School of Engineering and Applied Science (SEAS)
Ahmedabad University

BTech(ICT) Semester VI:Digital Signal Processing

Laboratory Assignment-2

Enrollment No:AU1841131 **Name:Mansi Dobariya**

AIM :: Lab2 helps to revise the concept of signal and systems. Solving the properties of signal like linear,auto,cross convolution and infinite sequence convolution of step signal in MATLAB . that's clearly shown in plots to understand better.

1. Find the linear convolution of following finite length sequences using available command and also without command by developing your own function. Plot required outputs along with input sequence.
   $a) x(n) = \{1, \mathbf{2}, 2, 1\},\ h(n) = \{1, -1, \mathbf{2}\}$
   $b) x(n) = \{-2, 0, 1, \mathbf{-1}, 3\},\ h(n) = \{1, \mathbf{2}, 0, -1\}$
   $c) x(n) = \{1, 2, 3, \mathbf{1}\},\ h(n) = \{\mathbf{1}, 2, 1, -1\}$
   $d) x(n) = \{\mathbf{9}, 1, 5, 4\},\ h(n) = \{0, \mathbf{2}, 2\}$
   Solution Problem-1

   (a) Matlab Script:

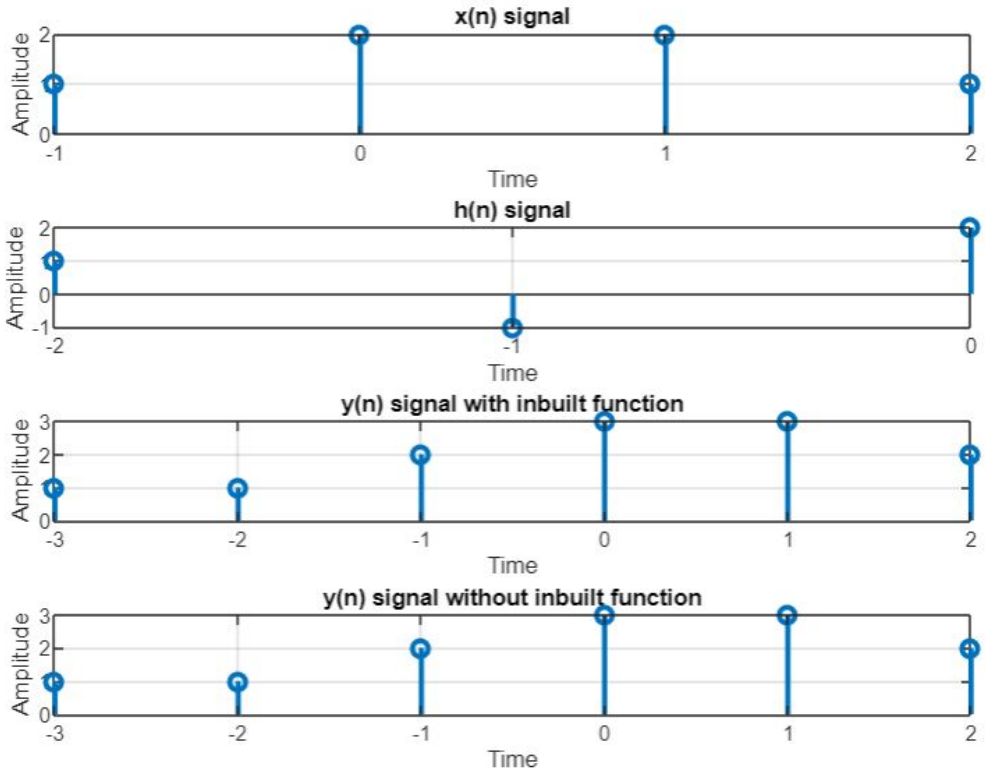
```
%linear y(n)=x(n)*h(n)
clc;
close all;
x=input('Enter the first array x(n) :');
xptr=input('Enter index of zero pointer x(n) :');
h=input('Enter the second array h(n) :');
hptr=input('Enter index of zero pointer h(n) :');
% -2 -1 0 1 2 3 so x=[-2:3]
% 0 1 2 3 so h=[0:3]
%range of y will be [-2+0:3+3]= [-2:6]
x_lower_index=1-xptr;        %index of least left sided element
x_upper_index=length(x)-xptr;%index of most right sided element
h_lower_index=1-hptr;        %index of least left sided element
h_upper_index=length(h)-hptr;%index of most right sided element
y_lower_index=x_lower_index+h_lower_index;  %lower index of convoltuion as y
y_upper_index=x_upper_index+h_upper_index; %upper index of convoltuion as y
m=[];
y=[];
subplot(4,1,1)
stem(x_lower_index:x_upper_index,x,'linewidth' , 2);
hold on;
grid on;
title('x(n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(x_lower_index:1:x_upper_index);

subplot(4,1,2)
stem(h_lower_index:h_upper_index,h,'linewidth' , 2);
hold on;
grid on;
title('h(n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(h_lower_index:1:h_upper_index);

```

```
convolution=conv(x,h);          %inbuilt func

subplot(4,1,3)
stem(y_lower_index:y_upper_index,convolution,'linewidth' , 2);
hold on;
grid on;
title('y(n) signal with inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);
%Matrix creation              %without inbuilt func
%h*x(i)
for i=1:length(x)
    g=h.*x(i);
    m=[m;g];%this sequence store array at new row so one matrix will be formed
end
%summation of diagonal(right) elements an store it into one array
%11 12 13 14
%21 22 23 24
%31 32 33 34
%r=3,c=4,k=7,diagonal_index_sum=2(1+1 first element) because max we have sum=k
[r c]=size(m);
k=r+c;
diagonal_index_sum=2;
element=0;
%column and row wise searching for sum=diagonal_index_sum
while(diagonal_index_sum<=k)
    for i=1:r
        for j=1:c
            if((i+j)==diagonal_index_sum)
               element=element+m(i,j);
            end
        end
    end
    diagonal_index_sum=diagonal_index_sum+1;%for next diagonal
    y=[y element];
    element=0;
end
disp(y);
subplot(4,1,4)
stem(y_lower_index:y_upper_index,y, 'linewidth' , 2);
hold on;
grid on;
title('y(n) signal without inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);

```

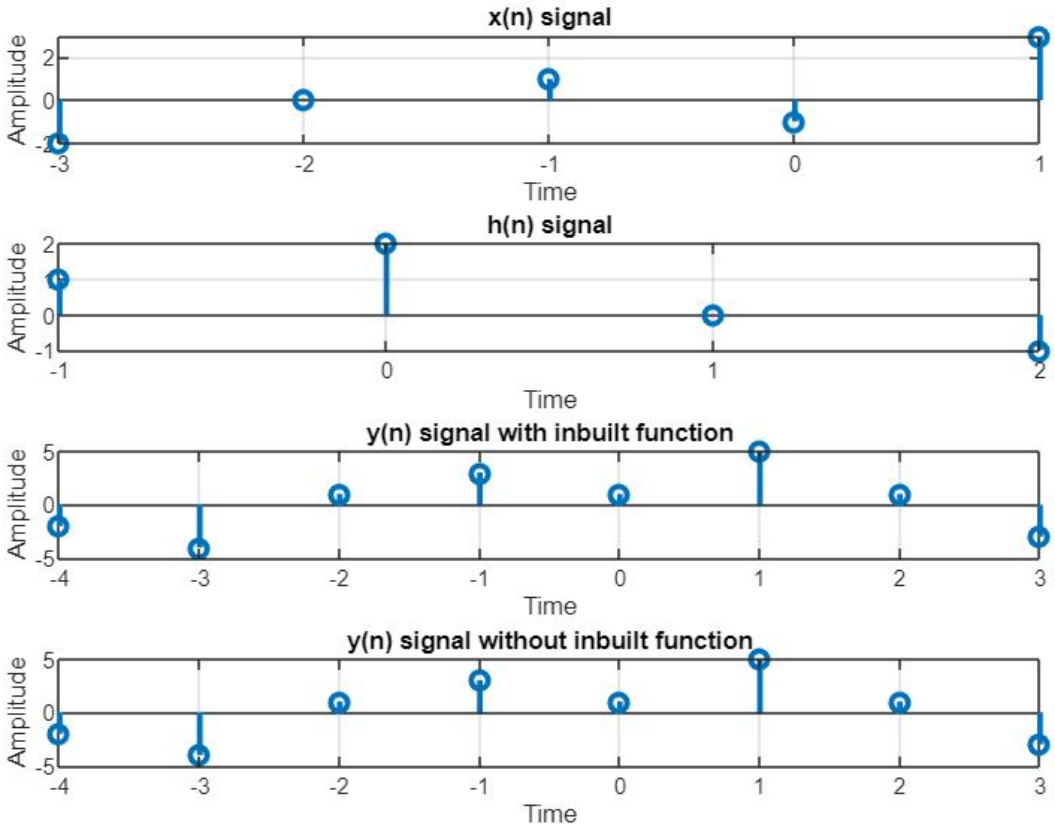
(b) Approach:
firstly,calculated lower and upper length from array given by user. Using these range calculated length of convolution array.multiplying one by one element of x(n) with whole array h(n) and storing into one matrix m.Then,I did addition of right diagonal from first element ($m_{1X1}, where\ diagonalIndexSum = 1 + 1 = 2$)to last element ($m_{rXc}$) .Sum of those element which satisfies diagonal index sum (2,3,4,..,r+c). After that plotted the value of y in appropriate range.

(c) Simulation Output:

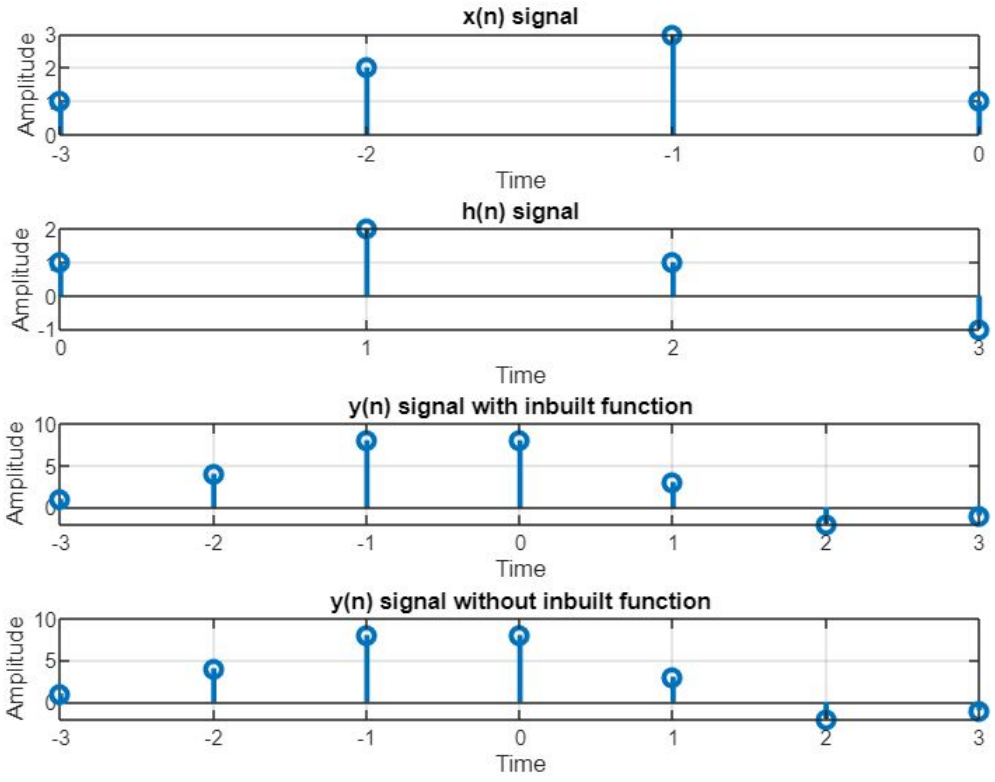
```
Enter the first array x(n) :
[1 2 2 1]
Enter index of zero pointer x(n) :
2
Enter the second array h(n) :
[1 -1 2]
Enter index of zero pointer h(n) :
3
     1     1     2     3     3     2
```



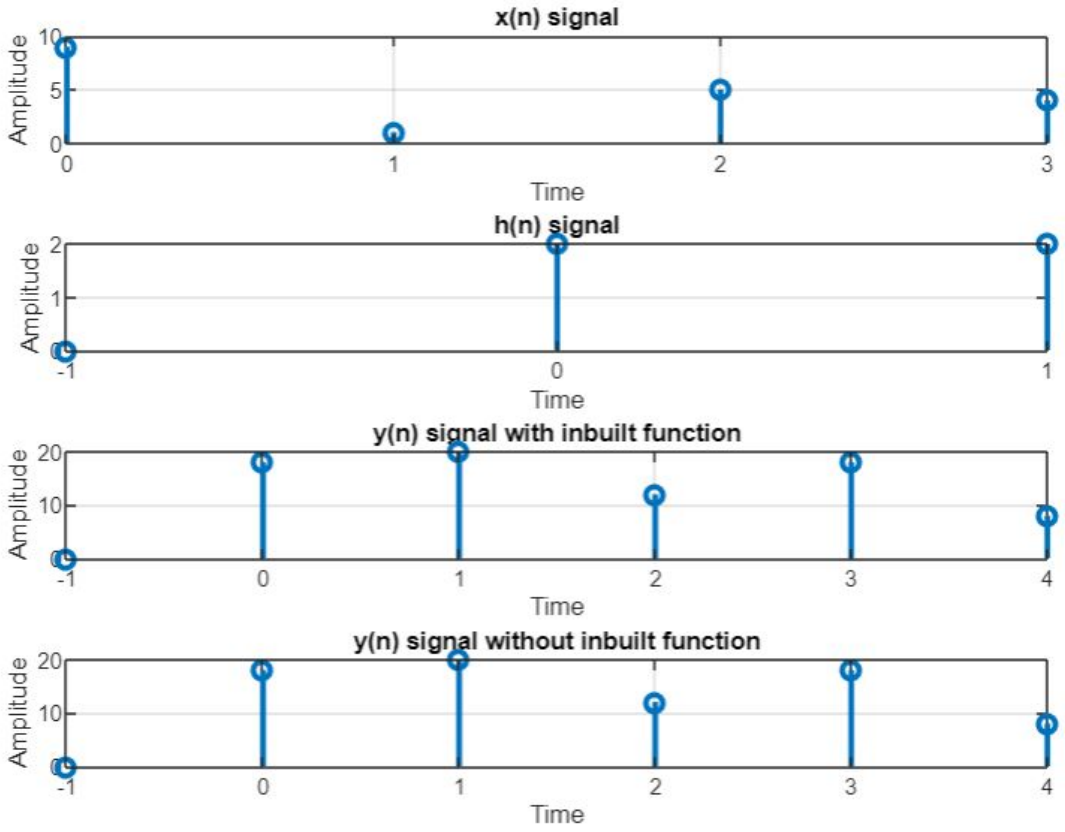
```
Enter the first array x(n) :
[-2 0 1 -1 3]
Enter index of zero pointer x(n) :
4
Enter the second array h(n) :
[1 2 0 -1]
Enter index of zero pointer h(n) :
2
    -2    -4     1     3     1     5     1    -3
```



```
Enter the first array x(n) :
[1 2 3 1]
Enter index of zero pointer x(n) :
4
Enter the second array h(n) :
[1 2 1 -1]
Enter index of zero pointer h(n) :
1
     1     4     8     8     3    -2    -1
```



```
Enter the first array x(n) :
[9 1 5 4]
Enter index of zero pointer x(n) :
1
Enter the second array h(n) :
[0 2 2]
Enter index of zero pointer h(n) :
2
     0    18    20    12    18     8
```



2. Find cross-correlation between two sequences using available command and also without command by developing your own function. Verify your program for following sequence and Plot required outputs
   $a) x(n) = \{1, \mathbf{2}, 2, 1\},\ h(n) = \{1, -1, \mathbf{2}\}$

Solution Problem-2

(a) Matlab Script:

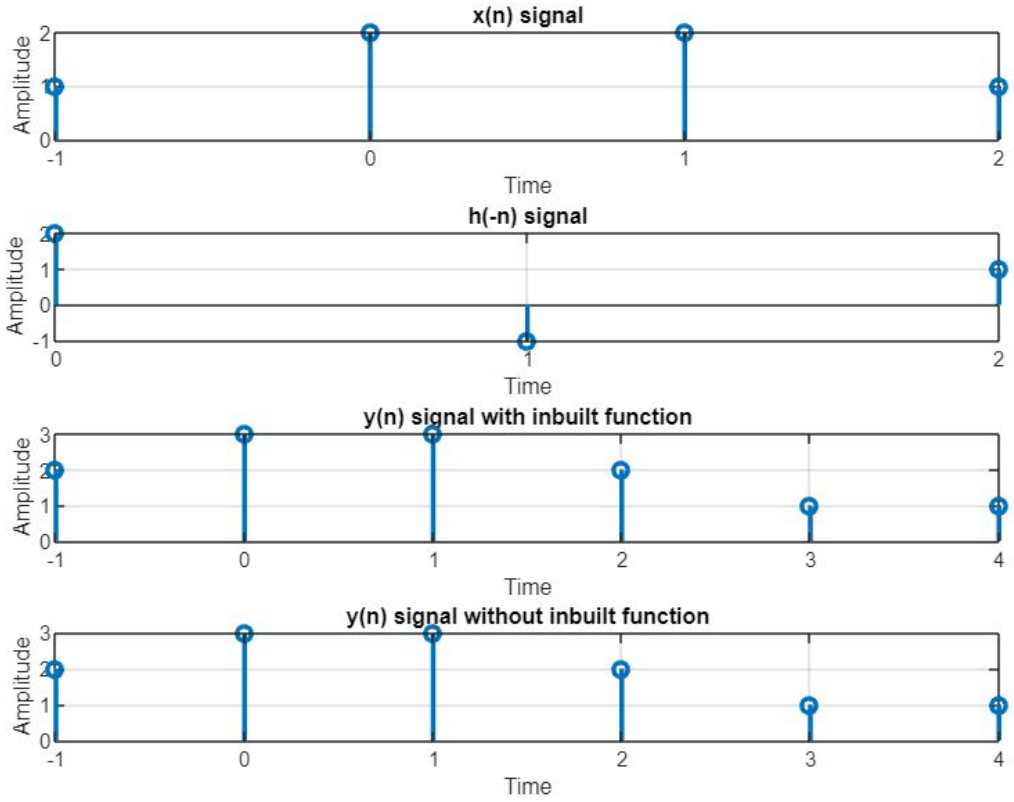
```
%cross y(n)=x(n)*h(-n)
clc;
close all;
x=input('Enter the first array x(n) :');
xptr=input('Enter index of zero pointer x(n) :');
h=input('Enter the second array h(n) :');
hptr=input('Enter index of zero pointer h(n) :');
% -2 -1 0 1 2 3 so x=[-2:3]
% 0 1 2 3 so h=[0:3]
%range of y will be [-2+0:3+3]= [-2:6]
x_lower_index=1-xptr;        %index of least left sided element
x_upper_index=length(x)-xptr;%index of most right sided element
h_lower_index=1-hptr;        %index of least left sided element
h_upper_index=length(h)-hptr;%index of most right sided element
%changing value of range -ve to +ve and vice-versa
dummy=h_lower_index;
h_lower_index=-h_upper_index
h_upper_index=-dummy;
%flip the array
h=fliplr(h);
y_lower_index=x_lower_index+h_lower_index; %lower index of convoltuion as y
y_upper_index=x_upper_index+h_upper_index; %upper index of convoltuion as y
m=[];
y=[];
subplot(4,1,1)
stem(x_lower_index:x_upper_index,x,'linewidth' , 2);
hold on;
grid on;
title('x(n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(x_lower_index:1:x_upper_index);

subplot(4,1,2)
stem(h_lower_index:h_upper_index,h,'linewidth' , 2);
hold on;
grid on;
title('h(-n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(h_lower_index:1:h_upper_index);

convolution=conv(x,h);        %inbuilt func

subplot(4,1,3)
stem(y_lower_index:y_upper_index,convolution,'linewidth' , 2);
hold on;
grid on;
title('y(n) signal with inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);
%Matrix creation              %without inbuilt func
%h*x(i)
for i=1:length(x)
    g=h.*x(i);
    m=[m;g];%this sequence store array at new row so one matrix will be formed
end
%summation of diagonal(right) elements an store it into one array
%11 12 13 14
%21 22 23 24
%31 32 33 34
%r=3,c=4,k=7,diagonal_index_sum=2(1+1 first element) because max we have sum=k
[r c]=size(m);
k=r+c;
diagonal_index_sum=2;
```

```
element=0;
%column and row wise searching for sum=diagonal_index_sum
while(diagonal_index_sum<=k)
    for i=1:r
        for j=1:c
            if((i+j)==diagonal_index_sum)
               element=element+m(i,j);
            end
        end
    end
    diagonal_index_sum=diagonal_index_sum+1;%for next diagonal
    y=[y element];
    element=0;
end
disp(y);
subplot(4,1,4)
stem(y_lower_index:y_upper_index,y,'linewidth' , 2);
hold on;
grid on;
title('y(n) signal without inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);


```

(b) Approach:
firstly,calculated lower and upper length from array given by user and folding the second signal h(n) , Using h(-n) these range calculated length of convolution array.multiplying one by one element of x(n) with whole array h(-n) and storing into one matrix m.Then,I did addition of right diagonal from first element ($m_{1X1}, where\ diagonalIndexSum = 1 + 1 = 2$)to last element ($m_{rXc}$) .Sum of those element which satisfies diagonal index sum (2,3,4,..,r+c). After that plotted the value of y in appropriate range.

(c) Simulation Output:

```
Enter the first array x(n) :
[1 2 2 1]
Enter index of zero pointer x(n) :
2
Enter the second array h(n) :
[1 -1 2]
Enter index of zero pointer h(n) :
3

    2     3     3     2     1     1
```



3. Find Auto-correlation of finite length sequences using available command and also without command by developing your own function. Plot required outputs along with input sequence.

Solution Problem-3

(a) Matlab Script:

```
%Auto corr y(n)=x(n)*x(-n)
clc;
close all;
x=input('Enter the first array x(n) :');
xptr=input('Enter index of zero pointer x(n) :');
%taking x's value as h and after folding these sequence so h will be x(-n)
h=x;
hptr=xptr;
x_lower_index=1-xptr;
x_upper_index=length(x)-xptr;
h_lower_index=1-hptr;
h_upper_index=length(h)-hptr;
%changing value of range -ve to +ve and vice-versa
dummy=h_lower_index;
h_lower_index=-h_upper_index;
h_upper_index=-dummy;
%flip the array
h=fliplr(h);
y_lower_index=x_lower_index+h_lower_index; %lower index of convoltuion as y
y_upper_index=x_upper_index+h_upper_index; %upper index of convoltuion as y
m=[];
y=[];
subplot(4,1,1)
stem(x_lower_index:x_upper_index,x,'linewidth' , 2);
hold on;
grid on;
title('x(n) signal');
xlabel('Time');
```

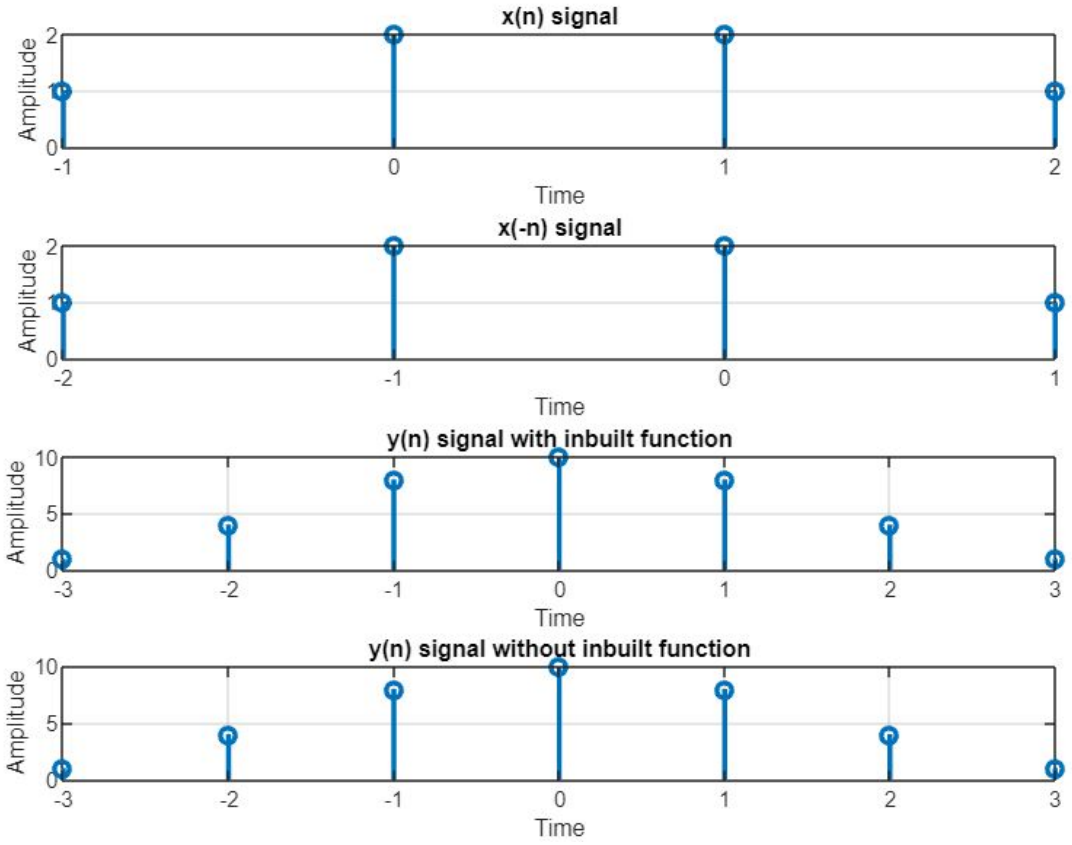
```
ylabel('Amplitude');
xticks(x_lower_index:1:x_upper_index);

subplot(4,1,2)
stem(h_lower_index:h_upper_index,h,'linewidth' , 2);
hold on;
grid on;
title('x(-n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(h_lower_index:1:h_upper_index);

convolution=xcorr(x,h);         %inbuilt func

subplot(4,1,3)
stem(y_lower_index:y_upper_index,convolution,'linewidth' , 2);
hold on;
grid on;
title('y(n) signal with inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);

%Matrix creation              %without inbuilt func
%h*x(i)
for i=1:length(x)
    g=h.*x(i);
    m=[m;g];%this sequence store array at new row so one matrix will be formed
end
%summation of diagonal(right) elements an store it into one array
%11 12 13 14
%21 22 23 24
%31 32 33 34
%r=3,c=4,k=7,diagonal_index_sum=2(1+1 first element) because max we have sum=k
[r c]=size(m);
k=r+c;
diagonal_index_sum=2;
element=0;
%column and row wise searching for sum=diagonal_index_sum
while(diagonal_index_sum<=k)
    for i=1:r
        for j=1:c
            if((i+j)==diagonal_index_sum)
               element=element+m(i,j);
            end
        end
    end
    diagonal_index_sum=diagonal_index_sum+1;%for next diagonal
    y=[y element];
    element=0;
end
disp(y);
subplot(4,1,4)
stem(y_lower_index:y_upper_index,y,'linewidth' , 2);
hold on;
grid on;
title('y(n) signal without inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);

```

(b) Approach:
firstly,calculated lower and upper length from array given by user and folding the first signal,x(-n) as h(n), Using h(n) these range calculated length of convolution array.multiplying one by one element of x(n) with whole array x(-n) and storing into one matrix m.Then,I did addition of right diagonal from first element

$(m_{1X1}, where\ diagonalIndexSum = 1 + 1 = 2)$to last element (m_{rXc}) .Sum of those element which satisfies diagonal index sum (2,3,4,..,r+c). After that plotted the value of y in appropriate range.

(c) Simulation Output:

```
Enter the first array x(n) :
[1 2 2 1]
Enter index of zero pointer x(n) :
2
    1     4     8    10     8     4     1
```



4. Find the linear convolution for following infinite length sequences and Plot required outputs.
   a. X(n) = u(n), h(n) = u(n)
   b. X(n) = cos $(2n\pi/4)$u(n) , h(n) = u(n)

Solution Problem-4

(a) Matlab Script a) X(n) = u(n), h(n) = u(n):

```
clc;
```

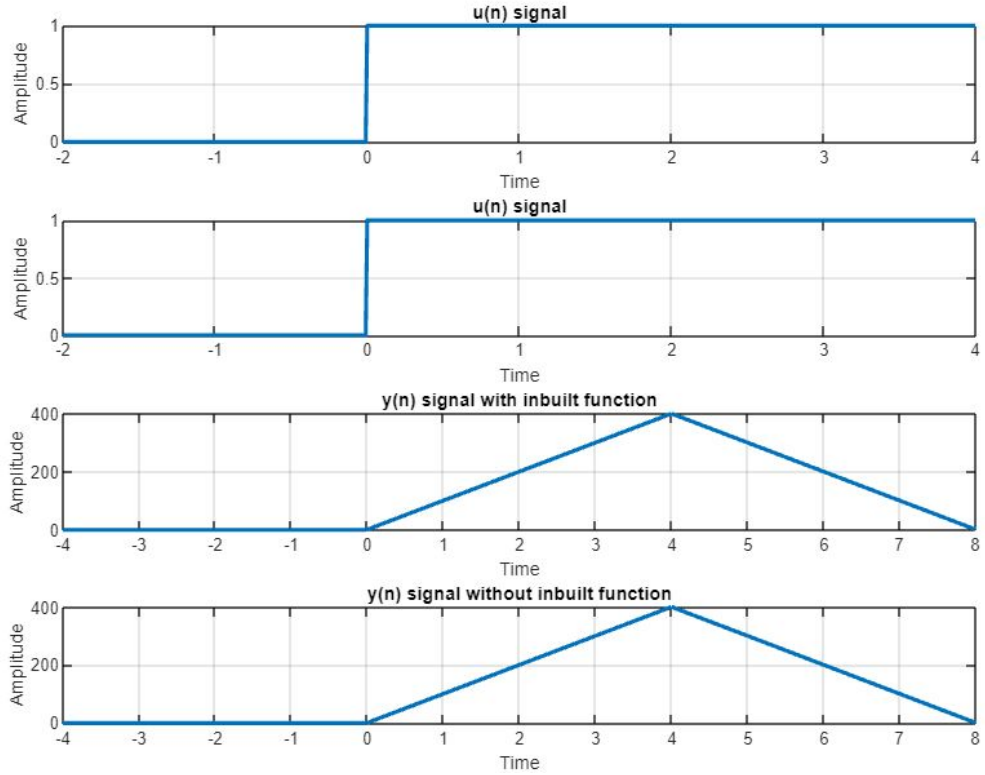
```
close all;
x_lower_index=input('Enter the lower index of Unit step signal :');
x_upper_index=input('Enter the upper index of Unit step signal :');
dn=0.01;
xn=x_lower_index:dn:x_upper_index;
x=zeros(1,length(xn));
h=zeros(1,length(xn));
for i=1:length(xn)
    x(i)=unit_step(xn(i));          %unit step function
    h(i)=x(i);                      %unit step function
end

y_lower_index=2*x_lower_index;  %lower index of convolution as y
y_upper_index=2*x_upper_index; %upper index of convolution as y
m=[];
y=[];
subplot(4,1,1)
plot(xn,x,'linewidth' , 2);
hold on;
grid on;
title('u(n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(x_lower_index:1:x_upper_index);

subplot(4,1,2)
plot(xn,h,'linewidth' , 2);
hold on;
grid on;
title('u(n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(x_lower_index:1:x_upper_index);

convolution=conv(x,h);          %inbuilt func

yn=y_lower_index:dn:y_upper_index;
subplot(4,1,3)
plot(yn,convolution,'linewidth' , 2);
hold on;
grid on;
title('y(n) signal with inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);
%Matrix creation              %without inbuilt func
%h*x(i)
for i=1:length(x)
    g=h.*x(i);
    m=[m;g];%this sequence store array at new row so one matrix will be formed
end
%summation of diagonal(right) elements an store it into one array
%11 12 13 14
%21 22 23 24
%31 32 33 34
%r=3,c=4,k=7,diagonal_index_sum=2(1+1 first element) because max we have sum=k
[r c]=size(m);
k=r+c;
diagonal_index_sum=2;
element=0;
%column and row wise searching for sum=diagonal_index_sum
while(diagonal_index_sum<=k)
    for i=1:r
        for j=1:c
            if((i+j)==diagonal_index_sum)
               element=element+m(i,j);
            end
        end
    end
    diagonal_index_sum=diagonal_index_sum+1;%for next diagonal
```

```
    y=[y element];
    element=0;
end
subplot(4,1,4)
plot(yn,y, 'linewidth' , 2);
hold on;
grid on;
title('y(n) signal without inbuilt function');
xlabel('Time');
ylabel('Amplitude');
xticks(y_lower_index:1:y_upper_index);
%function to calc unit step if value is greater or equal to 0 then return 1
    elsewhere 0
function z = unit_step(x)
    if x >= 0
        z=1;
    else
        z=0;
    end
end

```

(b) Approach a):
Taking lower and upper index from user and then calculated Unit Step signal through own function ,to calc unit step if value is greater or equal to 0 then return 1 elsewhere 0.Using this discrete signal, calculated length of convolution array and multiplying one by one element of x(n) with whole array h(n) and storing into one matrix m.Then,I did addition of right diagonal from first element ($m_{1X1}, where\ diagonalIndexSum = 1 + 1 = 2$)to last element ($m_{rXc}$) .Sum of those element which satisfies diagonal index sum (2,3,4,..,r+c). After that plotted the value of y in appropriate range.

(c) Simulation Output a):

```
Enter the lower index of Unit step signal :
-2
Enter the upper index of Unit step signal :
4
```



(d) Matlab Script b) X(n) = cos (2nπ/4)u(n) , h(n) = u(n):

```
clc;
close all;
x_lower_index=input('Enter the lower index of Unit step signal :');
x_upper_index=input('Enter the upper index of Unit step signal :');
dn=0.01;
xn=x_lower_index:dn:x_upper_index;
x=zeros(1,length(xn));
h=zeros(1,length(xn));
for i=1:length(xn)
    x(i)=unit_step(xn(i));              %first unit step function
    h(i)=cos(2*pi.*xn(i)./4).*x(i);     %cosine mulitply with unit step
end

y_lower_index=2*x_lower_index;  %lower index of convolution as y
y_upper_index=2*x_upper_index; %upper index of convolution as y
m=[];
y=[];
subplot(4,1,1)
plot(xn,x,'linewidth' , 2);
hold on;
grid on;
title('u(n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(x_lower_index:1:x_upper_index);

subplot(4,1,2)
plot(xn,h,'linewidth' , 2);
hold on;
grid on;
title('cos(2\pin/4)u(n) signal');
xlabel('Time');
ylabel('Amplitude');
xticks(x_lower_index:1:x_upper_index);

convolution=conv(x,h);          %inbuilt func
```

```
37
38 yn=y_lower_index:dn:y_upper_index;
39 subplot(4,1,3)
40 plot(yn,convolution,'linewidth' , 2);
41 hold on;
42 grid on;
43 title('y(n) signal with inbuilt function');
44 xlabel('Time');
45 ylabel('Amplitude');
46 xticks(y_lower_index:1:y_upper_index);
47 %Matrix creation              %without inbuilt func
48 %h*x(i)
49 for i=1:length(x)
50     g=h.*x(i);
51     m=[m;g];%this sequence store array at new row so one matrix will be formed
52 end
53 %summation of diagonal(right) elements an store it into one array
54 %11 12 13 14
55 %21 22 23 24
56 %31 32 33 34
57 %r=3,c=4,k=7,diagonal_index_sum=2(1+1 first element) because max we have sum=k
58 [r c]=size(m);
59 k=r+c;
60 diagonal_index_sum=2;
61 element=0;
62 %column and row wise searching for sum=diagonal_index_sum
63 while(diagonal_index_sum<=k)
64     for i=1:r
65         for j=1:c
66             if((i+j)==diagonal_index_sum)
67                element=element+m(i,j);
68             end
69         end
70     end
71     diagonal_index_sum=diagonal_index_sum+1;%for next diagonal
72     y=[y element];
73     element=0;
74 end
75 subplot(4,1,4)
76 plot(yn,y, 'linewidth' , 2);
77 hold on;
78 grid on;
79 title('y(n) signal without inbuilt function');
80 xlabel('Time');
81 ylabel('Amplitude');
82 xticks(y_lower_index:1:y_upper_index);
83 %function to calc unit step if value is greater or equal to 0 then return 1
       elsewhere 0
84 function z = unit_step(x)
85     if x >= 0
86         z=1;
87     else
88         z=0;
89     end
90 end
91
```

(e) Approach b):
Taking lower and upper index from user and then calculated Unit Step signal through own function ,to calc unit step if value is greater or equal to 0 then return 1 elsewhere 0 and cosine multiplication with resultant unit step signal gives second signal h(n).Using this discrete signal, calculated length of convolution array and multiplying one by one element of x(n) with whole array h(n) and storing into one matrix m.Then,I did addition of right diagonal from first element ($m_{1X1}, where\ diagonalIndexSum = 1 + 1 = 2$)to last element ($m_{rXc}$) .Sum of those element which satisfies diagonal index sum (2,3,4,..,r+c). After that plotted

the value of y in appropriate range.

(f) Simulation Output b):

```
Enter the lower index of Unit step signal :
-5
Enter the upper index of Unit step signal :
15
```