K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch: A2 **Roll No.: 1911027**

Experiment No. 4

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:

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

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:

LINEAR CONVOLUTION

* $x(n) = \{\underset{\uparrow}{1}, 2, 0.5, 1\}$

$h(n) = \{1, \underset{\uparrow}{2}, 1, -1\}$

$\Longrightarrow$ start of $x(n) = x_1 = 0$

start of $h(n) = h_1 = -1$

start of output sequence $y(n) = x_1 + h_1 = -1$

Total Number of samples in $x(n) = N_1 = 4$

Total Number of samples in $h(n) = N_2 = 4$

Total number of samples in output $y(n) = N_1 + N_2 - 1$

$= 4 + 4 - 1$

$N = 7$

Convolution of $x(n)$ and $h(n)$ is given by,

$$y(n) = x(n) * h(n) = \sum_{m=-\infty}^{+\infty} x(m)h(n-m)$$

Tabular method :-

m	-3	-2	-1	0	1	2	3	4	5	6
x(m)				1	2	0.5	1			
h(m)			1	2	1	-1				
h(-m)		-1	1	2	1					
h(-1-m) = h₁(m)	-1	1	2	1						
h(0-m) = h₀(m)		-1	1	2	1					
h(1-m) = h₁(m)			-1	1	2	1				

$h(2-m) = h_2(m)$				-1	1	2	1				
$h(3-m) = h_3(m)$					-1	1	2	1			
$h(4-m) = h_4(m)$						-1	1	2	1		
$h(5-m) = h_5(m)$							-1	1	2	1	

$\therefore$ for, $n = -1$,

$y(-1) = \sum_{m=-3}^{3} x(m)\, h_{-1}(m)$

$= 0+0+0+1+0+0+0$

$= 1$

for, $n = 0$

$y(0) = \sum_{m=-3}^{3} x(m)\, h_0(m)$

$= 0+0+2+2+0+0$

$= 4$

for, $n = 1$,

$y(1) = \sum_{m=-3}^{3} x(m)\, h_1(m)$

$= 0+1+4+0.5+0$

$= 5.5$

for, $n = 2$,

$y(2) = \sum_{m=0}^{3} x(m)\, h_2(m)$

$= -1+2+1+1$

$= 3$

for, $n = 3$,

$y(3) = \sum_{m=0}^{3} x(m)\, h_3(m)$

$= 0-2+0.5+2+0$

$= 0.5$

for $n = 4$,

$y(4) = \sum_{m=0}^{5} x(m)\, h_4(m)$

$= 0+0-0.5+1+0+0$

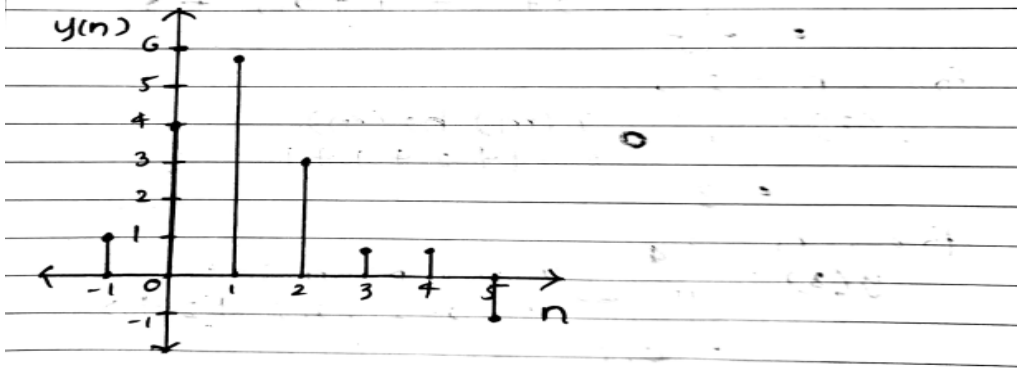
$= 0.5$

for $n = 5$,

$y(5) = \sum_{m=0}^{6} x(m)\, h_5(m) = 0+0+0-1+0+0+0$

$= -1$

$\therefore y(n) = \{1, 4, 5.5, 3, 0.5, 0.5, -1\}$



Example Of Circular Convolution:

CIRCULAR CONVOLUTION

* $x_1(n) = \{2, 1, 2, -1\}$
  $x_2(n) = \{1, 2, 3, 4\}$

⇒ Circular convolution of $x_1(n)$ and $x_2(n)$ is given by,

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)\, x_2((n-m))_N$$

where, m = dummy variable,

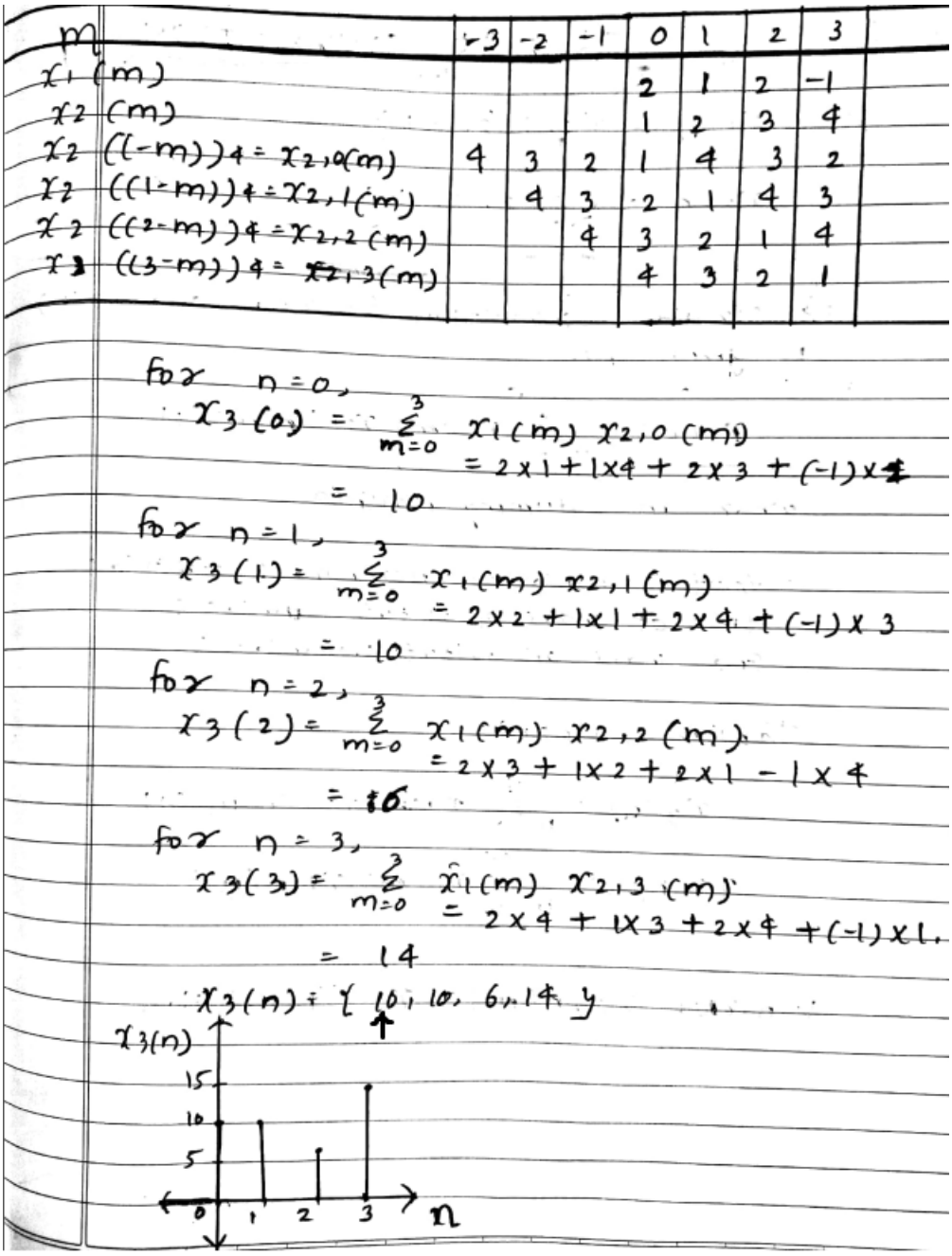
output sequence will also have 4 samples.
start index of output sequence = 0
end index of output sequence = 3.

Tabular method :-

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)\, x_2((n-m))_N$$

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)\, x_{2,n}(m)$$

where, $x_{2,n}(m) = x_2((n-m))_N$.



Implementation details along with screenshots:

Linear convolution:

Matrix method:

Code:

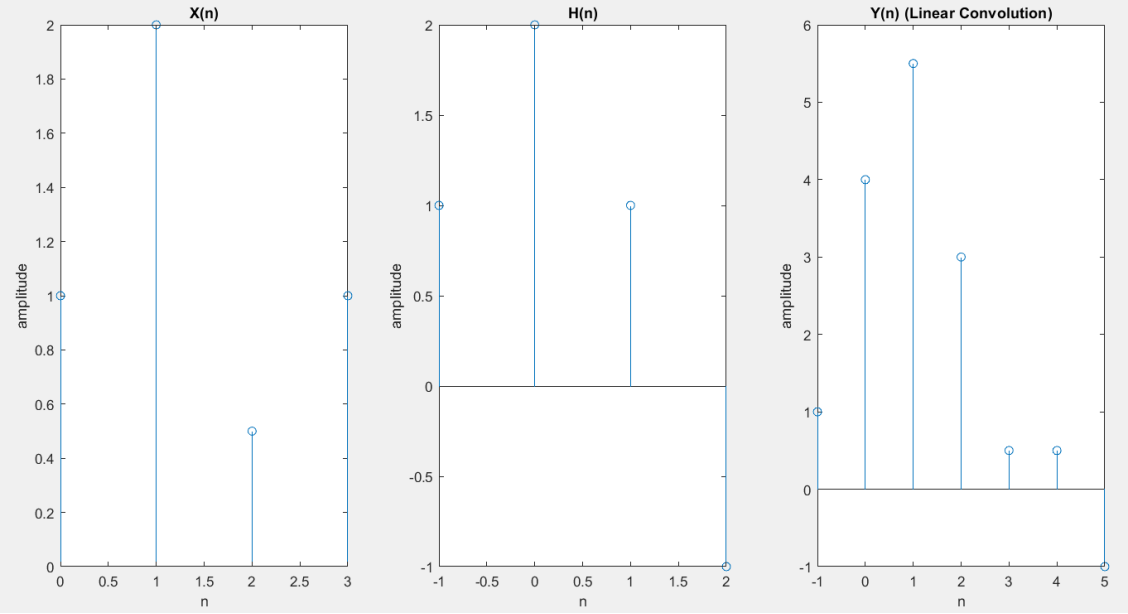
```
xl=input("Enter start of signal X: ");
```

```
hl=input("Enter start of singal H: ");
x=input("Enter amplitudes of signal X: ");
h=input("Enter amplitudes of signal H: ");

subplot(1,3,1);
figure(1);
stem(xl:1:xl+length(x)-1,x);
title('X(n)');
xlabel('n');
ylabel('amplitude');

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

xh=xl+length(x)-1;
hh=hl+length(h)-1;
nl=xl+hl;
nh=xh+hh;
index=1;
for i=nl:1:nh
   y(index)=0;
   index=index+1;
end
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(n) = ")
disp(y);
a=xl:1:xh;
b=hl:1:hh;
c=nl:1:nh;
subplot(1,3,3);
stem(c,y);
title('Y(n) (Linear Convolution)');
xlabel('n');
```

```
ylabel('amplitude');
```

Output:

```
>> EXP3
Enter start of signal X: 0
Enter start of singal H: -1
Enter amplitudes of signal X: [1 2 0.5 1]
Enter amplitudes of signal H: [1 2 1 -1]
Start index of output signal: -1
End index of output signal: 5
Total number of samples: 7
Y(n) =
    1.0000    4.0000    5.5000    3.0000    0.5000    0.5000   -1.0000
```



Tabular method:

Code:

```
xl=input("Enter start of signal X: ");
hl=input("Enter start of singal H: ");
x=input("Enter amplitudes of signal X: ");
h=input("Enter amplitudes of signal H: ");
```

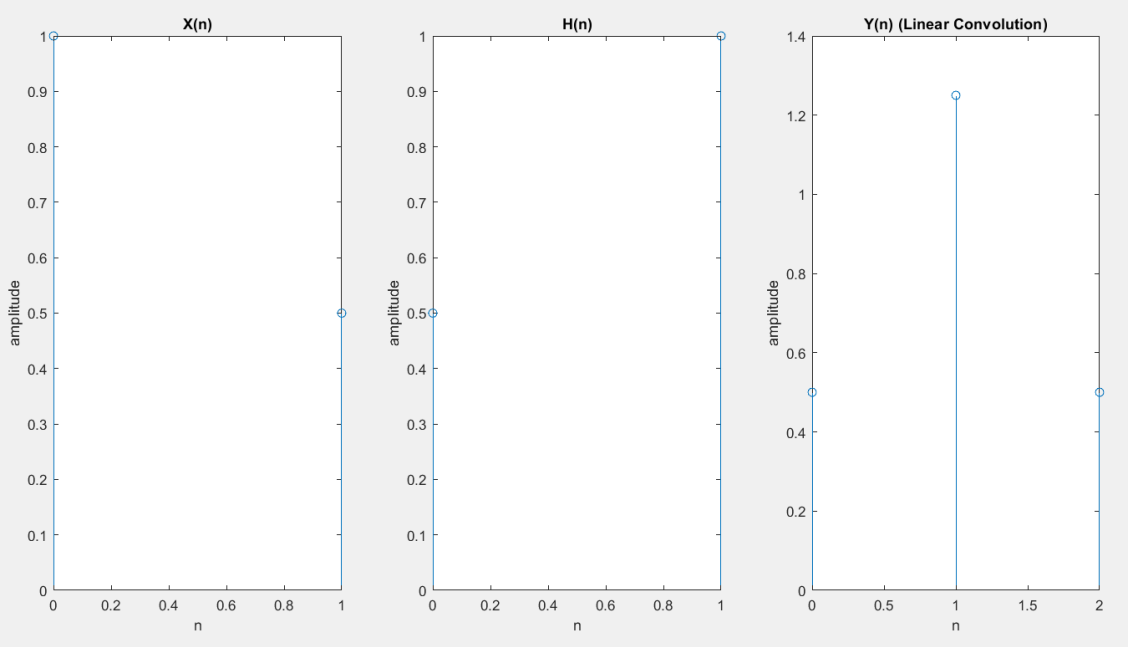
```
subplot(1,3,1);
figure(1);
stem(xl:1:xl+length(x)-1,x);
title('X(n)');
xlabel('n');
ylabel('amplitude');

subplot(1,3,2);
stem(hl:1:hl+length(h)-1,h);
title('H(n)');
xlabel('n');
ylabel('amplitude');
xh=xl+length(x)-1;
hh=hl+length(h)-1;
nl=xl+hl;
nh=xh+hh;
index=1;
for i=nl:1:nh
   y(index)=0;
   index=index+1;
end
h_start=xl-length(h)+1;
h_end=xh+length(h)-1;
ind=length(h);
temp_ind=1;
for i=h_start:1:h_end
   if(ind>0)
      final_h(temp_ind)=h(ind);
      ind=ind-1;
   else
      final_h(temp_ind)=0;
   end
   temp_ind=temp_ind+1;
end
ind=1;
for i=xl:1:xh
   x_index(ind)=i;
   ind=ind+1;
end
ind=1;
```

```
ind_fir_1=1;
disp("--------------------------------------------------------------------------------");
disp("Tabular method: ");
disp("--------------------------------------------------------------------------------");
for i=nl:1:nh
   sum=0;
   first_h=1;
   first_x=1;
   for j=h_start:1:h_end
      if(j>=xl & j<=xh)
         sum=sum+final_h(first_h)*x(first_x);
         first_x=first_x+1;
      end
      first_h=first_h+1;
   end
   y(ind)=sum;
   ind=ind+1;
   temp_final_h=final_h;
   index_temp_1=1;
   fprintf("\t\t\t");
   disp(x);
   disp(final_h);
   for k=1:1:ind_fir_1
      final_h(index_temp_1)=0;
      index_temp_1=index_temp_1+1;
   end
   mask_2=1;
   for k=h_start:1:h_end-ind_fir_1
      final_h(index_temp_1)=temp_final_h(mask_2);
      mask_2=mask_2+1;
      index_temp_1=index_temp_1+1;
   end
end
ind=1;
final_index(1)=0;
for i=nl:1:nh
   final_index(ind)=i;
   ind=ind+1;
end
subplot(1,3,3);
```

```
stem(final_index,y);
title('Y(n) (Linear Convolution)');
xlabel('n');
ylabel('amplitude');
fprintf("Start index of output signal: %d\n",nl);
fprintf("End index of output signal: %d\n",nh);
fprintf("Total number of samples: %d\n",length(y));
disp("Y(n) = ");
disp(y);
```

Output:

```
>> EXP3
Enter start of signal X: 0
Enter start of singal H: 0
Enter amplitudes of signal X: [1 0.5]
Enter amplitudes of signal H: [0.5 1]
-------------------------------------------------------------------------------------------
Tabular method:
-------------------------------------------------------------------------------------------
               1.0000    0.5000

    1.0000    0.5000         0         0

               1.0000    0.5000

         0    1.0000    0.5000         0

               1.0000    0.5000

         0         0    1.0000    0.5000

Start index of output signal: 0
End index of output signal: 2
Total number of samples: 3
Y(n) =
    0.5000    1.2500    0.5000
```



Circular convolution:

Matrix method:

Code:

```
xl=input("Enter start of signal X: ");
hl=input("Enter start of singal H: ");
x=input("Enter amplitudes of signal X: ");
h=input("Enter amplitudes of signal H: ");

index=1;
for i=xl:1:xl+length(x)-1
   index_x(index)=i;
   index=index+1;
end
index=1;
for i=hl:1:hl+length(h)-1
   index_h(index)=i;
   index=index+1;
end

subplot(1,3,1);
figure(1);
stem(index_x,x);
```

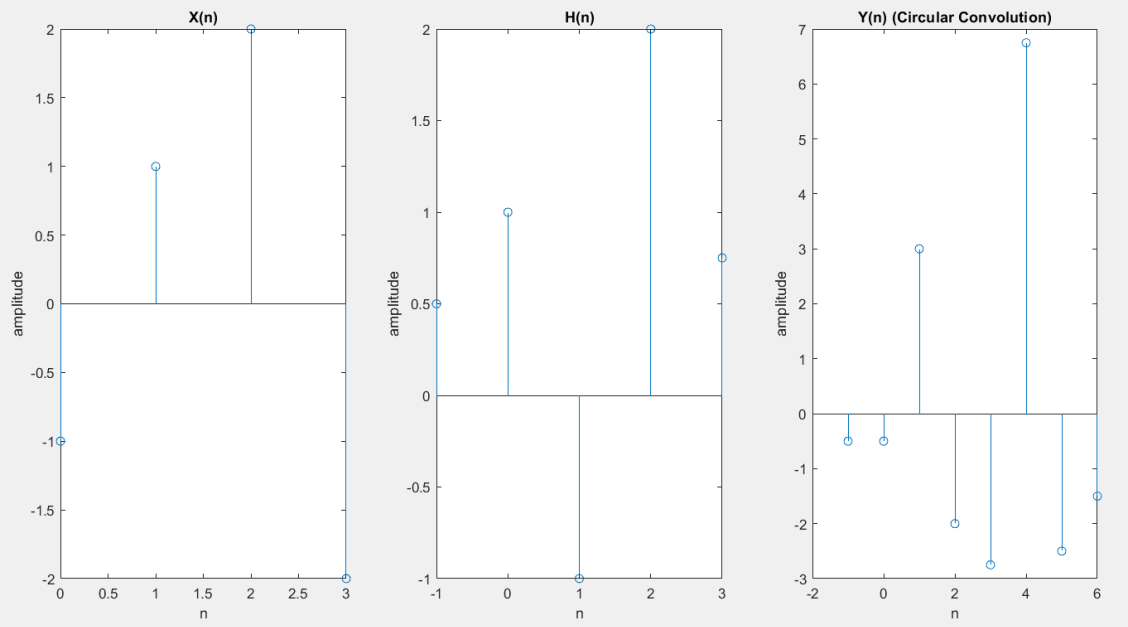
```
title('X(n)');
xlabel('n');
ylabel('amplitude');

subplot(1,3,2);
figure(1);
stem(index_h,h);
title('H(n)');
xlabel('n');
ylabel('amplitude');

index=1;
for i=0:1:10
   y(index)=2^i;
   index=index+1;
end
if xl==hl && length(x)==length(h)
   len=0;
   for i=1:1:11
      if y(i)==length(x)
         break
      end
      if(y(i)>length(x))
         len=y(i);
         break
      end
   end
   if(len~=0)
      for i=length(x):1:len-1
         x=[x,0];
         h=[h,0];
      end
   end
elseif xl==hl && length(x)~=length(h)
   len=0;
   for i=1:1:11
      if(y(i)>length(x) && y(i)>length(h))
         len=y(i);
         break
      end
```

```
    end
    for i=length(x):1:len-1
        x=[x,0];
    end
    for i=length(h):1:len-1
        h=[h,0];
    end
else
    temp=length(x)+length(h)-1;
    len=0;
    for i=1:1:11
        if(y(i)>=temp)
            len=y(i);
            break
        end
    end
    if(xl>hl)
        for i=1:1:xl-hl
            x=[0,x];
        end
        for i=length(x):1:len-1
            x=[x,0];
        end
        for i=length(h):1:len-1
            h=[h,0];
        end
    else
        for i=1:1:hl-xl
            h=[0,h];
        end
        for i=length(x):1:len-1
            x=[x,0];
        end
        for i=length(h):1:len-1
            h=[h,0];
        end
    end
end

%Matrix Method%
```

```
temp(1)=h(1);
for i=length(h):-1:2
   temp(length(h)-i+2)=h(i);
end
disp("Padded X(n): ");
disp(x);
disp("Padded H(n): ");
disp(h);
sum=0;
for i=1:1:length(h)
   sum=sum+x(i)*temp(i);
end
final(1)=sum;
for i=2:1:length(x)
   temp=circshift(temp,1);
   sum=0;
   for j=1:1:length(h)
      sum=sum+x(j)*temp(j);
   end
   final(i)=sum;
end
if xl<0 & hl<0
   final=circshift(final, -1*abs(xl-hl));
elseif xl<0
   final=circshift(final, xl);
elseif hl<0
   final=circshift(final, hl);
end
start_f=xl+hl;
end_f=start_f+length(final)-1;
fprintf("Start index of output sequence: %d\n",start_f);
fprintf("End index of output sequence: %d\n",end_f);
indices=start_f:1:end_f;
disp("Y(n) = ")
disp(final);
subplot(1,3,3);
stem(indices,final);
title('Y(n) (Circular Convolution)');
xlabel('n');
ylabel('amplitude');
```

Output:

```
>> EXP4
Enter start of signal X: 0
Enter start of singal H: -1
Enter amplitudes of signal X: [-1 1 2 -2]
Enter amplitudes of signal H: [0.5 1 -1 2 0.75]
Padded X(n):
     0    -1     1     2    -2     0     0     0

Padded H(n):
    0.5000    1.0000   -1.0000    2.0000    0.7500         0         0         0

Start index of output sequence: -1
End index of output sequence: 6
Y(n) =
   -0.5000   -0.5000    3.0000   -2.0000   -2.7500    6.7500   -2.5000   -1.5000
```



Tabular method:

Code:

```
xl=input("Enter start of signal X: ");
hl=input("Enter start of singal H: ");
x=input("Enter amplitudes of signal X: ");
h=input("Enter amplitudes of signal H: ");

index=1;
```

```
for i=xl:1:xl+length(x)-1
   index_x(index)=i;
   index=index+1;
end
index=1;
for i=hl:1:hl+length(h)-1
   index_h(index)=i;
   index=index+1;
end

subplot(1,3,1);
figure(1);
stem(index_x,x);
title('X(n)');
xlabel('n');
ylabel('amplitude');

subplot(1,3,2);
figure(1);
stem(index_h,h);
title('H(n)');
xlabel('n');
ylabel('amplitude');

index=1;
for i=0:1:10
   y(index)=2^i;
   index=index+1;
end
if xl==hl && length(x)==length(h)
   len=0;
   for i=1:1:11
      if y(i)==length(x)
         break
      end
      if(y(i)>length(x))
         len=y(i);
         break
      end
   end
```

```
    if(len~=0)
        for i=length(x):1:len-1
            x=[x,0];
            h=[h,0];
        end
    end
elseif xl==hl && length(x)~=length(h)
    len=0;
    for i=1:1:11
        if(y(i)>length(x) && y(i)>length(h))
            len=y(i);
            break
        end
    end
    for i=length(x):1:len-1
        x=[x,0];
    end
    for i=length(h):1:len-1
        h=[h,0];
    end
else
    temp=length(x)+length(h)-1;
    len=0;
    for i=1:1:11
        if(y(i)>=temp)
            len=y(i);
            break
        end
    end
    if(xl>hl)
        for i=1:1:xl-hl
            x=[0,x];
        end
        for i=length(x):1:len-1
            x=[x,0];
        end
        for i=length(h):1:len-1
            h=[h,0];
        end
    else
```

```
        for i=1:1:hl-xl
            h=[0,h];
        end
        for i=length(x):1:len-1
            x=[x,0];
        end
        for i=length(h):1:len-1
            h=[h,0];
        end
    end
end
%Graphical Method%
disp("Padded X(n): ");
disp(x);
disp("Padded H(n): ");
disp(h);
disp("-----------------------------------------------------------------------------------------");
disp("Tabular method: ");
disp("-----------------------------------------------------------------------------------------");
index_lh=1;
flag=1;
left_h(1)=99;
for i=hl:1:-1
    left_h(index_lh)=h(index_lh);
    index_lh=index_lh+1;
    flag=flag+1;
end
flag2=1;
index_temp=1;
for i=-1*hl+2:1:length(h)
    right_h(index_temp)=h(i);
    index_temp=index_temp+1;
    flag2=flag2+1;
end
index=1;
if(left_h(1)==99)
    final(index)=h(1);
    index=index+1;
    for i=length(right_h):-1:1
        final=[right_h(length(right_h)-i+1),final,right_h(i)];
```

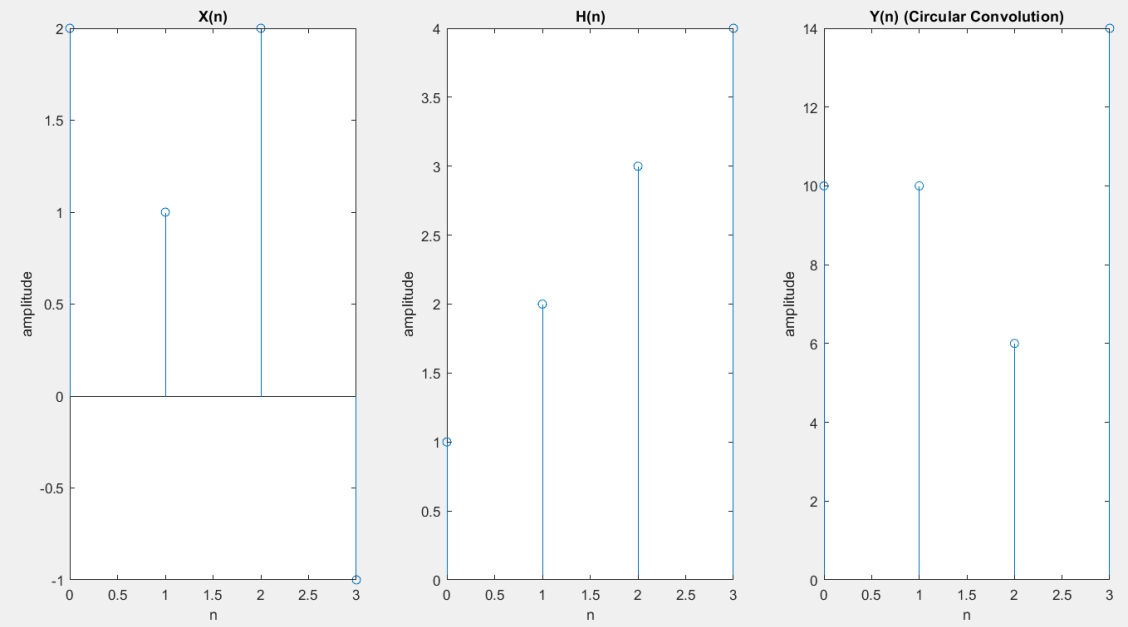
```
    end
else
    final(index)=h(length(left_h)+1);
    for i=1:1:length(left_h)
        final=[final,left_h(i)];
        index=index+1;
    end
    for i=length(right_h):-1:1
        final=[right_h(length(right_h)-i+1),final,right_h(i)];
    end
end

if(xl<0 | hl<0)
    for i=1:1:abs(xl-hl)-1
        final=circshift(final,-1);
    end
end
start_final=length(right_h)+1;
end_final=xl+length(x)+start_final-1;
index=1;
for i=1:1:length(x)
    it=1;
    sum=0;
    for j=start_final:1:end_final
        sum=sum+x(it)*final(j);
        it=it+1;
    end
    fprintf("\t\t\t\t  ")
    disp(x);
    disp(final);
    final_circ(index)=sum;
    index=index+1;
    final=circshift(final,1);
end
index=1;
if(hl<0)
    for i=hl:1:length(final_circ)-2
        indices(index)=i;
        index=index+1;
    end
```

```
else
   for i=hl:1:length(final_circ)-1
      indices(index)=i;
      index=index+1;
   end
end
subplot(1,3,3);
figure(1);
stem(indices,final_circ);
title('Y(n) (Circular Convolution)');
xlabel('n');
ylabel('amplitude');
fprintf("Start index of output sequence: %d\n",xl+hl);
fprintf("End index of output sequence: %d\n",xl+hl+length(final_circ)-1);
disp("Y(n) = ")
disp(final_circ);
```

Output:

```
>> EXP4
Enter start of signal X: 0
Enter start of singal H: 0
Enter amplitudes of signal X: [2 1 2 -1]
Enter amplitudes of signal H: [1 2 3 4]
Padded X(n):
     2     1     2    -1

Padded H(n):
     1     2     3     4

-------------------------------------------------------------------------------------
Tabular method:
-------------------------------------------------------------------------------------
                       2     1     2    -1

     4     3     2     1     4     3     2

                       2     1     2    -1

     2     4     3     2     1     4     3

                       2     1     2    -1

     3     2     4     3     2     1     4

                       2     1     2    -1

     4     3     2     4     3     2     1
```

```
Start index of output sequence: 0
End index of output sequence: 3
Y(n) =
    10    10     6    14
```



Correlation:

Cross correlation:

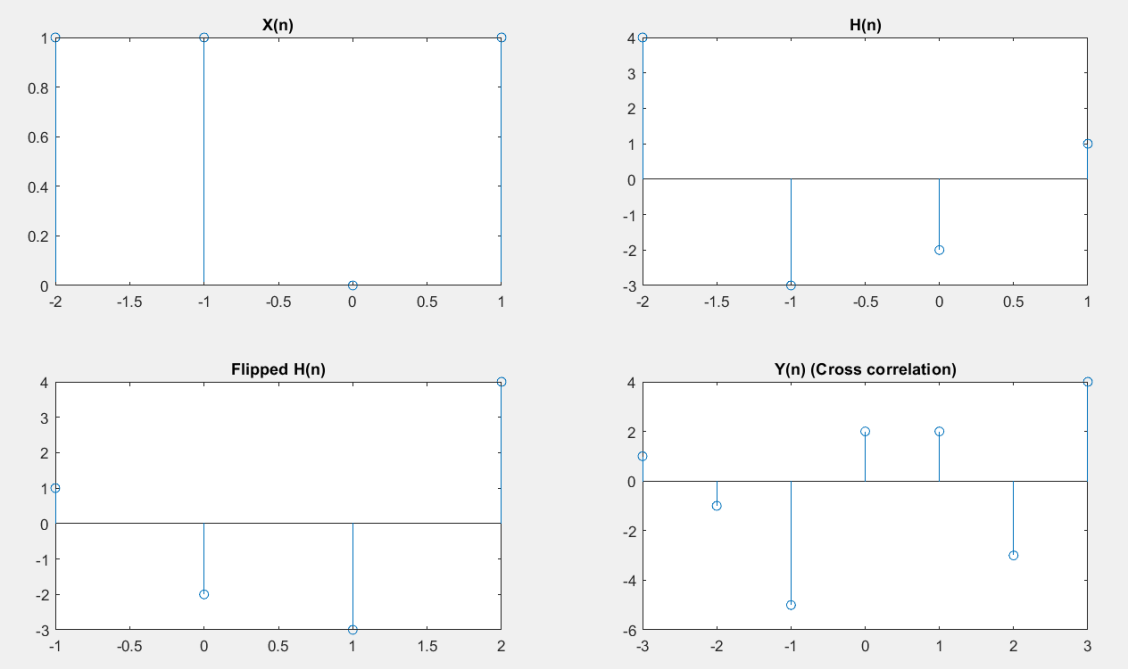
Code:

```
%Cross Correlation%
xl=input("Enter start of signal X: ");
hl=input("Enter start of singal H: ");
x=input("Enter amplitudes of signal X: ");
h=input("Enter amplitudes of signal H: ");
xh=xl+length(x)-1;
hh=hl+length(h)-1;

for i=1:1:length(h)
   new_h(i)=h(length(h)-i+1);
end
figure(1);
subplot(2,2,1);
```

```
stem([xl:xh], x);
title("X(n)");
subplot(2,2,2);
stem([hl:hh], h);
title("H(n)");
subplot(2,2,3);
stem([-1*hh:-1*hl], new_h);
title("Flipped H(n)");
nl=xl+(-1)*hh;
nh=xh+(-1)*hl;
disp("Flipped H(n): ");
disp(new_h);
fprintf("Start index of output sequence: %d\n",nl);
fprintf("End indedx of output sequence: %d\n",nh);
y=zeros(length(h)+length(x)-1,1)';
for i=1:1:length(x)
   for j=1:1:length(new_h)
      y(i+j-1)=y(i+j-1)+(x(i)*new_h(j));
   end
end
subplot(2,2,4);
stem([nl:nh], y);
title("Y(n) (Cross correlation)");
disp("Y(n): ");
disp(y);
```

Output:

```
>> EXP3Correlation
Enter start of signal X: -2
Enter start of singal H: -2
Enter amplitudes of signal X: [1 1 0 1]
Enter amplitudes of signal H: [4 -3 -2 1]
Flipped H(n):
     1    -2    -3     4

Start index of output sequence: -3
End indedx of output sequence: 3
Y(n):
     1    -1    -5     2     2    -3     4
```



Auto correlation:

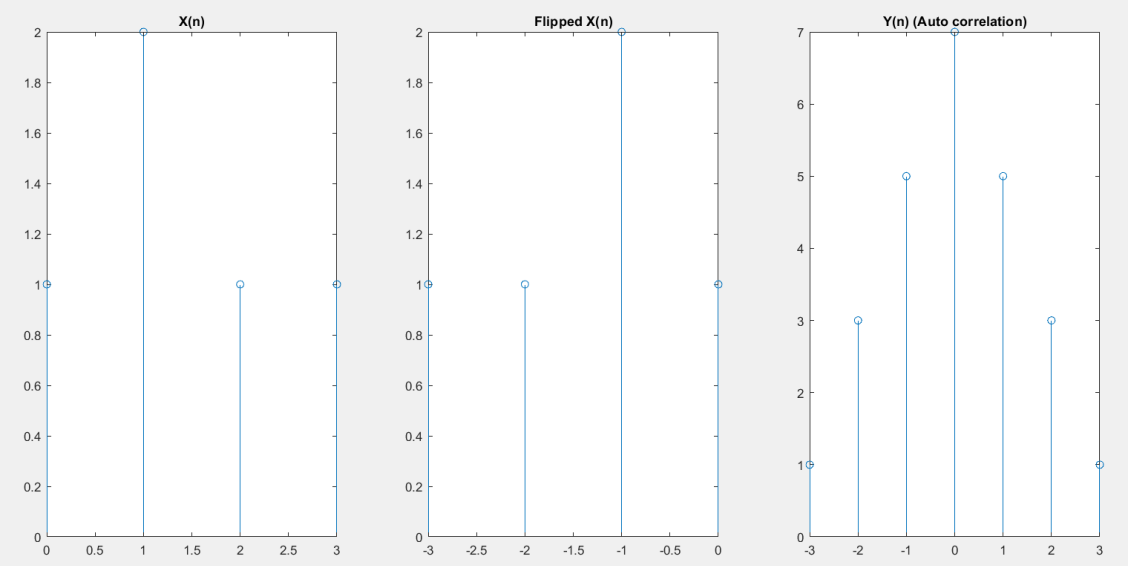
Code:

```
%Auto Correlation%
xl=input("Enter start of signal X: ");
x=input("Enter amplitudes of signal X: ");
xh=xl+length(x)-1;
hl=xl;
hh=xh;
h=x;
for i=1:1:length(h)
    new_h(i)=h(length(h)-i+1);
end
figure(2);
subplot(1,3,1);
stem([xl:xh], x);
title("X(n)");
subplot(1,3,2);
stem([-1*hh:-1*hl], new_h);
title("Flipped X(n)");
disp("Flipped X(n): ");
```

```
disp(new_h);
nl=xl+(-1)*hh;
nh=xh+(-1)*hl;
fprintf("Start index of output sequence: %d\n",nl);
fprintf("End indedx of output sequence: %d\n",nh);
y=zeros(length(h)+length(x)-1,1)';
for i=1:1:length(x)
   for j=1:1:length(new_h)
      y(i+j-1)=y(i+j-1)+(x(i)*new_h(j));
   end
end
subplot(1,3,3);
stem([nl:nh], y);
title("Y(n) (Auto correlation)");
disp("Y(n): ");
disp(y);
```

Output:

```
>> EXP3Correlation
Enter start of signal X: 0
Enter amplitudes of signal X: [1 2 1 1]
Flipped X(n):
     1     1     2     1

Start index of output sequence: -3
End indedx of output sequence: 3
Y(n):
     1     3     5     7     5     3     1
```



Conclusion:- Understood the concept of convolution and correlation. Explored various types of convolution and correlation such as linear convolution, circular convolution, cross correlation and auto correlation also explored various methods of calculating convolution such as graphical, tabular and matrix method. Implemented linear and circular convolution using matrix and tabular method and cross and auto correlation using matrix method.

Date: 27 / 4 / 2022 **Signature of faculty in-charge**

Post Lab Descriptive Questions

1. Explain the role of convolution in signal processing.

ANS) a) Characterizing a linear time-invariant (LTI) system in terms of its transfer function.

b) Determining the output of an LTI system when its input is known.

c) Synthesizing a new customizable pattern using the impulse response of a system.

d) Convolution can also be used for filtering the signals.

e) Synthesized Seismographs: Seismology is a branch of geophysics primarily concerned with the study of earthquakes and other cases of elastic waves traveling through the earth or even other planets.

2. Explain the difference between linear and circular convolution?

ANS)

Linear Convolution	Circular Convolution
1) Linear convolution is a mathematical operation done to calculate the output of any Linear-Time Invariant (LTI) system given its input and impulse response.	1) Circular convolution is essentially the same process as linear convolution. 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.
2) We can represent Linear Convolution as $y(n)=x(n)*h(n)$. Here, y(n) is the output, x(n) is the input signal, and h(n) is the impulse response of the LTI system.	2) 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.
3) In linear convolution, both the sequences may or may not be of equal sizes.	3) In circular convolution, both the sequences must be of equal sizes.
4) Graphically, when we perform linear convolution, there is a linear shift taking place.	4) Graphically, when we perform circular convolution, there is a circular shift taking place. Alternatively, we can call it rotation.
5) It is possible to find the response of a filter using linear convolution.	5) It is possible to find the response of a filter using circular convolution after zero padding.
6) Linear convolution may or may not result in a periodic output signal.	6) 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.

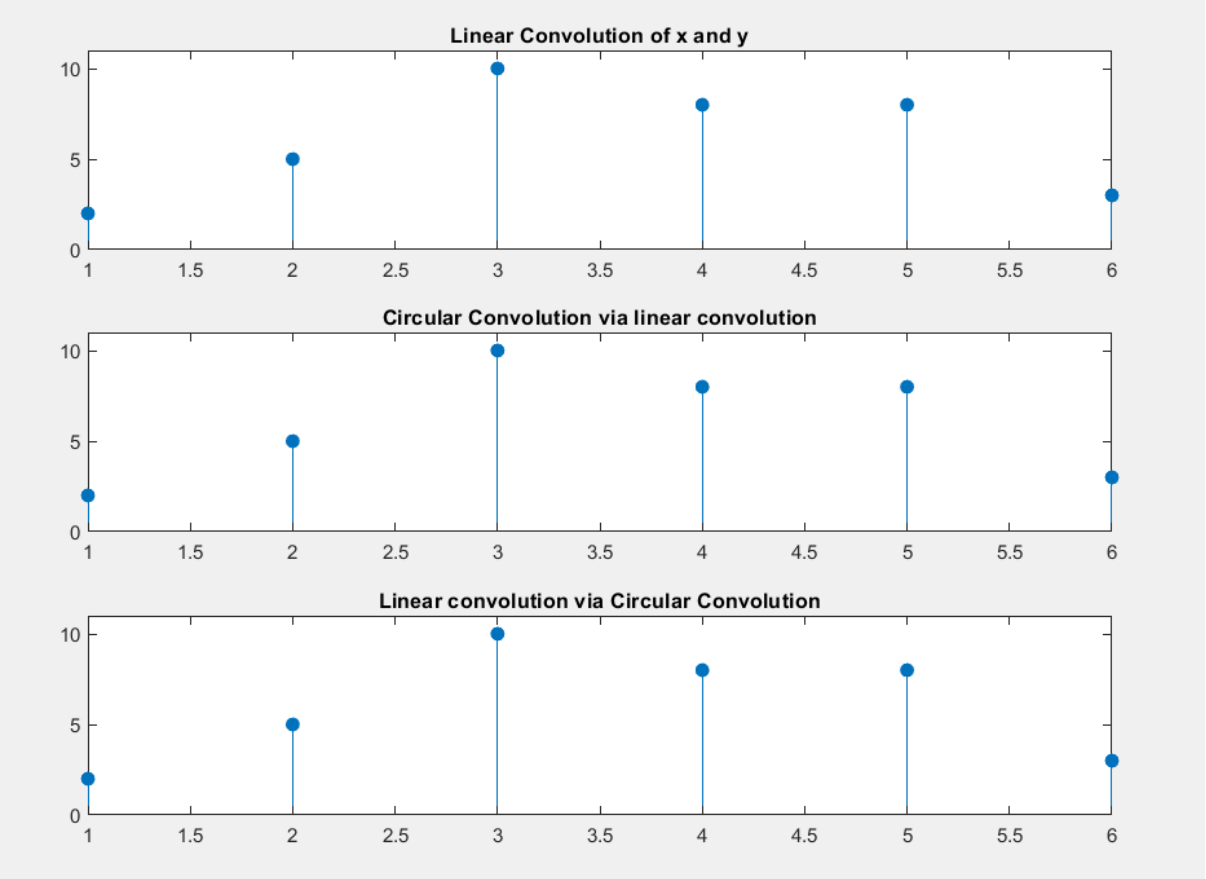
ANS) Linear and circular convolution are fundamentally different operations. However, there are conditions under which linear and circular convolution are equivalent. Establishing this equivalence has important implications. For two vectors, x and y, the circular convolution is equal to the inverse discrete 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.
1) Create two vectors, x and y, and compute the linear convolution of the two vectors.
2) Pad both vectors with zeros to length N1+N2-1. Obtain the DFT of both vectors, multiply the DFTs, and obtain the inverse DFT of the product.
3) Pad the vectors to length 12 and obtain the circular convolution using the inverse DFT of the product of the DFTs. Retain only the first N1+N2-1 elements to produce an equivalent result to linear convolution.

Code:

```
x = [2 1 2 1];
y = [1 2 3];
clin = conv(x,y);
xpad = [x zeros(1,6-length(x))];
ypad = [y zeros(1,6-length(y))];
ccirc = ifft(fft(xpad).*fft(ypad));
subplot(3,1,1)
stem(clin,'filled')
ylim([0 11])
title('Linear Convolution of x and y')
subplot(3,1,2)
stem(ccirc,'filled')
ylim([0 11])
title('Circular Convolution via linear convolution')
N = length(x)+length(y)-1;
xpad = [x zeros(1,12-length(x))];
ypad = [y zeros(1,12-length(y))];
ccirc = ifft(fft(xpad).*fft(ypad));
ccirc = ccirc(1:N);
ccirc2 = cconv(x,y,6);
subplot(3,1,3)
stem(ccirc2,'filled')
ylim([0 11])
title('Linear convolution via Circular Convolution')
```

Output:


Linear Convolution of x and y
Circular Convolution via linear convolution
Linear convolution via Circular Convolution
0
5
10
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6