# Title: GENERATION OF BASIC SIGNALS USING MATLAB

**AIM: To generate basic signals unit impulse, unit step, unit ramp signal and Exponential signals.**

**Objective: To generate basic signals like unit impulse, unit step, unit ramp signal and Exponential signals using MATlab.**

**Write a function in MATLAB to generate**

a) **x(t)= 4t2+3,**
b) **y(t)=3t+5,**
c) **x[n]= 2n2+1**
d) **y[n]=8n+3**

a) **x(t)= 4t2+3**

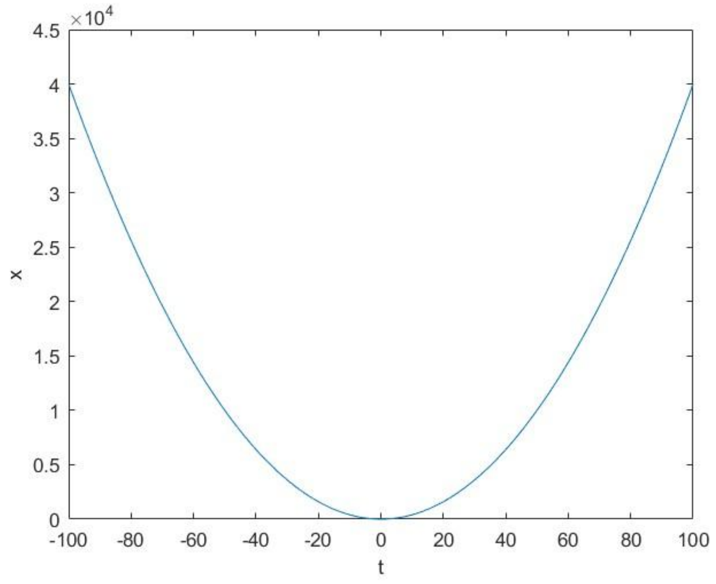
**Description**

In problem (a) x(t) is the continuous signal with dependent variable t is in the form of linear second order polynomial equation. The curve of this equation is a parabola

**Matlab Program**

```
t = linspace(-100,100,1000);
x = 4*t.*t+3;
plot(t,x);
xlabel('t');
ylabel('x');
```

**Output of the program**



**b) y(t)=3t+5**

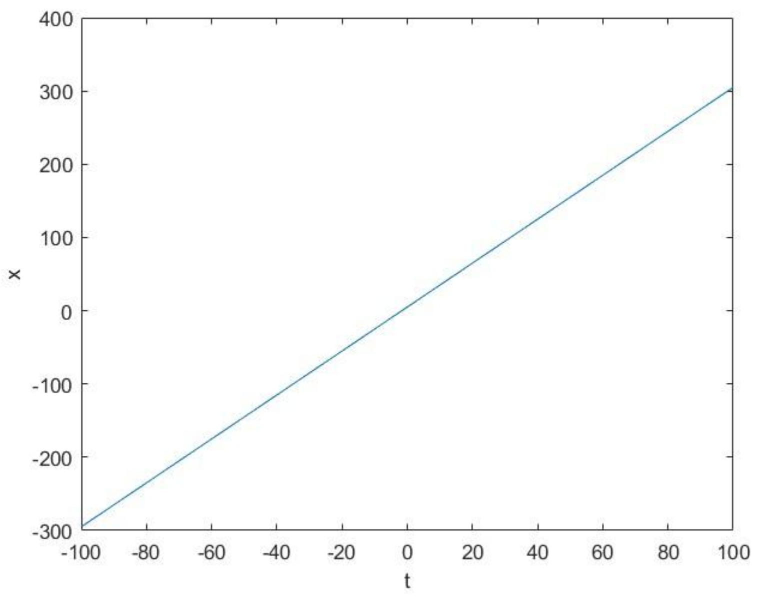
**Description**

In problem (b) x(t) is the continuous signal with dependent variable t is in the form of linear first order polynomial equation. The curve of this equation is a straight line.

**Matlab Program**

```
t = linspace(-100,100,1000);
x = 3*t+5;
plot(t,x);
xlabel('t');
ylabel('x');
```

**Output of the program**



**c) x[n]= 2n2+1**

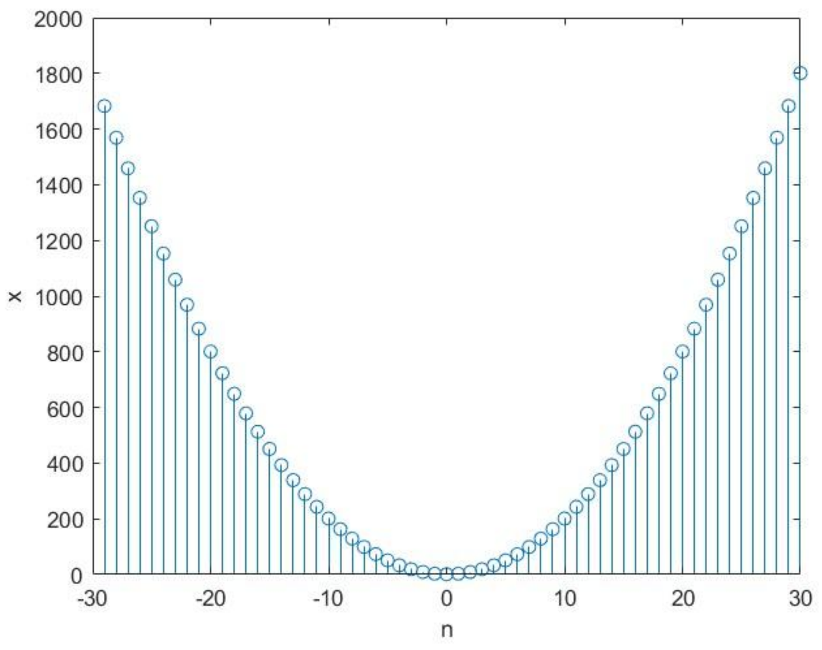
**Description**

In problem (c) x[n] is the discrete signal with dependent variable n where n is the integer. The equation (c) is in the form of a second order polynomial equation. The curve of this equation is a discrete sampled parabola.

**Matlab Program**

```
n=-29:30;
x = 2*n.*n+1;
stem(n,x);
xlabel('n');
ylabel('x');
```

**Output of the program**



d) y[n]=8n+3

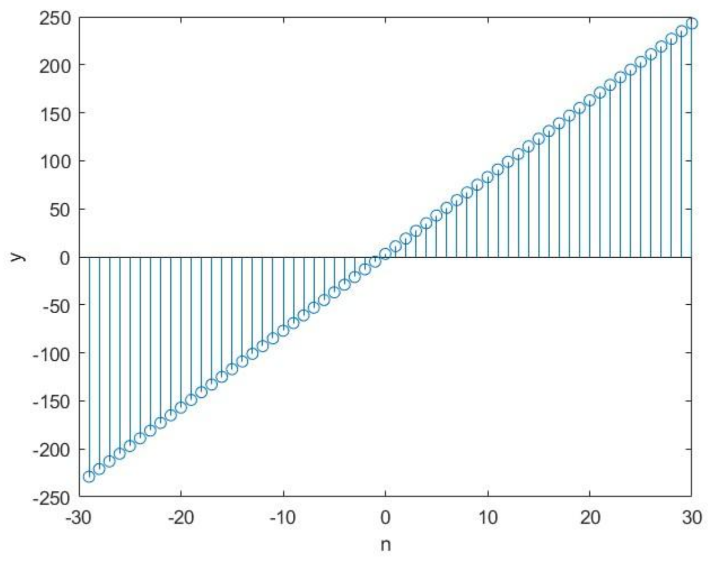
**Description**

In problem (d) x[n] is the discrete signal with dependent variable n where n is the integer. The equation (d) is in the form of a first order polynomial equation. The curve of this equation is a discrete sampled straight line.

**Matlab Program**

```
n=-29:30;
x = 8*n+3;
stem(n,x);
xlabel('n');
ylabel('x');
```

**Output of the program**



**Write a function in MATLAB to generate samples**

**(a) unit impulse**
**(b) unit step**
**(c) unit ramp**
**(d) square (amplitude 2, duty cycle 50%, frequency, sampling rate, total time duration)**
**(e) triangular (with the following input parameters: frequency, sampling rate, total time duration.)**
**(f) exponential (growing and decaying) and**
**(g) sinusoidal signal (with the following input parameters: amplitude, phase, frequency, sampling rate, total time duration.) in continuous and discrete time domain.**

**a) Unit impulse**

**Description**

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.

Continuous-Time Unit Impulse Signal

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

δ(t)={
1 for t=0
0 for t≠0 }

Hence, by the definition, the unit impulse signal has zero amplitude everywhere except at t = 0. At the origin (t = 0) the amplitude of the impulse signal is infinity so that the area under the curve is unity. The continuous-time impulse signal is also called Dirac Delta Signal.

Discrete-Time Unit Impulse Signal

The discrete-time unit impulse signal is denoted by δ(n) and is defined as −

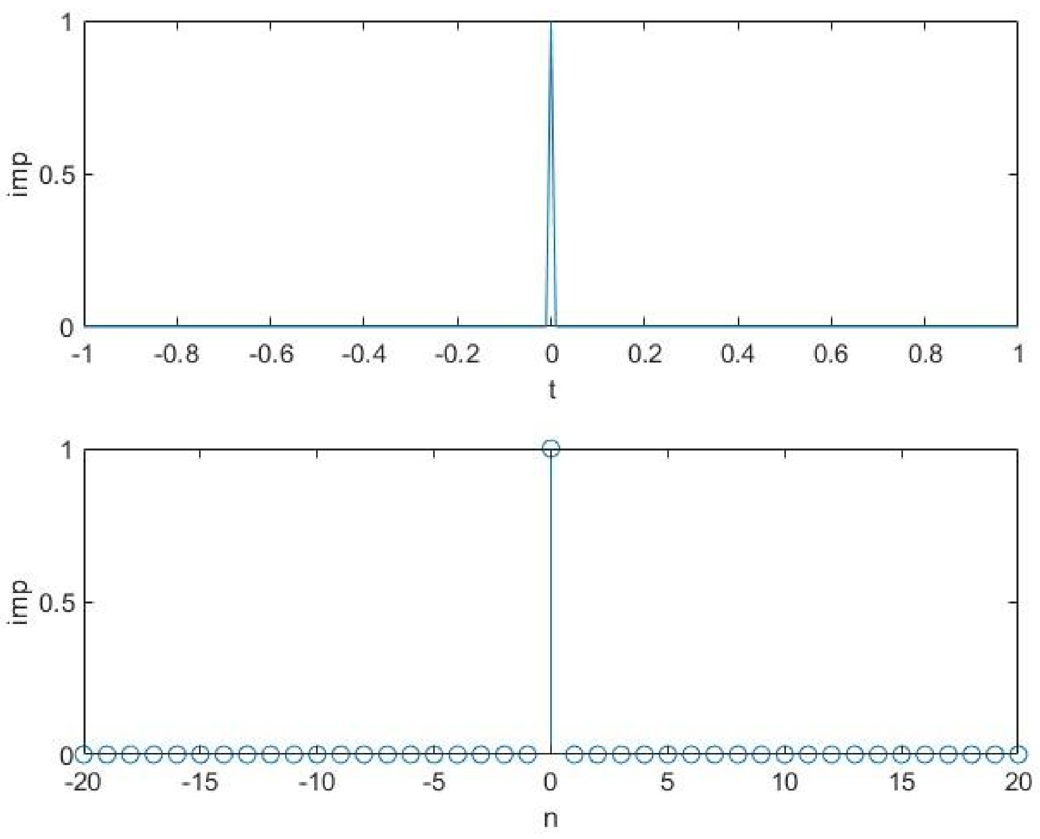
δ(n)={1 for n=0
0 for n≠0 }

The discrete-time signal is also called unit sample sequence

**Matlab Program**

```
t = (-1:0.01:1)';
imp = t==0;
subplot(2,1,1);
plot(t,imp);
xlabel('t');
ylabel('imp');
subplot(2,1,2);
n = -20:20;
imp1 = n==0;
stem(n,imp1);
xlabel('n');
ylabel('imp');
```

**Output of the program**



**b) Unit step**

**Description**

The unit step function, u(t), is defined as

u(t)={0 t<0

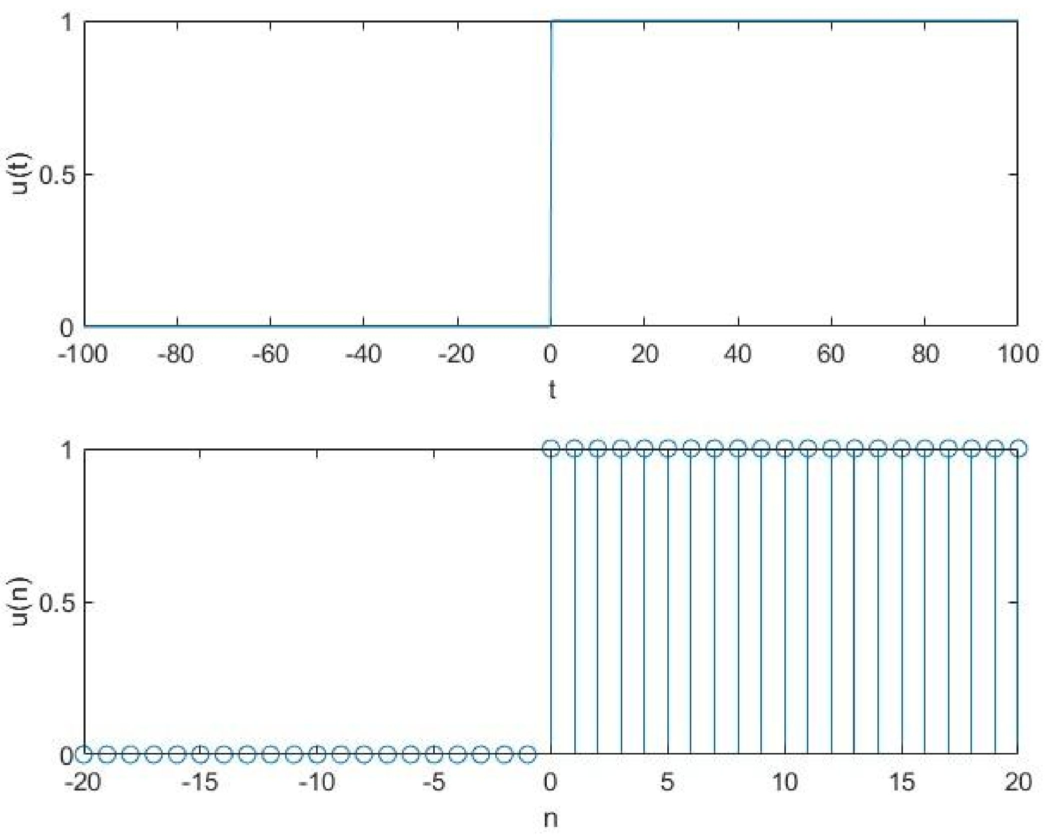
1 t>0 }

That is, u is a function of time t, and u has value zero when time is negative and value one when time is positive

**Matlab Program**

```
t = linspace(-100,100,1000);
unit = t>=0;
subplot(2,1,1);
plot(t,unit);
xlabel('t');
ylabel('u(t)');
n = -20:20;
uint1 = n>=0;
subplot(2,1,2);
stem(n,uint1);
xlabel('n');
ylabel('u(n)');
```

**Output of the program**



c) **Ramp**

**Description**

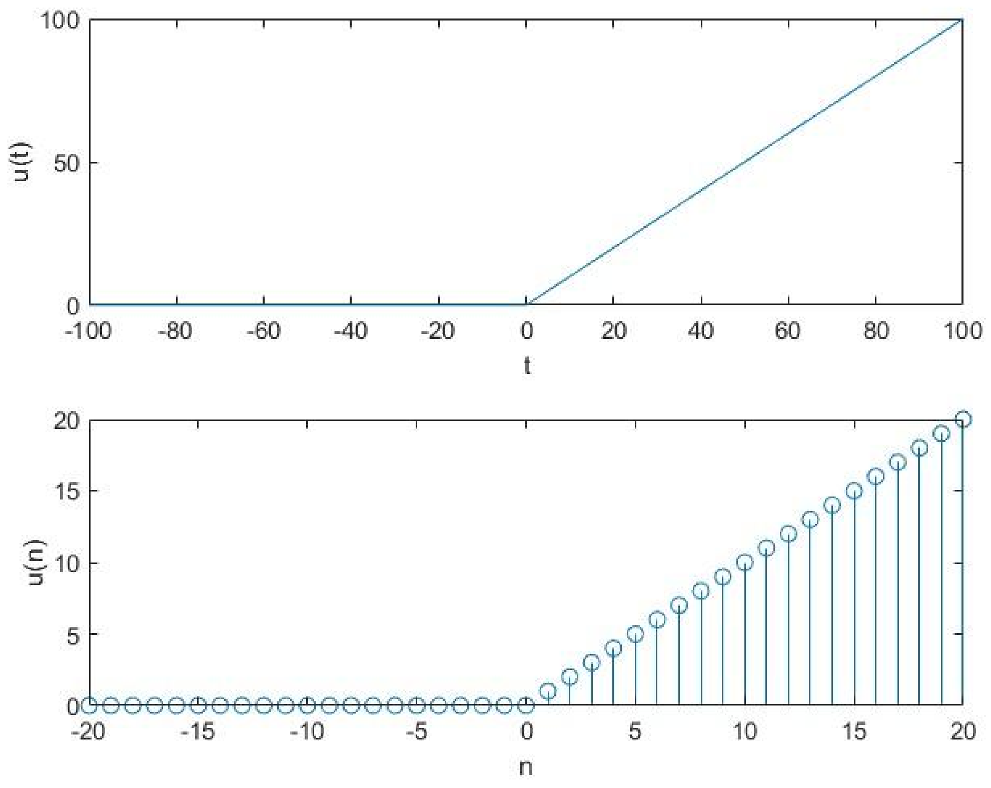
A ramp function or ramp signal is a type of standard signal which starts at *t* = 0 and increases linearly with time. The unit ramp function has unit slope**.**

**Matlab Program**

```
t = linspace(-100,100,1000);
unit = (t>=0).*t;
subplot(2,1,1);
```

```
plot(t,unit);
xlabel('t');
ylabel('u(t)');
n = -20:20;
uint1 = (n>=0).*n;
subplot(2,1,2);
stem(n,uint1);
xlabel('n');
ylabel('u(n)');
```

**Output of the program**



**d) Square (amplitude 2, duty cycle 50%, frequency, sampling rate, total time duration)**

**Description**

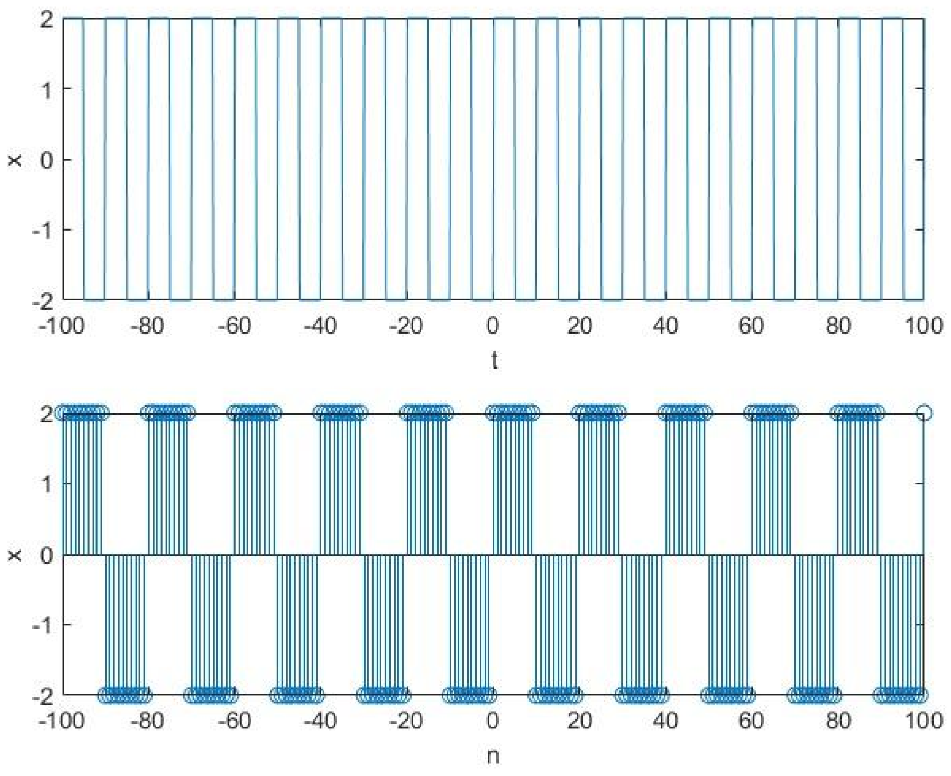
The square wave, also called a pulse train, or pulse wave, is a periodic waveform consisting of instantaneous transitions between two levels. The square wave is sometimes also called the Rademacher function.

**Matlab Program**

```
t = linspace(-100,100,1000); % time from -100 ms to 100 ms
f = 100;  %frequency in Hz
duty = 50; %duty cycle
amp = 2;
x = amp * square(2*pi*f*t,50)
```

```
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
f = .05;  % f reduce to .05 for sampling in 1hz
n = -100:100;
z = 2*square(2*pi*f*n,50);
subplot(2,1,2);
stem(n,z);
xlabel('n');
ylabel('x');
```

**Output of the program**



**e) Triangular (with the following input parameters: frequency, sampling rate, total time duration.)**

**Description**

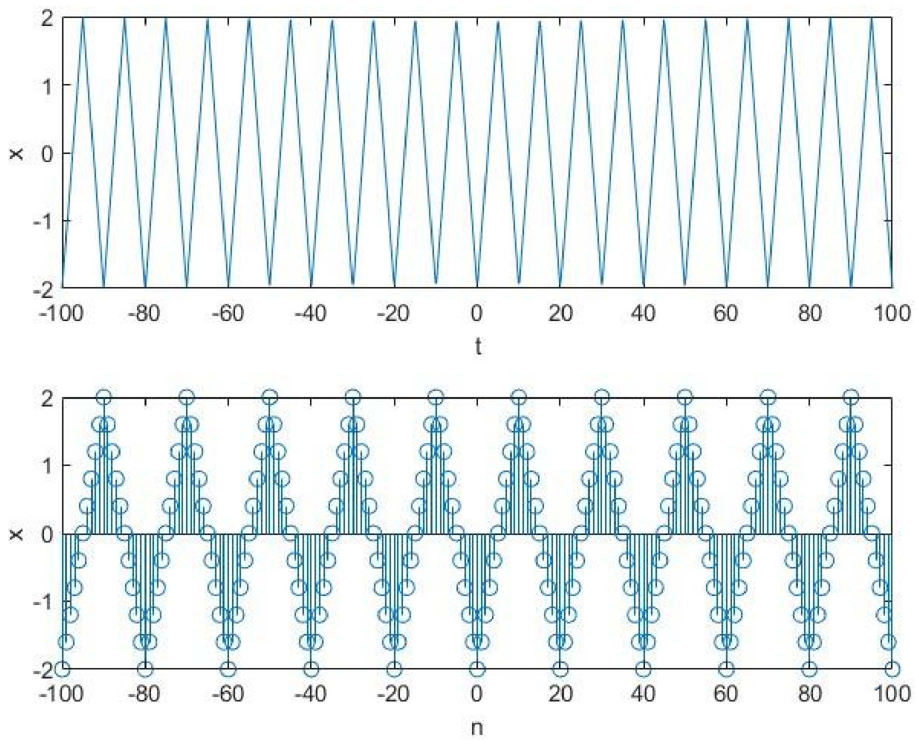
A triangular wave or triangle wave is a non-sinusoidal waveform named for its triangular shape. It is a periodic, piecewise linear, continuous real function.

**Matlab Program**

```
t = linspace(-100,100,1000); % time from -100 ms to 100 ms
f = 100;  %frequency in Hz
duty = 50; %duty cycle
amp = 2;
x = amp*sawtooth(2*pi*f*t,1/2);
subplot(2,1,1);
plot(t,x)
```

```
xlabel('t');
ylabel('x');
n = -100:100;
f = .05;  % f reduce to .05 for sampling in 1hz
z = amp*sawtooth(2*pi*f*n,1/2);
subplot(2,1,2);
stem(n,z)
xlabel('n');
ylabel('x');
```

**Output of the program**



### f) Exponential (growing and decaying)

Description

An exponential function is a Mathematical function in the form f (x) = a^x, where “x” is a variable and “a” is a constant which is called the base of the function and it should be greater than 0. The most commonly used exponential function base is the transcendental number e, which is approximately equal to 2.71828.

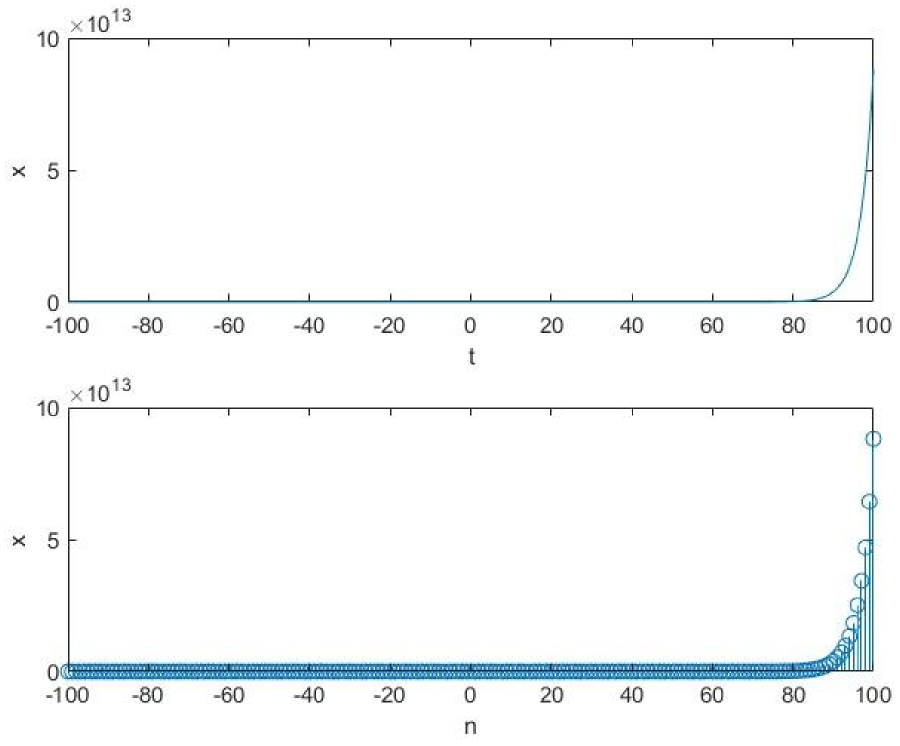
**exponential growing function is given by e^(+2*pi*f*t)**

**Matlab Program**

```
t = linspace(-100,100,1000); % time from -100 ms to 100 ms
f = 0.05;  %frequency in Hz
duty = 50; %duty cycle
amp = 2;
x = amp*exp(2*pi*f*t);
subplot(2,1,1);
```

```
plot(t,x)
xlabel('t');
ylabel('x');
n = -100:100;
f = .05;  % f reduce to .05 for sampling in 1hz
z = amp*exp(2*pi*f*n);
subplot(2,1,2);
stem(n,z)
xlabel('n');
ylabel('x');
```

**Output of the program**



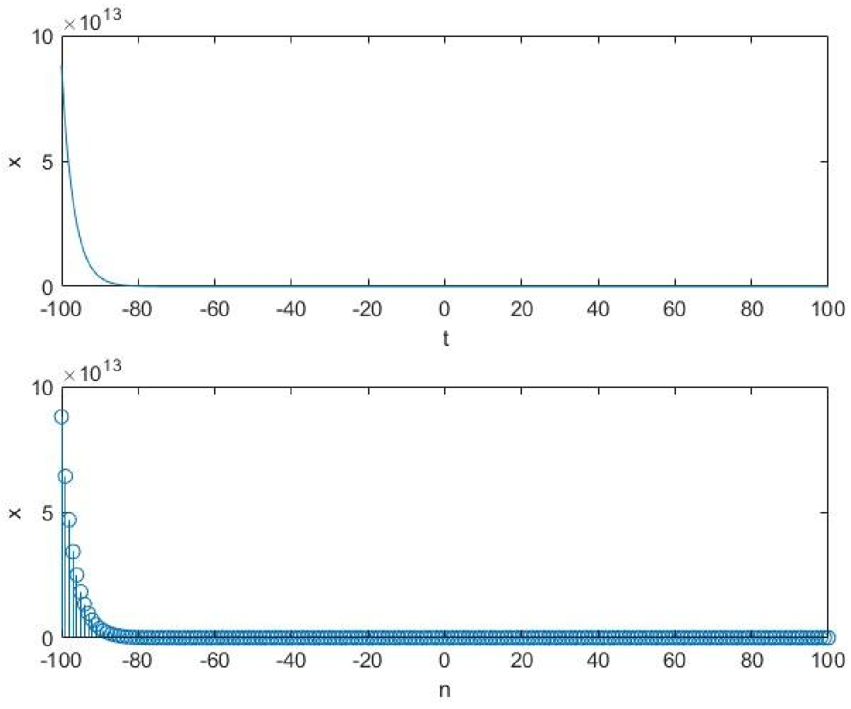
**exponential growing function is given by e^(-2*pi*f*t)**

**Matlab Program**

```
t = linspace(-100,100,1000); % time from -100 ms to 100 ms
f = 0.05;  %frequency in Hz
duty = 50; %duty cycle
amp = 2;
x = amp*exp(-2*pi*f*t);
subplot(2,1,1);
plot(t,x)
xlabel('t');
ylabel('x');
n = -100:100;
f = .05;  % f reduce to .05 for sampling in 1hz
z = amp*exp(-2*pi*f*n);
subplot(2,1,2);
stem(n,z)
xlabel('n');
```

```
ylabel('x');
```

**Output of the program**



**g) Sinusoidal signal (with the following input parameters: amplitude, phase, frequency, sampling rate, total time duration.) in continuous and discrete time domain.**

**Description**

A sinusoidal function or sinusoidal signal is a function that describes a smooth periodic oscillation.
Continuous-Time Sinusoidal Signal
A sinusoidal signal which is defined for every instant of time is called continuous-time sinusoidal signal. The continuous time sinusoidal signal is given as follows −
$x(t) = A \sin(\omega t + \varphi) = A \sin(2\pi f t + \varphi)$
Where,

- A is the amplitude of the signal. That is the peak deviation of the signal from zero.
- ω=2πf is the angular frequency in radians per seconds.
- f is the frequency of the signal in Hz.
- φ is the phase angle in radians.

All the continuous-time sinusoidal signals are periodic signals. The time period (T) of a continuous-time sinusoidal signal is given by,
T = 1/f =2*π/ω
Discrete-Time Sinusoidal Signal

A sinusoidal signal which is defined only at discrete instants of time is called discrete-time sinusoidal signal. The discrete-time sinusoidal signal is given as follows −
$x(n) = A \sin(\omega n + \varphi) = A \sin(2\pi f n + \varphi)$
Where,

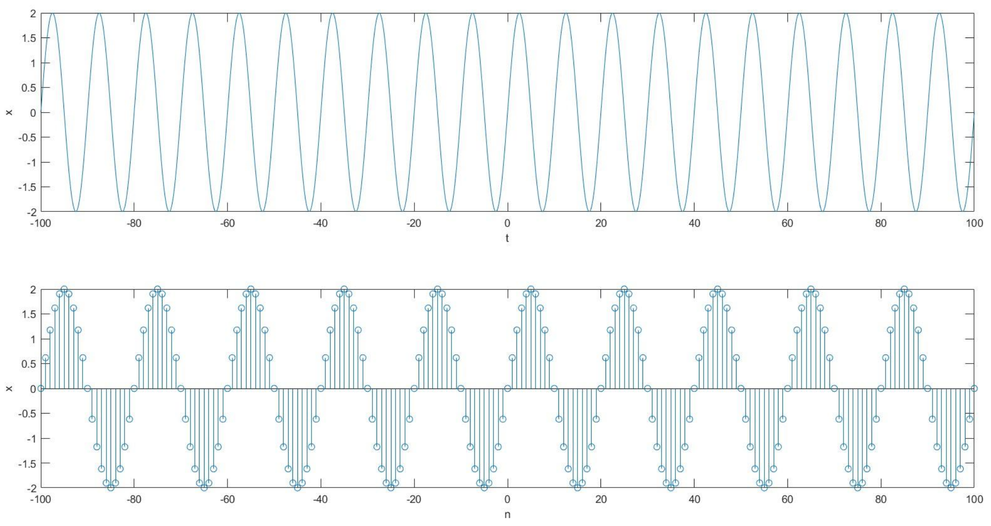
- A is the amplitude of the signal.
- ω=2πf is the angular frequency in radians per seconds.
- f is the frequency of the signal in Hz.
- φ is the phase angle in radians.
- n is an integer.

The time period of the discrete-time sinusoidal signal is given by,
N =2*π/ω*m where m is integer

**Matlab Program**

```
t = linspace(-100,100,1000); % time from -100 ms to 100 ms
f = 100;  %frequency in Hz
amp = 2;
x = amp * sin(2*pi*f*t)
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
f = .05;  % f reduce to .05 for samping in 1hz
n = -100:100;
z = 2*sin(2*pi*f*n);
subplot(2,1,2);
stem(n,z);
xlabel('n');
ylabel('x');
```

**Output of the program**



**Write a function in MATLAB to study transformation on**

**(a) amplitude scaling by factor of 3 (+ve and –ve)**
**(b) time-scaling (compression and dilation)**
**(c) time shifting (+ve and –ve factor)**

**(a) amplitude scaling by factor of 3 (+ve and –ve)**

**Description**

The process of rescaling the amplitude of a signal, i.e., the amplitude of the signal is either amplified or attenuated, is known as amplitude scaling. In the amplitude scaling operation on signals, the shape of the resulting signal remains the same as that of the original signal but the amplitude is altered (i.e., increased or decreased).

Let us take a function

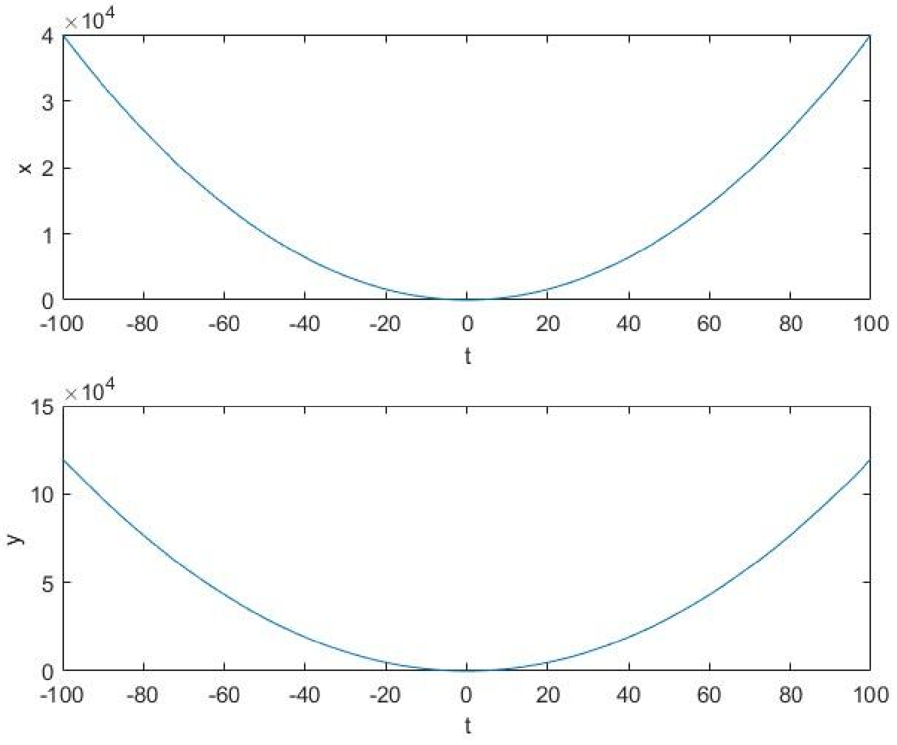
**x(t)= 4t^2+3**

**Positive scaling**

**Matlab Program**

```
t = linspace(-100,100,1000);
scale = 3;
x = 4*t.*t+3;
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(2,1,2);
plot(t,scale*x);
xlabel('t');
ylabel('y');
```

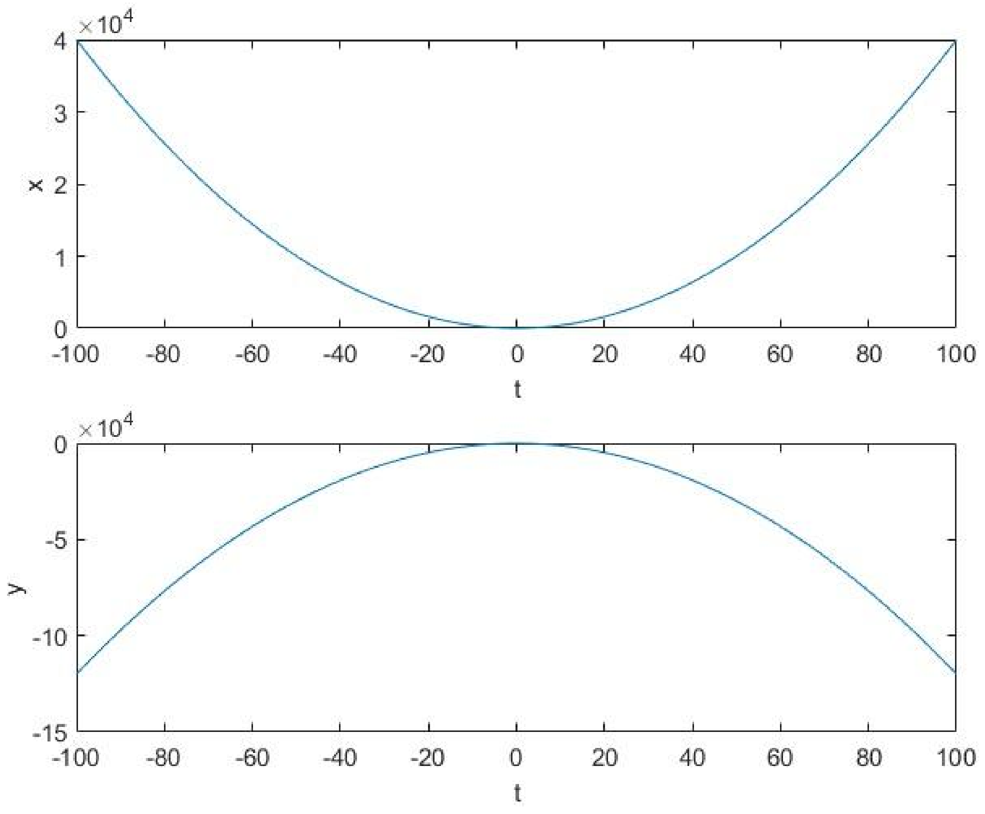
**Output of the program**



**Negative scaling**
**Matlab Program**

```
t = linspace(-100,100,1000);
scale = - 3;
x = 4*t.*t+3;
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(2,1,2);
plot(t,scale*x);
xlabel('t');
ylabel('y');
```

**Output of the program**



**(b) time-scaling (compression and dilation)**

**Description**

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 a signal may be time compression or time expansion depending upon the value of the constant or scaling factor.

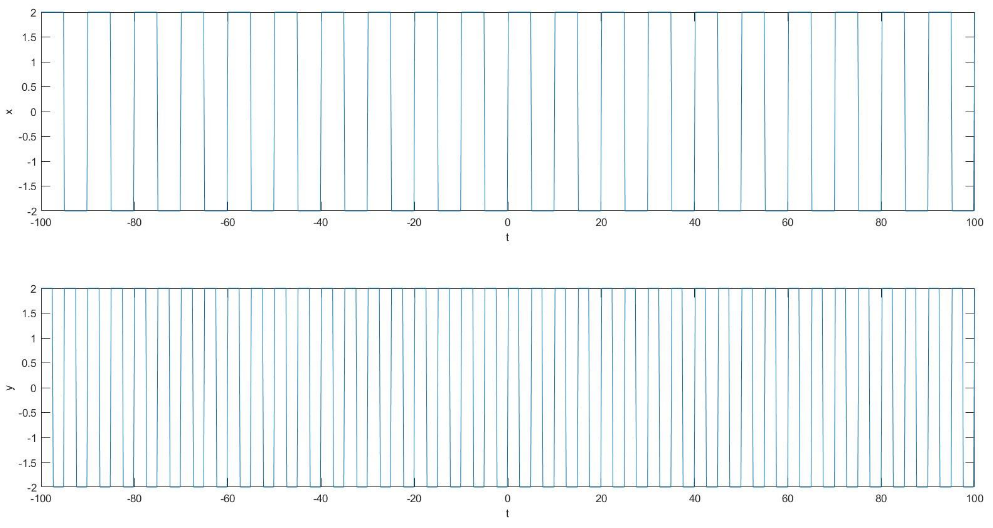
Let us take a function of square wave with frequency 100 Hz and amplitude 2V and duty 50%.

**Compression**

**Matlab Program**

```
t = linspace(-100,100,1000); % time from -100 ms to 100 ms
f = 100;  %frequency in Hz
duty = 50; %duty cycle
amp = 2;
scale =2;
x = amp * square(2*pi*f*t,50);
t1 = scale*t;
y = amp * square(2*pi*f*t1,50);
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(2,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
```

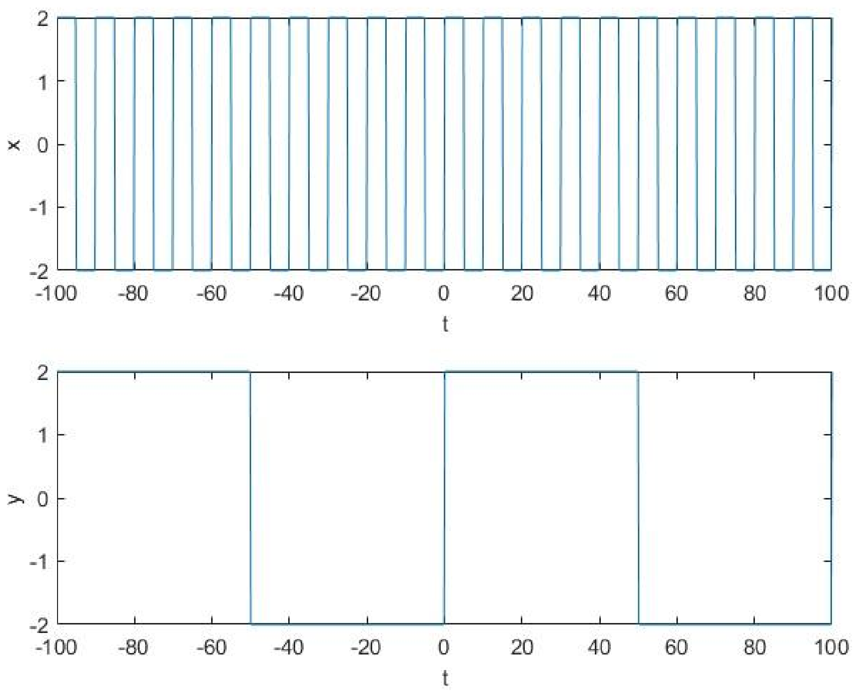
## Output of the program



## Dilation
## Matlab Program

```
t = linspace(-100,100,1000); % time from -100 ms to 100 ms
f = 100;  %frequency in Hz
duty = 50; %duty cycle
amp = 2;
scale =1/10;
x = amp * square(2*pi*f*t,50);
t1 = scale*t;
y = amp * square(2*pi*f*t1,50);
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(2,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
```

**Output of the program**



**(c) time shifting (+ve and –ve factor)**

**Description**

Time shifting or Shifting of a signal in time means that the signal may be either delayed in the time axis or advanced in the time axis.

The time shifting of a continuous time signal x(t) is represented as,

$y(t) = x(t - t0)$

The time-shifting of a signal results in the time delay or time advancement.

Let us take a function

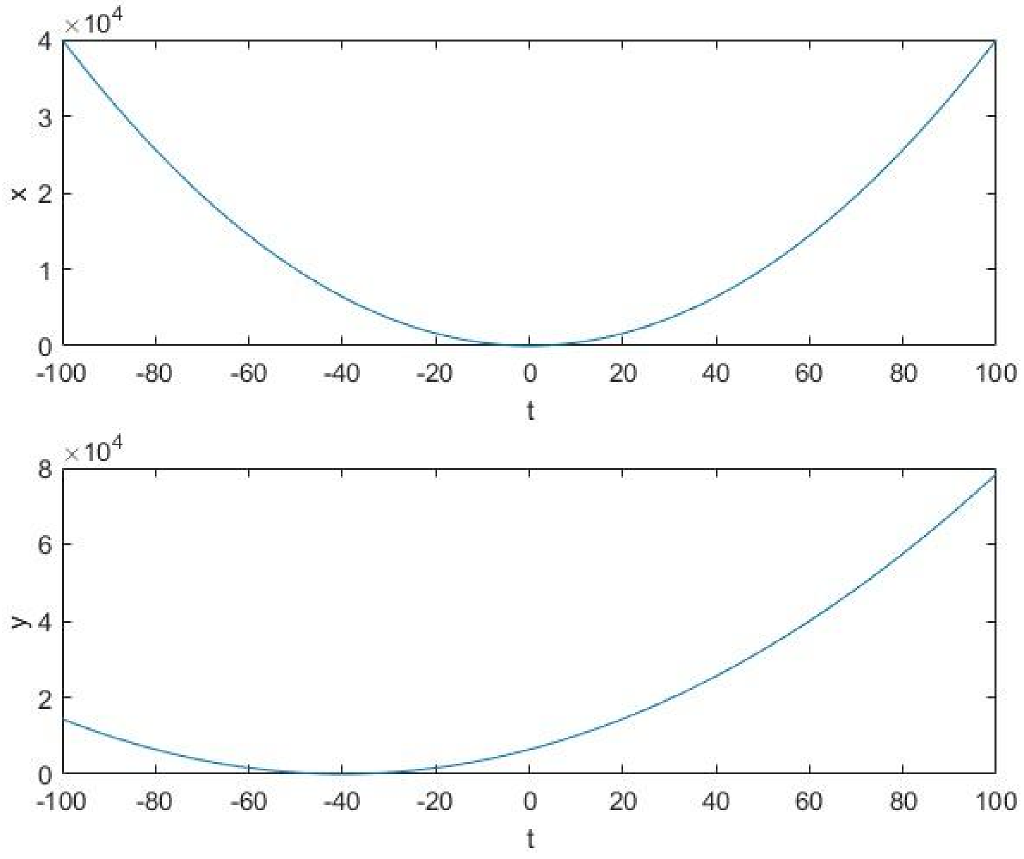
**x(t)= 4t^2+3**

**Time delay**

**Matlab Program**

```
t = linspace(-100,100,1000);
t0 = 40;
x = 4*t.*t+3;
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
t1 = t+t0;
y=4*t1.*t1+3;
subplot(2,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
```

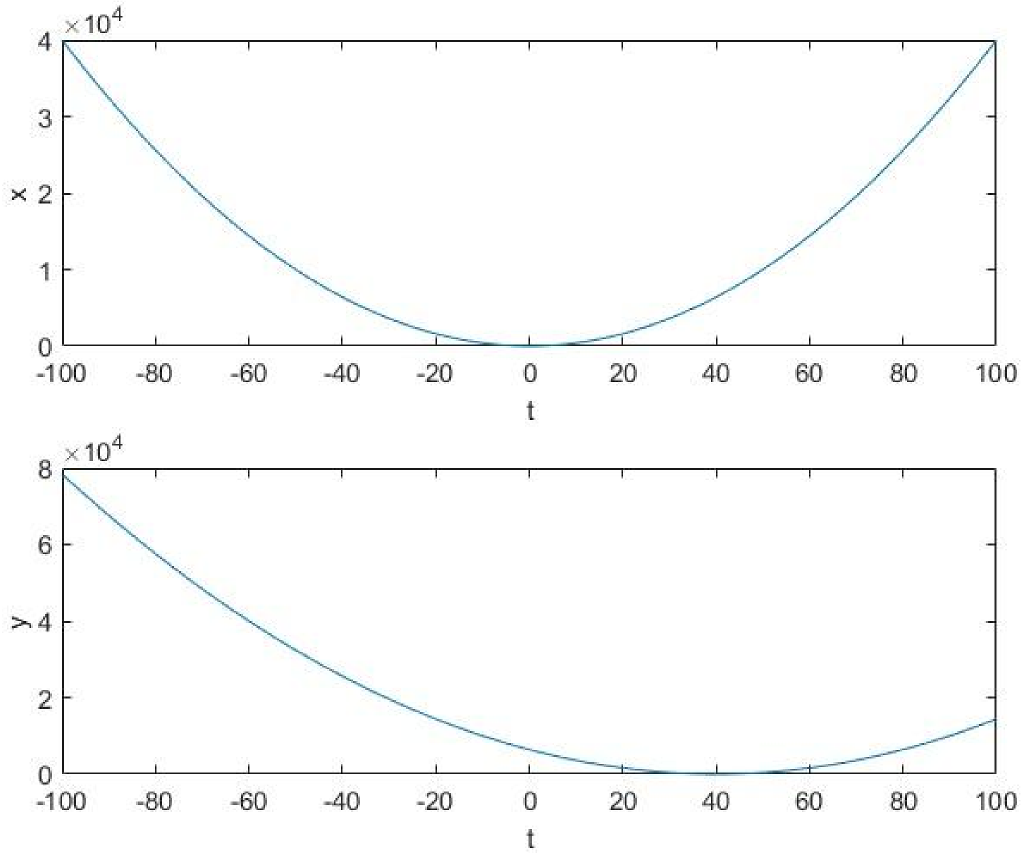
**Output of the program**



**Time delay**
**Matlab Program**

```
t = linspace(-100,100,1000);
t0 = 40;
x = 4*t.*t+3;
subplot(2,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
t1 = t-t0;
y=4*t1.*t1+3;
subplot(2,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
```

**Output of the program**



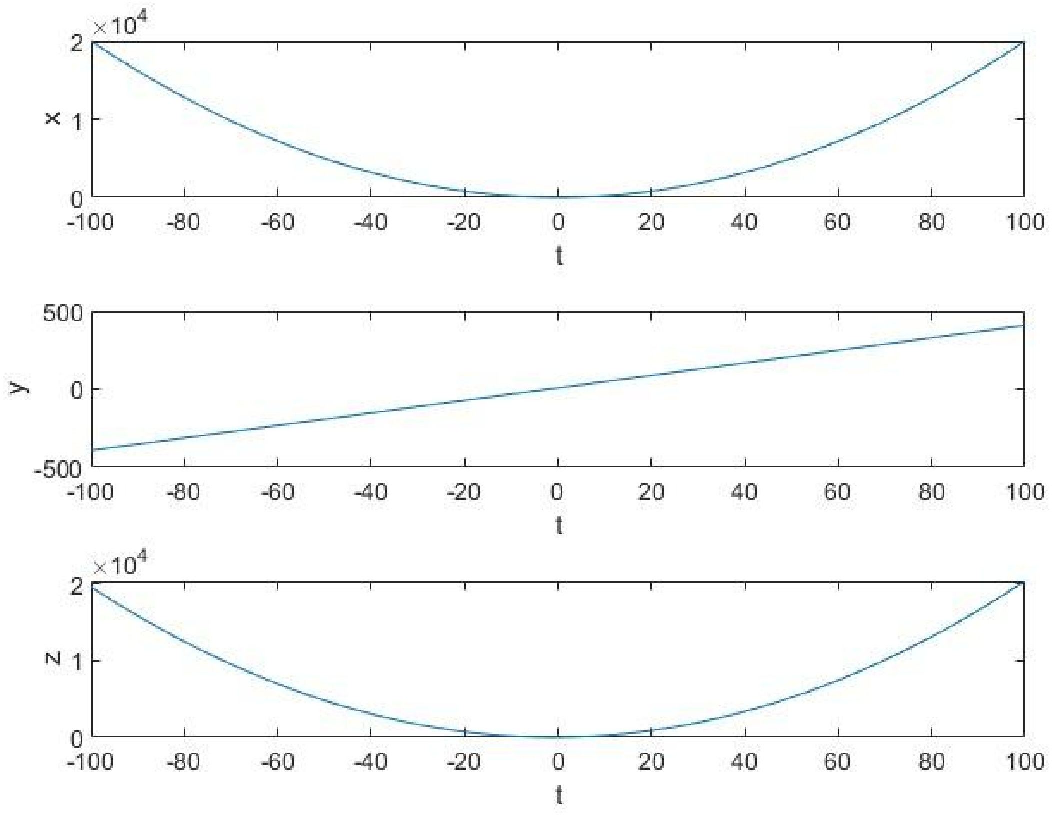
**Write a function in MATLAB to perform addition, subtraction, and multiplication between**

**(a) x(t)= 2t2+3 and y(t)=4t+8**
**(b)x[n]= 7n2+5 and y[n]=4n+3**

**(a) x(t)= 2t2+3 and y(t)=4t+8**
**Addition**
**Matlab Program**

```
t = linspace(-100,100,1000);
x = 2*t.*t+3;
y= 4*t+8;
z= x+y;
subplot(3,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(3,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
subplot(3,1,3);
plot(t,z);
xlabel('t');
ylabel('z');
```

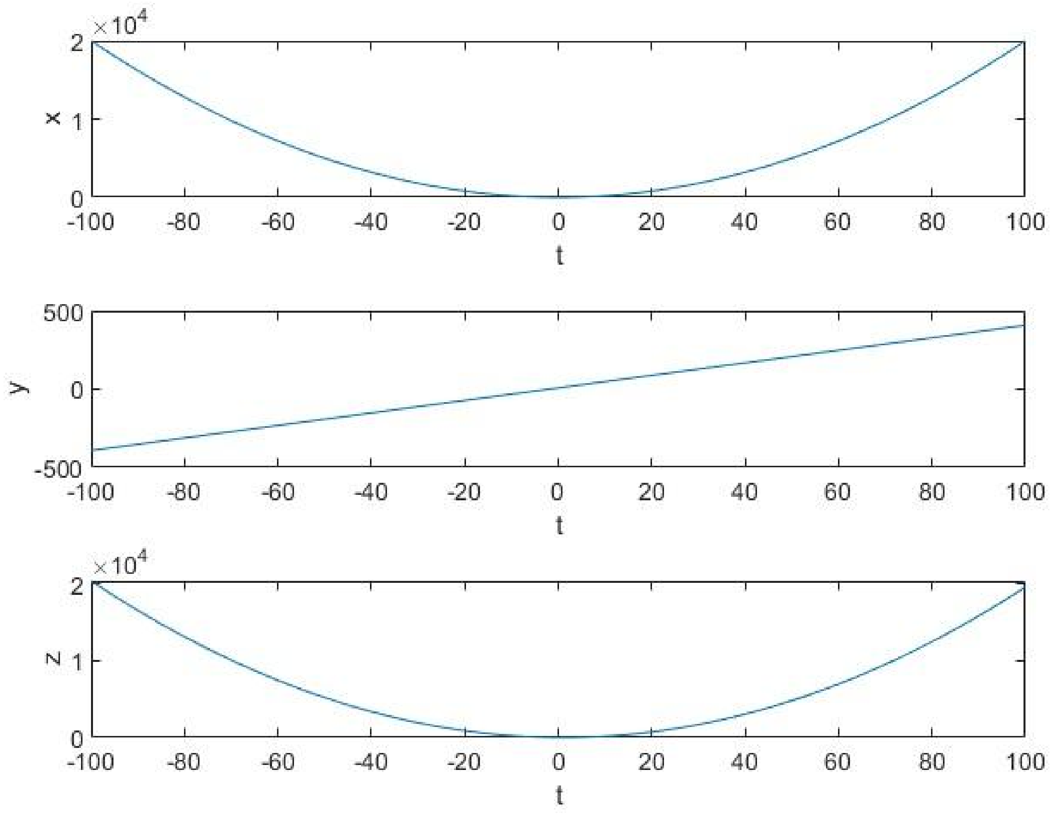
**Output of the program**



**Subtraction**
**Matlab Program**

```
t = linspace(-100,100,1000);
x = 2*t.*t+3;
y= 4*t+8;
z= x-y;
subplot(3,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(3,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
subplot(3,1,3);
plot(t,z);
xlabel('t');
ylabel('z');
```

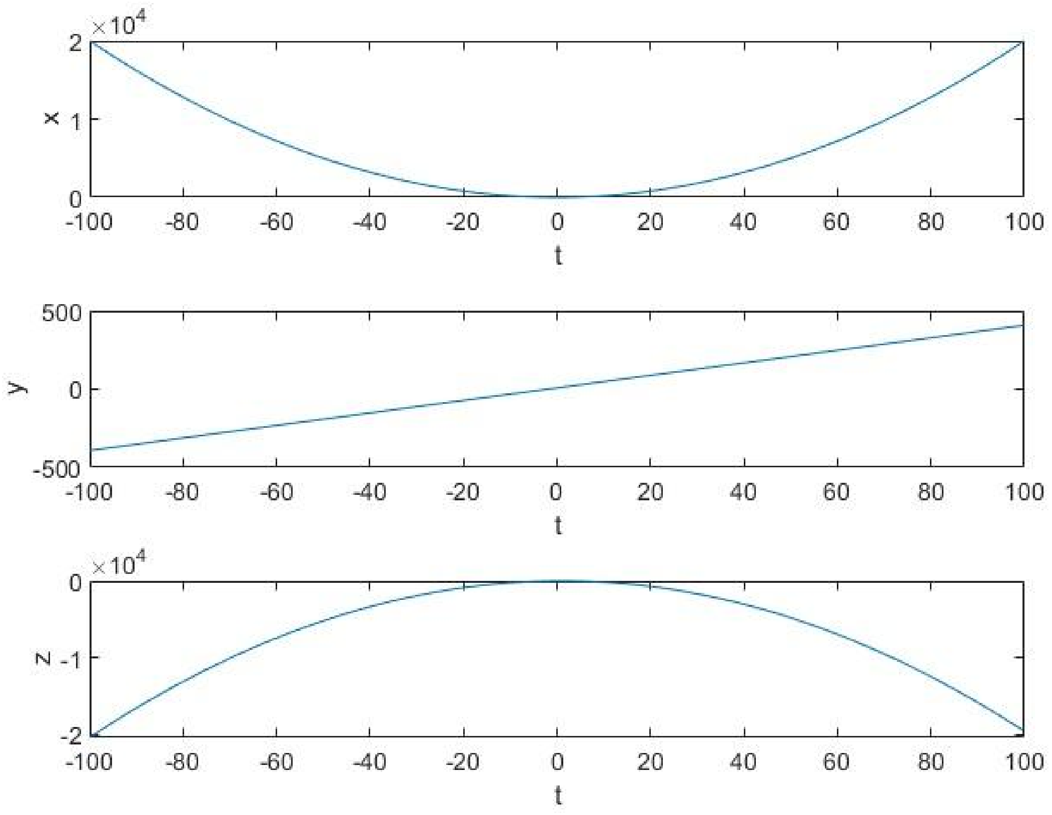
## Output of the progra



## Matlab Program

```
t = linspace(-100,100,1000);
x = 2*t.*t+3;
y= 4*t+8;
z= y-x;
subplot(3,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(3,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
subplot(3,1,3);
plot(t,z);
xlabel('t');
ylabel('z');
```

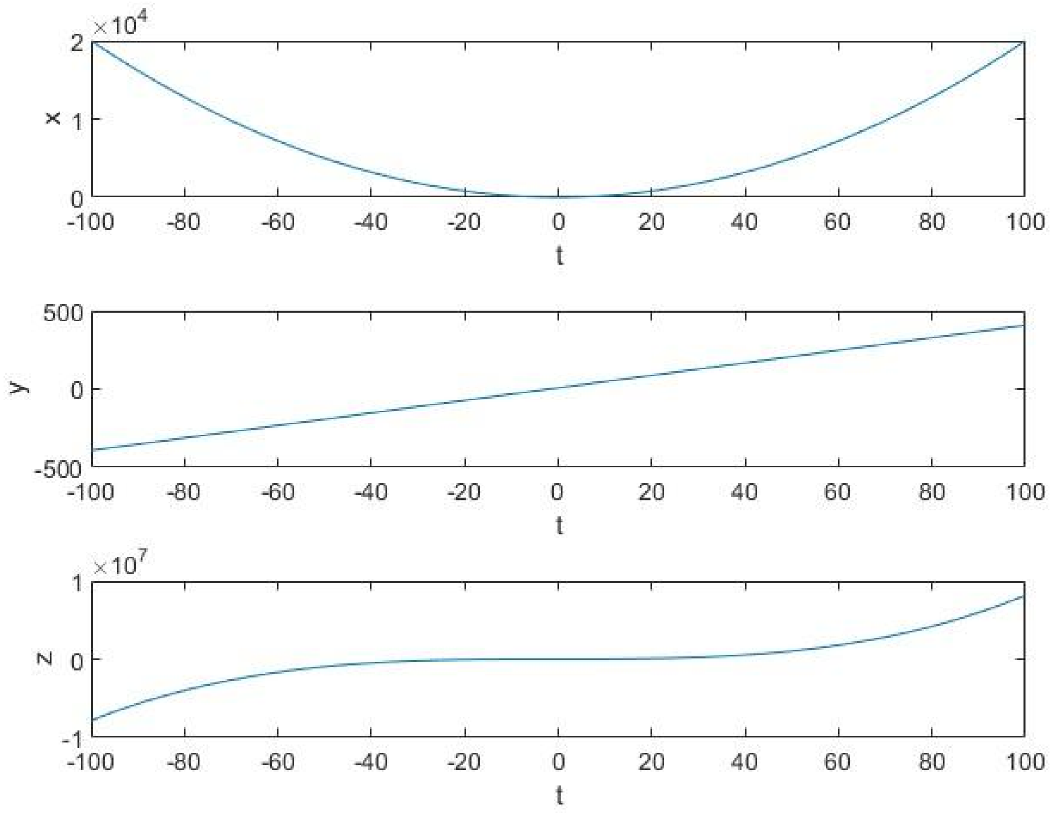
**Output of the program**



**Multiplication**
**Matlab Program**

```
t = linspace(-100,100,1000);
x = 2*t.*t+3;
y= 4*t+8;
z= y.*x;
subplot(3,1,1);
plot(t,x);
xlabel('t');
ylabel('x');
subplot(3,1,2);
plot(t,y);
xlabel('t');
ylabel('y');
subplot(3,1,3);
plot(t,z);
xlabel('t');
ylabel('z');
```

**Output of the program**

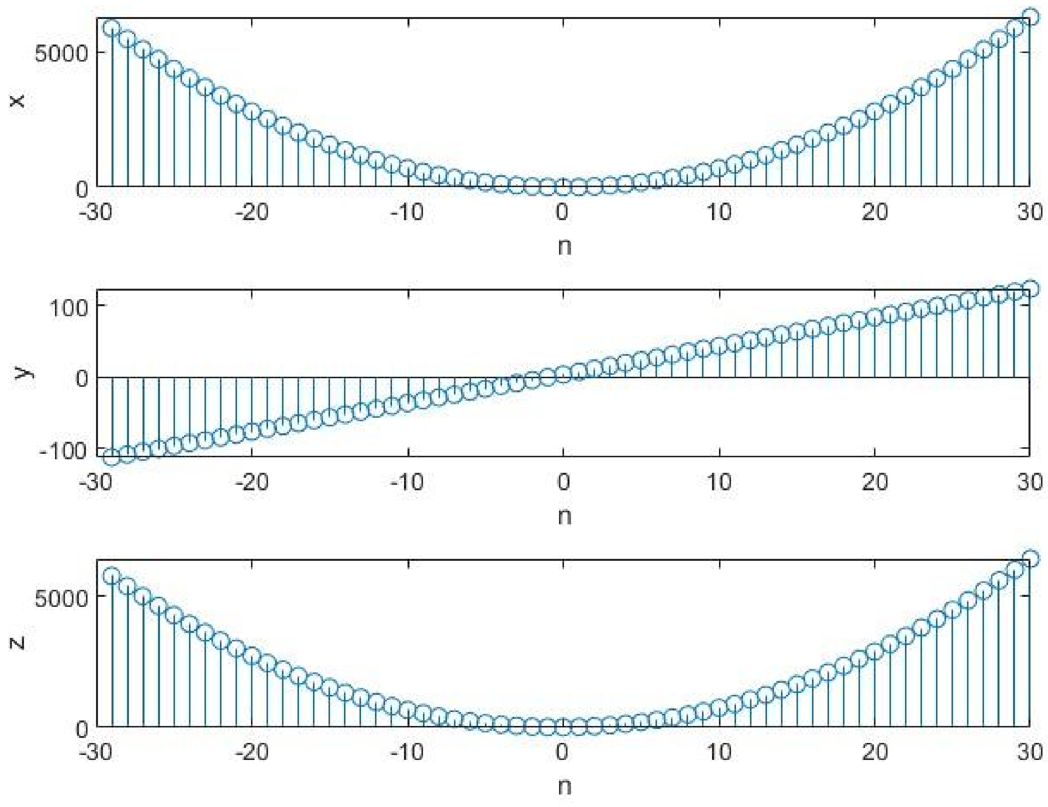


**(b)x[n]= 7n2+5 and y[n]=4n+3**

**Addition**
**Matlab Program**

```
n = -29:30;
x = 7*n.*n+5;
y= 4*n+3;
z= x+y;
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
stem(n,z);
xlabel('n');
ylabel('z');
```

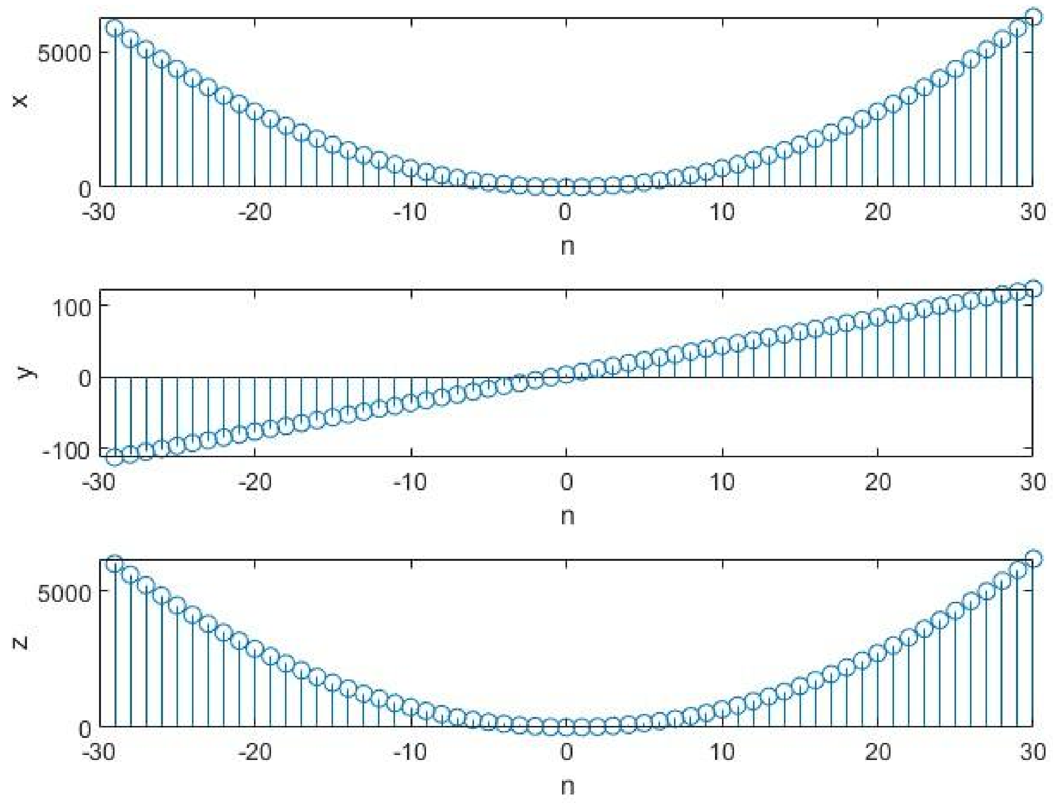
**Output of the program**



**Subtraction**
**Matlab Program**

```
n = -29:30;
x = 7*n.*n+5;
y= 4*n+3;
z= x-y;
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
stem(n,z);
xlabel('n');
ylabel('z');
```

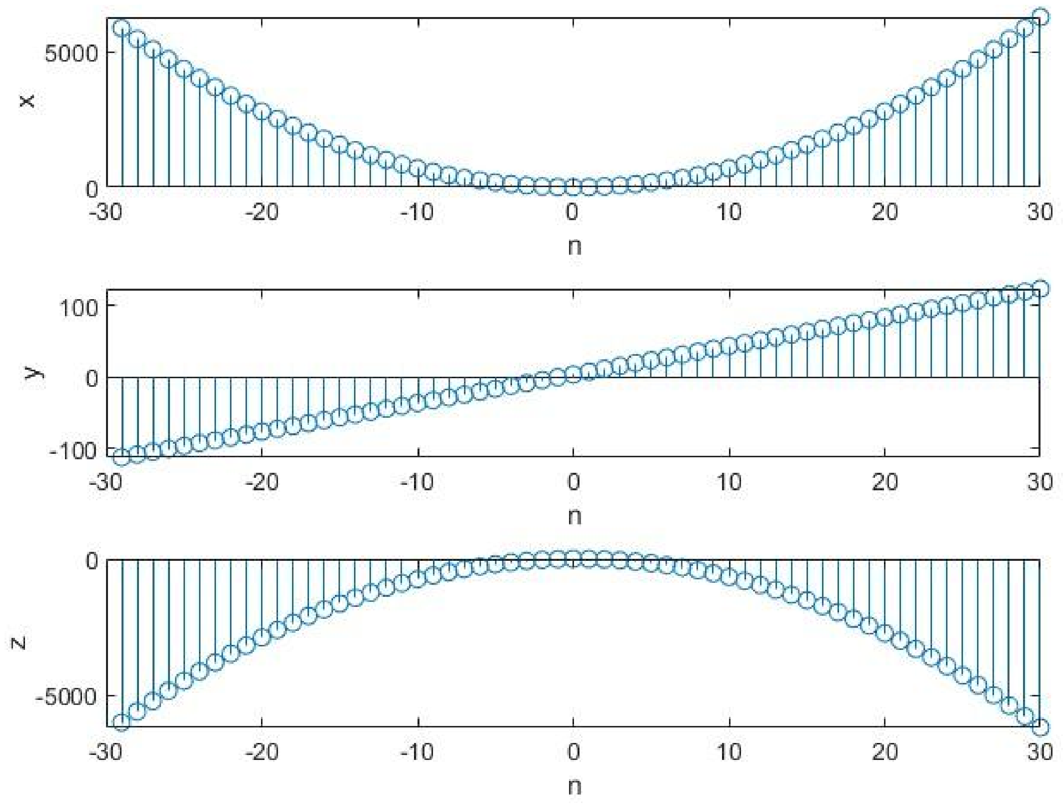
**Output of the program**



**Subtraction**
**Matlab Program**

```
n = -29:30;
x = 7*n.*n+5;
y= 4*n+3;
z= y-x;
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
stem(n,z);
xlabel('n');
ylabel('z');
```

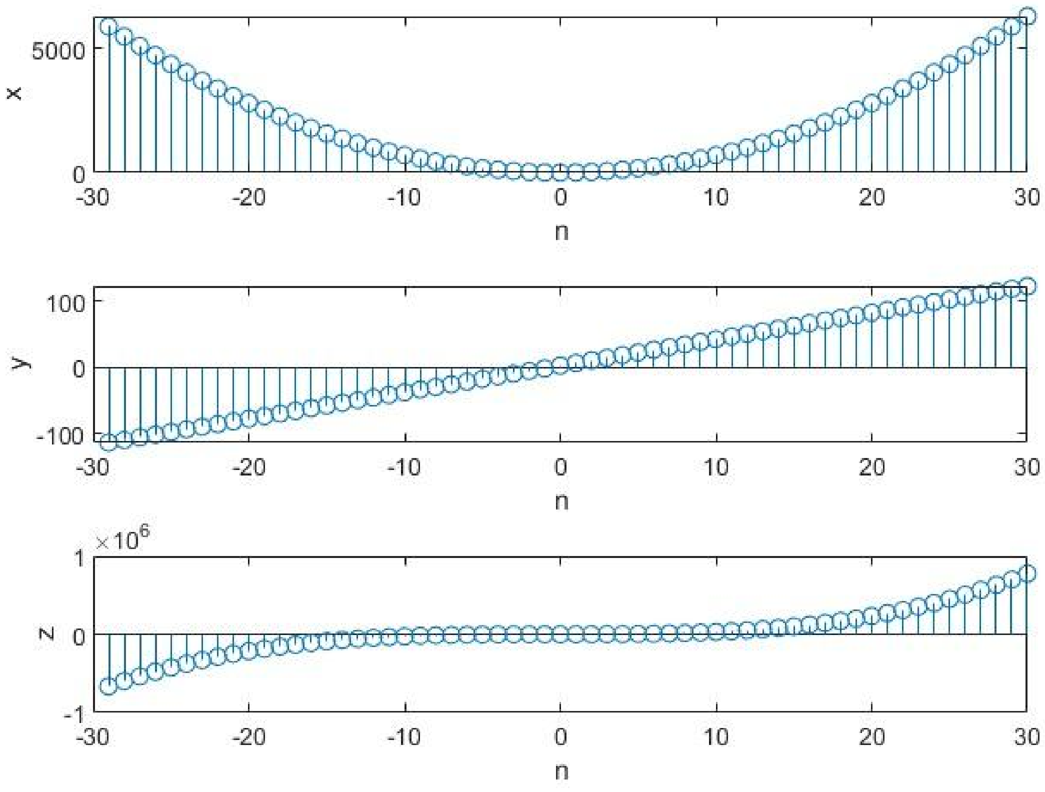
**Output of the program**



**Multiplication**
**Matlab Program**

```
n = -29:30;
x = 7*n.*n+5;
y= 4*n+3;
z= y.*x;
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
stem(n,z);
xlabel('n');
ylabel('z');
```

**Output of the program**



**Result : All the given signals are generated using matlab and various transformations on the given signal are studied.**

# Title:  Linear convolution and circular convolution

**AIM: To perform linear and circular convolution operations on a given pair of sequences .**

**Objective: To perform linear and circular convolution operations on a given pair of sequences using MATlab.**

**Convolution** is the process by which one may compute the overlap of two graphs.
Convolution is also interpreted as the area shared by the two graphs over time.
It is a blend between the two functions as one passes over the other.
So, given two functions F and G,
The convolution of the two is expressed and given by the following mathematical expression:

For continuous time,

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(u).g(t-u)du$$

For  discrete time,

$$f(n) * g(n) = \sum_{k=-\infty}^{\infty} f(k).g(n-k)$$

**Write function in MATLAB for linear convolution of two sequences given below without using built-in MATLAB function.**

**Description**

**Linear convolution** is a mathematical operation done to calculate the output of any **Linear-Time Invariant (LTI)** system given its input and impulse response.
It is applicable for both continuous and discrete-time signals.
We can represent Linear Convolution as
y(n)=x(n)*h(n)

Here, y(n) is the output (also known as convolution sum). x(n) is the input signal, and h(n) is the impulse response of the LTI system.

Graphically, when we perform linear convolution, there is a linear shift taking place.The formula for a linear convolution is given by.

$$\sum_{-\infty}^{\infty} x(k)h(n-k)$$

**Matlab function**

```
%{
The Function to find the Linear Convolution of two sequence
author :Sudip Biswas
%}
function c = convolution(a, b)
u = length(a);
v = length(b)
m =u +v -1;
a=[a,zeros(1,v-1)];
b=[b,zeros(1,u-1)];
for i = 1:1:m
   c(i)=0;
      for j = 1:1:i
              c(i) = c(i)+ a(j)*b((i-j+1));
      end
end
```

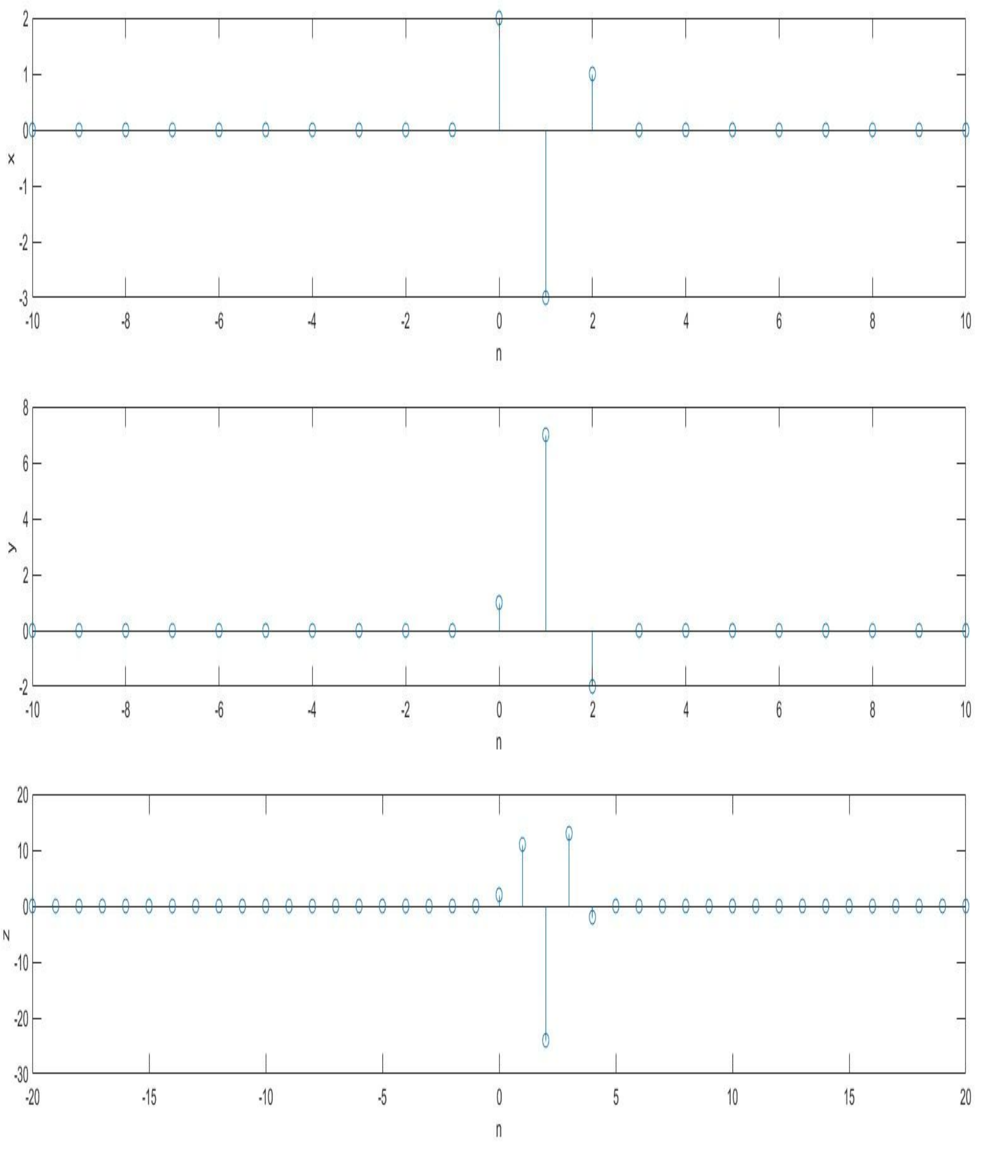
**a.**

$$x(n) = \begin{cases} 2 & n=0 \\ -3 & n=1 \\ 1 & n=2 \\ 0 & \text{otherwise} \end{cases} \qquad y(n) = \begin{cases} 1 & n=0 \\ 7 & n=1 \\ -2 & n=2 \\ 0 & \text{otherwise} \end{cases}$$

**Matlab Program**

```
n = -10:10;
n1 = -20:20;
x = [0 0 0 0 0 0 0 0 0 0 2 -3 1 0 0 0 0 0 0 0 0 ];
y=  [0 0 0 0 0 0 0 0 0 0 1 7 -2 0 0 0 0 0 0 0 0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = convolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

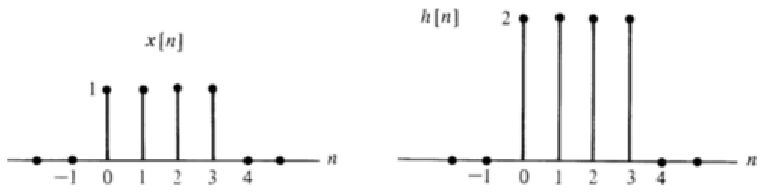
**Output of Program**



**Result:**

**[** 0 0 **<u>2</u>** 11 -24 13 -2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**]**

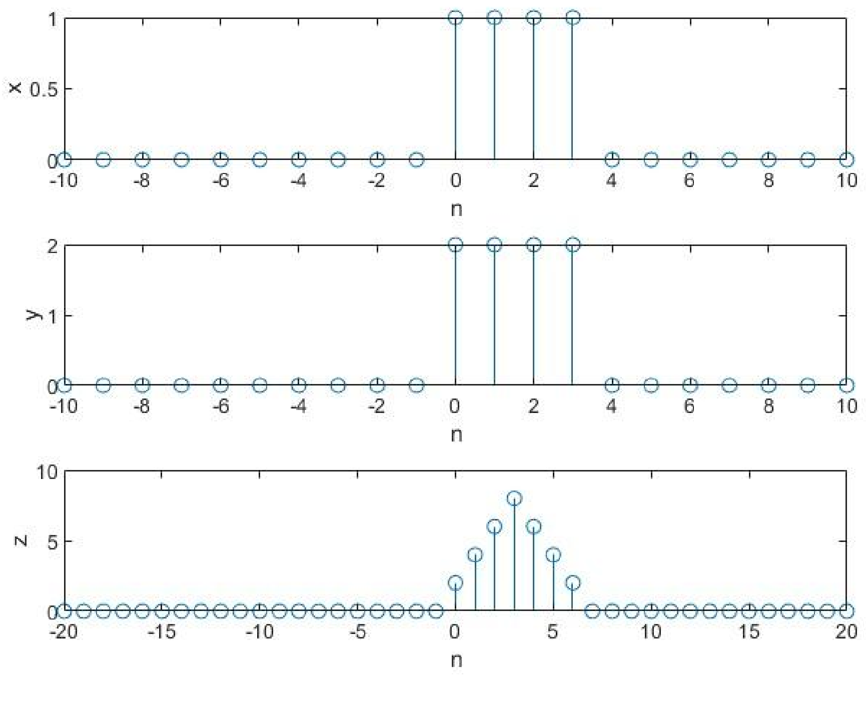
## b.



**Matlab Program**

```
n = -10:10;
n1 = -20:20;
x = [0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0 ];
y=  [0  0  0  0  0  0  0  0  0  0  2  2  2  2  0  0  0  0  0  0  0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = convolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

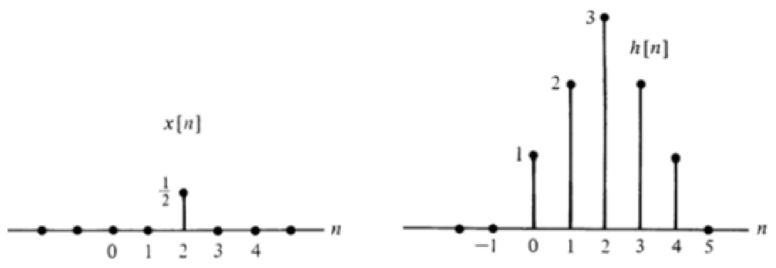
**Output of Program**



**Result:**

[0 0 <u>**2**</u> 4 6 8 6 4 2 0 0 0]

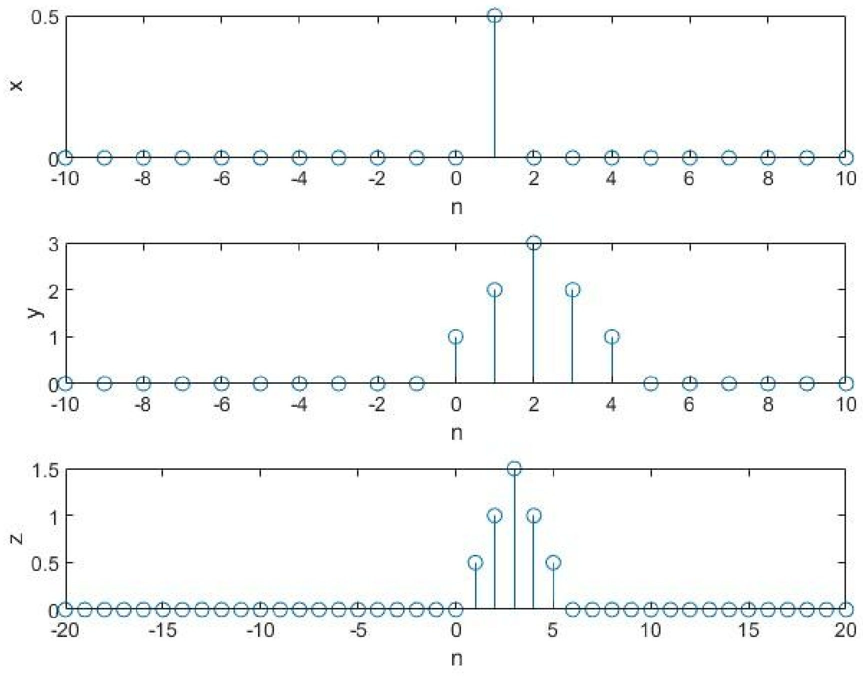
**c.**



**Matlab Program**

```
n = -10:10;
n1 = -20:20;
x = [0 0 0 0 0 0 0 0 0 0 0 .5 0 0 0 0 0 0 0 0 0 ];
y=  [0 0 0 0 0 0 0 0 0 0 1  2 3 2 1 0 0 0 0 0 0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = convolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

**Output of Program**



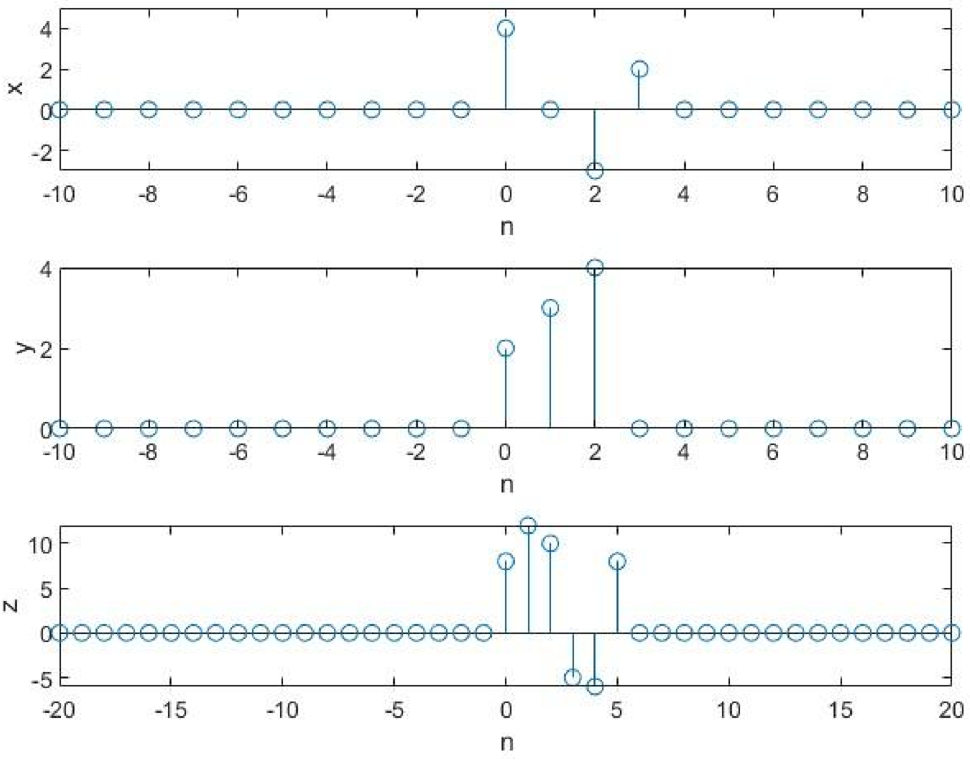
**Result:**
**[0 <u>0</u> 0.5 1 1.5 1 0.5 0 0 0]**

**d.** X[n]=4δ[n]-3δ[n-2]+2δ[n-3]; h[n]={2,4,3}

**Matlab Program**

```
n = -10:10;
n1 = -20:20;
x = [0 0 0 0 0 0 0 0 0 0 4 0 -3 2 0 0 0 0 0 0 0 ];
y=  [0 0 0 0 0 0 0 0 0 0 2 3  4 0 0 0 0 0 0 0 0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = convolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

**Output of Program**



**Result:**
**[0 <u>8</u> 12 10 -5 -6 8 0 0 0]**

**Write a function in MATLAB for circular convolution of two sequences given below without using built–in function.**

**Description**

**Circular convolution** is essentially the same process as linear convolution. Just like linear convolution, it involves the operation of folding a sequence, shifting it, multiplying it with another sequence, and summing the resulting products. However, in circular convolution, the signals are all periodic. Thus the shifting can be thought of as actually being a rotation. Since the values keep repeating because of the periodicity. Hence, it is known as circular convolution.

Circular convolution is also applicable for both continuous and discrete-time signals.

We can represent Circular Convolution as

y(n)=x(n)⊕h(n)

Here y(n) is a periodic output, x(n) is a periodic input, and h(n) is the periodic impulse response of the LTI system.

**Matlab Function**

```
%{
The Function to find the Circular Convolution of two sequence
author :Sudip Biswas
%}
function c = circonvolution(a, b)
u = length(a);
v = length(b)
n =max(u,v);
if(u>v)
b=[b,zeros(1,u-v)];
else
a=[a,zeros(1,v-u)];
end
for i =0:n-1
   c(i+1)=0;
   for j = 0:n-1
       k = mod((i-j),n);
       c(i+1) = c(i+1) + a(j+1)*b(k+1);
   end
end
```

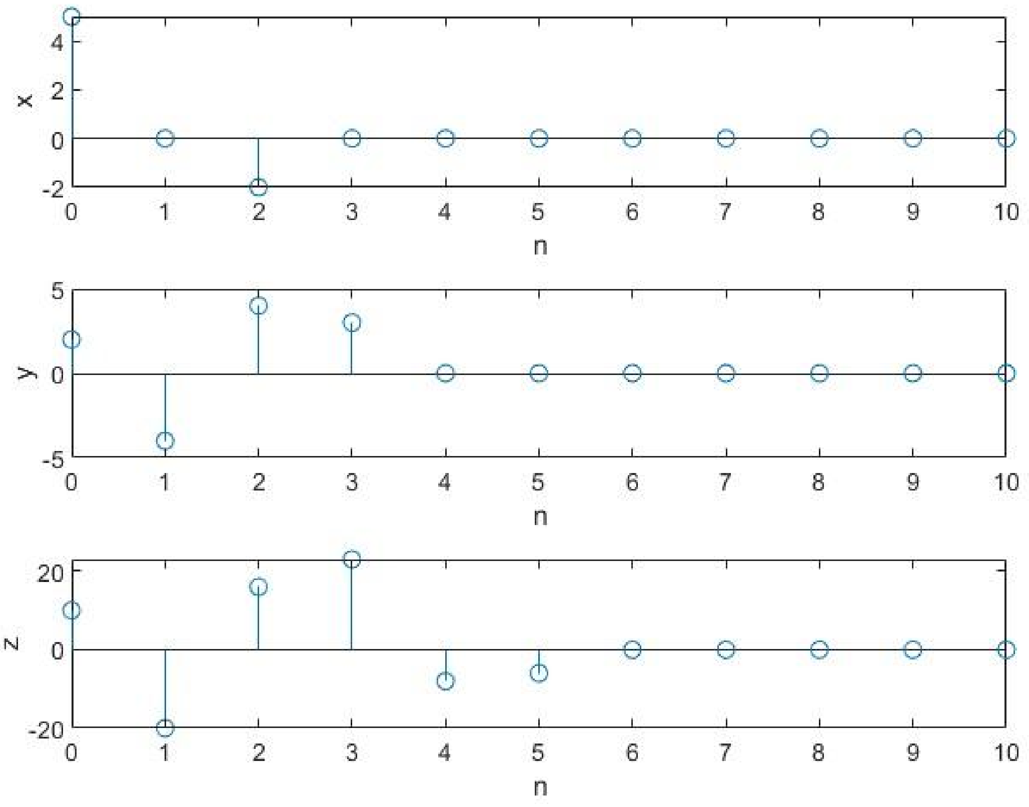
(a) X1[n]={5,0,-2}; X2[n]={2,-4,4,3}

**Matlab Program**

```
n = 0:10;
n1 = 0:10;
x = [5  0 -2 0 0 0 0 0 0 0 0 ];
y=  [2 -4  4 3 0 0 0 0 0 0 0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
```

```
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = circonvolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

**Output of program**



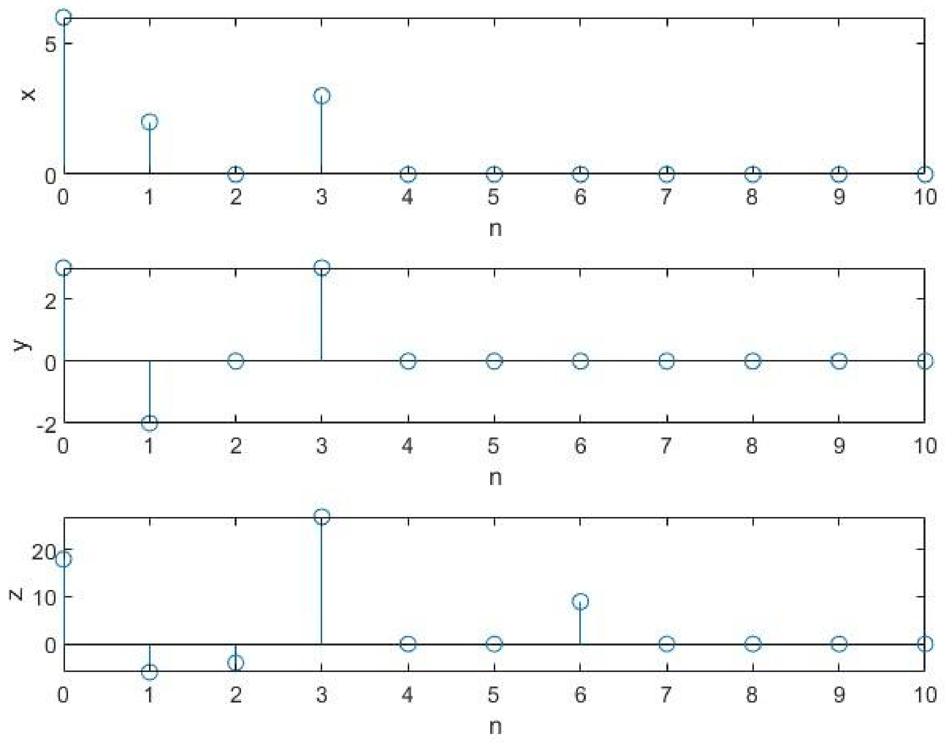
**Result:**
**[<u>10</u>** -20 16 23 -8 -6 0 0 0 0 0**]**

**(b) X1[n]={6,2,0,3}; X2[n]=3δ[n]-2δ[n-1]+3δ[n-3]**

**Matlab Program**

```
n = 0:10;
n1 = 0:10;
x = [6  2  0 3 0 0 0 0 0 0 0 ];
y=  [2 -2  0 3 0 0 0 0 0 0 0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = circonvolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

**Output of program**



**Result:**

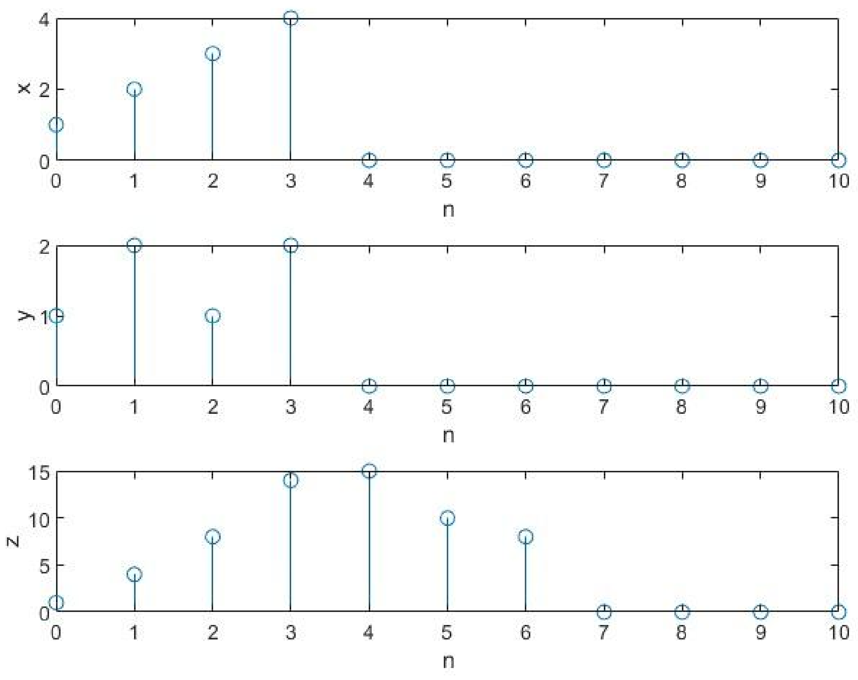
**[<u>18</u>   -6   -4   27   0   0   9   0   0   0   0]**

**(c) X1[n] = {1, 2, 3, 4, 0, 0, 0}; X2[n] = {1, 2, 1, 2, 0, 0, 0}**

**Matlab Program**

```
n = 0:10;
n1 =0:10;
x = [1 2  3 4 0 0 0 0 0 0 0 ];
y=  [1 2  1 2 0 0 0 0 0 0 0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = circonvolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

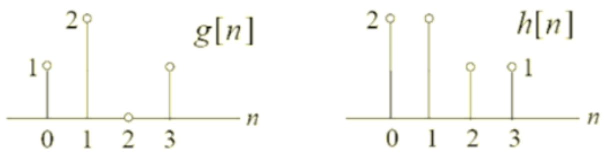
**Output of program**



**Result:**

**[1** 4 8 14 15 10 8 0 0 0 **0]**

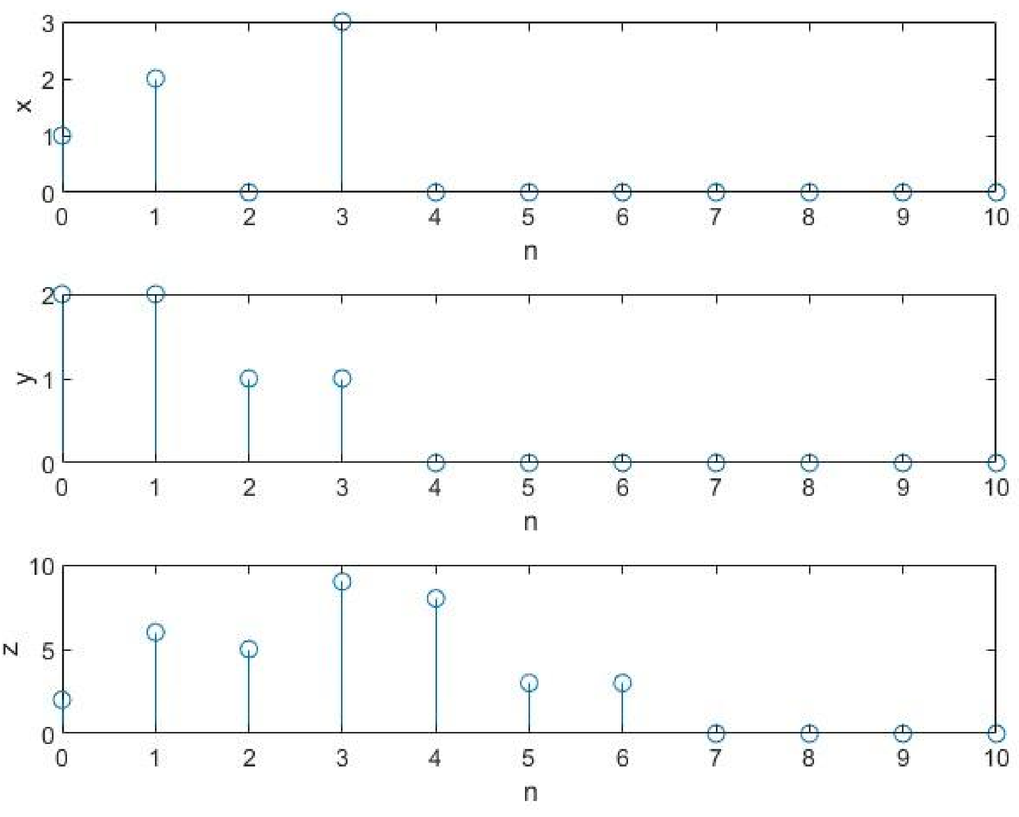
**d.**



**Matlab Program**

```
n = 0:10;
n1 = 0:10;
x = [1 2  0 3 0 0 0 0 0 0 0 ];
y=  [2 2  1 1 0 0 0 0 0 0 0 ];
subplot(3,1,1);
stem(n,x);
xlabel('n');
ylabel('x');
subplot(3,1,2);
stem(n,y);
xlabel('n');
ylabel('y');
subplot(3,1,3);
z = circonvolution(x,y);
stem(n1,z);
xlabel('n');
ylabel('z');
```

**Output of program**



**Result: [2    6    5    9    8    3    3    0    0    0    0]**