain must be 68 +5% and the input impedance must be approximately 10KΩ.

9. Conclusion:

In this experiment, the motive was to introduce OP-AMP and their behaviour (frequency response).

s communication with MIMO systems will give better results by using of antennae arrays, equalizer, diversity technique and OFDM technique on both sides of the communication link.

: