## EXPERIMENT NO. 1

Write a MATLAB program to generate Unit Impulse, Ramp, Unit Step and Exponential.

### AIM:

To generate Unit Impulse, Ramp, Unit Step and Exponential.

### Theory:

An **ideal impulse signal** is a signal that is zero everywhere but at the origin (t = 0), it is infinitely high. Although, the area of the impulse is finite. The unit impulse signal is the most widely used standard signal used in the analysis of signals and systems.

The continuous-time unit impulse signal is denoted by δ(t) and is defined as

$$\delta(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases}$$

The discrete-time unit impulse signal is denoted by δ[n] and is defined as

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

The **step signal or step function** is that type of standard signal which exists only for positive time and it is zero for negative time. In other words, a signal x(t) is said to be step signal if and only if it exists for $t \geq 0$ and zero for $t < 0$. The step signal is an important signal used for analysis of many systems.

The continuous-time unit step signal is denoted by u(t) and is defined as

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

The discrete-time unit step signal is denoted by u[n] and is defined as

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

A **ramp function** or **ramp signal** is a type of standard signal which starts at $t = 0$ and increases linearly with time. The unit ramp function has unit slope.

The continuous-time unit ramp signal is denoted by r(t) and is defined as

$$r(t) = \begin{cases} t & t \geq 0 \\ 0 & t < 0 \end{cases}$$

The discrete-time unit ramp signal is denoted by r[n] and is defined as

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

An **exponential signal** or **exponential function** is a function that literally represents an exponentially increasing or decreasing series.

A continuous time real exponential signal is defined as

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

A discrete-time real exponential sequence is defined as

$x[n] = a^n$ for all n

**Relationship between Unit Impulse Signal and Unit Step Signal**

The time integral of unit impulse signal is a unit step signal. In other words, the time derivative of a unit step signal is a unit impulse signal, i.e.,

$\int_{-\infty}^{\infty} \delta(t)dt = u(t)$ and $\delta(t) = \frac{d}{dt} u(t)$

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in figure window.

**Program:**

```
clc;
clear;
close all;
t=0:1:50;
t1=-50:1:50;
% generation of unit impulse
d=[zeros(1,50),1,zeros(1,50)];
% generation of ramp
r=0:50;
% generation of unit step
s=[zeros(1,50),ones(1,51)];
% generation of exponential
e=exp(0.1*t);

subplot(221);
plot(t1,d);
title('unit impulse');
```

```
xlabel('time');
ylabel('magnitude');
ylim([0,2]);
subplot(222);
plot(t,r);
title('ramp');
xlabel('time');
ylabel('magnitude');
subplot(223);
plot(t1,s);
title('unit step');
xlabel('time');
ylabel('magnitude');
ylim([0,2]);
subplot(224);
plot(t,e);
title('exponential');
xlabel('time');
ylabel('magnitude');
```

**Output:**

The plots are shown in Fig.1

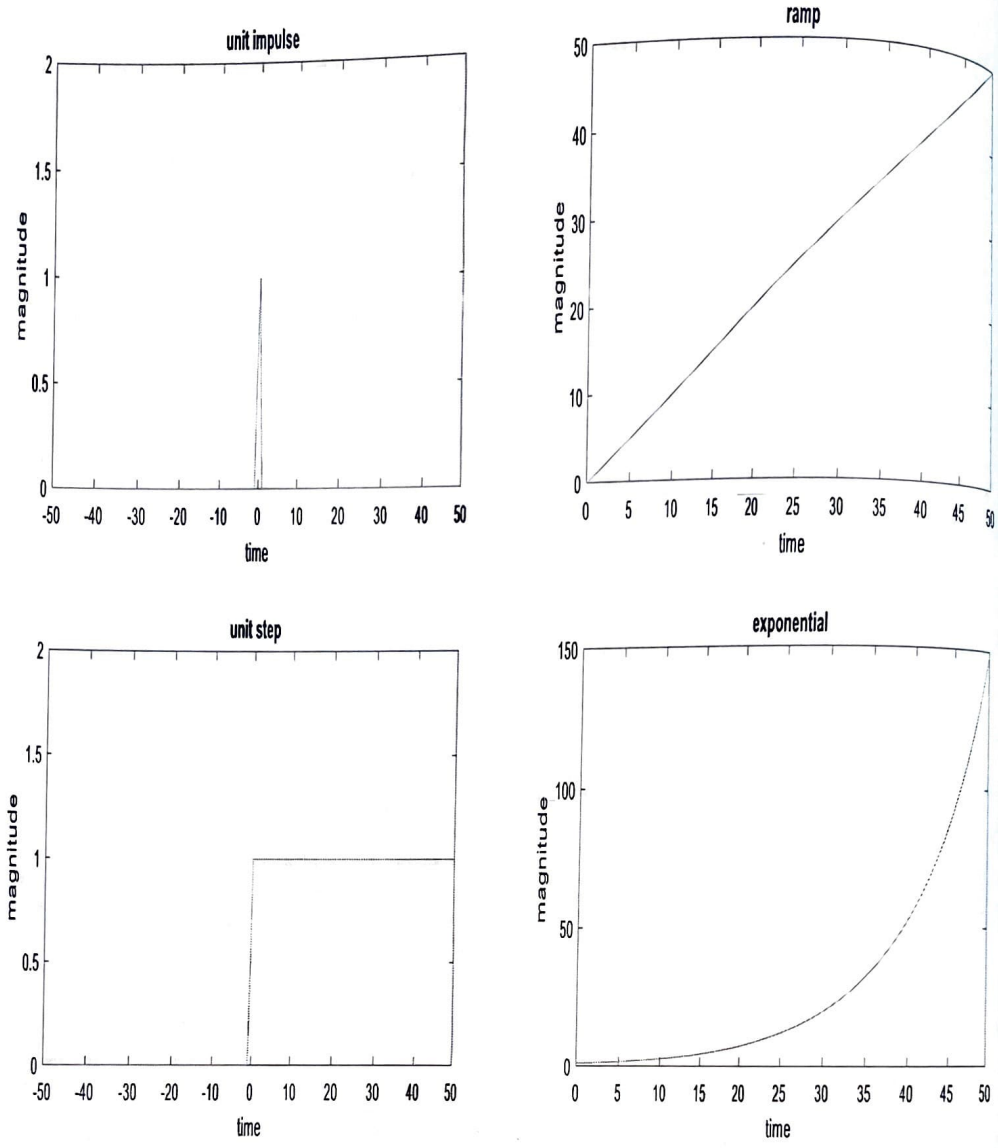


**Fig.1: Plots of Unit Impulse, Ramp, Unit Step and Exponential**

**Result:**

Unit Impulse, Ramp, Unit Step and Exponential signals are generated.

**EXPERIMENT NO. 2**

Write a MATLAB program to implement folding, time shifting, and time scaling of a given signal.

**AIM:**

To implement folding, time shifting, and time scaling of a given signal.

**Theory:**

The **time reversal or folding** of a signal is also called as the reflection of the signal about the time origin (or t = 0). Time reversal of a signal is a useful operation on signals in convolution.

If the independent variable t is replaced by '-t', this operation is known as time reversal of the signal about the y-axis, or amplitude axis. This can be achieved by taking the mirror image of the signal x(t) about y-axis 'or' by rotating x(t) by 180° about the y-axis.

**Time shifting** means, shifting of signals in the time domain. Mathematically, it can be written as. $x(t) \rightarrow y(t+k)$ This K value may be positive or it may be negative. According to the sign of k value, we have two types of shifting named as Right shifting and Left shifting.

The process of multiplying a constant to the time axis of a signal is known as **time scaling** of the signal. The time scaling of signal may be time compression or time expansion depending upon the value of the constant or scaling factor.

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in figure window.

**Program:**

```
clc;
clear;
close all;
t=0:10;
x=[0 1 2 1 0 1 2 0 1 2 1];
```

```
% time reversal (folding)
figure; subplot(211);
plot(t,x);
title('x(t)');
xlabel('Time');
ylabel('Amplitude');
axis([-10 10 0 2]);
y=fliplr(x);
t1=-10:0;
subplot(212);
plot(t1,y);
title('x(-t)');
xlabel('Time');
ylabel('Amplitude');
axis([-10 10 0 2]);

% time shifting
figure;
subplot(311);
plot(t,x);
title('x(t)');
xlabel('Time');
ylabel('Amplitude');
axis([-2 12 0 2]);
subplot(312);
plot(t+2,x);
title('x(t-2)');
xlabel('Time');
ylabel('Amplitude');
axis([-2 12 0 2]);
subplot(313);
plot(t-2,x);
title('x(t+2)');
```

```
xlabel('Time');
ylabel('Amplitude');
axis([-2 12 0 2]);

% time scaling
t=0:0.01:8*pi;
x=sin(t);
figure;
subplot(311);
plot(t,x);
title('x(t)');
xlim([0,max(t)]);
xlabel('Time');
ylabel('Amplitude');
y=sin(t/2);
subplot(312);
plot(t,y);
title('x(t/2)');
xlim([0,max(t)]);
xlabel('Time');
ylabel('Amplitude');
z=sin(t*2);
subplot(313);
plot(t,z);
title('x(2t)');
xlim([0,max(t)]);
xlabel('Time');
ylabel('Amplitude');
```

**Output:**

The plots are shown in Fig.2.1, Fig.2.2 and Fig.2.3.

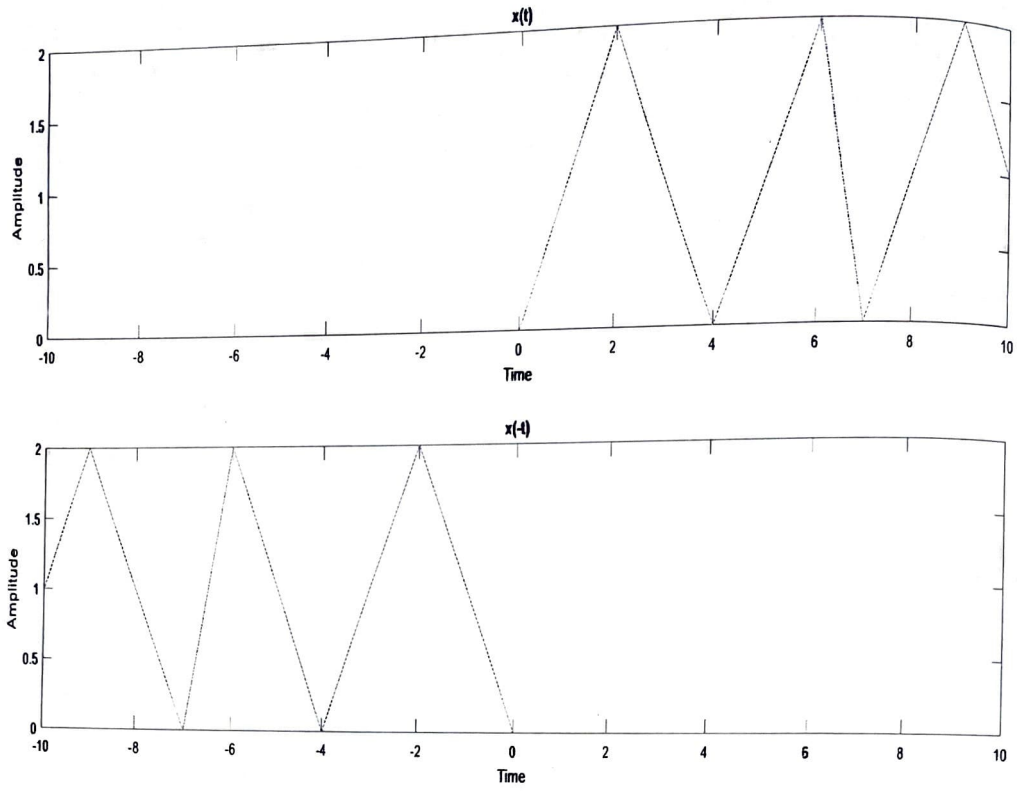


**Fig.2.1: Plot of Time Reversal**

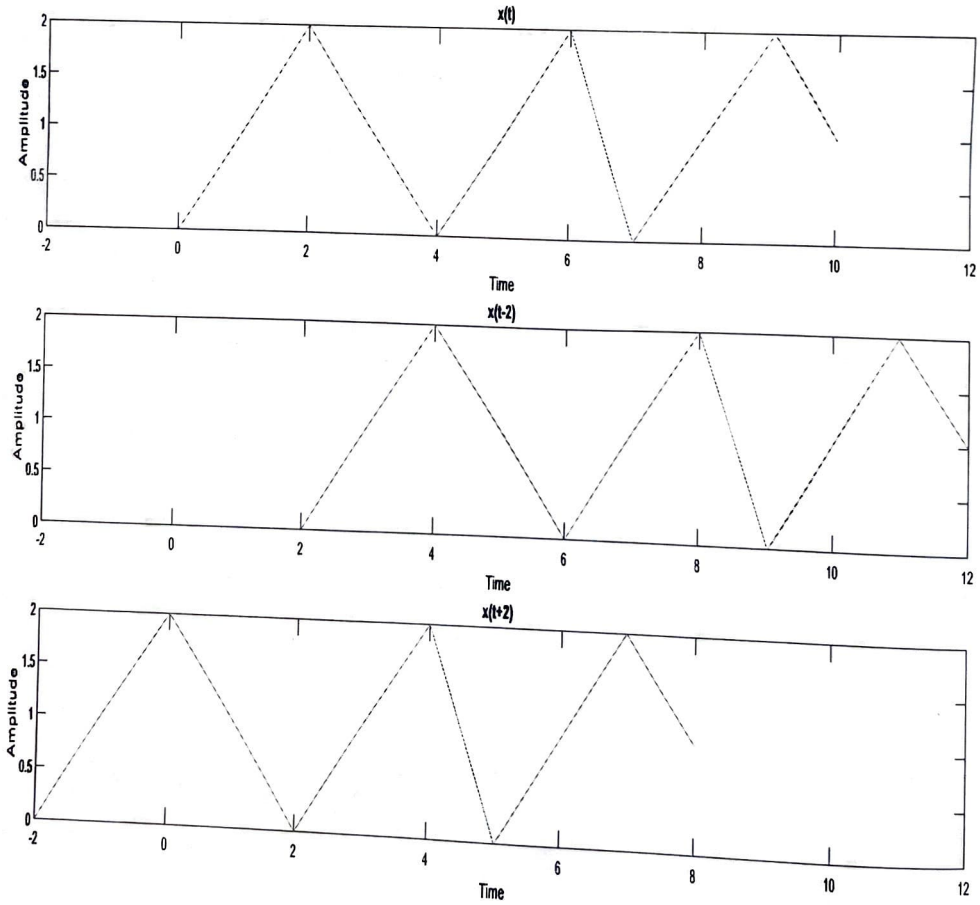


**Fig.2.2: Plot of Time Shifting**

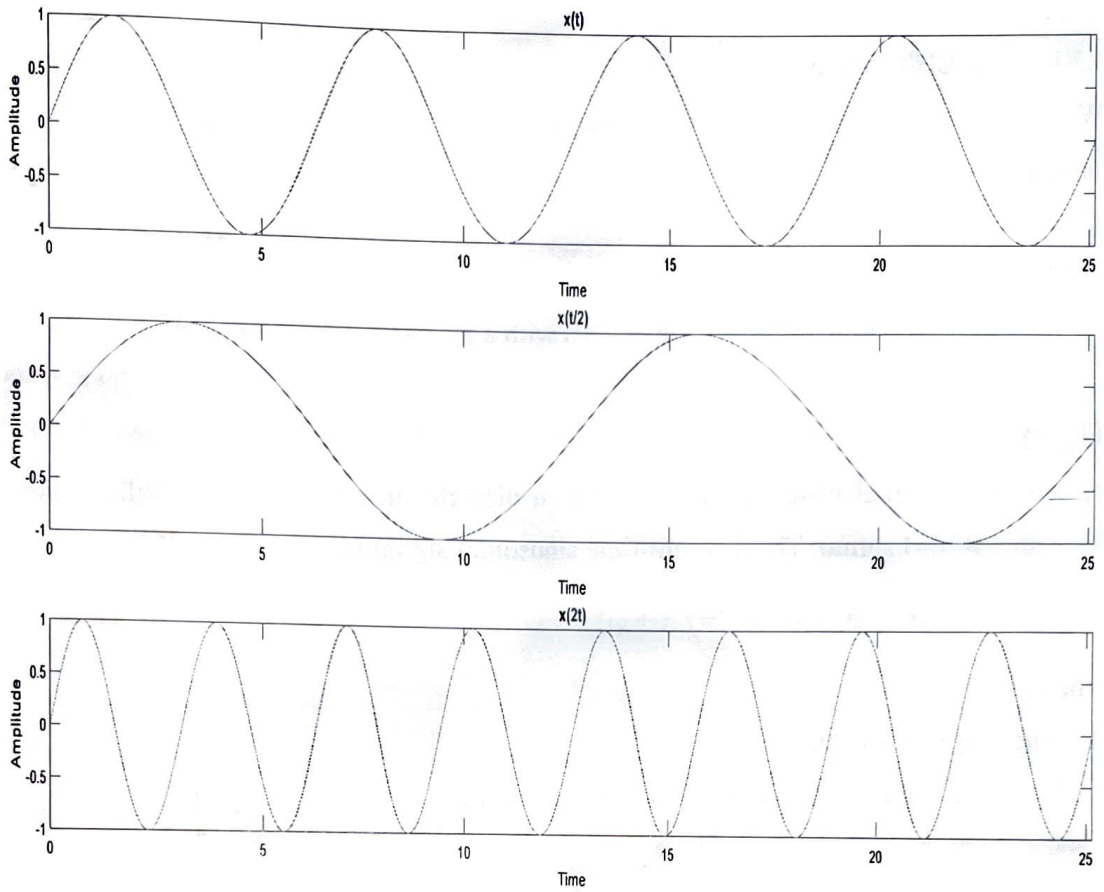


**Fig.2.3: Plot of Time Scaling**

**Result:**

Folding, time shifting, and time scaling of a given signal are implemented.

**EXPERIMENT NO. 3**

Write a MATLAB program to generate discrete sine and cosine signals with a given sampling frequency.

**AIM:**

To generate discrete sine and cosine signals with a given sampling frequency.

**Theory:**

A sinusoidal signal which is defined only at discrete instants of time is called **discrete-time sinusoidal signal**. The discrete-time sinusoidal signal is

$$x(n) = A\sin(\omega n + \varphi) = A\sin(2\pi f n + \varphi)$$

where,

A is the amplitude of the signal.

$\omega=2\pi f$ is the angular frequency in radians per seconds.

f is the frequency of the signal in Hz.

$\varphi$ is the phase angle in radians.

n is an integer.

A cosine signal which is defined only at discrete instants of time is called **discrete-time cosine signal**. The discrete-time cosine signal is

$$x(n) = A\cos(\omega n + \varphi) = A\cos(2\pi f n + \varphi)$$

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in figure window.

**Program:**

```
clear;
close all;
clc;
tfinal=0.1;
% signal frequency
```

```
fd=100;

% sampling frequency
fs=1000;
n=0:1/fs:tfinal;
x1=sin(2*pi*n*fd);
x2=cos(2*pi*n*fd);
subplot(211);
stem(n,x1);
title('Discrete sine wave');
xlabel('time');
ylabel('amplitude');
subplot(212);
stem(n,x2);
title(' Discrete cosine wave');
xlabel('time');
ylabel(' amplitude');
```

**Output:**

The plots are shown in Fig.3

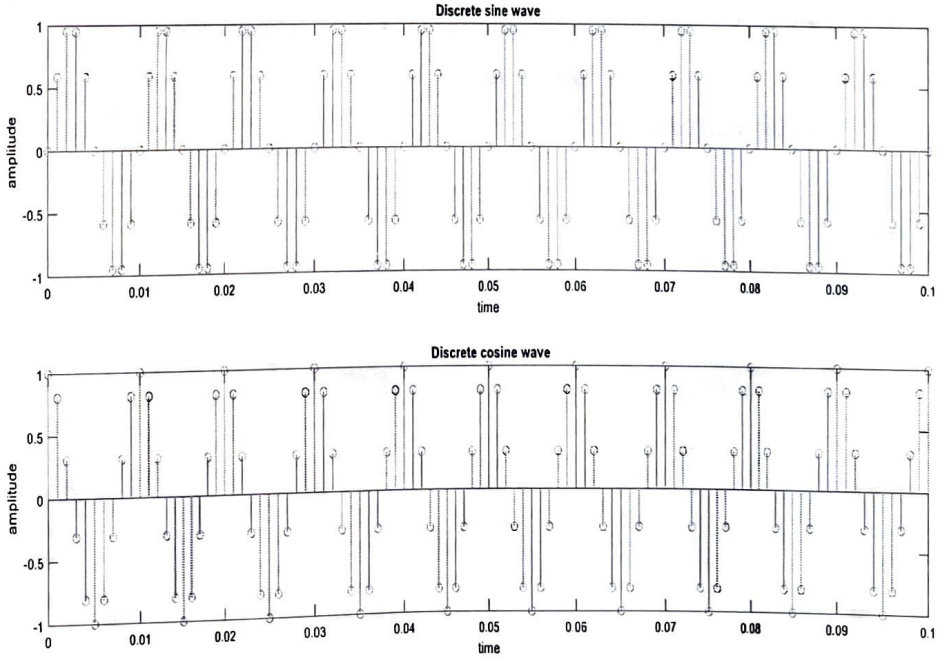


**Fig.3: Plot of Discrete sine wave and Discrete cosine wave**

**Result:**

Discrete sine and cosine signals are generated.

**EXPERIMENT NO. 4**

Write a MATLAB program to perform linear convolution between two vectors using MATLAB.

**AIM:**

To perform linear convolution between two vectors.

**Theory:**

The output y[n] of a LTI (linear time invariant) system can be obtained by convolving the input x[n] with the system's impulse response h[n].

The convolution sum is $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k]$.

x[n] and h[n] can be both finite or infinite duration sequences.

If both the sequences are of finite duration, then we can use the MATLAB function '*conv*' to evaluate the convolution sum to obtain the output y[n].

The length of y[n] = xlength + hlength -1.

The conv function assumes that the two sequences begin at n=0 and is invoked by y=conv(x,h).

Even if one of the sequences begin at other values of n, say n=-3,or n=2; then we need to provide a beginning and end point to y[n] which are ybegin=xbegin+hbegin and yend=xend+hend respectively.

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.
3) Enter the input sequences.
4) Observe the results in command window and figure window.

**Program:**

```
clc;
clear;
close all;
x1=input('Enter the first sequence');
```

```
x2=input('Enter the second sequence');

y1=conv(x1,x2);
disp('Linearly Convolved Sequence is');
disp(y1);

n=0:length(x1)-1;
subplot(311);
stem(n,x1);
title('First Sequence');
xlabel('n');
ylabel('x1(n)');
n=0:length(x2)-1;
subplot(312);
stem(n,x2);
title('Second Sequence');
xlabel(' n');
ylabel('x2(n)');

n=0:length(y1)-1;
subplot(313);
stem(n,y1);
title('Linearly Convolved Sequence');
xlabel('n');
ylabel('y1(n)');
```

**Output:**

Enter the first sequence[1 2 3 4]

Enter the second sequence[2 5 -1 3]

Linearly Convolved Sequence is

2 9 15 24 23 5 12

The plots are shown in Fig.4.

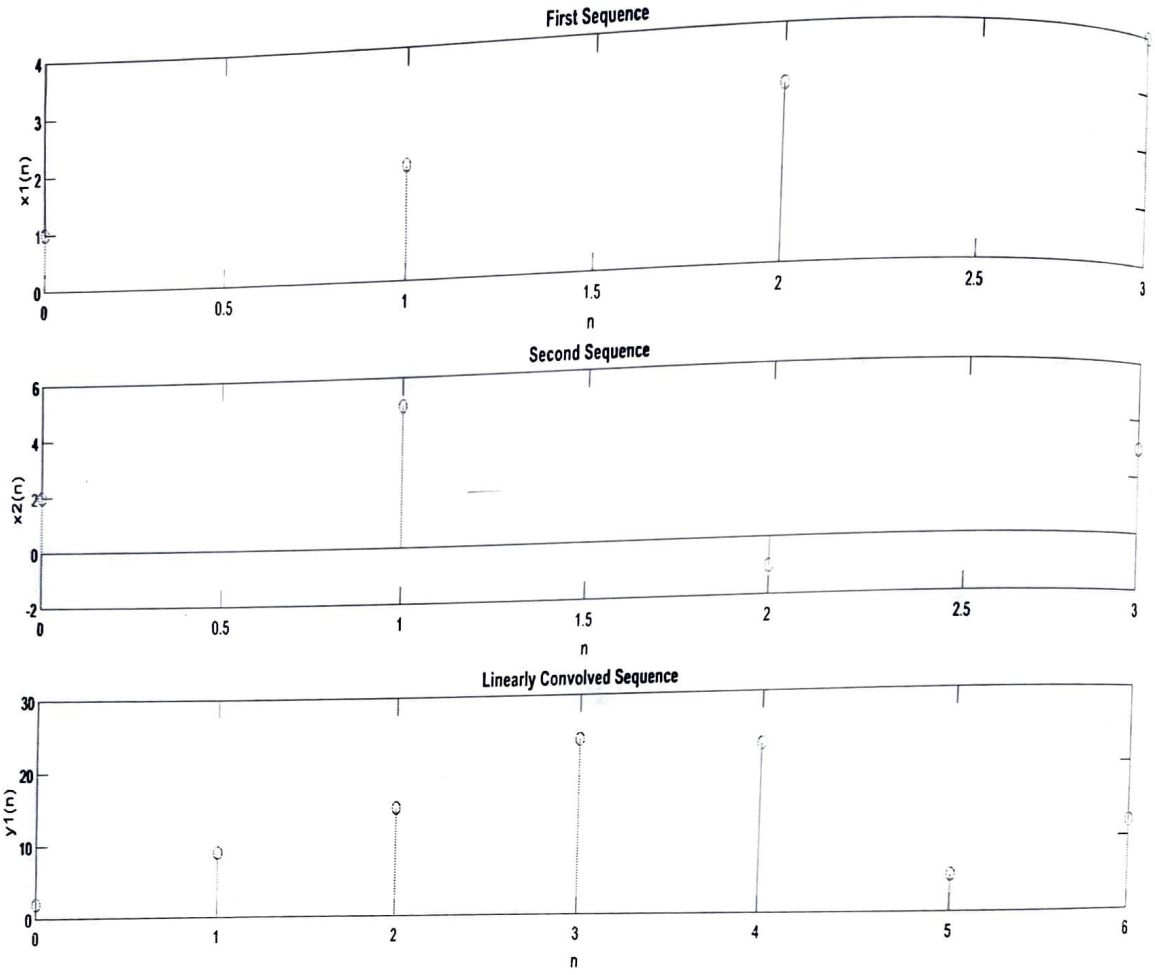


**Fig.4: Plot of Linearly Convolved Sequences**

**Result:**

Linear convolution between two vectors are performed.

**EXPERIMENT NO. 5**

Write a MATLAB program to obtain Z-Transform of a given signal.

**AIM:**

To obtain Z-Transform of a given signal.

**Theory:**

In mathematics and signal processing, the Z-transform converts a discrete-time signal, which is a sequence of real or complex numbers, into a complex frequency-domain (z-domain or z-plane) representation. It can be considered as a discrete-time equivalent of the Laplace transform (s-domain).

The Z-transform (ZT) is a mathematical tool which is used to convert the difference equations in time domain into the algebraic equations in z-domain. To solve these difference equations which are in time domain, they are converted first into algebraic equations in z-domain using the Z-transform, then the algebraic equations are manipulated in z-domain and the result obtained is converted back into time domain using the inverse Z-transform.

The Z-transform may be of two types viz. **unilateral (or one-sided)** and **bilateral (or two-sided).**

Mathematically, if x[n] is a discrete-time signal or sequence, then its *bilateral or two-sided Z-transform* is defined as

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

Where, z is a complex variable and it is given by,

$z = re^{j\omega}$

Also, the *unilateral or one-sided z-transform* is defined as

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

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.

3) Observe the results in command window.

**Program:**

```
clear;
close all;
clc;
syms n;

% finding z transform
a=2;
x=a^n;
X1=ztrans(x);
disp('z tranform of a^n a>1');
disp(X1);

a=0.5;
x=a^n;
X2=ztrans(x);
disp('z tranform of a^n 0<a<1');
disp(X2);

a=2;
x=1+n;
X3=ztrans(x);
disp('z tranform of 1+n');
disp(X3);

sig1 = sin(n);
disp('z transform of sine signal')
X4=ztrans(sig1);
disp(X4);

sig2 = cos(n);
disp('z transform of cosine signal')
X5=ztrans(sig2);
```

```
disp(X5);
```

**Output:**

z tranform of a^n a>1

z/(z - 2)

z tranform of a^n 0<a<1

z/(z - 1/2)

z tranform of 1+n

z/(z - 1) + z/(z - 1)^2

z transform of sine signal

(z*sin(1))/(z^2 - 2*cos(1)*z + 1)

z transform of cosine signal

(z*(z - cos(1)))/(z^2 - 2*cos(1)*z + 1)

**Result:**

Z-Transform of a given signal are obtained.

## EXPERIMENT NO. 6

Write a MATLAB program to obtain inverse Z-Transform of a given system.

### AIM:

To obtain inverse Z-Transform of a given system.

### Theory:

The **inverse Z-transform** is defined as the process of finding the time domain signal x[n] from its Z-transform X(z).

The inverse Z-transform is denoted as

$$x[n] = Z^{-1}(X(z)) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(z)Z^n\,dz$$

Where, z is a complex variable and it is given by,

$z = re^{j\omega}$

### Procedure:

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in command window and figure window.

### Program:

```
clear;
close all;
clc;
syms z;
Fz = z/(z^2 - .5*z + .06);

% inverse z-transform
fn = iztrans(Fz);
pretty(fn);
num = 1;
den = [1,-0.5,0.06];
n = 0:100;
```

```
fn = impulse(num,den,10);

% Plot
subplot(211);
stem(n,fn);
grid on;
title('Discrete Time Sequence ');
ylabel('f(n)');
xlabel('n');
```

**Output:**

```
  / 3 \n      / 1 \n
10 | -- |  - 10 | - |
  \ 10 /      \ 5 /
```

The plot is shown in Fig.6.

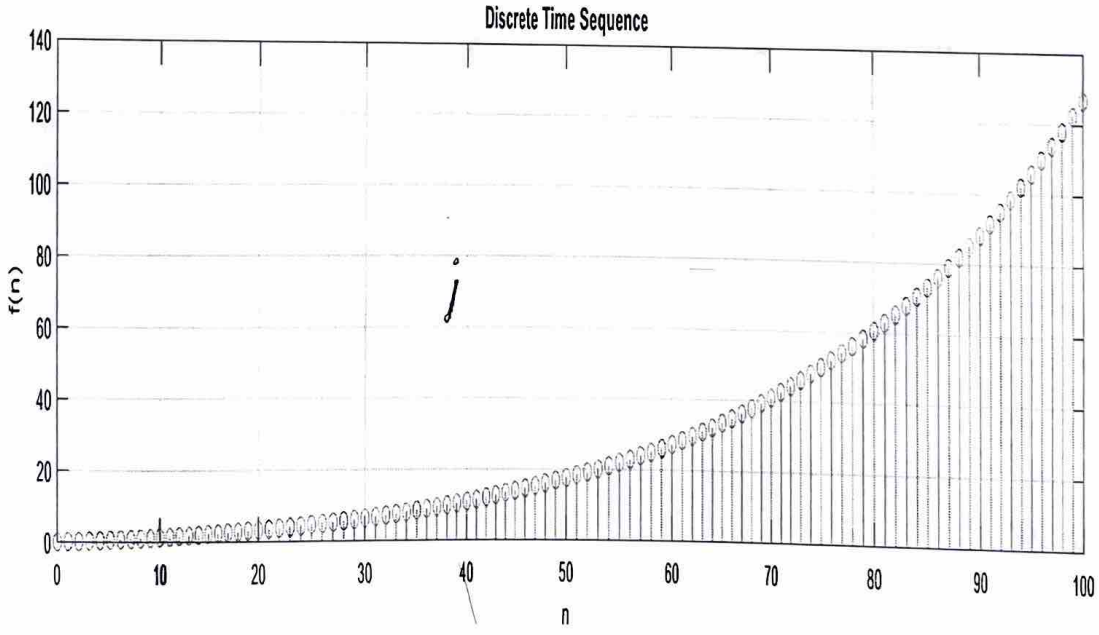


**Fig.6: Plot of Inverse Z-Transform**

**Result:**

Inverse Z-Transform of a given system is obtained.

**EXPERIMENT NO. 7**

Write a MATLAB program to determine impulse response of a given system.

**AIM:**

To determine impulse response of a given system.

**Theory:**

The impulse response for an LTI system is the output y[n] when the input is unit impulse signal δ[n].

In other words, when x[n]= δ[n], h[n] = y[n].

Given the system equation, you can find the impulse response just by feeding x[n] = δ[n] into the system.

If the system is linear and time-invariant, then you can use the impulse response to find the output for any input, using a method called convolution.

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in figure window.

**Program:**

```
clc;
clear;
close all;
N=50;
b=1;
a=[1,-0.5];

% To find Impulse Response
x=[1,zeros(1,N-1)];
n=0:1:N-1;
y=filter(b,a,x);
subplot(211);
```

```
stem(n,x);
xlabel('n');
ylabel('u(n)');
title('Impulse Input');
subplot(212);
stem(n,y);
xlabel('n');
ylabel('y(n)');
title('Impulse Response');
```

**Output:**

The plot is shown in Fig.7.

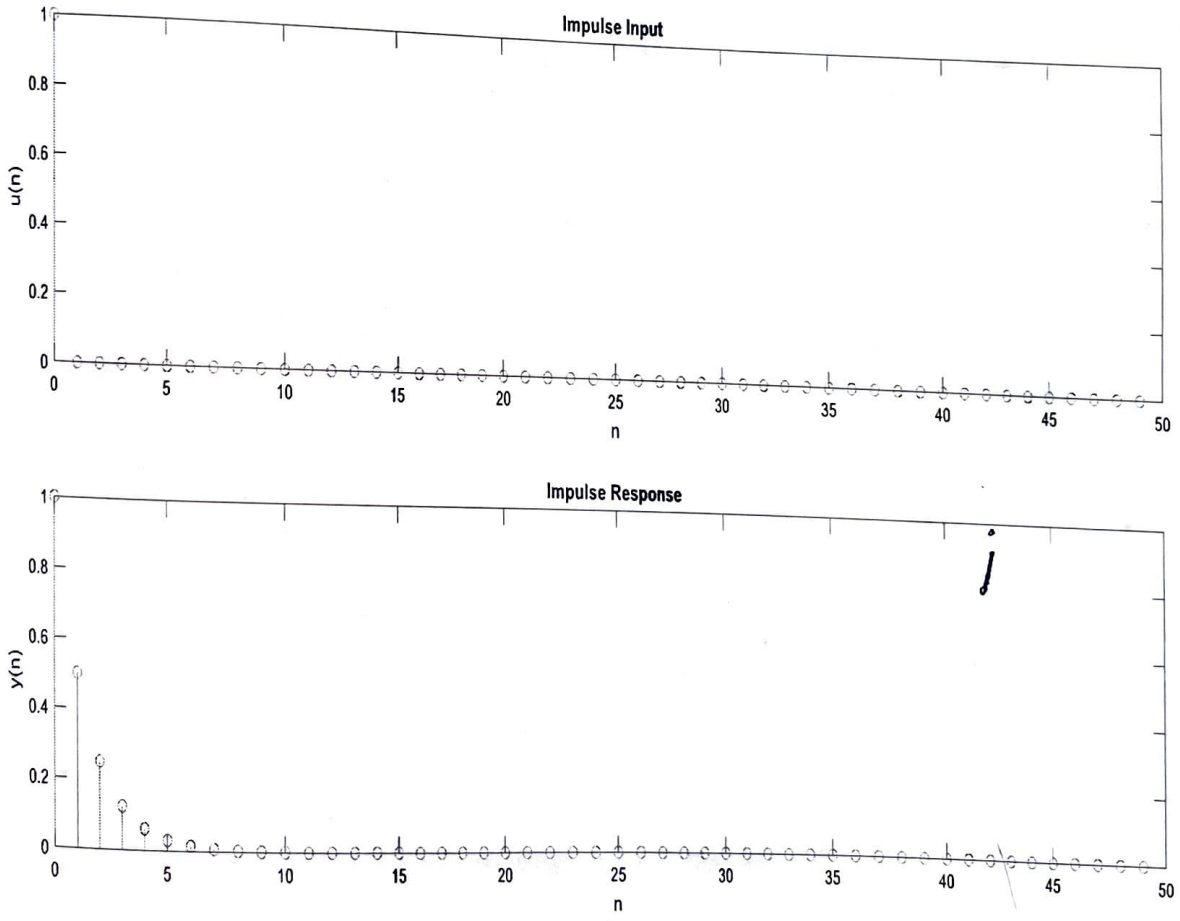


**Fig.7: Plot of Impulse Response**

**Result:**

Impulse Response of a given system is determined.

**EXPERIMENT NO. 8**

Write a MATLAB program to determine step response of a given system.

**AIM:**

To determine step response of a given system.

**Theory:**

The step response for an LTI system is the output y[n] when the input is the unit step signal u[n].

In other words, when x[n]= u[n], h[n] = y[n].

Given the system equation, you can find the step response just by feeding x[n] = u[n] into the system.

The step response of a discrete LTI system is the convolution of the unit step with the impulse response.

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in figure window.

**Program:**

```
clc;
clear;
close all;
N=50;
b=1;
a=[1,-0.5];

% To find step Response
x=[ones(1,N)];
n=0:1:N-1;
y=filter(b,a,x);
subplot(211);
```

```
stem(n,x);
xlabel('n');
ylabel('u(n)');
title('Step Input');
subplot(212);
stem(n,y);
xlabel('n');
ylabel('y(n)');
title('Step Response');
```

**Output:**

The plot is shown in Fig.8.

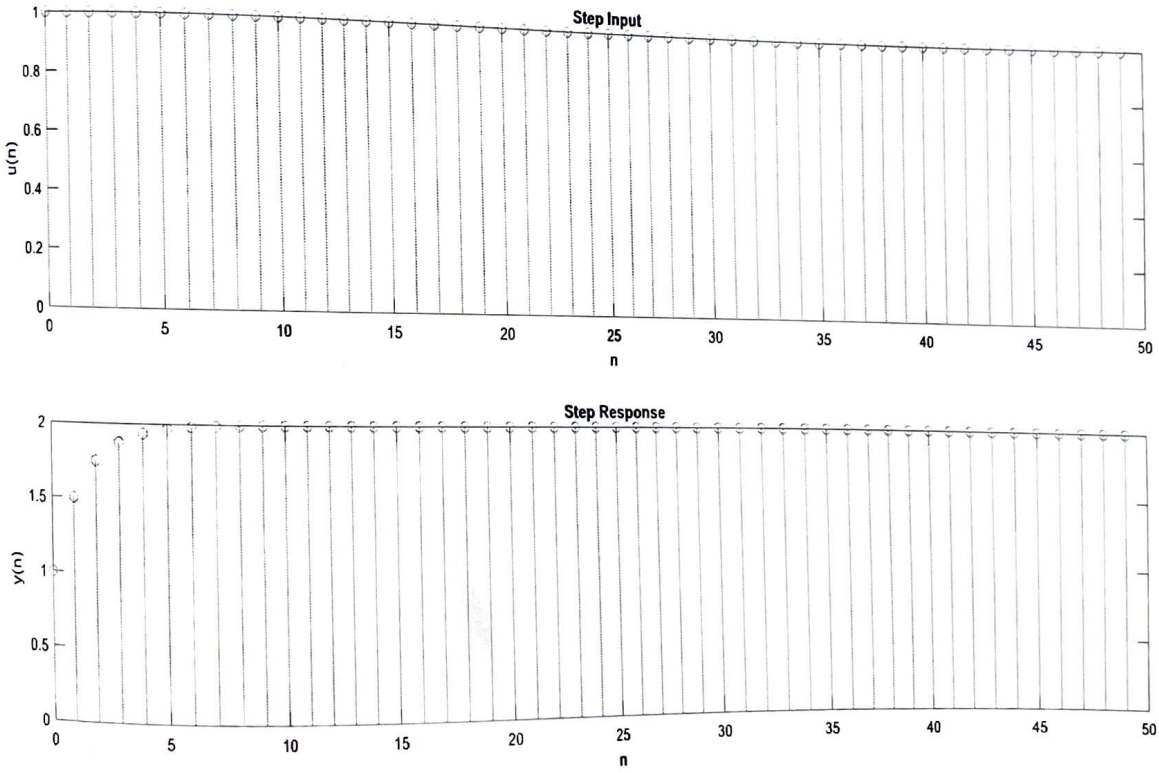


**Fig.8: Plot of Step Response**

**Result:**

Step Response of a given system is determined.

**EXPERIMENT NO. 9**

Write a MATLAB program to obtain partial fraction expansion of a given equation.

**AIM:**

To obtain partial fraction expansion of a given equation.

**Theory:**

Partial fraction expansion (also called partial fraction decomposition) is performed whenever we want to represent a complicated fraction as a sum of simpler fractions. This occurs when working with Laplace or Z-Transform in which we have methods of efficiently processing simpler fractions.

**Procedure:**

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in command window.

**Program:**

```
clc;
clear;
close all;
syms z
Fz = (z+1)/(z^2 - 5*z + 6);
P1=partfrac(Fz);
disp(P1);
```

**Output:**

```
4/(z - 3) - 3/(z - 2)
```

**Result:**

Partial fraction expansion of a given equation is obtained.

## EXPERIMENT NO. 10

Write a MATLAB program to obtain pole zero plot of a given system.

### AIM:

To obtain pole zero plot of a given system.

### Theory:

A pole–zero plot is a graphical representation of a rational transfer function in the complex plane which helps to convey certain properties of the system.

Poles and Zeros of a transfer function are the frequencies for which the value of the denominator and numerator of transfer function becomes infinite and zero respectively. The values of the poles and the zeros of a system determine whether the system is stable, and how well the system performs.

Poles are the roots of the denominator of a transfer function. Zeros are the roots of the nominator of a transfer function.

### Procedure:

1) Type the program in the editor window and save it.
2) Run the program.
3) Observe the results in figure window.

### Program:

```
clc;
clear;close all;
syms z;
F=(12*z^2 - z)/(6*z^2 - z - 1);
[num,den] = numden(F);
Ts = 0.1; % Sampling period
H = tf(sym2poly(num),sym2poly(den),Ts);
pzmap(H);
```

### Output:

The plot is shown in Fig.10.

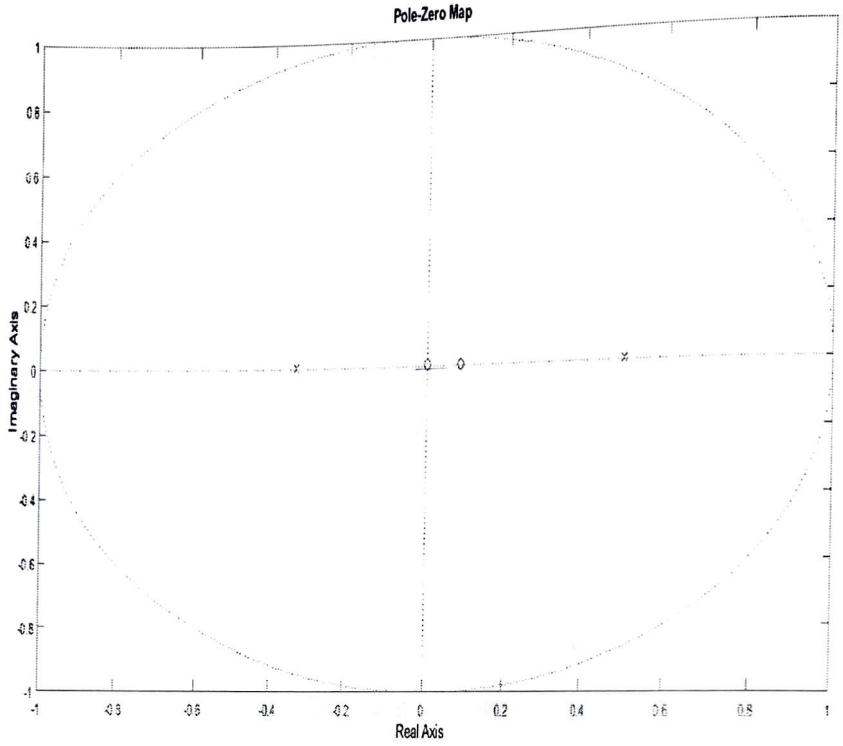


**Fig.10: Pole Zero Plot**

**Result:**

Pole zero plot of a given system is obtained.

$j$