K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Batch: A2 Roll No.: 1911031
Experiment No. 04
Grade: AA / AB / BB / BC / CC / CD /DD
Signature of the Staff In-charge with date

Title: Write a program to Compute linear and circular convolution of two discrete time signal sequences using Matlab.

Objective: To familiarize the beginner to MATLAB by introducing the basic features and commands of the program.

Expected Outcome of Experiment:

CO	Outcome
CO3	To understand the concept of convolution and perform different convolution operations on the given input signals.

Books/ Journals/ Websites referred:

1. http://www.mathworks.com/support/
2. www.math.mtu.edu/~msgocken/intro/intro.html
3. www.mccormick.northwestern.edu/docs/efirst/matlab.pdf
4. A.Nagoor Kani “Digital Signal Processing”, 2nd Edition, TMH Education.

Pre Lab/ Prior Concepts:

Convolution

Discrete time convolution is a method of finding response of linear time invariant system. It is based on the concepts of linearity and time invariance and assumes that the system information

is known in terms of its impulse response h[n].

Convolution is defined as

$$Y[n] = \sum_{k=-\infty}^{\infty} h[k]x\,[n-k] = h[n]*x[n]$$

Convolution consists of folding, shifting, Multiplication and summation operations.

Circular Convolution

Circular convolution between two length N sequences can be carried out as shown by the expression below:

$$y_C[n] = \sum_{m=0}^{N-1} g[m]h\left[\langle n-m\rangle_N\right]$$

Since the above operation involves two length-N sequences it is referred to as the N-point circular convolution and denoted by:

$$y_c[n] = g[n] \textcircled{N} h[n]$$

As in linear convolution circular convolution is commutative.

i.e.

$$g[n] \textcircled{N} h[n] \equiv h[n] \textcircled{N} g[n]$$

Example Of Linear Convolution:

Determine the response of the LTI system whose input x(n) and impulse response h(n) are given by,

$x(n) = \{1, 2, 0.5, 1\}$ and $h(n) = \{1, 2, 1, -1\}$

When $n = -1$; $y(-1) = \sum_{m=-\infty}^{+\infty} x(m)\, h(-1-m) = \sum_{m=-\infty}^{+\infty} x(m)\, h_{-1}(m) = \sum_{m=-\infty}^{+\infty} v_{-1}(m)$

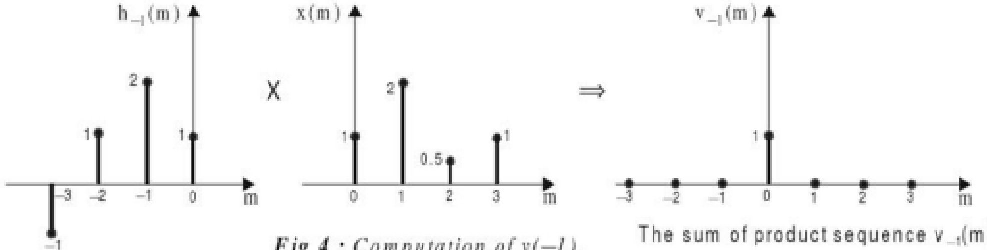


Fig 4 : Computation of y(−1).

The sum of product sequence $v_{-1}(m)$ gives $y(-1)$. $\therefore\ y(-1) = 1$

When $n = 0$; $y(0) = \sum_{m=-\infty}^{+\infty} x(m)\, h(0-m) = \sum_{m=-\infty}^{+\infty} x(m)\, h_0(m) = \sum_{m=-\infty}^{+\infty} v_0(m)$

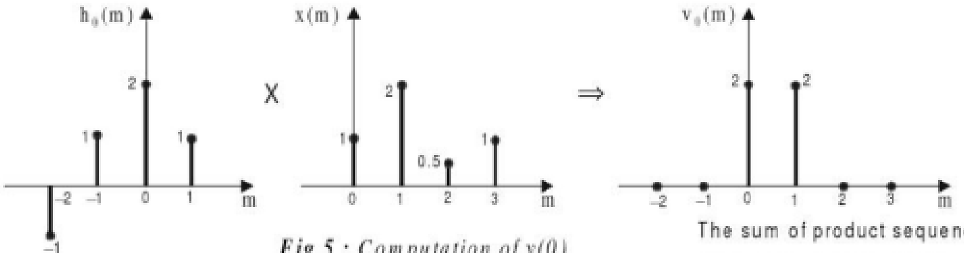


Fig 5 : Computation of y(0).

The sum of product sequence $v_0(m)$ gives $y(0)$. $\therefore\ y(0) = 2 + 2 = 4$

When $n = 1$; $y(1) = \sum_{m=-\infty}^{+\infty} x(m)\, h(1-m) = \sum_{m=-\infty}^{+\infty} x(m)\, h_1(m) = \sum_{m=-\infty}^{+\infty} v_1(m)$

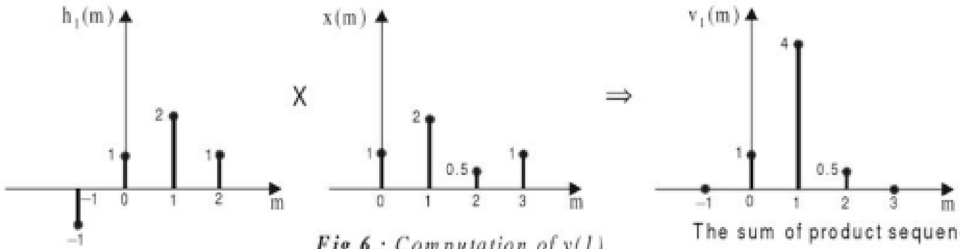


Fig 6 : Computation of y(1).

The sum of product sequence $v_1(m)$ gives $y(1)$. $\therefore\ y(1) = 1 + 4 + 0.5 = 5.5$

When $n = 2$; $y(2) = \sum_{m=-\infty}^{+\infty} x(m)\, h(2-m) = \sum_{m=-\infty}^{+\infty} x(m)\, h_2(m) = \sum_{m=-\infty}^{+\infty} v_2(m)$

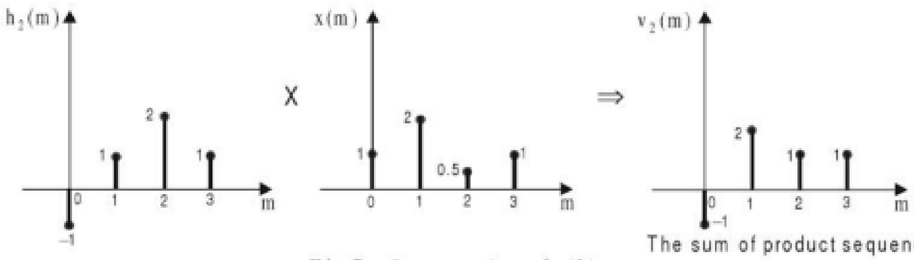


Fig 7 : Computation of y(2).

The sum of product sequence $v_2(m)$ gives $y(2)$. $\therefore\ y(2) = -1 + 2 + 1 + 1 = 3$

When n = 3 ; $y(3) = \sum_{m=-\infty}^{+\infty} x(m)\, h(3-m) = \sum_{m=-\infty}^{+\infty} x(m)\, h_3(m) = \sum_{m=-\infty}^{+\infty} v_3(m)$

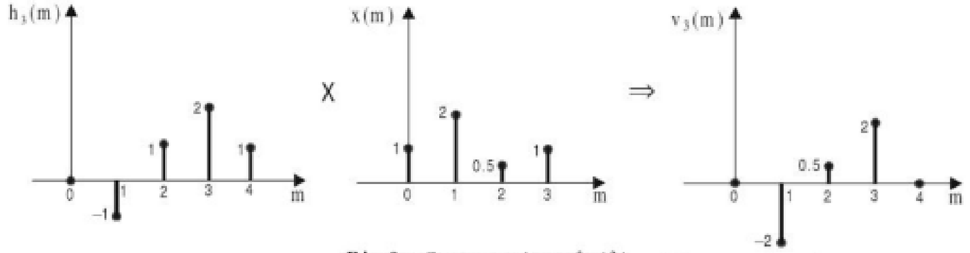


Fig 8 : Computation of y(3).

The sum of product sequence $v_3(m)$ gives y(3). $\therefore$ $y(3) = -2 + 0.5 + 2 = 0.5$

When n = 4 ; $y(4) = \sum_{m=-\infty}^{+\infty} x(m)\, h(4-m) = \sum_{m=-\infty}^{+\infty} x(m)\, h_4(m) = \sum_{m=-\infty}^{+\infty} v_4(m)$

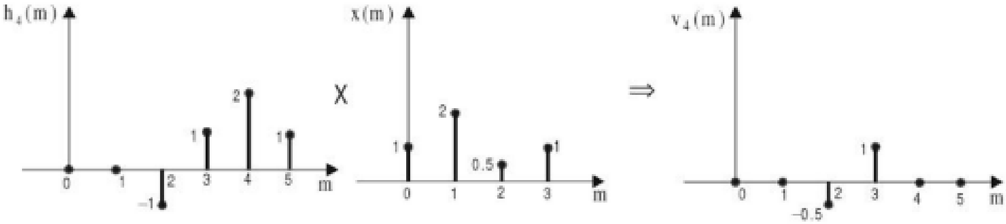


Fig 9 : Computation of y(4).

The sum of product sequence $v_4(m)$ gives y(4). $\therefore$ $y(4) = -0.5 + 1 = 0.5$

When n = 5 ; $y(5) = \sum_{m=-\infty}^{+\infty} x(m)\, h(5-m) = \sum_{m=-\infty}^{+\infty} x(m)\, h_5(m) = \sum_{m=-\infty}^{+\infty} v_5(m)$

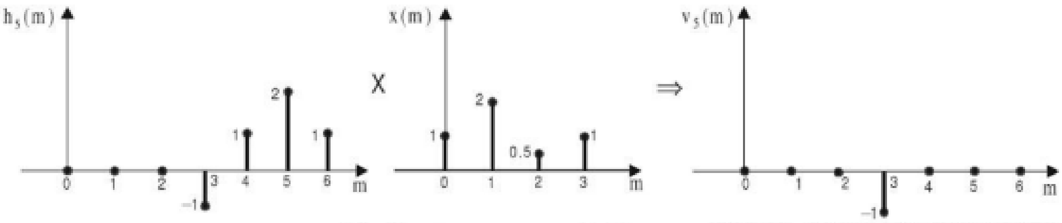


Fig 10 : Computation of y(5).

The sum of product sequence $v_5(m)$ gives y(5). $\therefore$ $y(5) = -1$

The output sequence, $y(n) = \{1,\ \underset{\uparrow}{4},\ 5.5,\ 3,\ 0.5,\ 0.5,\ -1\}$

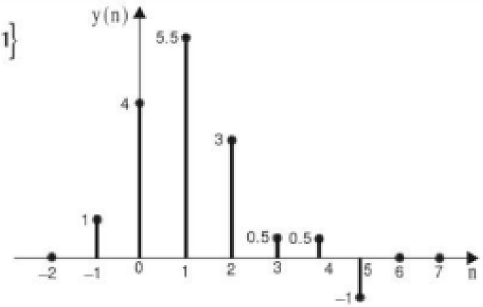


Fig 11 : Graphical representation of y(n).

Example Of Circular Convolution:

Perform circular convolution of the two sequences, $x_1(n) = \{2, 1, 2, -1\}$ and $x_2(n) = \{1, 2, 3, 4\}$

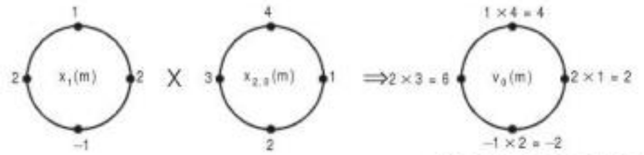
The given sequences are 4-point sequences . \ N = 4.

Each sample of $x_3(n)$ is given by sum of the samples of product sequence defined by the equation,

$$x_3(n) = \sum_{m=0}^{3} x_1(m)\ x_{2,n}(m) = \sum_{m=0}^{3} v_n(m) \quad ; \quad \text{where } v_n(m) = x_1(m)\ x_{2,n}(m) \qquad .....(1)$$

Using the above equation (1), graphical method of computing each sample of $x_3(n)$ are shown in fig 5 to fig 8.

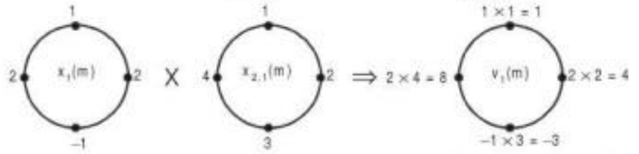
When $n = 0$; $x_3(0) = \sum_{m=0}^{3} x_1(m)\ x_2((0-m))_4 = \sum_{m=0}^{3} x_1(m)\ x_{2,0}(m) = \sum_{m=0}^{3} v_0(m)$



The sum of samples of $v_0(m)$ gives $x_3(0)$

Fig 5: Computation of $x_3(0)$. $\therefore\ x_3(0) = 2 + 4 + 6 - 2 = 10$

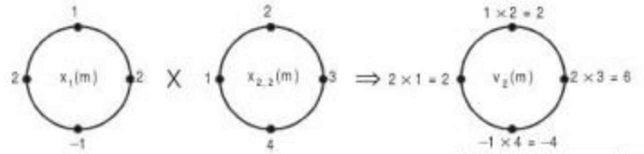
When $n = 1$; $x_3(1) = \sum_{m=0}^{3} x_1(m)\ x_2((1-m))_4 = \sum_{m=0}^{3} x_1(m)\ x_{2,1}(m) = \sum_{m=0}^{3} v_1(m)$



The sum of samples of $v_1(m)$ gives $x_3(1)$

Fig 6: Computation of $x_3(1)$. $\therefore\ x_3(1) = 4 + 1 + 8 - 3 = 10$

When $n = 2$; $x_3(2) = \sum_{m=0}^{3} x_1(m)\ x_2((2-m))_4 = \sum_{m=0}^{3} x_1(m)\ x_{2,2}(m) = \sum_{m=0}^{3} v_2(m)$



The sum of samples of $v_2(m)$ gives $x_3(2)$

Fig 7: Computation of $x_3(2)$. $\therefore\ x_3(2) = 6 + 2 + 2 - 4 = 6$

When $n = 3$; $x_3(3) = \sum_{m=0}^{3} x_1(m)\ x_2((3-m))_4 = \sum_{m=0}^{3} x_1(m)\ x_{2,3}(m) = \sum_{m=0}^{3} v_3(m)$

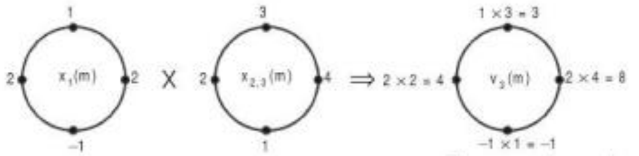


Fig 8: Computation of $x_3(3)$. The sum of samples of $v_3(m)$ gives $x_3(3)$

$\therefore\ x_3(3) = 8 + 3 + 4 - 1 = 14$

\ $x_3(n) = \{10, 10, 6, 14\}$

Implementation details along with screenshots:

Linear Convolution

Code:

Matrix Method:

```
xl = 0;
hl = -1;
x = [1, 2, 0.5, 1];
h = [1, 2, 1, -1];
k = 1;
nl = xl + hl;
xh = xl+length(x)-1;
hh = hl+length(h)-1;
nh = (xh) + (hh);
y = [0, 0, 0, 0, 0, 0, 0];
for i=1:1:length(x)
    for j=1:1:length(h)
        y(i+j-1) = y(i+j-1) + x(i)*h(j);
    end
end

% disp(y);

a = xl:1:xh;
b = hl:1:hh;
c = nl:1:nh;

figure(2)
subplot(1, 3, 1)
stem(a, x)
title("x(n)");

subplot(1, 3, 2)
stem(b, h)
title("h(n)");

subplot(1, 3, 3)
stem(c, y)
title("y(n)");
```

Graphical Method

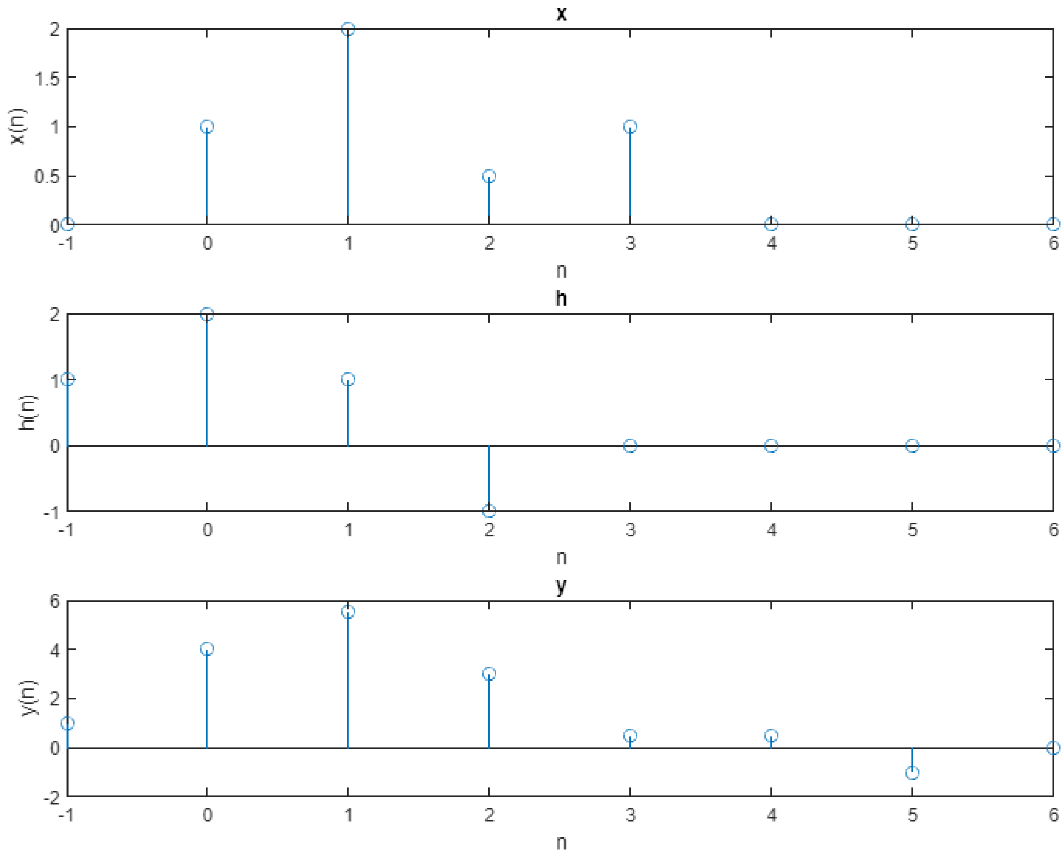
```
,n1,xl]= get_vector('x');
```

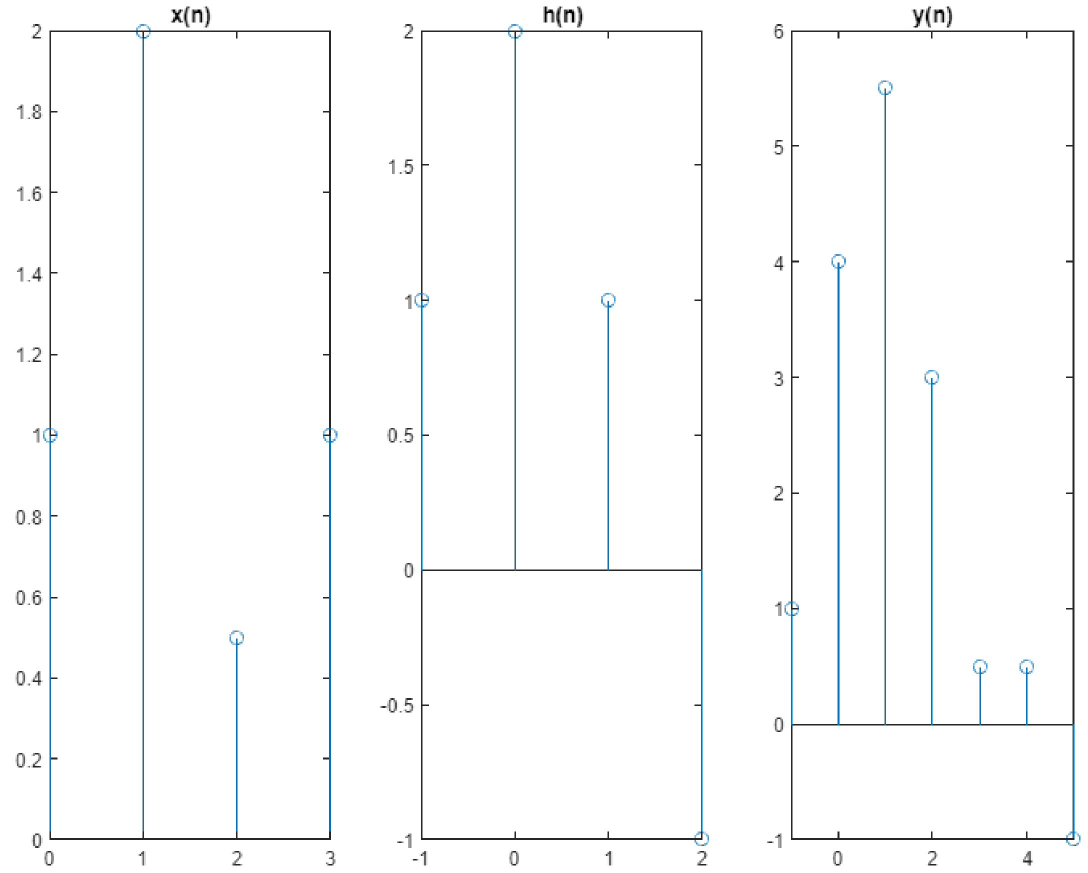
```
disp('');
[h,n2,hl]= get_vector('h');
disp('');

if xl==hl && n1~=n2
   temp = zeros(1,max([n1 n2]));
   temp(1:n1) = x;
   x = temp;
   temp = zeros(1,max([n1 n2]));
   temp(1:n2) = h;
   h = temp;
end

if xl~=hl
   point = 1;
   while n1+n2-1>point
      point=point*2;
   end

   temp = zeros(1,point);
   temp(relu(hl-xl)+1:relu(hl-xl)+n2) = h;
   h = temp;
   temp = zeros(1,point);
   temp(relu(xl-hl)+1:relu(xl-hl)+n1) = x;
   x = temp;
end

xl = min([xl hl]);
hl = xl;

n1 = length(x);
n2 = length(h);

matrix = zeros(n2,n2);

for i=1:n2
   matrix(:,i) = h;
   h = circshift(h,1);
end

y = (matrix*x.').';
y = circshift(y,-1*relu(-hl));

disp('Result of convolution (y = x*h) : ');
disp(y);
```

```
subplot(3,1,1);
stem(xl:xl+n1-1,x);
title('x');
xlabel('n');
ylabel('x(n)');

subplot(3,1,2);
stem(hl:hl+n2-1,h);
title('h');
xlabel('n');
ylabel('h(n)');

subplot(3,1,3);
stem(xl:xl+n1-1,y);
title('y');
xlabel('n');
ylabel('y(n)');

function val = relu(i)
   val = 0;
   if i>0
      val = i;
   end
end

function [x,n,start] = get_vector(s)
   n = input(['Enter the size of ' s ' : ']);
   x = zeros(1,n);
   for i=1:n
      x(i) = input(['Enter element ' num2str(i,'%d') ' of ' s ' : ']);
   end
   start = input(['Enter the start of ' s ' : ']);
end
```

Output:

```
>> Exp3
Enter the size of x :
4
Enter element 1 of x :
1
Enter element 2 of x :
2
Enter element 3 of x :
0.5
Enter element 4 of x :
1
Enter the start of x :
0
Enter the size of h :
4
Enter element 1 of h :
1
Enter element 2 of h :
2
Enter element 3 of h :
1
Enter element 4 of h :
-1
Enter the start of h :
-1
Result of convolution (y = x*h) :
    1.0000    4.0000    5.5000    3.0000    0.5000    0.5000   -1.0000         0
```


x
x(n)
n
h
h(n)
n
y
y(n)
n


x(n)
h(n)
y(n)

Circular Convolution:

Code:

Matrix Method:

```
x_1n = 4;
x_1l = 0;
x_1h = 3;
x_1 = [2, 1, 2, -1];

x_2n = 4;
x_2l = 0;
x_2h = 3;
x_2 = [1, 2, 3, 4];

figure(1);
x_index = x_1l:1:x_1n-abs(x_1l)-1;
subplot(1, 3, 1);
stem(x_index, x_1);
title("x1(n)");
xlabel("n");
ylabel("amplitude");

x_index = x_2l:1:x_2n-abs(x_2l)-1;
subplot(1, 3, 2);
stem(x_index, x_2);
title("x2(n)");
xlabel("n");
ylabel("amplitude");

npower = 0;

if x_1l == x_2l
    if x_1n > x_2n
        npower = ceil(log2(x_1n));
    else
        npower = ceil(log2(x_2n));
    end
    for i=1:((2^npower) - x_1n)
        x_1 = [x_1, 0];
    end
    for i=1:((2^npower) - x_2n)
        x_2 = [x_2, 0];
    end
else
```

```
    npower = log2((x_1n + x_2n - 1));
    count = 0;
    if x_1l > x_2l
        FRONT_PAD = abs(x_1l - x_2l);
        for i=1:FRONT_PAD
            x_1 = [0, x_1];
        end
        count = 2^npower - FRONT_PAD - x_1n;
        for i=1:count
            x_1 = [x_1, 0];
        end
        for i=1:((2^npower) - x_2n)
            x_2 = [x_2, 0];
        end
    else
        FRONT_PAD = abs(x_1l - x_2l);
        for i=1:FRONT_PAD
            x_2 = [0, x_2];
        end
        count = 2^npower - FRONT_PAD - x_2n;
        for i=1:count
            x_2 = [x_2, 0];
        end
        for i=1:((2^npower) - x_1n)
            x_1 = [x_1, 0];
        end
    end
end
disp(x_1);
disp(x_2);
%
%
% Matrix Method %

x_2 = x_2(end:-1:1);
matrix_2 = [circshift(x_2,[1,1])];
for i=2:1:length(x_2)
    matrix_2 = [matrix_2; circshift(x_2,[1,1*i])];
end
disp(matrix_2);

x_1  = x_1.';
x_3 = matrix_2*x_1;
x_3 = circshift(x_3, -abs(x_1l-x_2l));
disp(x_3);
```

```
x_index = x_2l:1:x_2n-abs(x_2l)-1;
subplot(1, 3, 3);
stem(x_index, x_3);
title("x3(n)");
xlabel("n");
ylabel("amplitude");
```

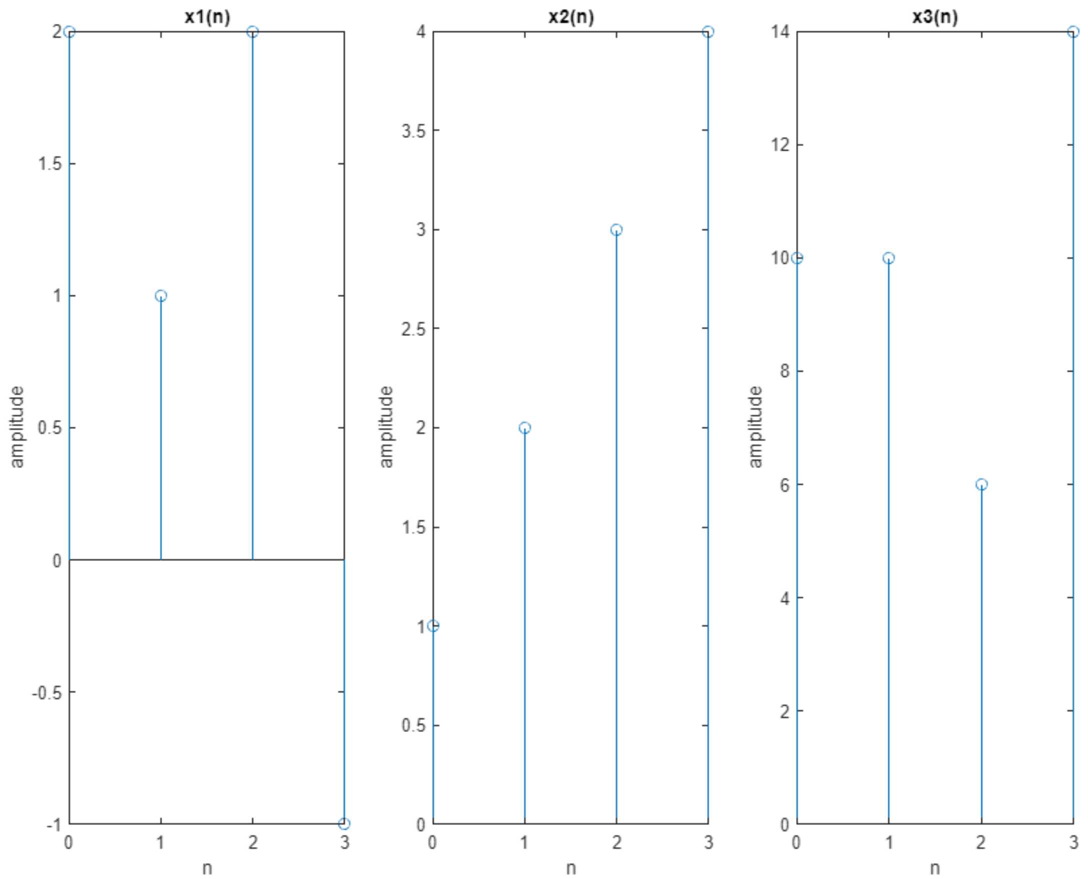
Graphical Method

```
x_1n = 4;
x_1l = 0;
x_1h = 3;
x_1 = [2, 1, 2, -1];

x_2n = 4;
x_2l = 0;
x_2h = 3;
x_2 = [1, 2, 3, 4];

figure(1);
x_index = x_1l:1:x_1n-abs(x_1l)-1;
subplot(1, 3, 1);
stem(x_index, x_1);
title("x1(n)");
xlabel("n");
ylabel("amplitude");

x_index = x_2l:1:x_2n-abs(x_2l)-1;
subplot(1, 3, 2);
stem(x_index, x_2);
title("x2(n)");
xlabel("n");
ylabel("amplitude");

npower = 0;
if x_1l == x_2l
   if x_1n > x_2n
      npower = ceil(log2(x_1n));
   else
      npower = ceil(log2(x_2n));
   end
   for i=1:((2^npower) - x_1n)
      x_1 = [x_1, 0];
   end
   for i=1:((2^npower) - x_2n)
```

```
        x_2 = [x_2, 0];
    end
else
    npower = log2((x_1n + x_2n - 1));
    count = 0;
    if x_1l > x_2l
        FRONT_PAD = abs(x_1l - x_2l);
        for i=1:FRONT_PAD
            x_1 = [0, x_1];
        end
        count = 2^npower - FRONT_PAD - x_1n;
        for i=1:count
            x_1 = [x_1, 0];
        end
        for i=1:((2^npower) - x_2n)
            x_2 = [x_2, 0];
        end
    else
        FRONT_PAD = abs(x_1l - x_2l);
        for i=1:FRONT_PAD
            x_2 = [0, x_2];
        end
        count = 2^npower - FRONT_PAD - x_2n;
        for i=1:count
            x_2 = [x_2, 0];
        end
        for i=1:((2^npower) - x_1n)
            x_1 = [x_1, 0];
        end
    end
end
disp(x_1);
disp(x_2);


% Graphical method %

for i=2:1:length(x_2)/2
    temp = x_2(length(x_2) - i + 2);
    x_2(length(x_2) - i + 2) = x_2(i);
    x_2(i) = temp;
end

final = [];
for i=1:1:length(x_2)
    disp(x_2);
```

```
    final = [final, x_1*x_2'];
    x_2 = circshift(x_2,[1,1]);
end
final = circshift(final, [1, -abs(x_1l-x_2l)]);

disp(x_1);
disp(x_2);
disp(final);

start = 0;
if x_1l < x_2l
    start = x_1l;
else
    start = x_2l;
end

endi = length(x_1) - 1 - abs(start);

figure(1);
x_index = start:1:endi;
subplot(1, 3, 3);
stem(x_index, final);
title("x3(n)");
xlabel("n");
ylabel("amplitude");
```

Output:



Correlation

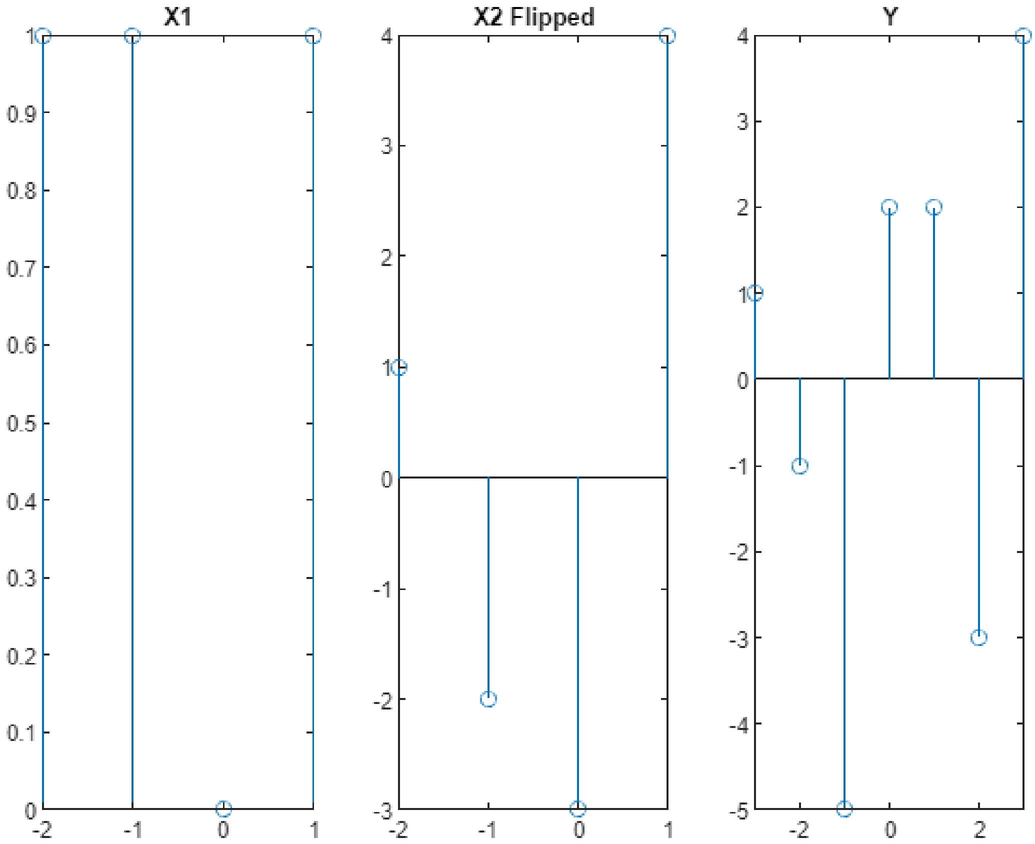
Cross:

```
x_1n = 4;
x_1l = -2;
x_1h = 1;
x_1 = [1, 1, 0, 1];

x_2n = 4;
x_2l = -2;
x_2h = 1;
x_2 = [4, -3, -2, 1];

disp(x_1);
disp(x_2);

for i=4:-1:1
    x_2_flipped(5-i) = x_2(i);
end
disp(x_2_flipped);
```

```
x_2 = x_2_flipped;
% disp(x_2);

y = zeros(1, x_1n+x_2n-1);

for i=1:1:length(x_1)
    for j=1:1:length(x_2)
        y(i+j-1) = y(i+j-1) + x_1(i)*x_2(j);
    end
end
disp(y);

yn = x_1n+x_2n-1;
yl = x_1l + (-1)*x_2h;
yh = x_1h + (-1)*x_2l;
% disp(yl);
% disp(yh);
figure(1);
subplot(1, 3, 1);
stem(x_1l:1:x_1h, x_1);
title("X1");
subplot(1, 3, 2);
stem(x_2l:1:x_2h, x_2);
title("X2 Flipped");
subplot(1, 3, 3);
stem(yl:1:yh, y);
title("Y");
```

Output:

```
>> Exp3
     1     1     0     1

     4    -3    -2     1

     1    -2    -3     4

     1    -1    -5     2     2    -3     4
```



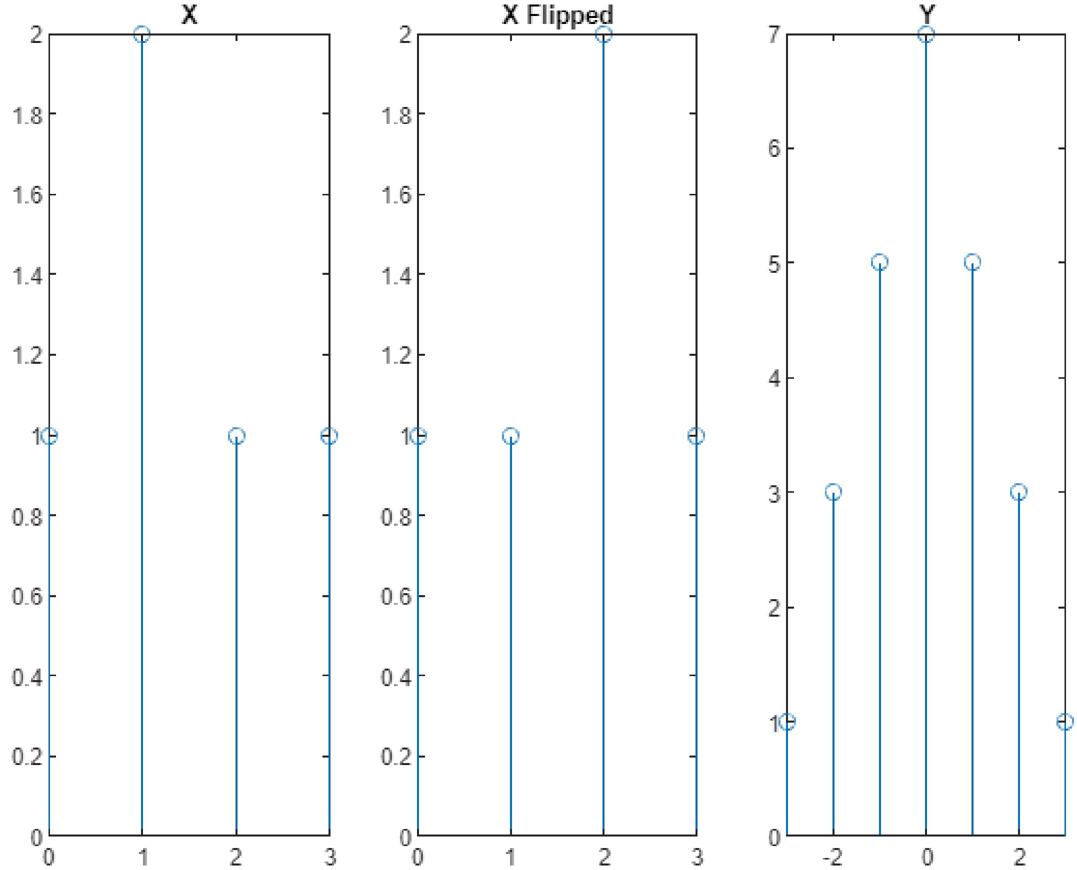
Auto:

```
x_1n = 4;
x_1l = 0;
x_1h = 3;
x_1 = [1, 2, 1, 1];

x_2n = 4;
x_2l = 0;
x_2h = 3;

disp(x_1);

for i=4:-1:1
    x_2(5-i) = x_1(i);
end
disp(x_2);

% disp(x_2);

y = zeros(1, x_1n+x_2n-1);
```

```
for i=1:1:length(x_1)
    for j=1:1:length(x_2)
        y(i+j-1) = y(i+j-1) + x_1(i)*x_2(j);
    end
end
disp(y);

yn = x_1n+x_2n-1;
yl = x_1l + (-1)*x_2h;
yh = x_1h + (-1)*x_2l;
% disp(yl);
% disp(yh);

figure(2);
subplot(1, 3, 1);
stem([x_1l:1:x_1h], x_1);
title("X");
subplot(1, 3, 2);
stem([x_2l:1:x_2h], x_2);
title("X Flipped");
subplot(1, 3, 3);
stem([yl:1:yh], y);
title("Y");
```

Output:

```
>> Exp3
     1     2     1     1

     1     1     2     1

     1     3     5     7     5     3     1
```



Conclusion:- Successfully implemented linear and circular convolution in matlab

Date: 25-04-2022 **Signature of faculty in-charge**

Post Lab Descriptive Questions

1. Explain the role of convolution in signal processing.

In mathematics (in particular, functional analysis), convolution is a mathematical operation on two functions (f and g) that produces a third function ({\displaystyle f*g}f*g) that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it. It is defined as the integral of the product of the two functions after one is reversed and shifted. The integral is evaluated for all values of shift, producing the convolution function.

Some features of convolution are similar to cross-correlation: for real-valued functions, of a continuous or discrete variable, it differs from cross-correlation ({\displaystyle f\star g}f\star g) only in that either f(x) or g(x) is reflected about the y-axis; thus it is a cross-correlation of f(x) and g(−x), or f(−x) and g(x).[A] For complex-valued functions, the cross-correlation operator is the adjoint of the convolution operator.

Convolution has applications that include probability, statistics, acoustics, spectroscopy, signal processing and image processing, geophysics, engineering, physics, computer vision and differential equations.[1]

The convolution can be defined for functions on Euclidean space and other groups.[citation needed] For example, periodic functions, such as the discrete-time Fourier transform, can be defined on a circle and convolved by periodic convolution. (See row 18 at DTFT § Properties.) A discrete convolution can be defined for functions on the set of integers.

Generalizations of convolution have applications in the field of numerical analysis and numerical linear algebra, and in the design and implementation of finite impulse response filters in signal processing.[citation needed]

Computing the inverse of the convolution operation is known as deconvolution.

2. Explain the difference between linear and circular convolution?

Linear Convolution	**Circular Convolution**
Linear convolution is a mathematical operation done to calculate the output of any Linear-Time Invariant (LTI) system given its input and impulse response.	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 (We'll see what this means in a minute). 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.
It is applicable for both continuous and discrete-time signals.	Circular convolution is also 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.	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.
In linear convolution, both the sequences (input and impulse response) may or may not be of equal sizes. That is, they may or may not have the same number of samples. Thus the output, too, may or may not have the same number of samples as any of the inputs.	In circular convolution, both the sequences (input and impulse response) must be of equal sizes. They must have the same number of samples. Thus the output of a circular convolution has the same number of samples as the two inputs.
For example, consider the following signals: x(n): [1,2,3]	For the given example, circular convolution is possible only after modifying the signals via a method known as *zero padding.* In zero padding,

h(n): [1,2,3,4,5] As you can see, the number of samples in the input and Impulse response signals is not the same. Still, linear convolution is possible. Here's how you calculate the number of samples in the output of linear convolution. L = M + N – 1 Where M is the number of samples in x(n). N is the number of samples in h(n). For the above example, the output will have (3+5-1) = 7 samples.	zeroes are appended to the sequence that has a lesser size to make the sizes of the two sequences equal. Thus, for the given sequence, after zero-padding: x(n) = [1,2,3,0,0] Now both x(n) and h(n) have the same lengths. So circular convolution can take place. And the output of the circular convolution will have the same number of samples. i.e., 5.
Graphically, when we perform linear convolution, there is a linear shift taking place. Check out the formula for a convolution. $\sum_{-\infty}^{\infty} x(k)h(n-k)$ There is a folding of the IR sequence, shifting it by n, multiplying it with another sequence (input), and summing the resulting products.	Graphically, when we perform circular convolution, there is a circular shift taking place. Alternatively, we can call it rotation.
It is possible to find the response of a filter using linear convolution.	It is possible to find the response of a filter using circular convolution after zero padding. I fact, we will be doing this in overlap-save and overlap-add methods — two essential topics in our digital signal processing course.
Linear convolution may or may not result in a periodic output signal.	The output of a circular convolution is always periodic, and its period is specified by the periods of one of its inputs.

3. Explain with the help of an example the steps required to transform linear convolution with circular convolution and vice-versa.

Linear and circular convolution are fundamentally different operations. However, there are conditions under which linear and circular convolution are equivalent. Establishing this equivalence has important implications. For two vectors, x and y, the circular convolution is equal to the inverse discrete Fourier transform (DFT) of the product of the vectors' DFTs. Knowing the conditions under which linear and circular convolution are equivalent allows you to use the DFT to efficiently compute linear convolutions.

The linear convolution of an N-point vector, x, and an L-point vector, y, has length N + L - 1.

For the circular convolution of x and y to be equivalent, you must pad the vectors with zeros to length at least N + L - 1 before you take the DFT. After you invert the product of the DFTs, retain only the first N + L - 1 elements.

Create two vectors, x and y, and compute the linear convolution of the two vectors.

x = [2 1 2 1];

y = [1 2 3];

clin = conv(x,y);

The output has length 4+3-1.

Pad both vectors with zeros to length 4+3-1. Obtain the DFT of both vectors, multiply the DFTs, and obtain the inverse DFT of the product.

xpad = [x zeros(1,6-length(x))];

ypad = [y zeros(1,6-length(y))];

ccirc = ifft(fft(xpad).*fft(ypad));

The circular convolution of the zero-padded vectors, xpad and ypad, is equivalent to the linear convolution of x and y. You retain all the elements of ccirc because the output has length 4+3-1.

Plot the output of linear convolution and the inverse of the DFT product to show the equivalence.

```
subplot(2,1,1)

stem(clin,'filled')

ylim([0 11])

title('Linear Convolution of x and y')

subplot(2,1,2)

stem(ccirc,'filled')

ylim([0 11])

title('Circular Convolution of xpad and ypad')
```