Title: Linear convolution and circular convolution using DFT and IDFT

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

Objective: To perform linear and circular convolution operations on a given pair of sequences using DFT and IDFT in 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,

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

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.

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

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
$y(n)=x(n) \oplus h(n)$
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.

Write a function in MATLAB to calculate DFT and IDFT of a sequence without using a built–in function.

DFT : The discrete Fourier transform (DFT) is the primary transform used for numerical computation in digital signal processing. It is very widely used for spectrum analysis, fast convolution, and many other applications. The DFT transforms N discrete-time samples to the same number of discrete frequency samples, and is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\left(i\frac{2\pi nk}{N}\right)}$$

Matlab function to perform DFT

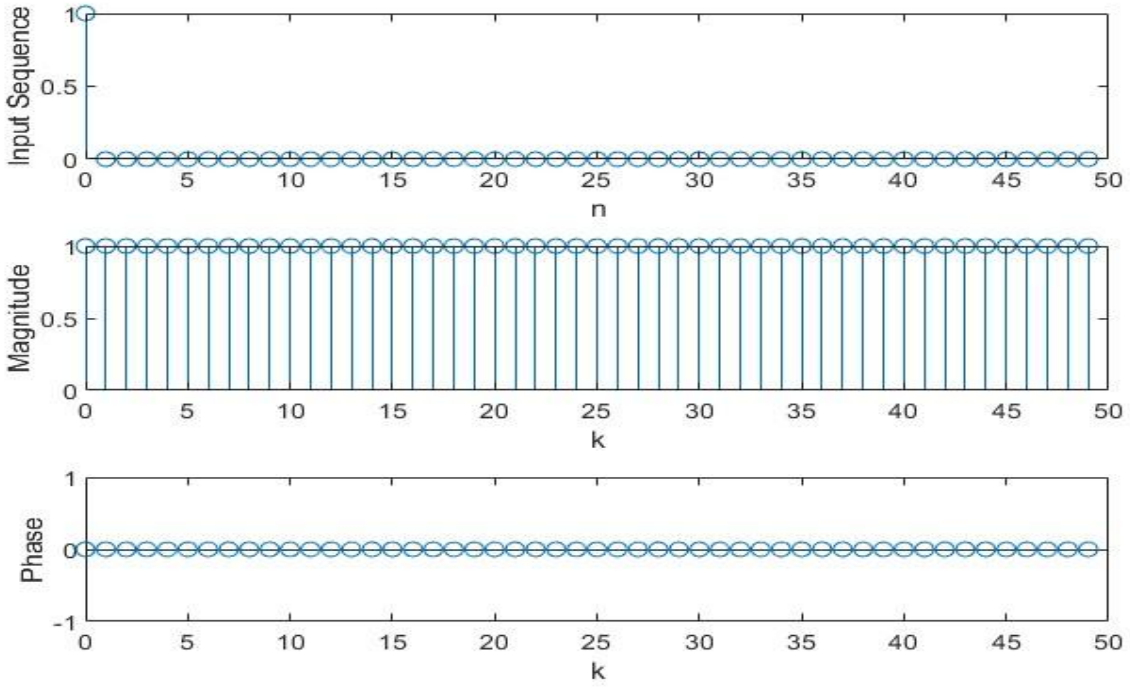
```
%{
      To Calculate the DFT of given sequence
      Author : Sudip Biswas
%}
function xk = DFTFun(x)
N = length(x);
n1=0:N-1;
for k =1:N
   xk(k)=0;
   for n=1:N
       W = (-1i*2*pi*(n-1)*(k-1))/N;
       xk(k) = xk(k)+ x(n)*exp(W);
   end
end
subplot(3,1,1);
stem(n1,x);
ylabel("Input Sequence");
xlabel("n");
subplot(3,1,2);
stem(n1,abs(xk));
ylabel("Magnitude");
xlabel("k");
subplot(3,1,3);
stem(n1,angle(xk));
ylabel("Phase");
xlabel("k");
```

Test Function

a) Unit Impulse

```
% Unit Impulse
x = [1,zeros(1,49)];
DFTFun(x);
```

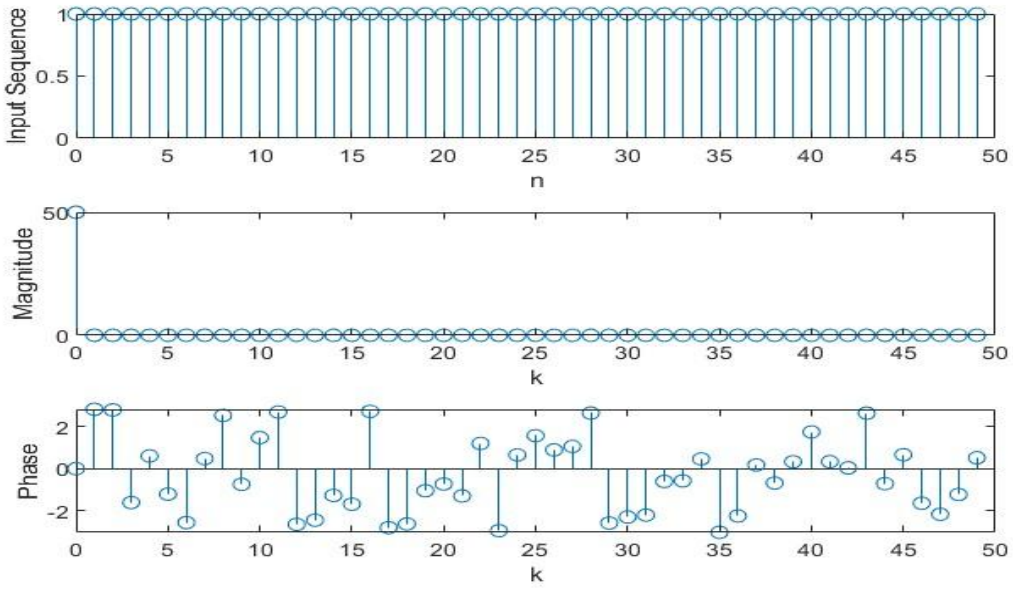
Output of Program



b) Unit step

```
% Unit Step
x = ones(1,50);
DFTFun(x);
```

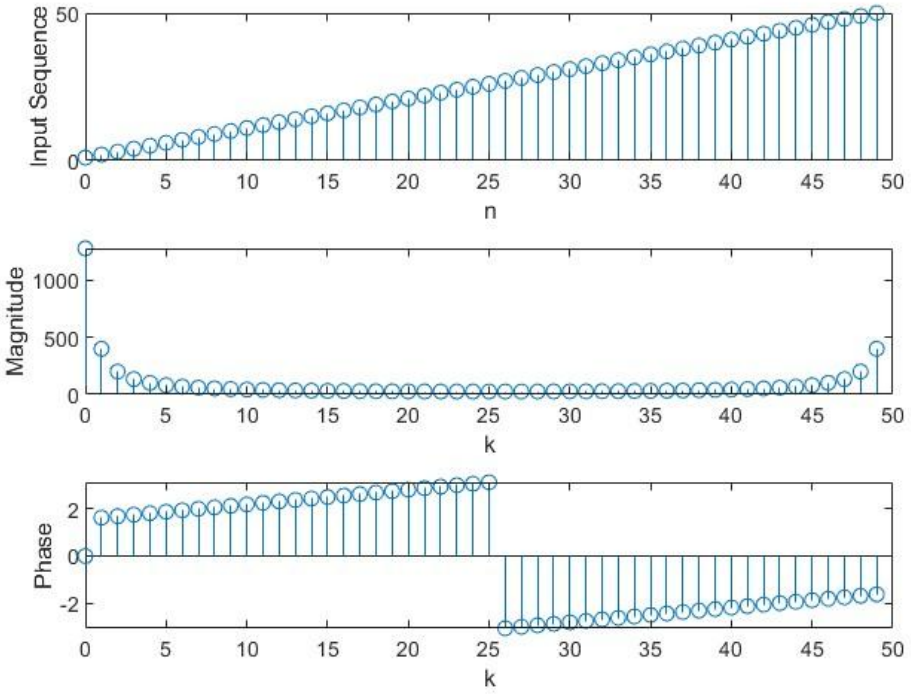
Output of Program



c) Unit ramp

```
% Unit Ramp
x = [1:50];
DFTFun(x);
```

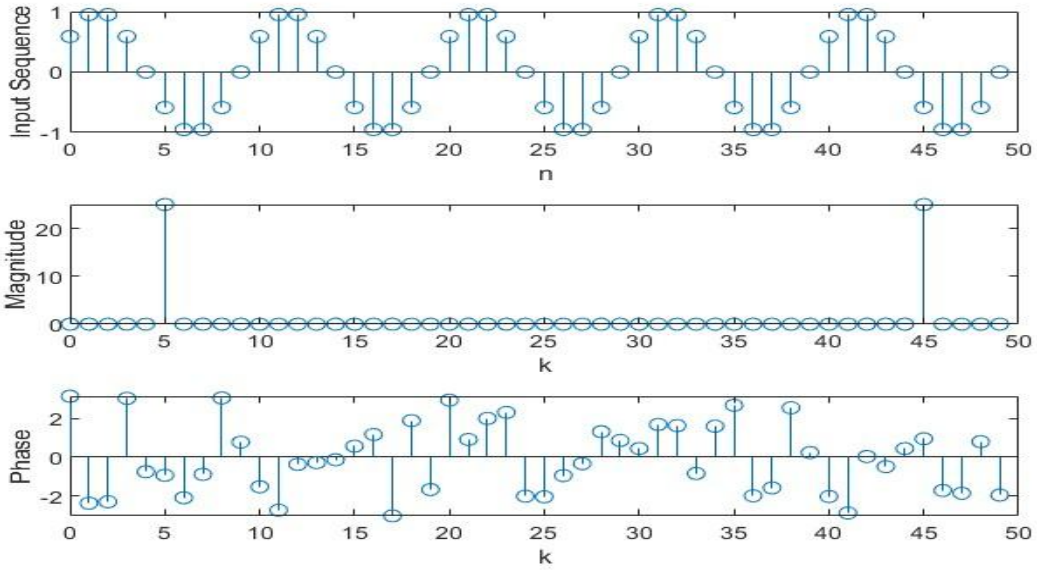
Output of Program



d) Sine

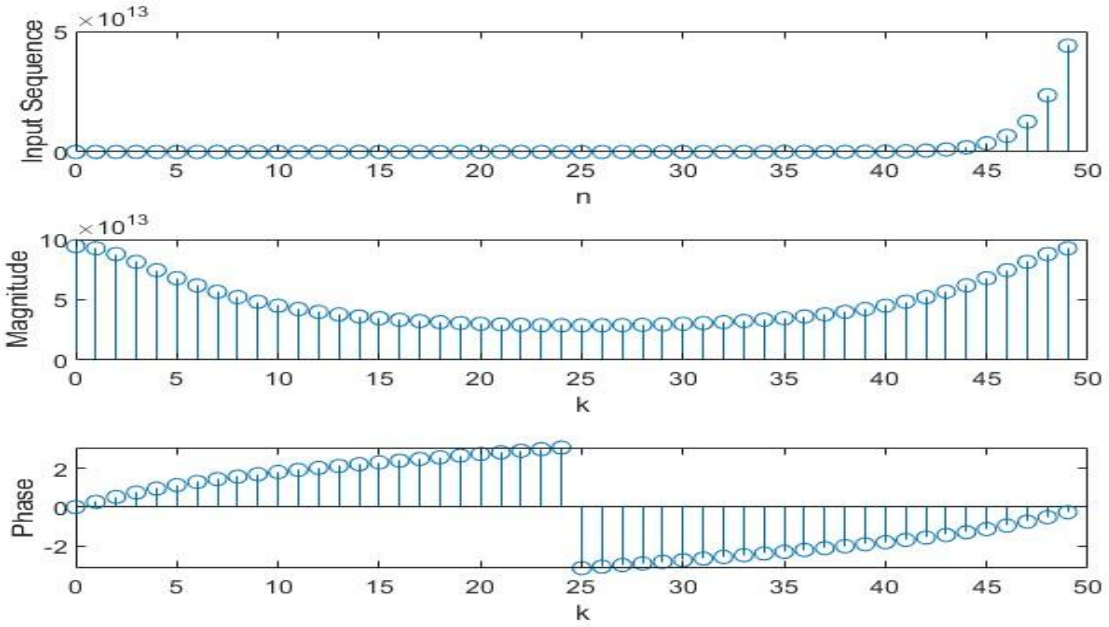
```
% sine
x = 1:50;
y=sin(2*pi*x/10);
DFTFun(y);
```

Output of Program



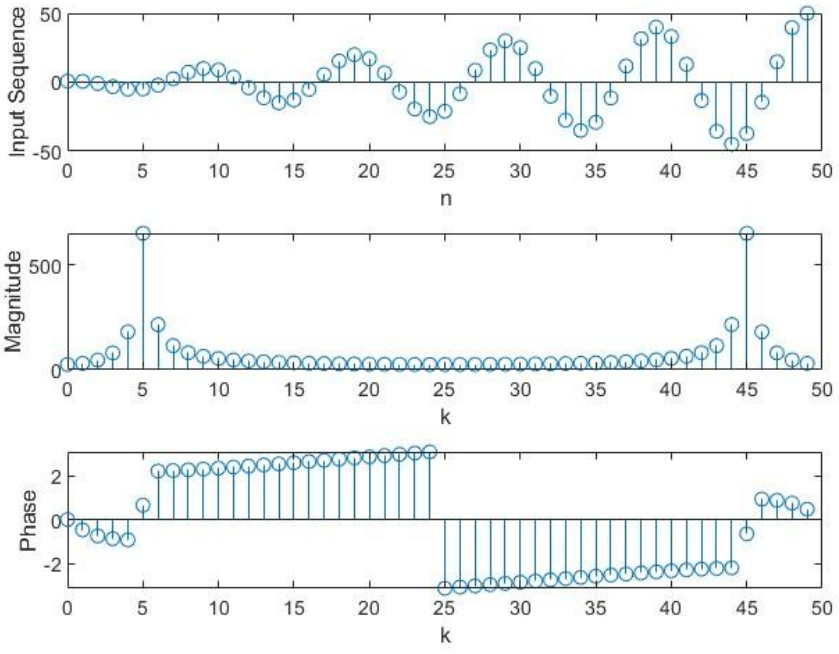
e) Exponential

```
% exponential
x = 1:50;
y=exp(2*pi*x/10);
DFTFun(y);
```



f) Modulation with Ramp

```
% modulation with ramp
x = 1:50;
y=cos(2*pi*x/10);
z=x.*y;
DFTFun(z);
```



The **Inverse DFT (IDFT)** transforms N discrete-frequency samples to the same number of discrete-time samples. The IDFT has a form very similar to the DFT,

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i \frac{2\pi nk}{N}}$$

Matlab function to perform IDFT

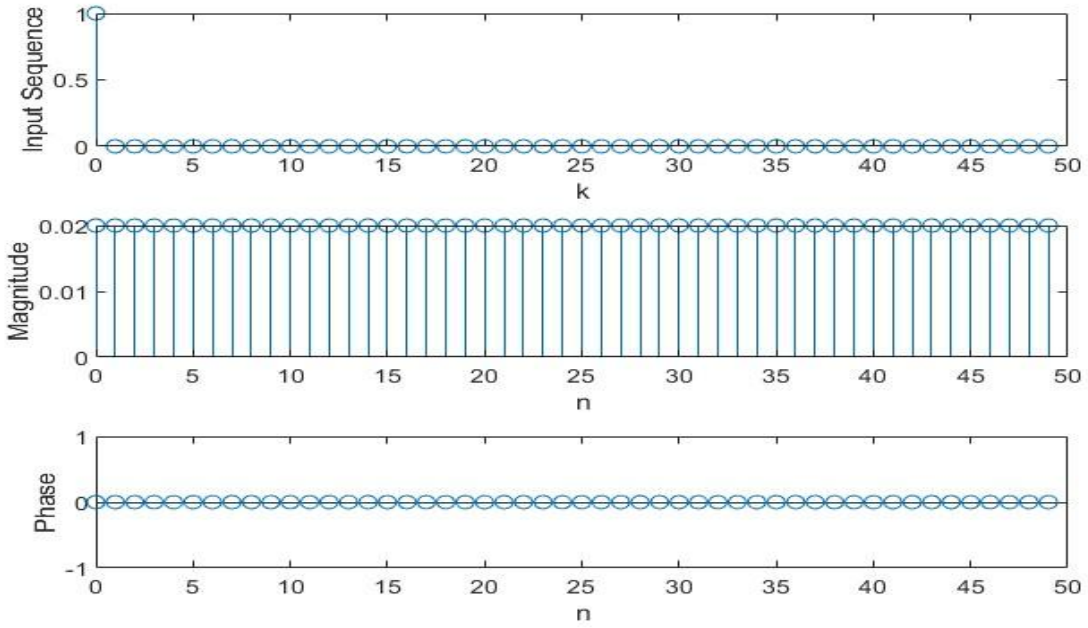
```
%{
To Calculate the inverse DFT of given sequence
Author : Sudip Biswas
%}
function xn = iDFTFun(x)
N = length(x);
idflen=0:N-1;
for k =1:N
   xn(k)=0;
   for n=1:N
       W = (1i*2*pi*(n-1)*(k-1))/N;
       xn(k) = xn(k)+ x(n)*exp(W);
   end
   xn(k) = xn(k)/N;

end
subplot(3,1,1);
stem(idflen,x);
ylabel("Input Sequence");
xlabel("k");
subplot(3,1,2);
stem(idflen,abs(xn));
ylabel("Magnitude");
xlabel("n");
subplot(3,1,3);
stem(idflen,angle(xn));
ylabel("Phase");
xlabel("n");
```

Test Function

(a) Unit Impulse

```
% Unit Impulse
x = [1,zeros(1,49)];
iDFTFun(x);
```

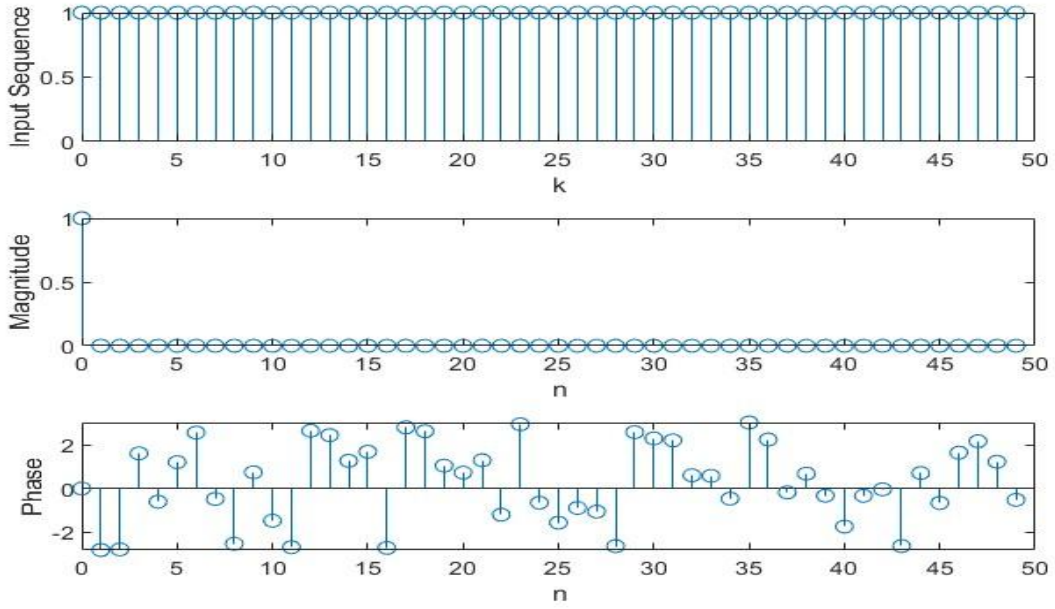
Output of Program



(b) Unit Impulse

```
% Unit Step
x = ones(1,50);
iDFTFun(x);
```

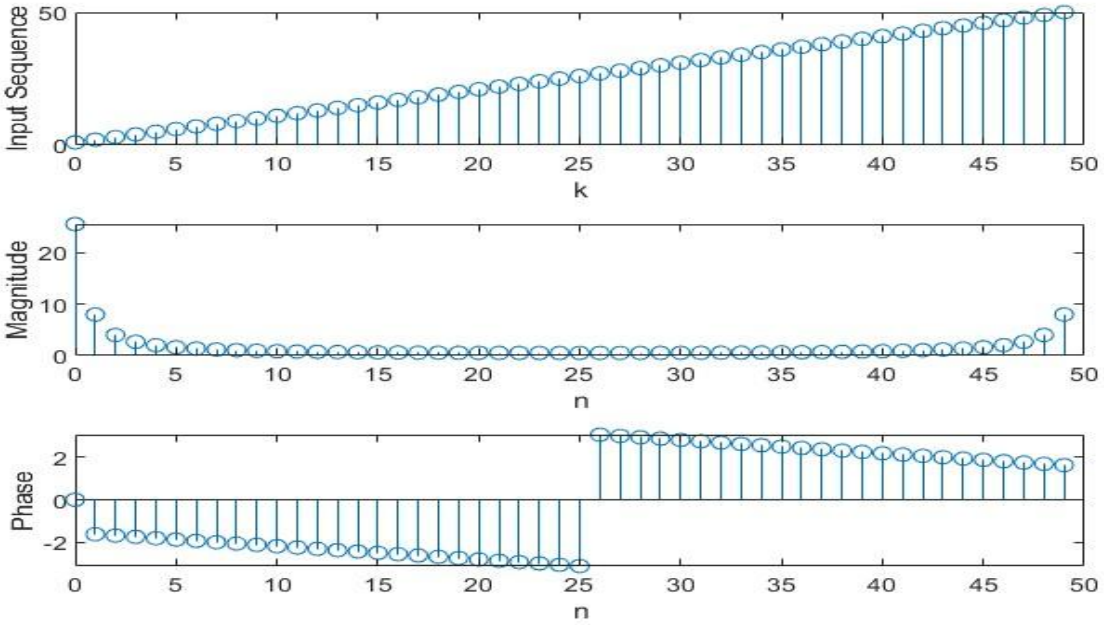
Output of Program



(c) Unit Ramp

```
% Unit Ramp
x = [1:50];
iDFTFun(x);
```

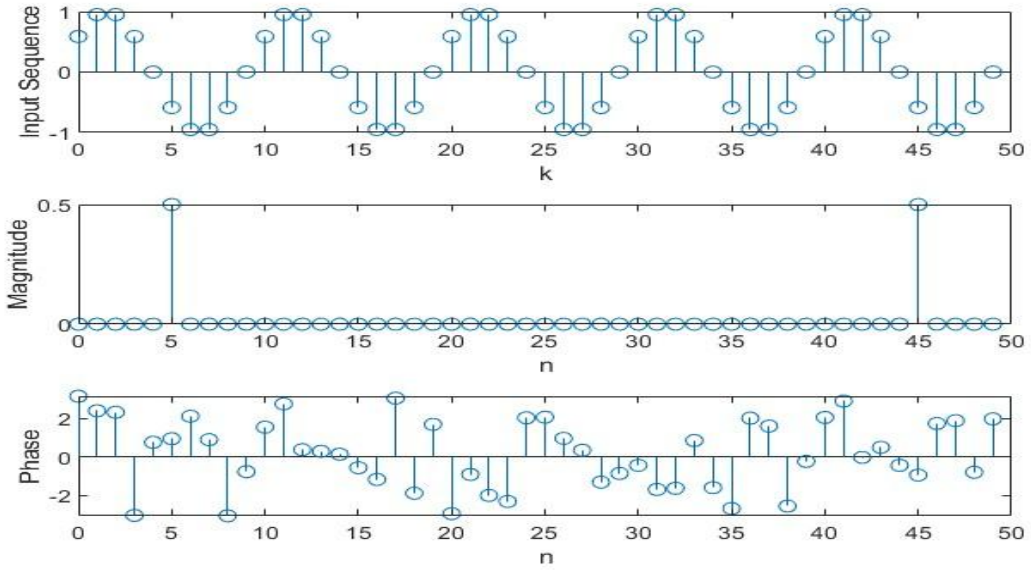
Output of Program



(d) Sin

```
%sin
x = 1:50;
y=sin(2*pi*x/10);
iDFTFun(y);
```

Output of Program



Use MATLAB code written for (1) to calculate linear convolution of the following sequence.

Let x1(n) and x2(n) be the input sample
Then let X1(K) be the DFT of x1(n) and X2(K) be the DFT of x2(n) then
We Know that multiplication in frequency domain is convolution in time domain
Therefore

X3(K) = X1(K) x X2(K)

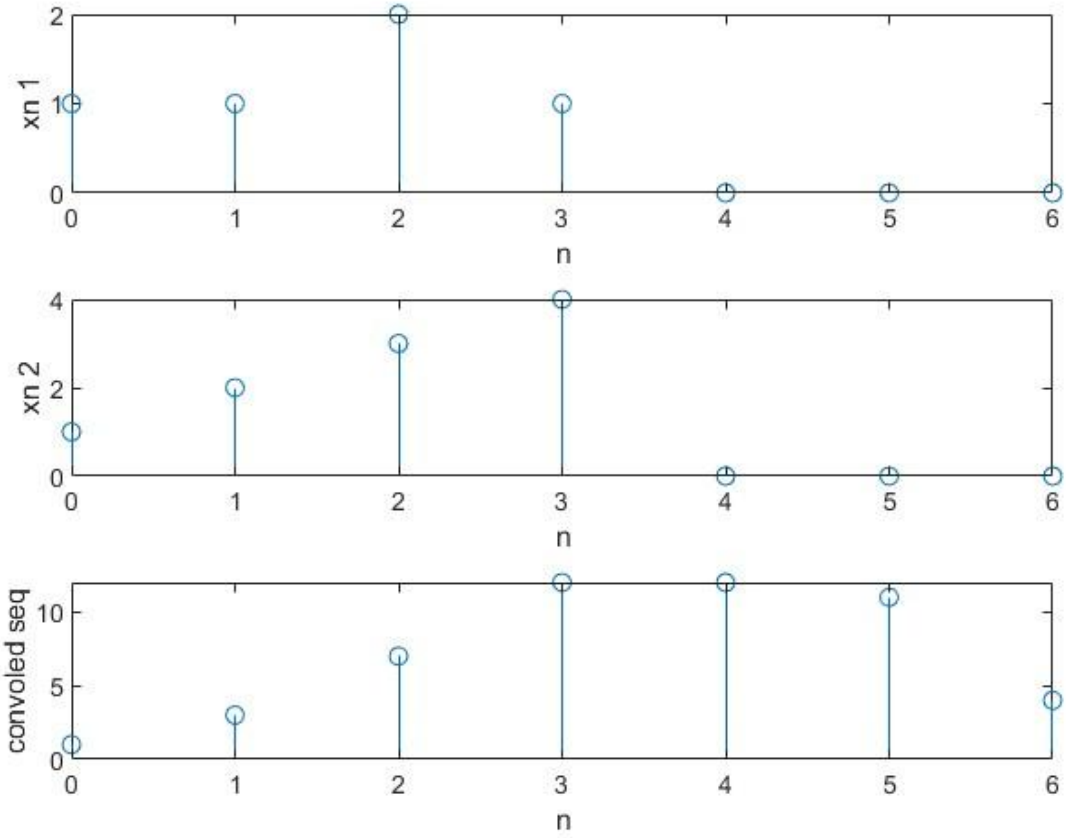
IDFT of X3(K) gives the convolution of x1(n) and x2(n)
Let M be the length of x1(n)
And N be the length of x2(n)
For Linear convolution Length = M+N-1

(a) x1[n]=[1,1,2,1] and x2 [n]=[1,2,3,4]

Matlab Program

```
x1=[1,1,2,1];
x2=[1,2,3,4];
len1 = length(x1);
len2 = length(x2);
len = len1+len2-1;
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
xlabel("n");
ylabel("convoled seq");
```

Output of Program



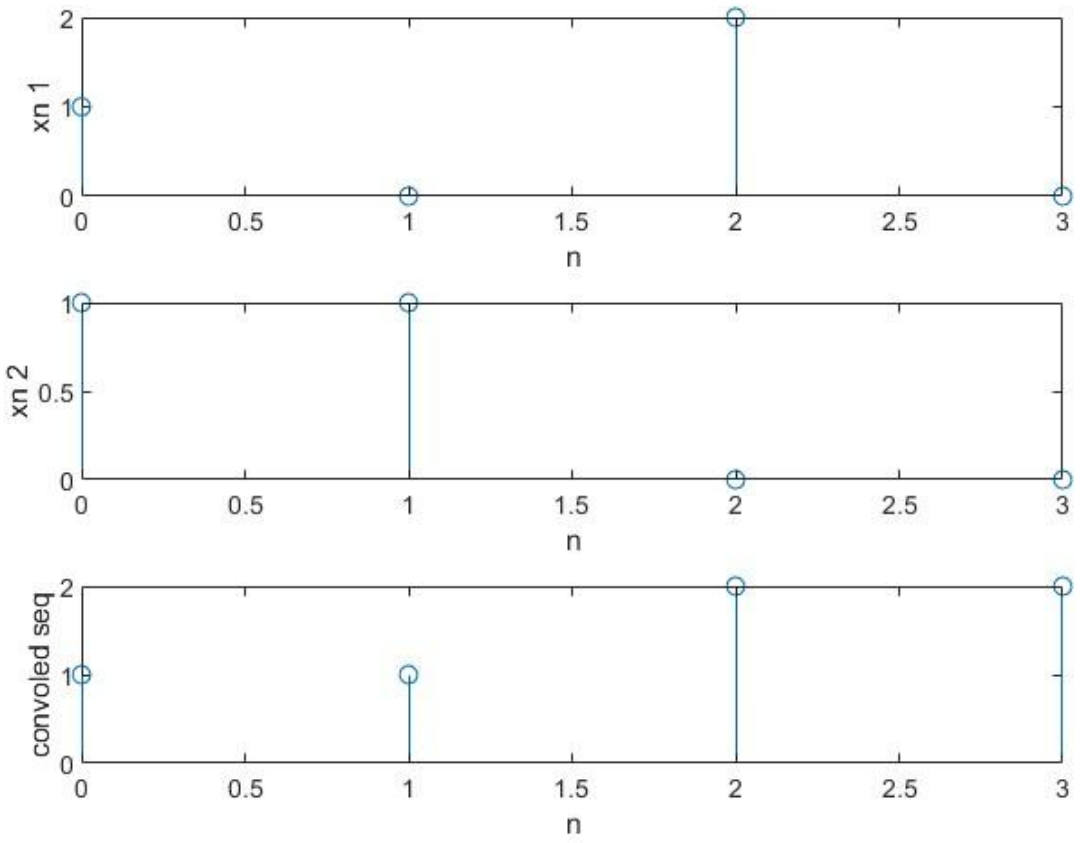
(b) x1[n]=[1,0,2] and x2[n]=[1,1]

Matlab Program

```
x1=[1,0,2] ;
x2=[1,1];
len1 = length(x1);
len2 = length(x2);
len = len1+len2-1;
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
```

```
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
xlabel("n");
ylabel("convoled seq");
```

Output of Program



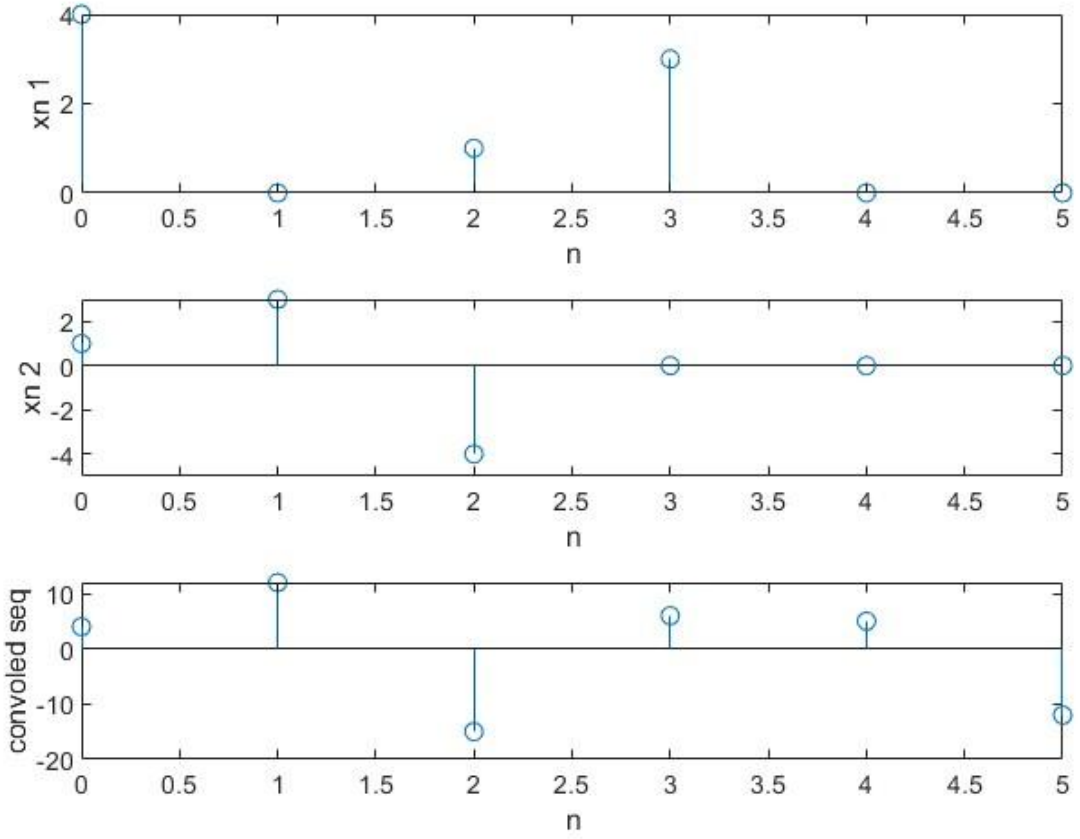
(c) x1[n]=[4,1,0,3]; x2[n]= δ[n]+3δ[n-1]-4δ[n-2]

Matlab Program

```
x1=[4,0,1,3] ;
x2=[1,3,-4];
len1 = length(x1);
len2 = length(x2);
len = len1+len2-1;
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
```

```
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
xlabel("n");
ylabel("convoled seq");
```

Output of Program



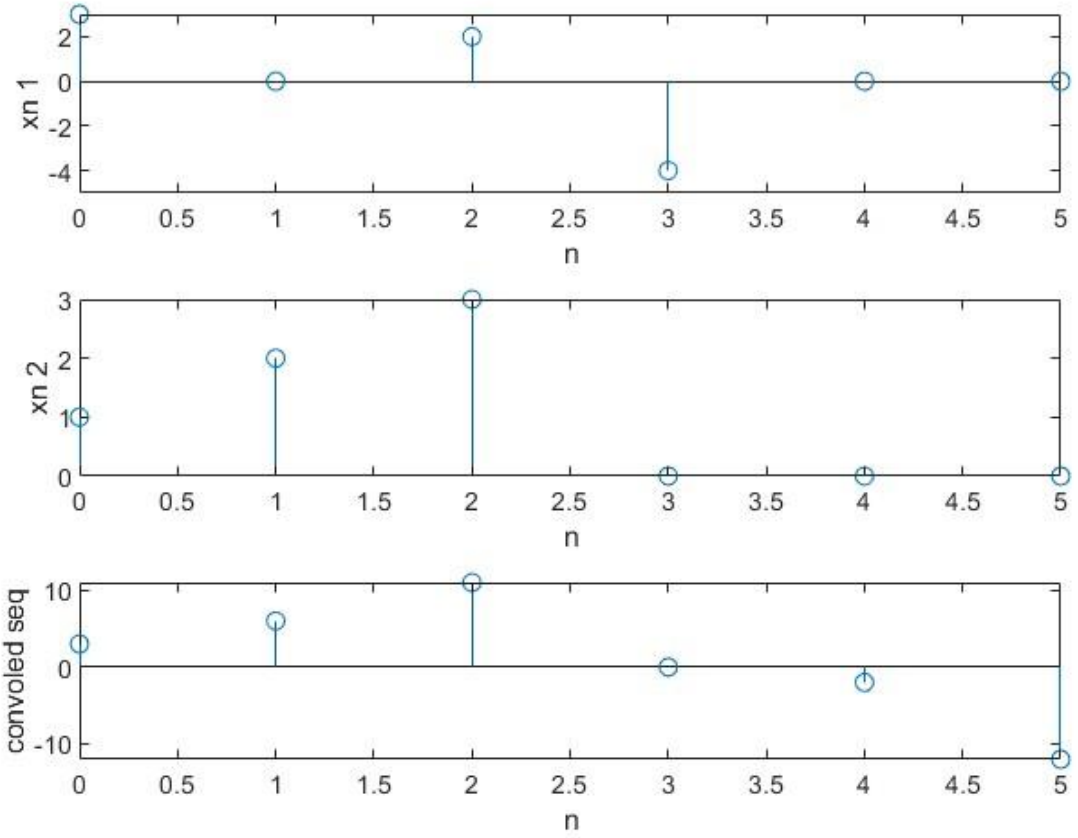
(d) x1[n]=3δ[n]+2δ[n-2]-4δ[n-3]; x2[n]=[1,3,2]

Matlab Program

```
x1=[3,0,2,-4] ;
x2=[1,2,3];
len1 = length(x1);
len2 = length(x2);
len = len1+len2-1;
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
```

```
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
xlabel("n");
ylabel("convoled seq");
```

Output of Program



Use MATLAB code written for (1) to calculate circular convolution of the following sequence.

Let x1(n) and x2(n) be the input sample
Then let X1(K) be the DFT of x1(n) and X2(K) be the DFT of x2(n) then
We Know that multiplication in frequency domain is convolution in time domain
Therefore

$$X3(K) = X1(K) \times X2(K)$$

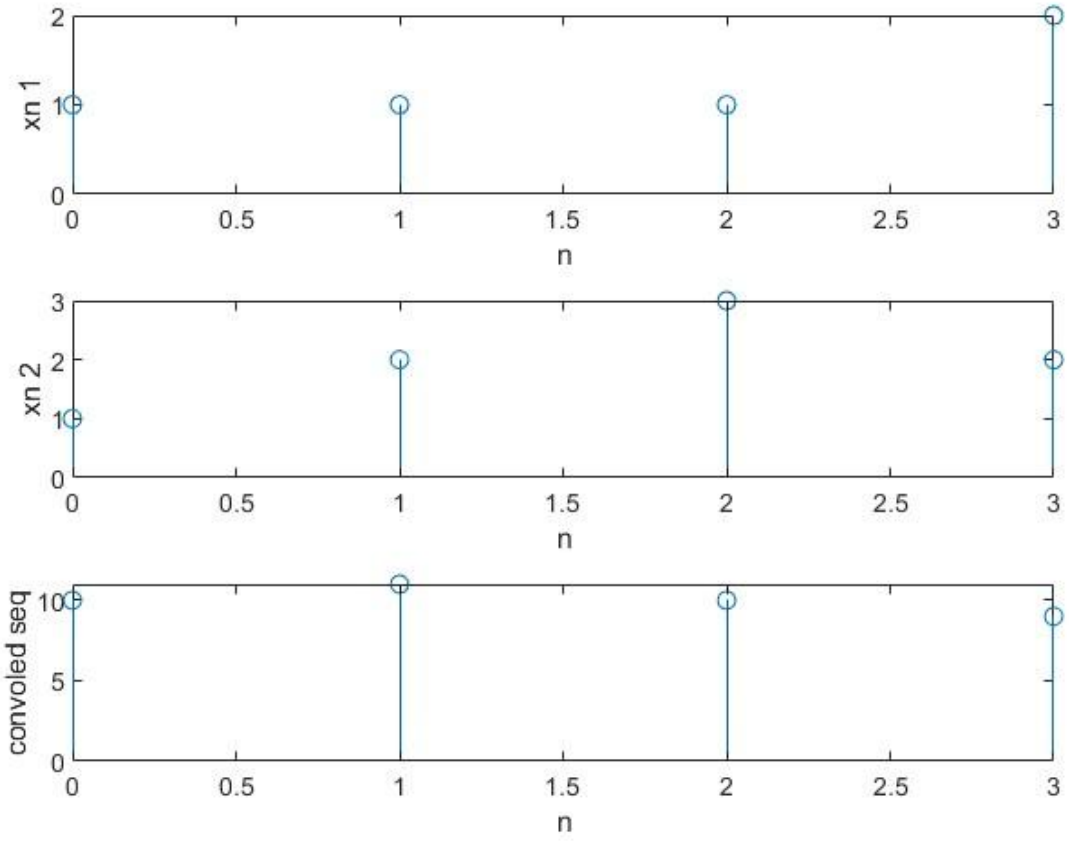
IDFT of X3(K) gives the convolution of x1(n) and x2(n)
Let M be the length of x1(n)
And N be the length of x2(n)
For Linear convolution Length = MAX(M,N)

(a) x1[n]=[1,1,1,2] and x2 [n]=[1,2,3,2]

Matlab Program

```
x1=[1,1,1,2] ;
x2=[1,2,3,2];
len1 = length(x1);
len2 = length(x2);
len = max(len2,len1);
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
xlabel("n");
ylabel("convoled seq");
```

Output of Program



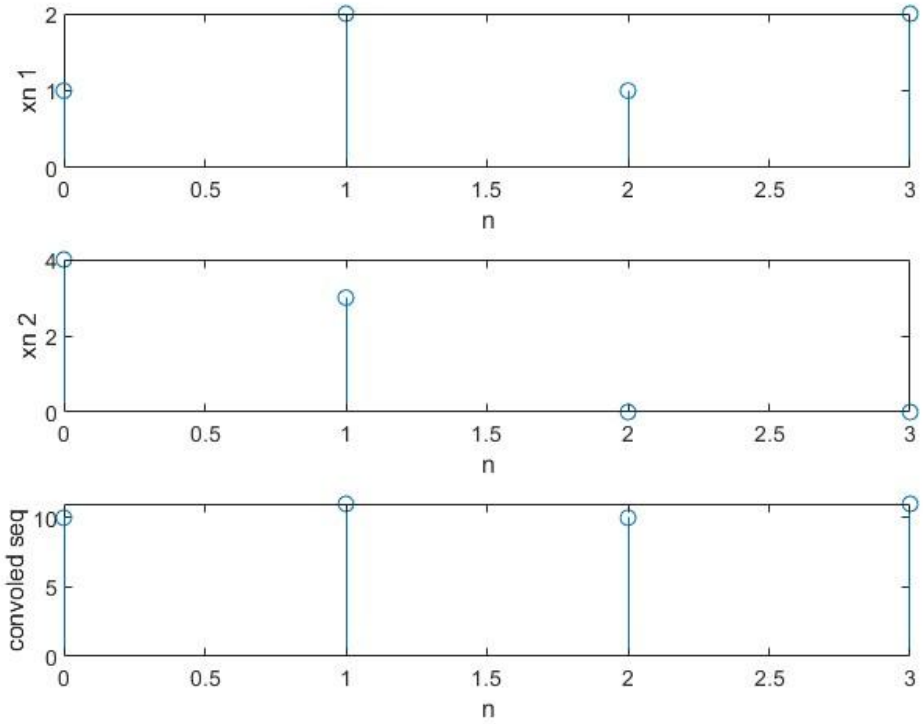
(b) x1[n]=[1,2,1,2] and x2[n]=[4,3]

Matlab Program

```
x1=[1,2,1,2] ;
x2=[4,3];
len1 = length(x1);
len2 = length(x2);
len = max(len2,len1);
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
```

```
xlabel("n");
ylabel("convoled seq");
```

Output of Program



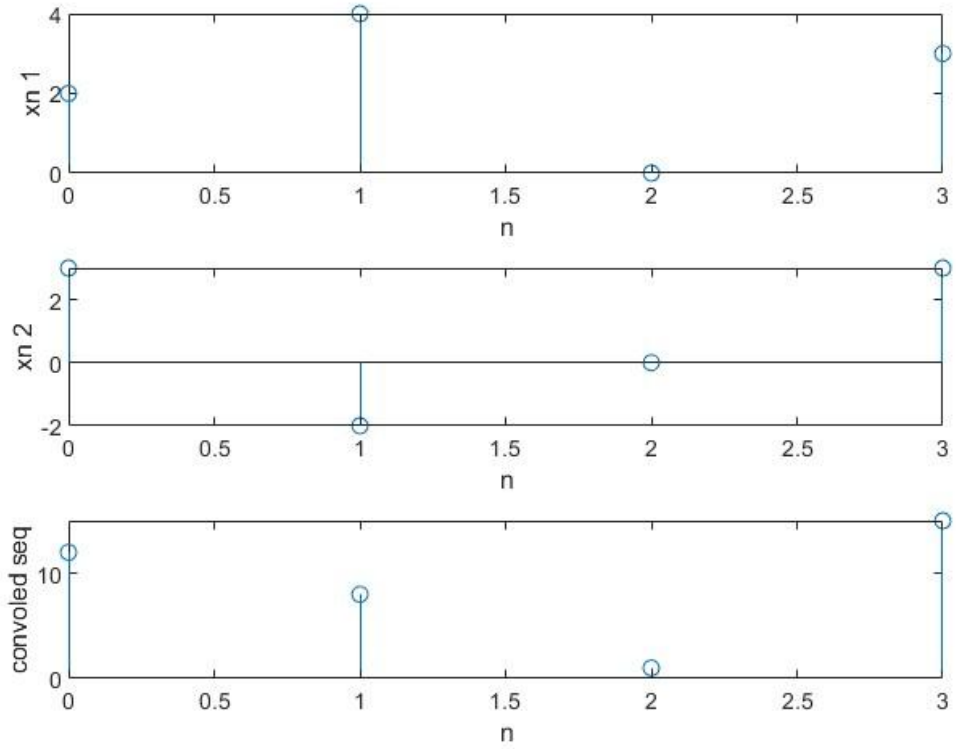
(c) x1[n]=[2,4,0,3]; x2[n]= 3δ[n]-2δ[n-1]+3δ[n-3]

Matlab Program

```
x1=[2,4,0,3] ;
x2=[3,-2,0,3];
len1 = length(x1);
len2 = length(x2);
len = max(len2,len1);
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
xlabel("n");
```

```
ylabel("convoled seq");
```

Output of Program



(d) x1[n] = δ[n]-2δ[n-1]+3δ[n-2]; x2[n]=[1,3,4]

Matlab Program

```
x1=[1,-2,3] ;
x2=[1,3,4];
len1 = length(x1);
len2 = length(x2);
len = max(len2,len1);
x2 = [x2,zeros(1,len-len2)];
x1 = [x1,zeros(1,len-len1)];
n=0:len-1;
xk1 = DFTFun(x1);
xk2 = DFTFun(x2);
xkc = xk1.*xk2;
xnc = iDFTFun(xkc);
subplot(3,1,1);
stem(n,x1);
xlabel("n");
ylabel("xn 1");
subplot(3,1,2);
stem(n,x2);
xlabel("n");
ylabel("xn 2");
subplot(3,1,3);
stem(n,xnc);
xlabel("n");
```

```
ylabel("convoled seq");
```

Output of Program