Signals & Systems

EEE-223

Lab # 06

Name	Muhammad Haris Irfan
Registration Number	FA18-BCE-090
Class	BCE-4A
Instructor's Name	Muhammad Bilal Qasim

LAB # 06

Analysis of Discrete LTI Systems using Convolution Sum

Lab 06- Analysis of Discrete LTI Systems using Convolution Sum

Pre-Lab Tasks

6.1 Discrete Time Convolution:

An LTI discrete-time system is (equivalently to the continuous-time case) completely describes by impulse response, which is usually denoted by $h[n]$. The impulse response of an LTI discrete-time system is the output of the system when the unit impulse sequence (or Kronecker delta function) is applied as input. Even though the unit impulse input consists of one no-zero term, the impulse response signal $h[n]$ usually consists of more than one non-zero elements. The explanation is that the system is dynamic (with memory); that is, the system responds over various time instances to the output applied at $n = 0$. The knowledge of the impulse response $h[n]$ of a system allows the computation of the response $y[n]$ of the system to any input signal $x[n]$. The output signal is computed by discrete time convolution. The mathematical expression (6.1) of discrete time convolution is

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

The convolution between two discrete time signals is computed by using the MATLAB command conv.

6.1.1 The Command conv:

In MATLAB, the command conv allows the direct computation of the convolution between two signals. In order to illustrate the conv command There are three rules that have to be applied for successful computation of the convolution between two continuous-time signals.

- First rule: Two signals (input and impulse response) should be defined in the same time interval. Hence both the signals are defined in the time interval $a \le n \le b$.
- Second rule: When a signal consists of multiple parts, the time intervals in which each part is defined must not overlap. For example,

$$h[n] = \begin{cases} 1-n & 0 \le n \le 1 \\ 0 & 1 < n \le 2 \end{cases}$$

Note that the equality at $n = 1$ is placed only in the upper part.

Having defined the input and impulse response signals the response of the system can be computed by convoluting the two signals. The convolution is implemented by the command y=conv(x,h). The response of the system is now computed and the only thing left to do is to plot it. However, the number of elements of the output vector y is not equal to the number of elements of vectors x or h.

The precise relationship is length(y)=length(x)+length(h)-1. To overcome this, the third rule must be applied.

- Third rule: The output of the system is plotted in the double time interval of the one in which the output and impulse response signals are defined.

6.1.2 The Unit Impulse Sequence as Input to a System:

In this section, the impulse response of a discrete time system is discussed in detail. More specifically, it is established that if the unit impulse sequence $\delta[n]$ is the input to a system, the output of the system is the impulse response $h[n]$ of the system.

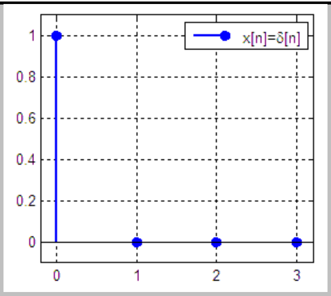
Example:

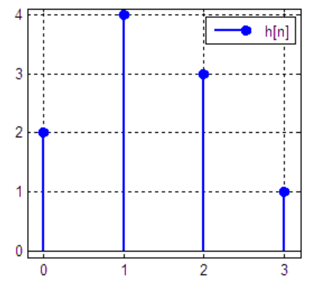
Suppose that a discrete time system is described by the impulse response $h[n]=[2\ 4\ 1\ 3], 0 \le n \le 3$. Compute the response of the system to the input signal

$$x[n]=\delta[n]=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases}$$

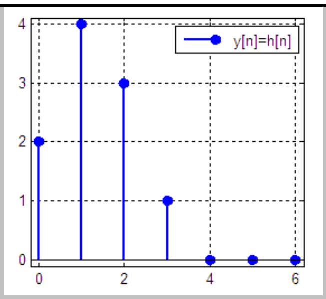
The response of the system is computed by convoluting the impulse response $h[n]$ with the input signal $\delta[n]$. Recall that $\delta[n]$ must be defined in the same time interval to $h[n]$; that is, $0 \le n \le 3$. For illustration purposes both signals are plotted.

Commands	**Results**	**Comments**

x=[1 0 0 0];
 	
n=0:3;
 	
stem(n,x,'fill','linewidth',2),grid on
 	
axis([-0.2 3.2 -0.1 1.1])
 	
legend('x[n]=\delta[n]')
		Input signal $x[n]=\delta[n]$ plotted in $0 \le n \le 3$.

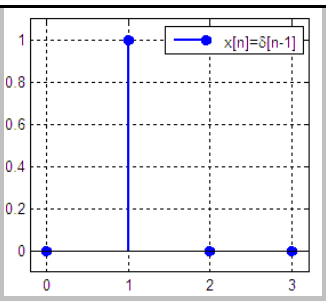
h=[2 4 3 1]; **n=0:3;** **stem(n,h,'fill','linewidth',2),grid on** **axis([-0.2 3.2 -0.1 4.1])** **legend('h[n]')**		Impulse response $h[n]=[2\ 4\ 1\ 3]$ plotted in $0 \le n \le 3$.

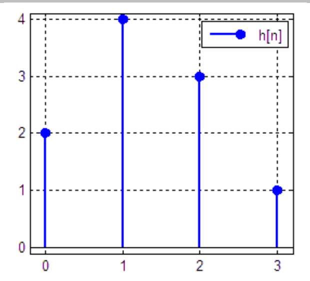
The number of elements of the output vector $y[n]$ compared to the number of elements of the input and impulse response vectors. The exact relationship is length (y)=length(x)+length(h)-1. Therefore the output of the system is plotted in the double time interval from input and impulse response signals.

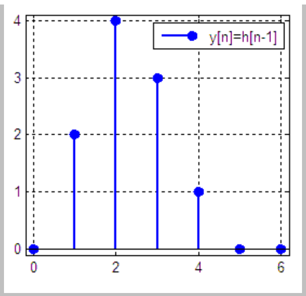
Commands	**Results**	**Comments**
y=conv(x,h); **stem(0:6,y,'fill','linewidth',2),grid on** **axis([-0.2 6.2 -0.1 4.1])** **legend('y[n]=h[n]')**		The output $y[n]$ is computed and plotted for $0 \le n \le 6$. As expected the output and impulse response signals are the same.

The output and impulse response signals are the same; thus the identity property $\left(h[n]*\delta[n]=h[n]\right)$ clearly stands. The system under consideration is a linear shift invariant system. If at a linear and shift invariant system, with impulse response $h[n]$, the input signal $\delta[n-k]$, i.e., a shifted unit impulse sequence is applied, then the response of the system is also shifted by k units.

Commands	Results	Comments

x=[0 1 0 0];
n=0:3;
stem(n,x,'fill','linewidth',2),grid on
axis([-0.2 3.2 -0.1 1.1])
legend('x[n]=\delta[n-1]')
		The shifted unit impulse sequence $x[n]=\delta[n-1]$ is the input to the system.

h=[2 4 3 1];
n=0:3;
stem(n,h,'fill','linewidth',2),grid on
axis([-0.2 3.2 -0.1 4.1])
legend('h[n]')
		Impulse response $h[n]=[2\ 4\ 1\ 3]$, $0\le n\le 3$.

y=conv(x,h);
stem(0:6,y,'fill','linewidth',2),grid on
axis([-0.2 6.2 -0.1 4.1])
legend('y[n]=h[n-1]')
		The output of the system is its impulse response shifted by 1 unit to the right.

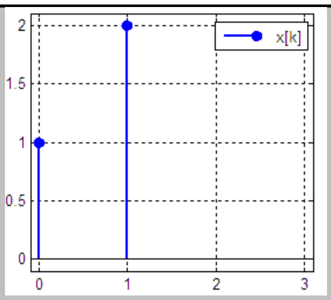
As expected, the response $y[n]$ of the system to the input signal $\delta[n-1]$ is $h[n-1]$. More generally, the response of a linear shift invariant system to a shifted unit impulse sequence $\delta[n-k]$ shifted by k units version of impulse response, i.e., the output of the system is $h[n-k]$.

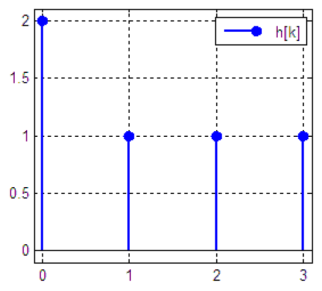
6.1.3 Computation of Discrete Time Convolution:

The process that has to be followed to analytically derive the convolution sum (equation 6.1) is as follows. First, the input and impulse response signals are plotted in the k-axis. One of the two signals is reversed about the amplitude axis and its reflection is shifted from $-\infty$ to $+\infty$ by changing appropriately the value of n. The output of the system is computed from the overlapping values of $x[n]$ and $h[n-k]$ according to the convolution sum $y[n]=x[n]*h[n]=\sum_{k=-\infty}^{+\infty} x[k]h[n-k]$. Te convolution procedure for two discrete time signals is demonstrated by using the signals $x[n]=[1\ \ 2], 0\leq n\leq 1$ and $h[n]=[2\ \ 1\ \ 1\ \ 1], 0\leq n\leq 3$.

Step 1: The two signals are plotted at the k-axis.

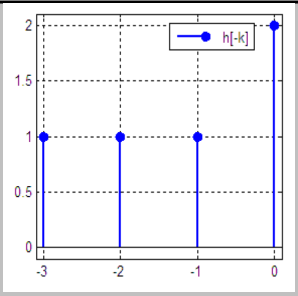
Commands	Results	Comments

`kx=[0 1];`
`x=[1 2];`
`stem(kx,x,'fill','linewidth',2),grid on`
`axis([-0.1 3.1 -0.1 2.1])`
		Input signal $x[k]$.

`kh=0:3;`
`h=[2 1 1 1];`
`stem(kh,h,'fill','linewidth',2),grid on`
`axis([-0.1 3.1 -0.1 2.1])`
`legend('h[k]')`
		Impulse response signal $h[k]$.

Step 2: The reflected version of $h[k]$, namely, $h[-k]$ is plotted.

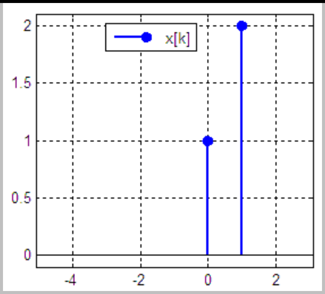
Commands	Results	Comments

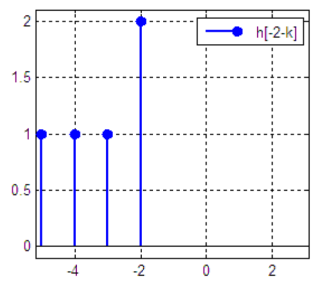
stem(-kh,h,'fill','linewidth',2) ,grid on
 	
axis([-3.1 0.1 -0.1 2.1])
 	
legend('h[-k]')
		The reflected signal $h[-k]$.

Step 3: The signal $h[n-k]$ is shifted from $-\infty$ to $+\infty$ by changing appropriately the value of n. The output if the system is computed at each shift through equation 6.1, in which only the values of overlapping parts of $x[n]$ and $h[n-k]$ are considered. In discrete time case we will compute the sum of $x[n]$ and $h[n-k]$. There are three stages, first there is zero overlap, next there is overlap and finally there is no overlap.

- First Stage: Zero Overlap. The stage occurs for $n \le -1$.

Commands	**Results**	**Comments**

stem(kx,x,'fill','linewidth',2),grid on
 	
axis([-5.1 3.1 -0.1 2.1])
 	
legend('x[k]')
		Input signal $x[k]$.

n=-2;
 	
stem(-kh+n,h,'fill','linewidth',2), grid on
 	
axis([-5.2 3.1 -0.1 2.1])
 	
legend('h[-2-k]')
		Impulse response signal $h[n-k]$ for $n=-2$.

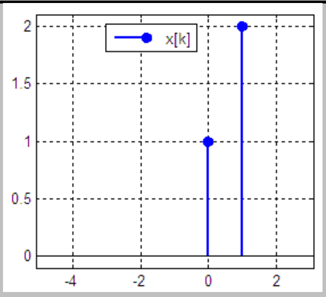
Obviously the two signals do not overlap. Thus the output is $y[n]=0,\ n \le -1$.

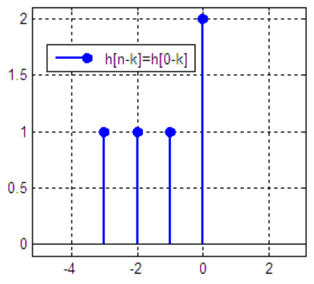
- Second Stage: Overlap. The signals $x[n]$ and $h[n-k]$ overlap for $0 \le n \le 4$.

In order to compute the output, the two signals are plotted for $n = 0, 1, \ldots, 4$. The output is computed through the convolution sum by taking into account only the overlapping samples.

For $n = 0$,

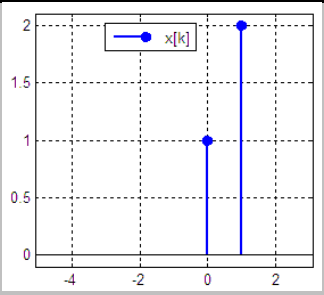
Commands	Results

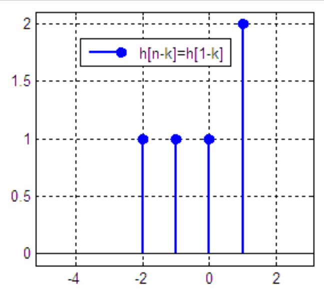
stem(kx,x,'fill','linewidth',2),grid on
 	
axis([-5.1 3.1 -0.1 2.1])
 	
legend('x[k]')
	

n=0;
 	
stem(-kh+n,h,'fill','linewidth',2),grid on
 	
axis([-5.2 3.1 -0.1 2.1])
 	
legend('h[n-k]=h[0-k]')
	

For $n = 0$, the signals $x[k]$ and $h[n-k] = h[0-k]$ have one overlapping sample at $k = 0$. The output of the system for $n = 0$; that is, $y[0] = (1)(2)$, where 1 is the value of $x[k]$ and 2 is the value of $h[n-k]$ at the overlapping sample. In order to understand better the output calculation, notice that according to the definition of the convolution sum $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$, the overlapping samples of $x[k]$ and $h[n-k]$ are multiplied. The sum is applied when more than one samples overlap. This is the case for $n = 1$.

Commands	Results

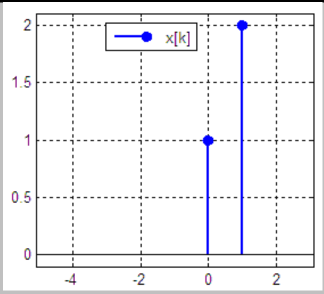
stem(kx,x,'fill','linewidth',2),grid on
 	
axis([-5.1 3.1 -0.1 2.1])
 	
legend('x[k]')
	

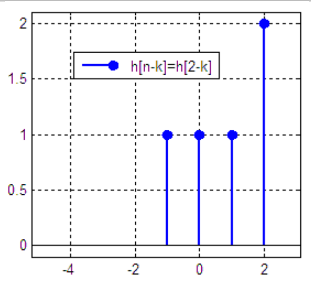
n=1;
 	
stem(-kh+n,h,'fill','linewidth',2),grid on
 	
axis([-5.2 3.1 -0.1 2.1])
 	
legend('h[n-k]=h[0-k]')
	

For $n=1$, the signals $x[k]$ and $h[n-k]=h[1-k]$ overlap with two samples, namely, at $k=0$ and at $k=1$. The output for $n=1$, that is, $y[n]=y[1]$ is computed as $y[1]=(2)(2)+(1)(1)$, where 2 is the value of $x[k]$ and $h[n-k]$ at $k=1$, while 1 is the value of $x[k]$ and $h[n-k]$ at $k=0$. Hence $y[1]=5$.

For $n=2$, we have

Commands	**Results**

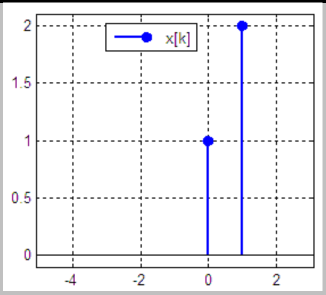
stem(kx,x,'fill','linewidth',2),grid on
 	
axis([-5.1 3.1 -0.1 2.1])
 	
legend('x[k]')
	

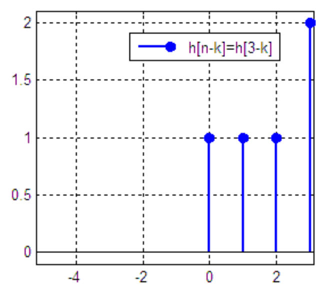
n=2;
 	
stem(-kh+n,h,'fill','linewidth',2),grid on
 	
axis([-5.2 3.1 -0.1 2.1])
 	
legend('h[n-k]=h[2-k]')
	

For $n=2$, the signals $x[k]$ and $h[n-k]=h[2-k]$ overlap at two points, at $k=0$ and at $k=1$. The output for $n=2$, that is, $y[n]=y[2]$ is computed as $y[2]=(2)(1)+(1)(1)$, where 2 is the value of $x[k]$ and 1 is the value of $h[n-k]$ at $k=1$, while 1 is the value of $x[k]$ and $h[n-k]$ at $k=0$. Hence $y[2]=3$.

For $n=3$, we have

Commands	Results

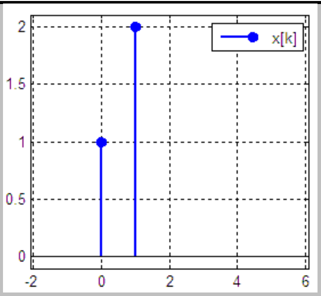
stem(kx,x,'fill','linewidth',2),grid on
 	
axis([-5.1 3.1 -0.1 2.1])
 	
legend('x[k]')
	

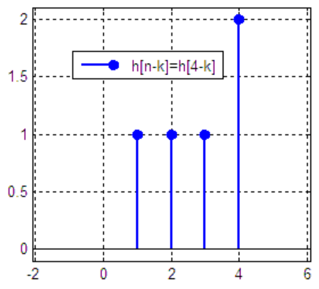
n=3;
 	
stem(-kh+n,h,'fill','linewidth',2),grid on
 	
axis([-5.2 3.1 -0.1 2.1])
 	
legend('h[n-k]=h[3-k]')
	

The output for $n=3$, i.e., $y[n]=y[3]$ is computed as $y[3]=(2)(1)+(1)(1)=3$.

For $n=4$,

Commands	Results

stem(kx,x,'fill','linewidth',2),grid on
 	
axis([-2.1 6.1 -0.1 2.1])
 	
legend('x[k]')
	

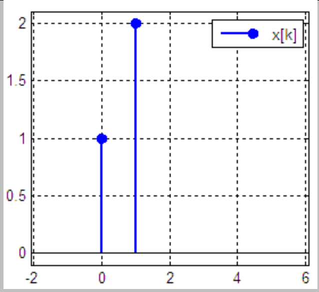
n=4;
 	
stem(-kh+n,h,'fill','linewidth',2),grid on
 	
axis([-2.1 6.1 -0.1 2.1])
 	
legend('h[n-k]=h[4-k]')
	

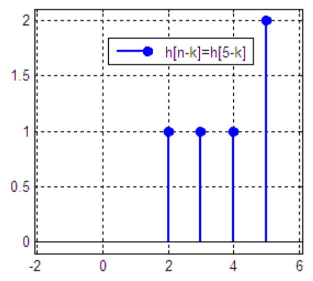
For $n = 4$, there is only one overlap at $k = 1$. Thus the output for $n = 4$ is $y[4] = (2)(1) = 2$.

- Third Stage: Zero Overlap. For $n \geq 5$, the input and impulse response signals do not overlap, hence $y[n] = 0,\ n \geq 5$.

For $n = 5$, we have

Commands	**Results**

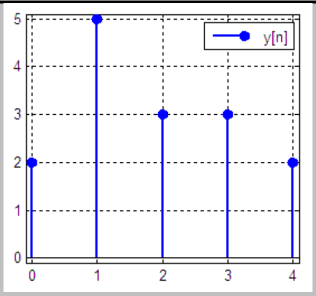
stem(kx,x,'fill','linewidth',2),grid on
 	
axis([-2.1 6.1 -0.1 2.1])
 	
legend('x[k]')
	

n=5;
 	
stem(-kh+n,h,'fill','linewidth',2),grid on
 	
axis([-2.1 6.1 -0.1 2.1])
 	
legend('h[n-k]=h[5-k]')
	

Combining the derived results we conclude that the response of the system with impulse response $h[n] = [2\ 1\ 1\ 1],\ 0 \leq n \leq 3$ to the input signal $x[n] = [1\ 2],\ 0 \leq n \leq 1$ is $y[n] = [2\ 5\ 3\ 3\ 2],\ 0 \leq n \leq 4$. The output signal is plotted in figure below.

Commands	**Results**

```
n=0:4;

y=[2 5 3 3 2];

stem(n,y,'fill','linewidth',2),grid on

axis([-0.1 4.1 -0.1 5.1])

legend('y[n]')
```



There are two basic principles that should be considered while computing the convolution between two discrete time signals $x[n]$ and $h[n]$ using conv command.

1. Suppose that the length of vector x is N and the length of vector of h is M. The outcome of command y=conv(x,h) is a vector y of length $M+N-1$. In other words, length(y)=length(x)+length(h)-1.
2. If non zeros values of $x[n]$ are in the interval $[a_x, b_x]$ and non zero values of $h[n]$ are in the interval $[a_h, b_h]$ then the non zero values of the output $y[n]$ are in the interval $[a_x + a_h, b_x + b_h]$.

In-Lab Tasks

Task 01: Compute and plot the convolution $y[n]=x[n]*h[n]$ by any of the two procedures, where

$$x[n]=\left(\frac{1}{3}\right)^{-n}u[-n-1] \text{ and } h[n]=u[n-1]$$

```
n = -5:5;
u = (n<=1);
x = 0.3.^-n.*u;
subplot(3,1,1)
stem(n,x,'LineWidth',2)
axis([-5.5 5.5 -0.5 4])
legend('x[n]','Location','NorthEastOutside')

h=zeros(1,11);
h(7:11)=1;
subplot(3,1,2)
stem(n,h,'LineWidth',2)
axis([-5.5 5.5 -0.5 1.5])
legend('h[n]','Location','NorthEastOutside')

y =conv(h,x);
subplot(3,1,3)
stem(-5:15,y,'LineWidth',2)
legend('y[n]=x[n]*h[n]','Location','NorthEastOutside')
```

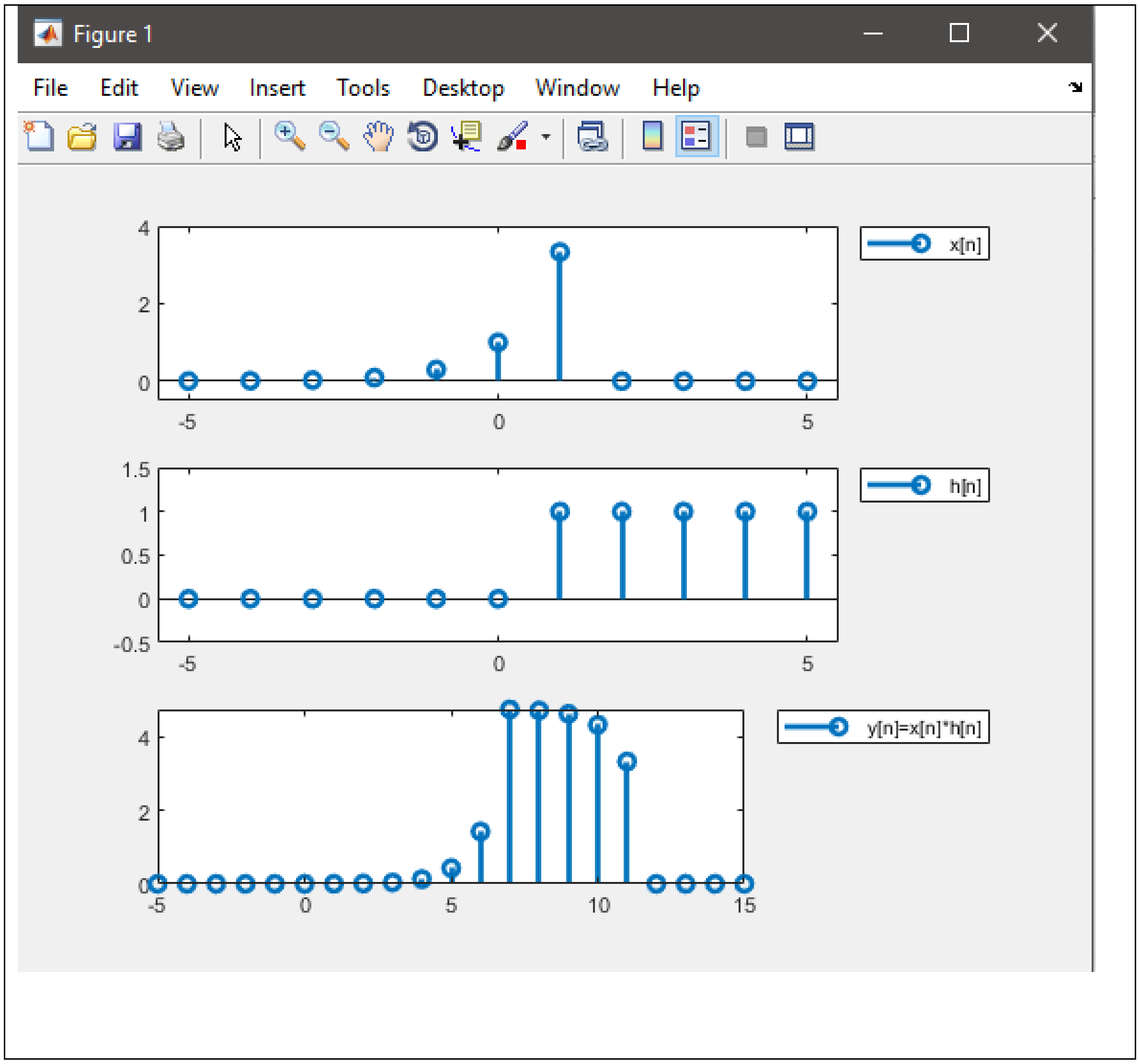


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
x[n]
h[n]
y[n]=x[n]*h[n]

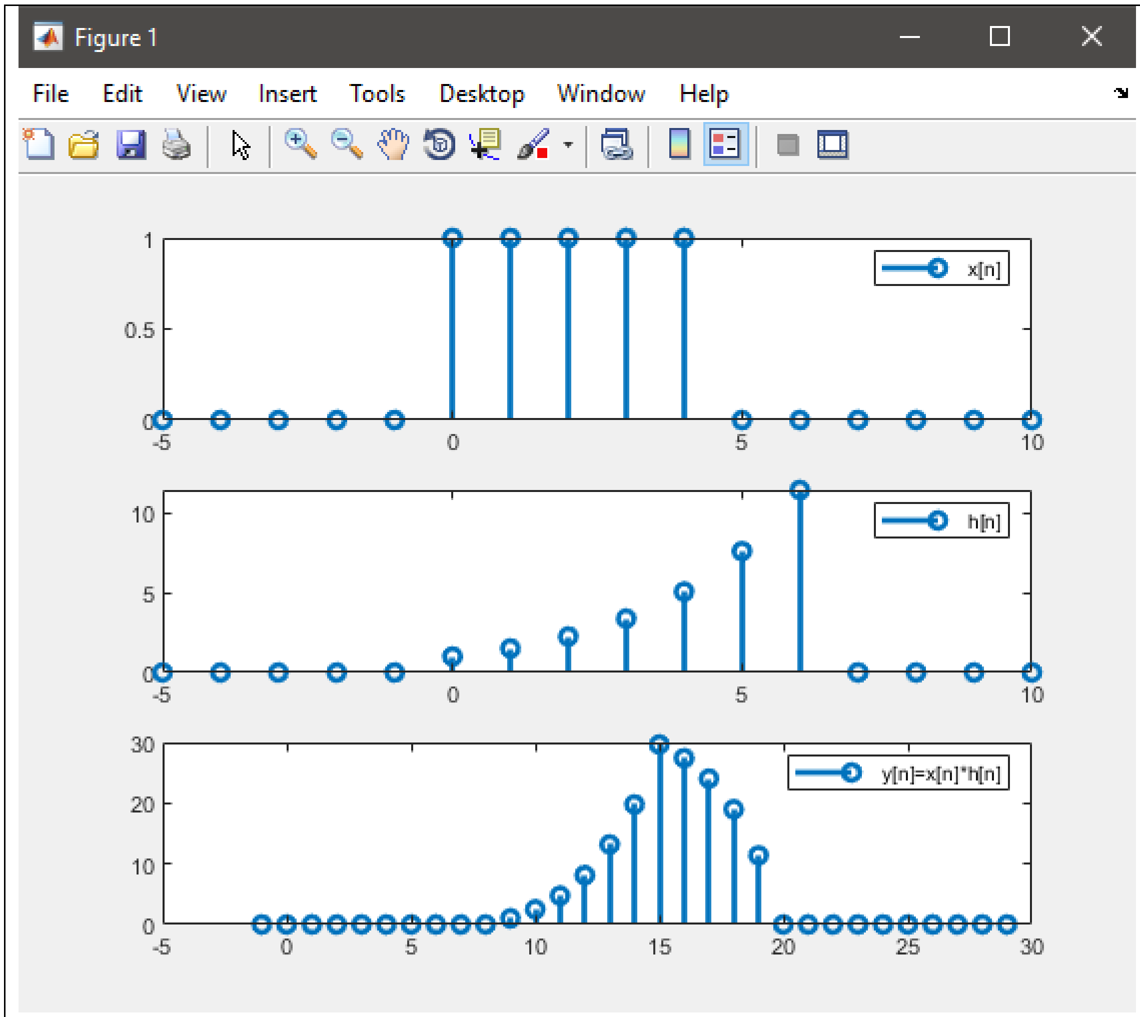
Task 02: Compute and plot the convolution of following signals (by both procedures)

$$x[n] = \begin{cases} 1 & 0 \le n \le 4 \\ 0 & elsewhere \end{cases}$$

and

$$h[n] = \begin{cases} 1.5^n & 0 \le n \le 6 \\ 0 & elsewhere \end{cases}$$

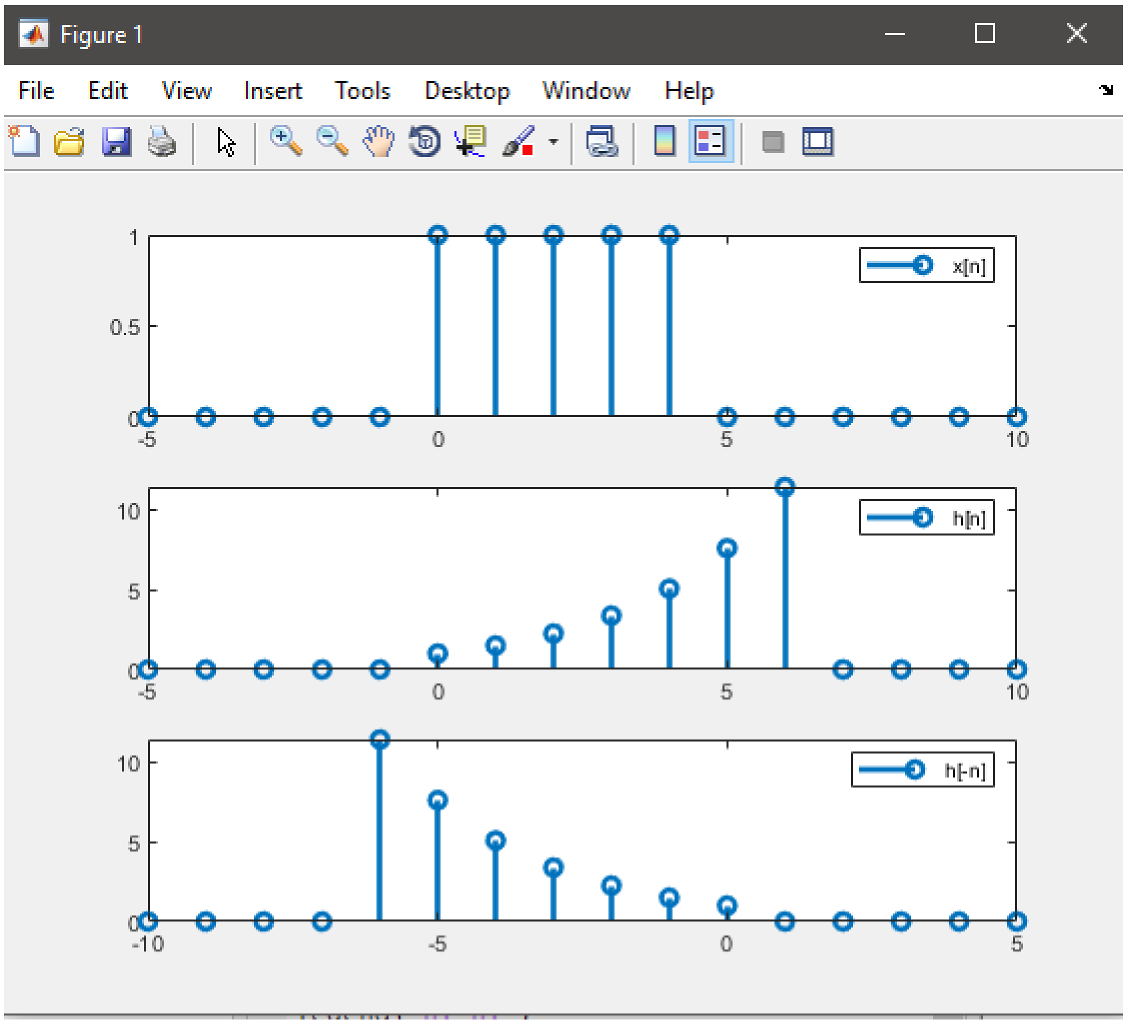
```
n = -5:10;
x = [0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0];

subplot(3,1,1)
stem(n,x,'LineWidth',2)
legend('x[n]')

subplot(3,1,2)
a1 = (n>=0)&(n<=6);
a2 = 1.5.^n;
h = a1.*a2;
stem(n,h,'LineWidth',2)
legend('h[n]')

y = conv(x,h);
subplot(3,1,3)
stem(-1:29,y,'LineWidth',2)
legend('y[n]=x[n]*h[n]')
```



Other Method:

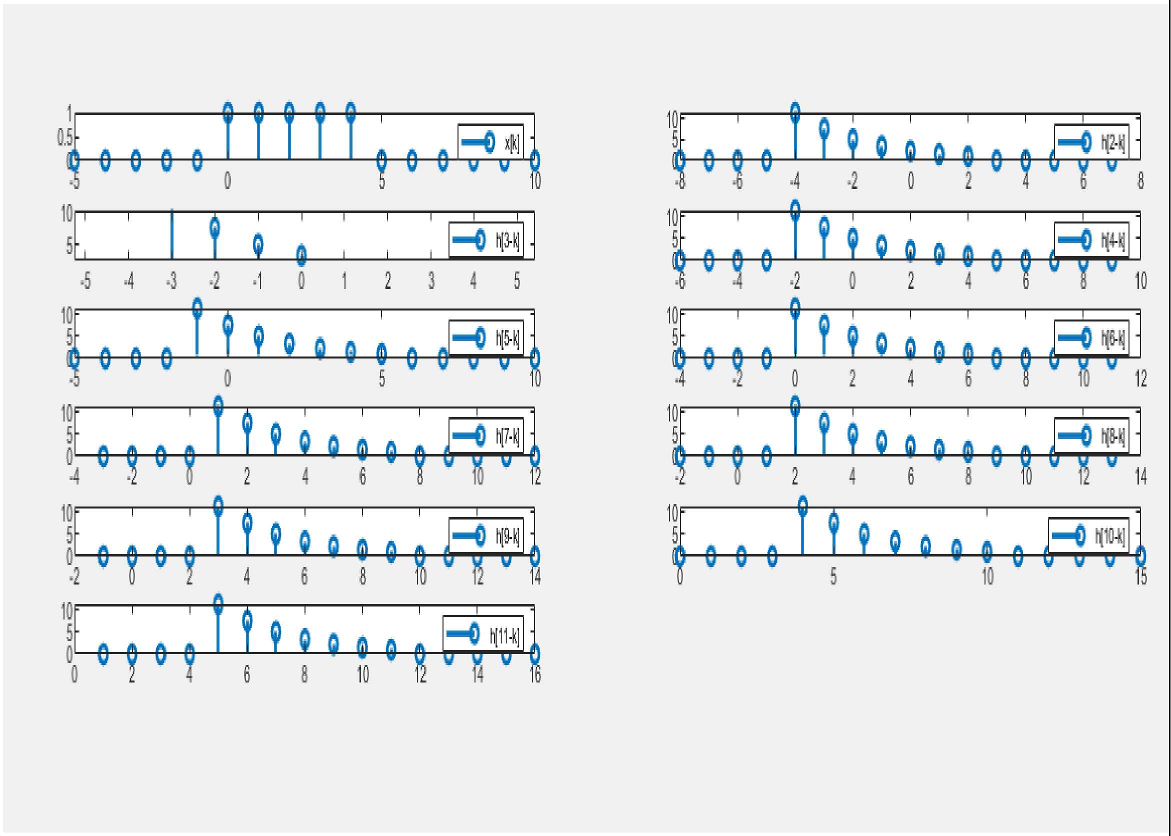
```
n = -5:10;
x = [0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0];
a1 = 1.5.^n;
a2 = (n>=0)&(n<=6);
h = a1.*a2;

subplot(3,1,1)
stem(n,x,'LineWidth',2)
legend('x[n]')
```

```
subplot(3,1,2)
stem(n,h,'LineWidth',2)
legend('h[n]')

subplot(3,1,3)
stem(-n,h,'LineWidth',2)
legend('h[-n]')
```



```
k = -5:10;   % changing axis
n = 0;       % delay
subplot(12,2,1)
stem(k,x,'LineWidth',2)
legend('x[k]')
```

```
subplot(12,2,2)
stem(-k+n,h,'LineWidth',2)
legend('h[0-k]')
n=2;        % delay
subplot(12,2,2)
stem(-k+n,h,'LineWidth',2)
legend('h[2-k]')

n=3;        % delay
subplot(12,2,3)
stem(-k+n,h,'LineWidth',2)
legend('h[3-k]')

n=4;        % delay
subplot(12,2,4)
stem(-k+n,h,'LineWidth',2)
legend('h[4-k]')

n=5;        % delay
subplot(12,2,5)
stem(-k+n,h,'LineWidth',2)
legend('h[5-k]')

n=6;        % delay
subplot(12,2,6)
stem(-k+n,h,'LineWidth',2)
legend('h[6-k]')

n=7;        % delay
subplot(12,2,7)
stem(-k+n,h,'LineWidth',2)
legend('h[7-k]')

n=8;        % delay
subplot(12,2,8)
stem(-k+n,h,'LineWidth',2)
legend('h[8-k]')

n=9;        % delay
subplot(12,2,9)
stem(-k+n,h,'LineWidth',2)
legend('h[9-k]')

n=10;        % delay
subplot(12,2,10)
stem(-k+n,h,'LineWidth',2)
legend('h[10-k]')

n=11;        % delay
subplot(12,2,11)
stem(-k+n,h,'LineWidth',2)
legend('h[11-k]')
```



Convolution

```
%convolution
y = [1 2.5 4.75 8.13 13.19 19.78 29.67 27.42 24.05 18.98 11.39];
yn = 0:10;
stem(yn,y,'LineWidth',2)
title('y[n]=x[n]*h[n]')
```

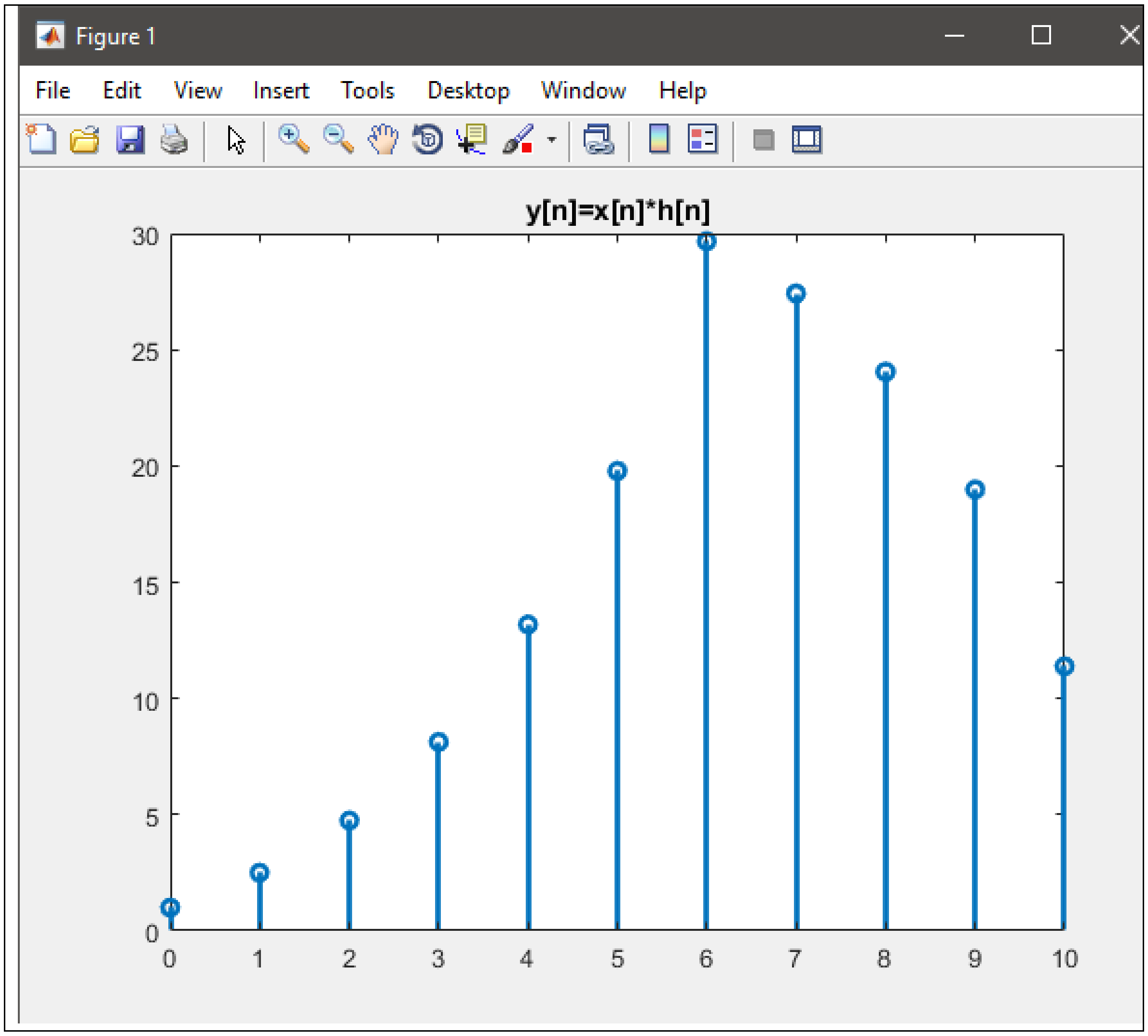


Figure 1
File Edit View Insert Tools Desktop Window Help
y[n]=x[n]*h[n]
30
25
20
15
10
5
0
0 1 2 3 4 5 6 7 8 9 10

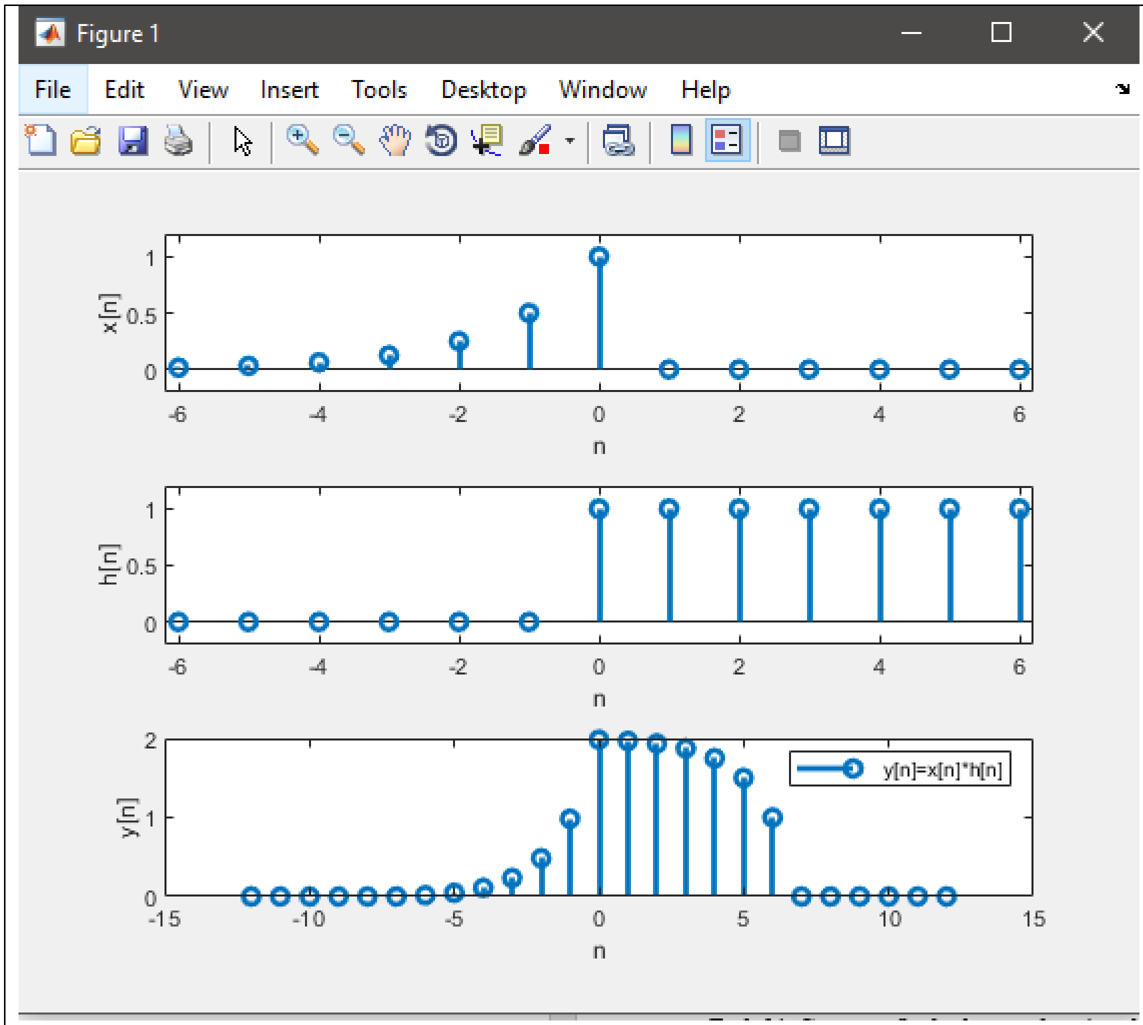
Task 03: Consider and LTI system with input $x[n]$ and unit impulse response $h[n]$ specified as

$x[n] = 2^n u[-n]$

$h[n] = u[n]$

Compute the response of the system (by both methods) where we have $-6 \le n \le 6$.

```
n = -6:6;
h = [0 0 0 0 0 0 1 1 1 1 1 1 1];
x = 2.^n.*(n<=0);
subplot(3,1,1)
stem(n,x,'LineWidth',2)
axis([-6.2 6.2 -0.2 1.2])
xlabel('n')
ylabel('x[n]')

subplot(3,1,2)
stem(n,h,'LineWidth',2)
axis([-6.2 6.2 -0.2 1.2])
xlabel('n')
ylabel('h[n]')

y = conv(h,x);
yn = -12:12;
subplot(3,1,3)
stem(yn,y,'LineWidth',2)
xlabel('n')
ylabel('y[n]')
legend('y[n]=x[n]*h[n]')
```

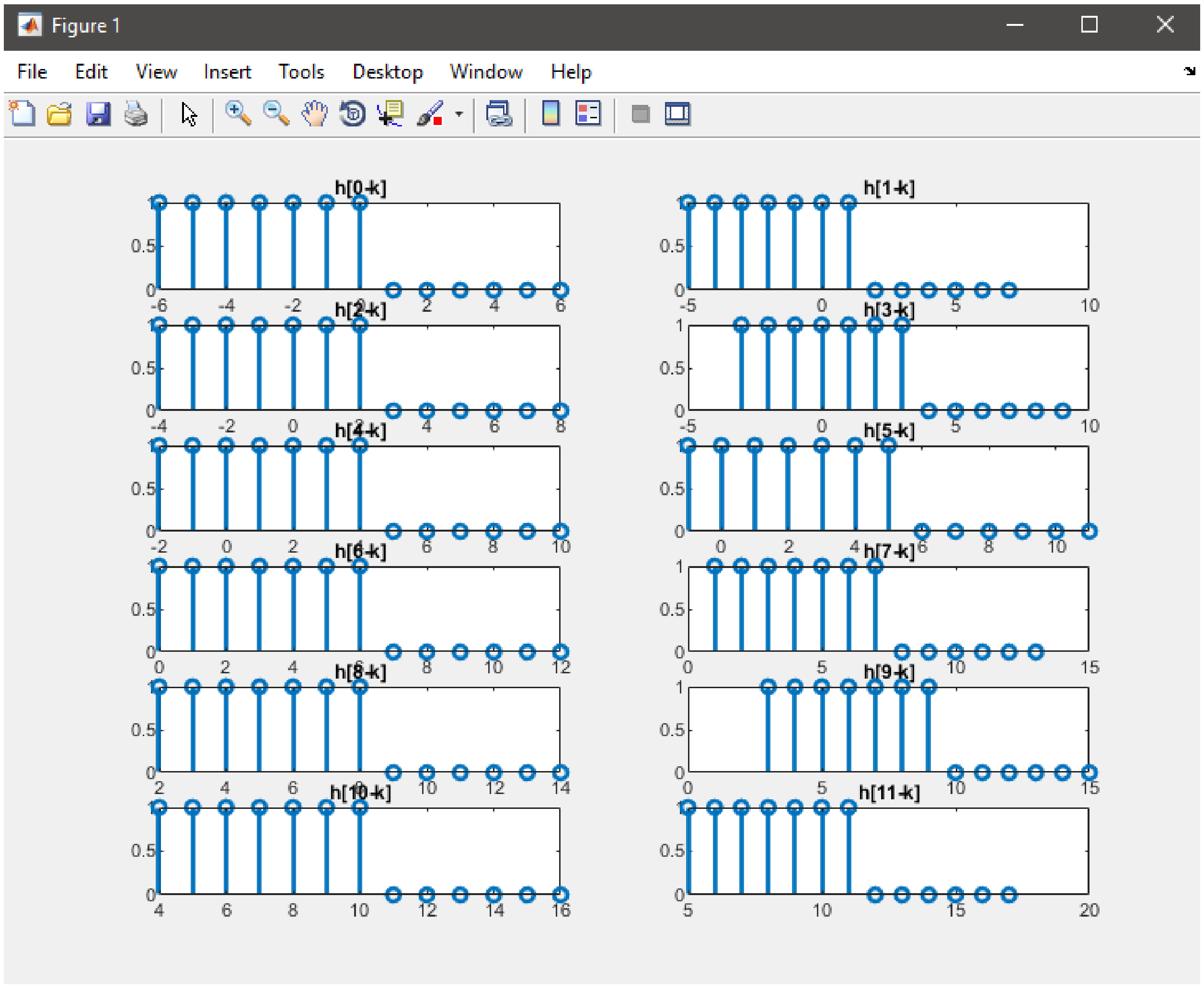


Other Method:

```
k = -6:6;          % changing axis
x = 2.^k.*(k<=0);
subplot(3,1,1)
stem(k,x,'LineWidth',2)
legend('x[k]')

h = [0 0 0 0 0 0 1 1 1 1 1 1 1];
subplot(3,1,2)
stem(k,h,'LineWidth',2)
legend('h[k]')

subplot(3,1,3)
stem(-k,h,'LineWidth',2)
legend('h[-k]')
```

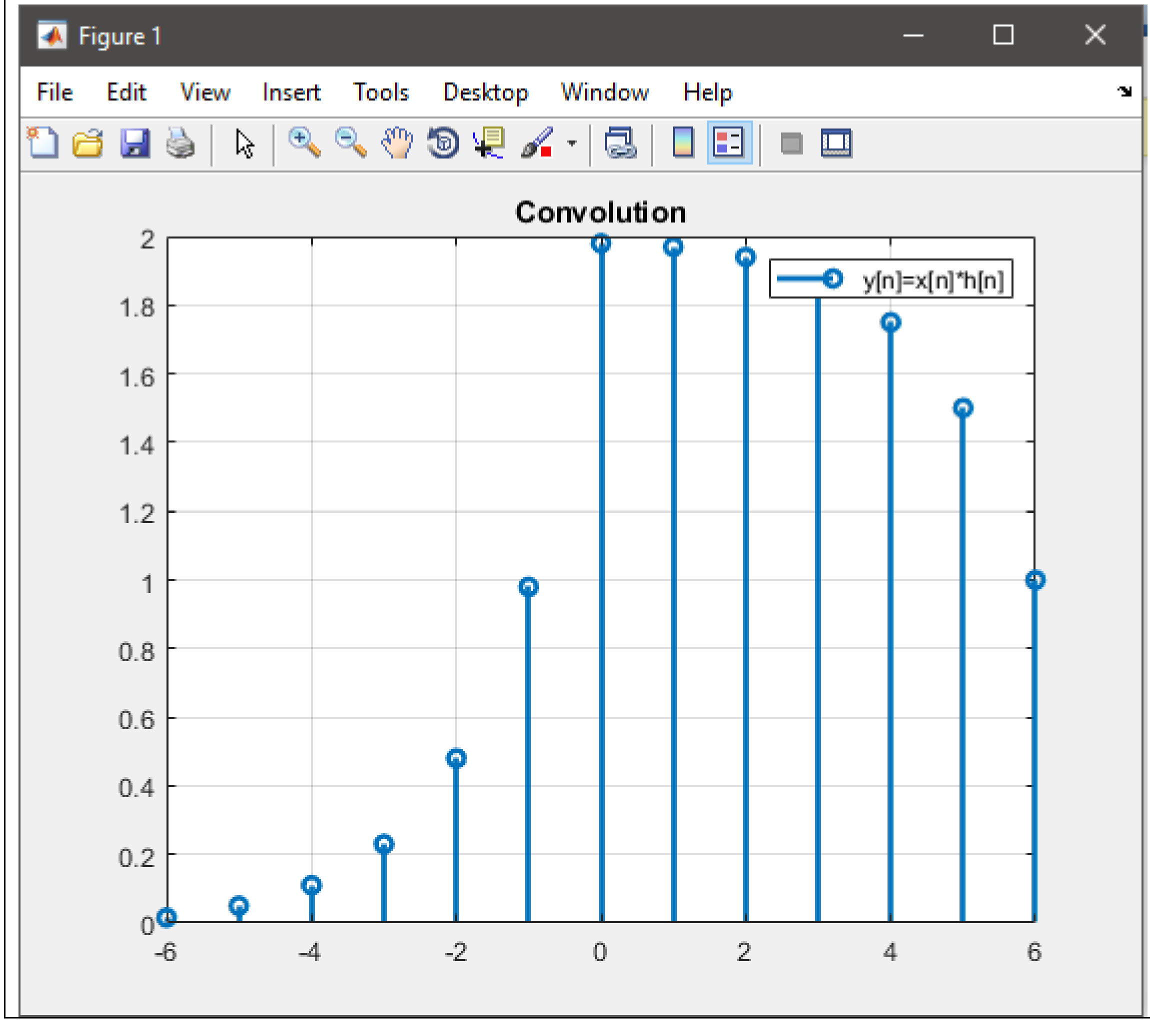
```
shift=0;                 % No shift
subplot(6,2,1)
stem(-k+shift,h,'LineWidth',2)
title('h[0-k]')


shift=1;                 % 1 unit shift
subplot(6,2,2)
stem(-k+shift,h,'LineWidth',2)
title('h[1-k]')



shift=2;                 % 2 unit shift
subplot(6,2,3)
stem(-k+shift,h,'LineWidth',2)
title('h[2-k]')

shift=3;                 % 3 unit shift
subplot(6,2,4)
stem(-k+shift,h,'LineWidth',2)
title('h[3-k]')

shift=4;                 % 4 unit shift
subplot(6,2,5)
stem(-k+shift,h,'LineWidth',2)
title('h[4-k]')

shift=5;                 % 5 unit shift
subplot(6,2,6)
stem(-k+shift,h,'LineWidth',2)
title('h[5-k]')

shift=6;                 % 6 unit shift
subplot(6,2,7)
stem(-k+shift,h,'LineWidth',2)
title('h[6-k]')

shift=7;                 % 7 unit shift
subplot(6,2,8)
stem(-k+shift,h,'LineWidth',2)
title('h[7-k]')

shift=8;                 % 8 unit shift
subplot(6,2,9)
stem(-k+shift,h,'LineWidth',2)
title('h[8-k]')

shift=9;                 % 9 unit shift
subplot(6,2,10)
stem(-k+shift,h,'LineWidth',2)
title('h[9-k]')
```

```
shift=10;               % 10 unit shift
subplot(6,2,11)
stem(-k+shift,h,'LineWidth',2)
title('h[10-k]')



shift=11;               % 11 unit shift
subplot(6,2,12)
stem(-k+shift,h,'LineWidth',2)
title('h[11-k]')
```



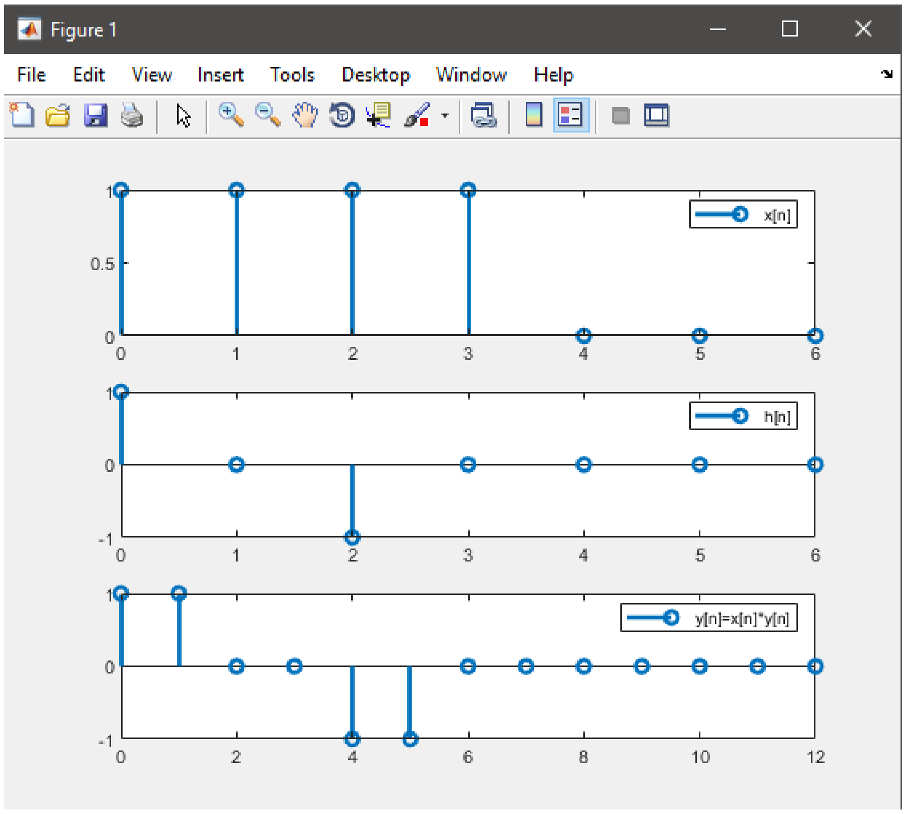
```
%Convolution
y=[ 0.0156 0.05 0.11 0.23 0.48 0.98 1.98 1.97 1.94 1.87 1.75 1.5 1.0];
yn=-6:6;
```

```
stem(yn,y,'LineWidth',2)
title('Convolution')
legend('y[n]=x[n]*h[n]')
grid on
```



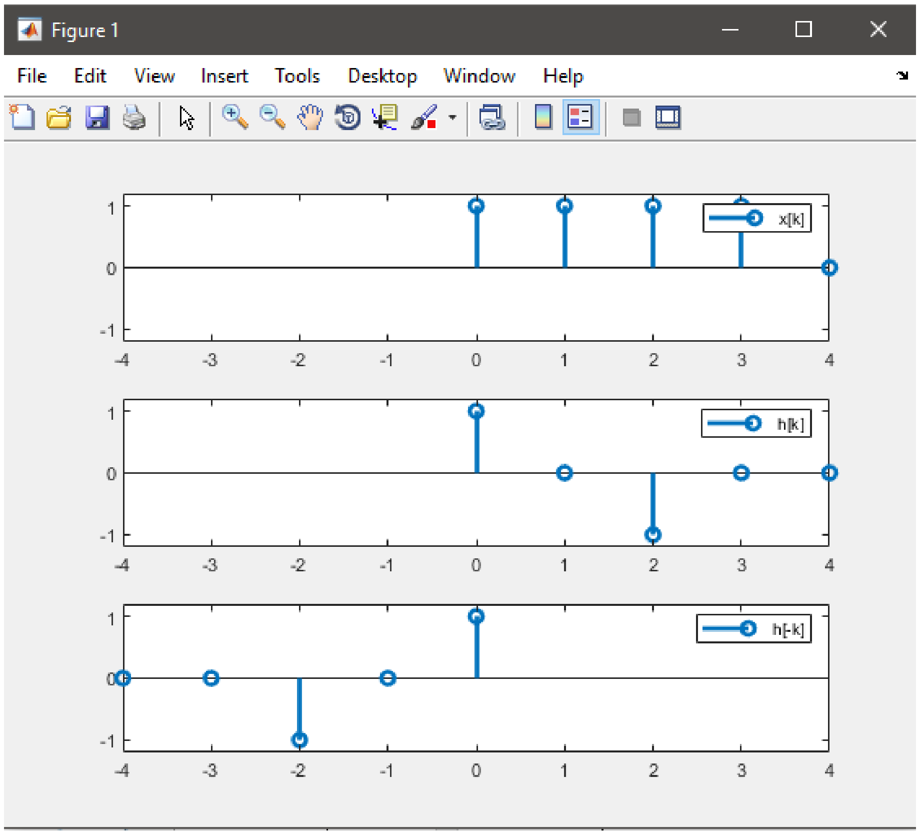
Task 04: Compute (by both procedures) and graph the convolution $y[n]=x[n]*h[n]$, where $x[n]=u[n]-u[n-4]$ and $h[n]=\delta[n]-\delta[n-2]$.

```
n = 0:6;
h = [1 0 -1 0 0 0 0];
x = [1 1 1 1 0 0 0];
subplot(3,1,1)
stem(n,x,'LineWidth',2)
legend('x[n]')

subplot(3,1,2)
stem(n,h,'LineWidth',2)
legend('h[n]')

y = conv(x,h);
subplot(3,1,3)
stem(0:12,y,'LineWidth',2)
legend('y[n]=x[n]*y[n]')
```



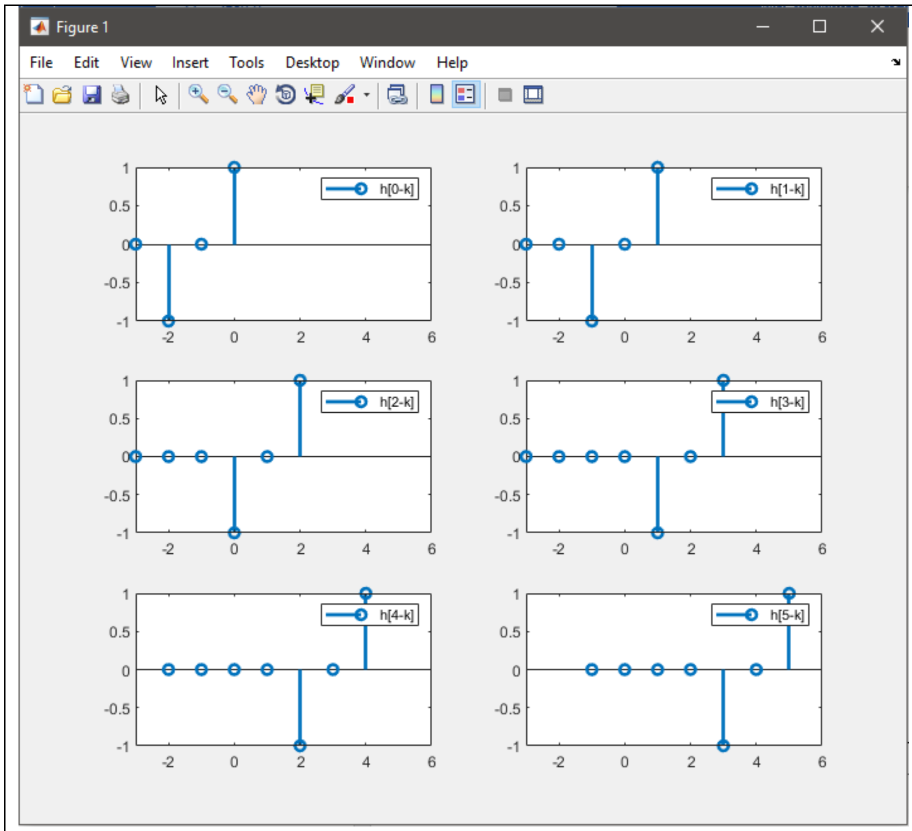
Other Method

```
k = 0:6;        %changing axis
x = [1 1 1 1 0 0 0];
h = [1 0 -1 0 0 0 0];
subplot(3,1,1)
stem(k,x,'LineWidth',2)
legend('x[k]')
axis([-4 4 -1.2 1.2])

subplot(3,1,2)
stem(k,h,'LineWidth',2)
legend('h[k]')
axis([-4 4 -1.2 1.2])

subplot(3,1,3)
stem(-k,h,'LineWidth',2)
legend('h[-k]')
axis([-4 4 -1.2 1.2])
```



```
n=0;        %delay
subplot(3,2,1)
stem(-k+n,h,'LineWidth',2)
```

```
legend('h[0-k]')
xlim([-3 6])


n=1;      %delay
subplot(3,2,2)
stem(-k+n,h,'LineWidth',2)
legend('h[1-k]')
xlim([-3 6])

n=2;      %delay
subplot(3,2,3)
stem(-k+n,h,'LineWidth',2)
legend('h[2-k]')
xlim([-3 6])

n=3;      %delay
subplot(3,2,4)
stem(-k+n,h,'LineWidth',2)
legend('h[3-k]')
xlim([-3 6])

n=4;      %delay
subplot(3,2,5)
stem(-k+n,h,'LineWidth',2)
legend('h[4-k]')
xlim([-3 6])

n=5;      %delay
subplot(3,2,6)
stem(-k+n,h,'LineWidth',2)
legend('h[5-k]')
xlim([-3 6])
```



```
%Convolution
y = [1 1 0 0 -1 -1 0];
yn = 0:6;
stem(yn,y,'LineWidth',2)
title('Convolution')
legend('y[n]=x[n]*h[n]')
```

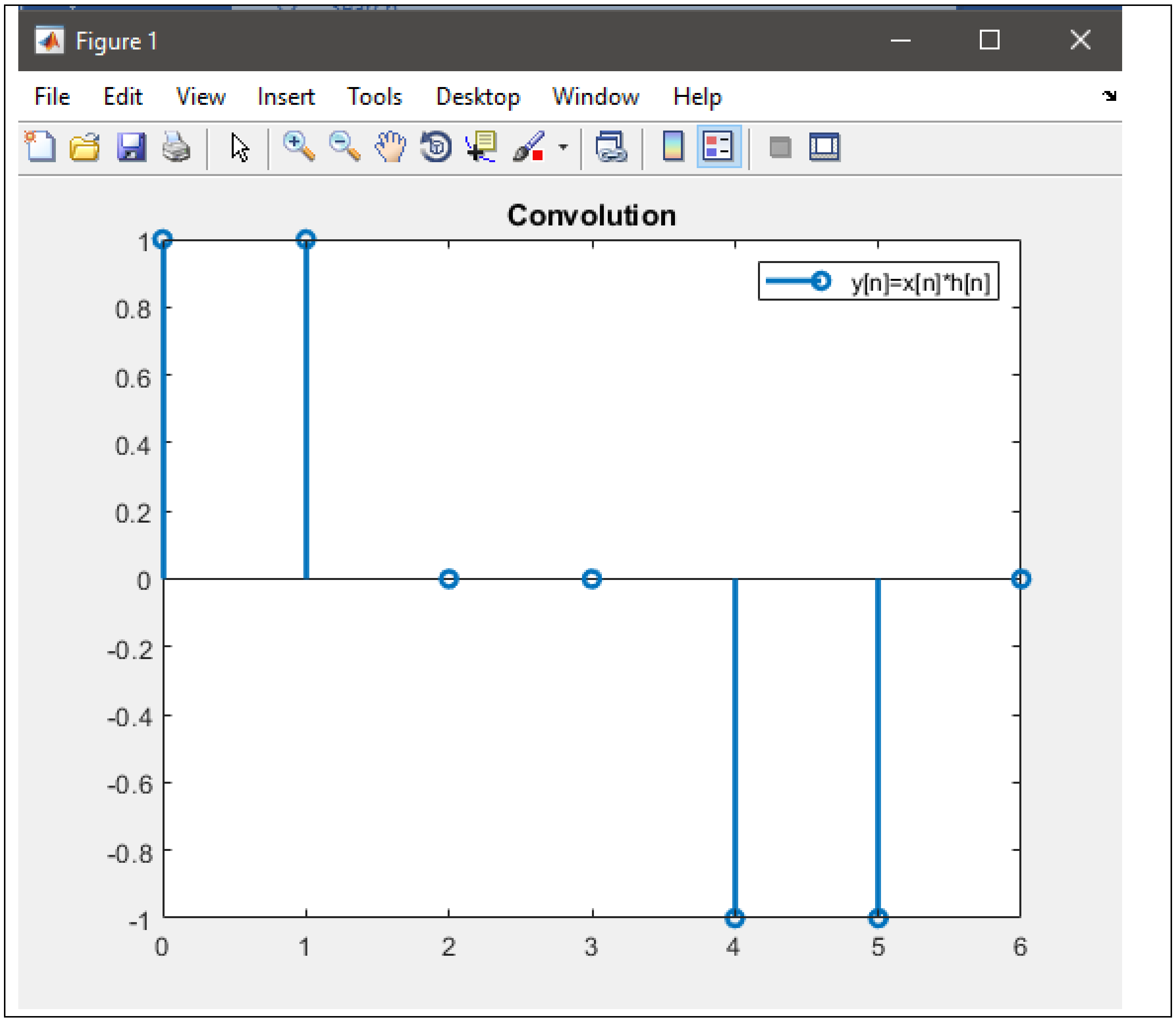


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Convolution
y[n]=x[n]*h[n]
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6

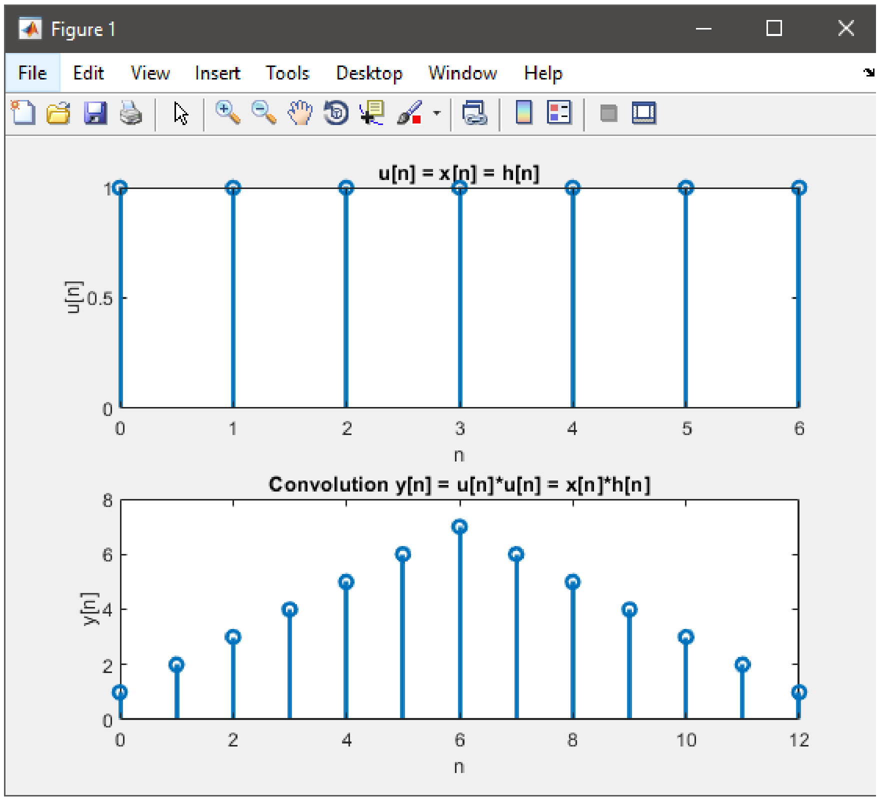
Task 05: Compute (by both procedures) and graph the convolution $y[n]$, where

(a) $y[n] = u[n] * u[n], 0 \le n \le 6$

(b) $y[n] = 3\delta[n-4] * (3/4)^n u[n], 0 \le n \le 5$

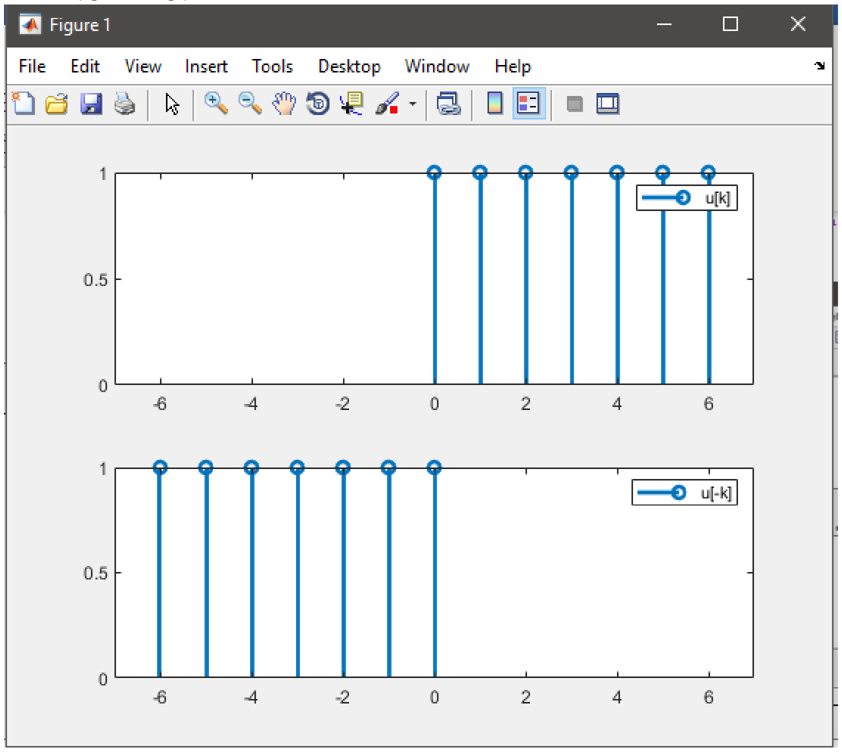
A)

```
n = 0:6;
u = ones(size(n));

subplot(2,1,1)
stem(n,u,'linewidth',2)
title('u[n] = x[n] = h[n]')
xlabel('n')
ylabel('u[n]')

y = conv(u,u);
subplot(2,1,2)
stem(0:12,y,'LineWidth',2)
title('Convolution y[n] = u[n]*u[n] = x[n]*h[n]')
xlabel('n')
ylabel('y[n]')
```

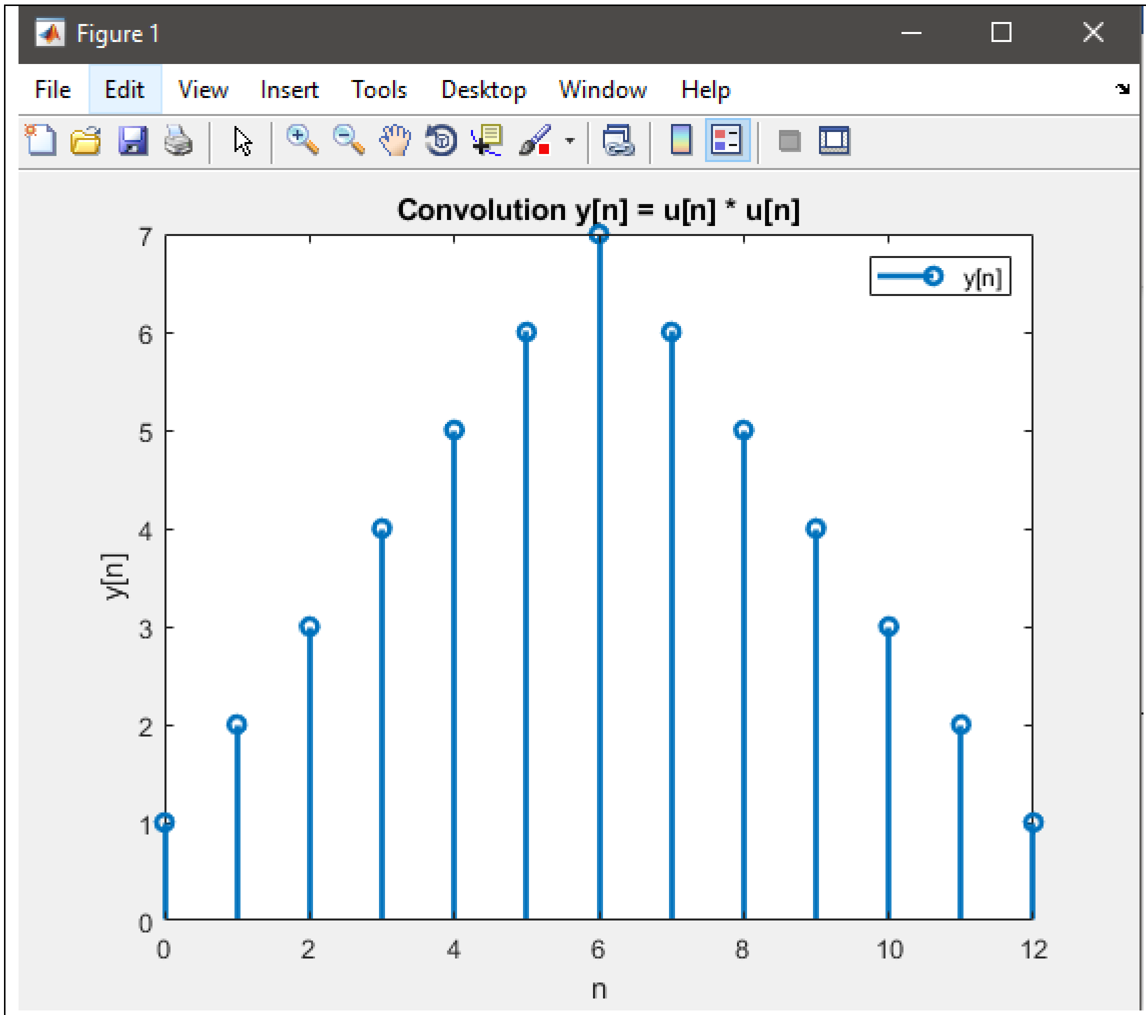


Other Method

```
k = 0:6;
u = ones(size(k));
subplot(2,1,1)
stem(k,u,'LineWidth',2)
legend('u[k]')
xlim([-7 7]);

subplot(2,1,2)
stem(-k,u,'LineWidth',2)
legend('u[-k]')
xlim([-7 7]);
```



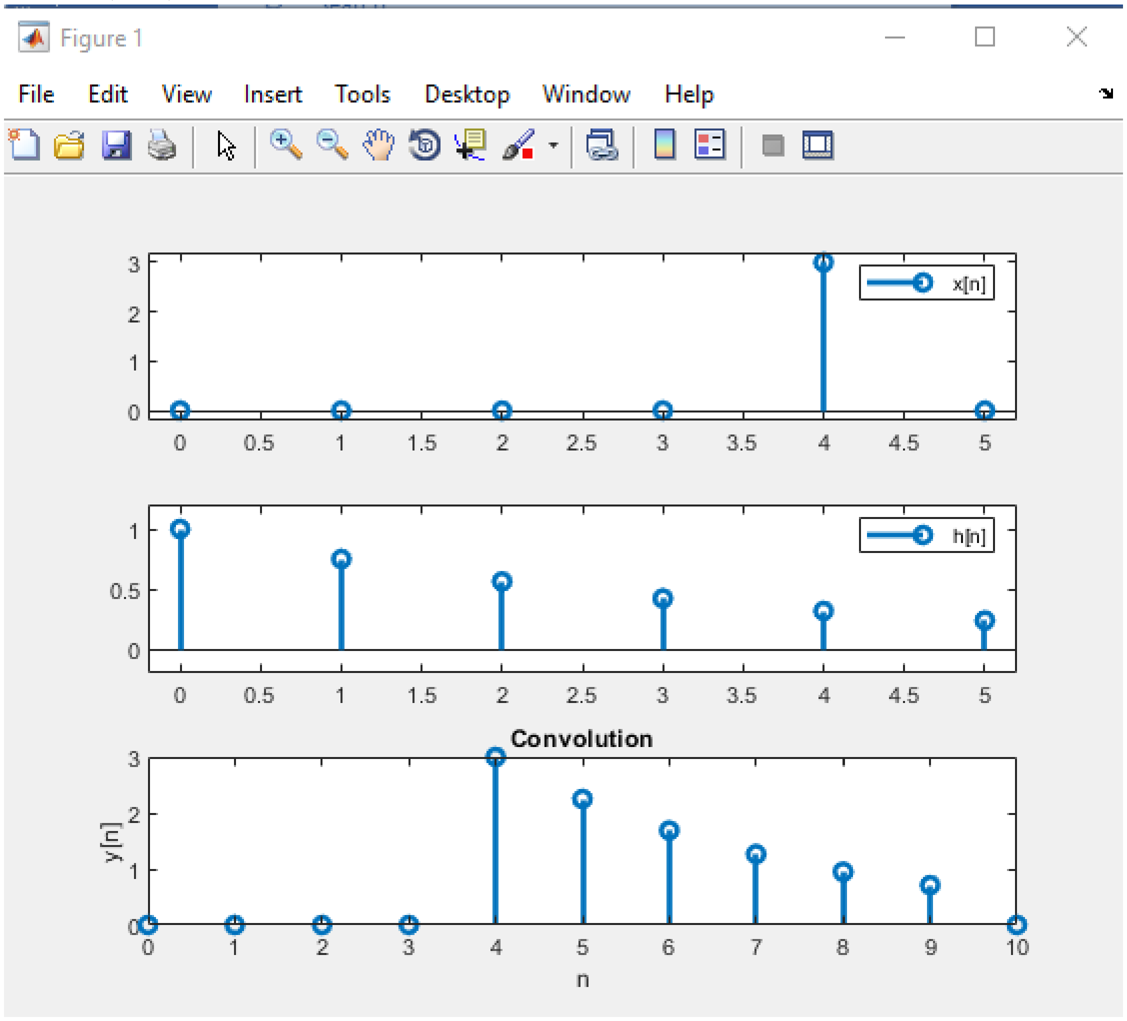
```
%Convolution
y = [1 2 3 4 5 6 7 6 5 4 3 2 1];
yn = 0:12;
stem(yn,y,'LineWidth',2)
legend('y[n]')
title('Convolution y[n] = u[n] * u[n]')
xlabel('n')
ylabel('y[n]')
```



B)

```
n = 0:5;
h = (3./4).^n.*(n>=0);
x = [0 0 0 0 3 0];
subplot(3,1,1)
stem(n,x,'LineWidth',2)
legend('x[n]')
axis([-0.2 5.2 -0.2 3.2])

subplot(3,1,2)
stem(n,h,'LineWidth',2)
legend('h[n]')
axis([-0.2 5.2 -0.2 1.2])
```

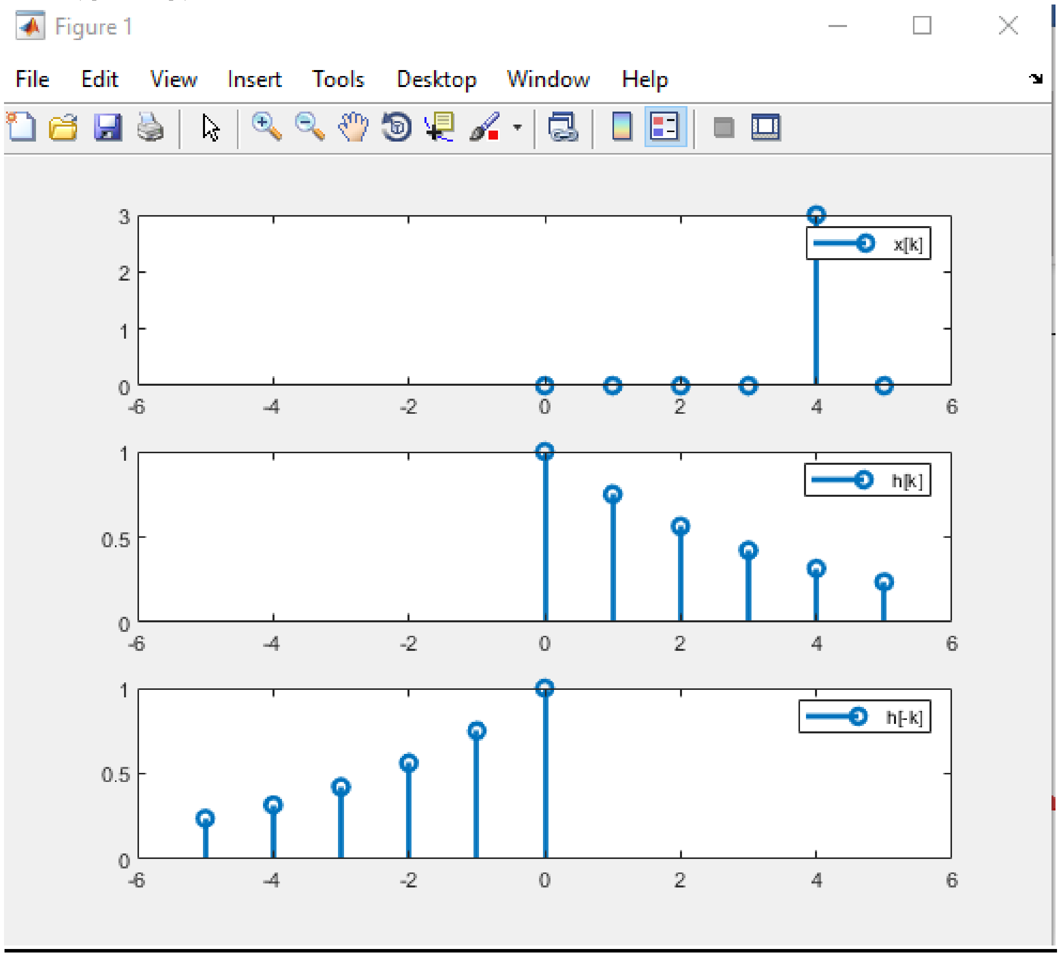
```
y = conv(x,h);
subplot(3,1,3)
stem(0:10,y,'LineWidth',2)
title('Convolution')
ylabel('y[n]')
xlabel('n')
```



Other Method:

```
k = 0:5;
x = [0  0  0  0  3  0];
h = (3./4).^k.*(k>=0);
subplot(3,1,1)
stem(k,x,'LineWidth',2)
legend('x[k]')
xlim([-6  6])

subplot(3,1,2)
stem(k,h,'LineWidth',2)
legend('h[k]')
```

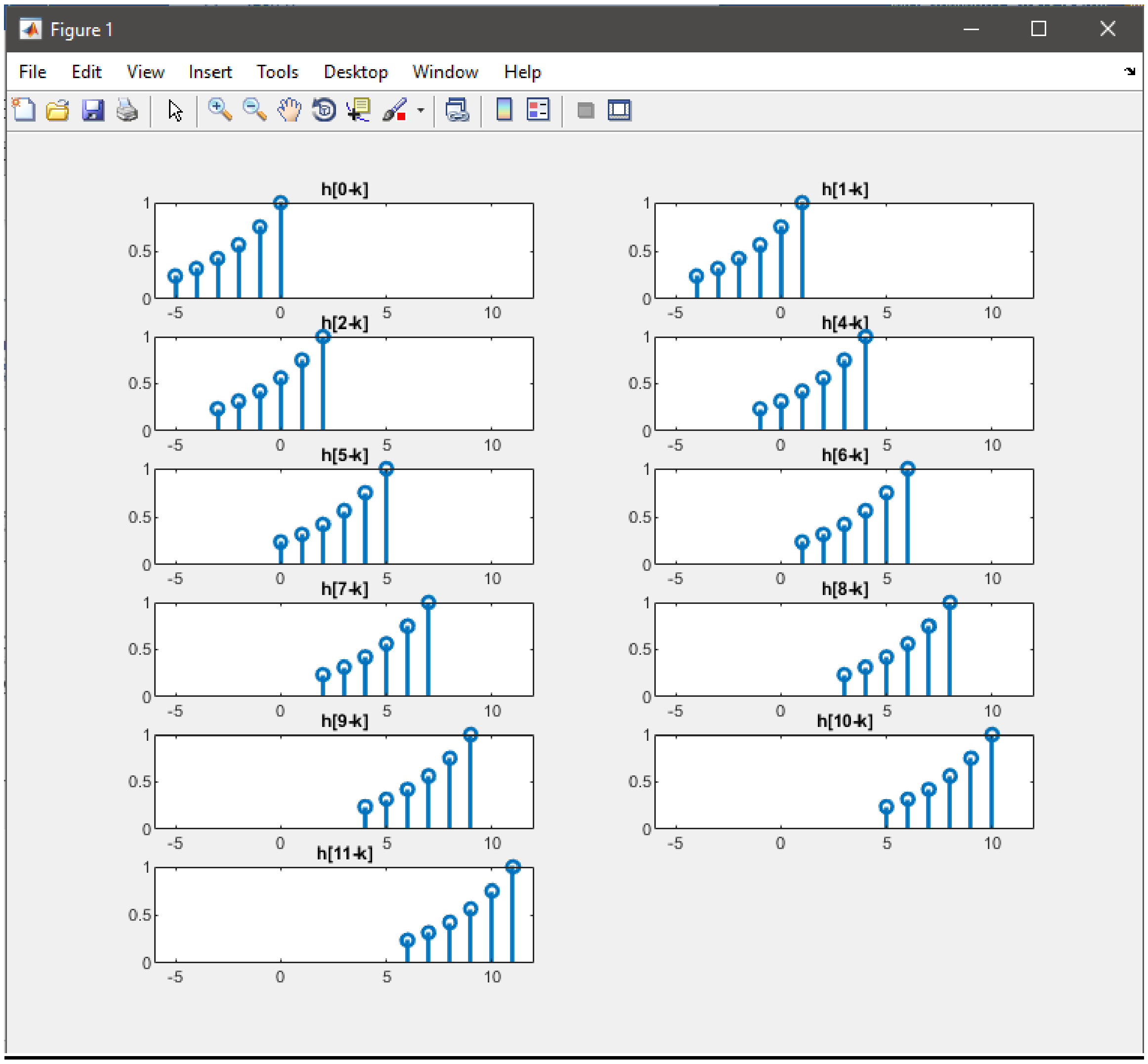
```
xlim([-6 6])

subplot(3,1,3)
stem(-k,h,'LineWidth',2)
legend('h[-k]')
xlim([-6 6])
```



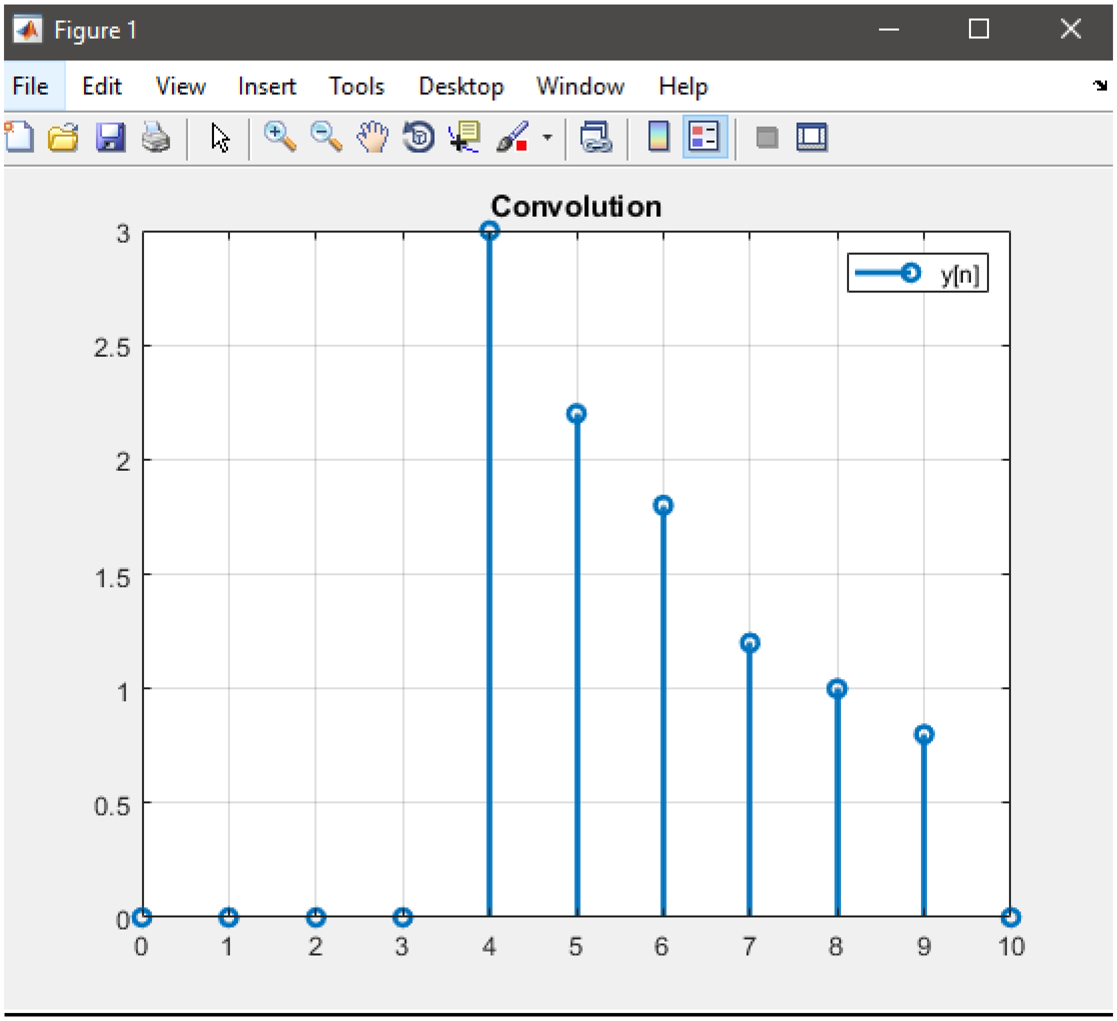
```
n=0;          %delay
subplot(6,2,1)
stem(-k+n,h,'LineWidth',2)
title('h[0-k]')
xlim([-6 12])

n=1;          %delay
subplot(6,2,2)
stem(-k+n,h,'LineWidth',2)
title('h[1-k]')
xlim([-6 12])

n=2;          %delay
```

```
subplot(6,2,3)
stem(-k+n,h,'LineWidth',2)
title('h[2-k]')
xlim([-6 12])

n=3;         %delay
subplot(6,2,4)
stem(-k+n,h,'LineWidth',2)
title('h[3-k]')
xlim([-6 12])

n=4;         %delay
subplot(6,2,4)
stem(-k+n,h,'LineWidth',2)
title('h[4-k]')
xlim([-6 12])

n=5;         %delay
subplot(6,2,5)
stem(-k+n,h,'LineWidth',2)
title('h[5-k]')
xlim([-6 12])

n=6;         %delay
subplot(6,2,6)
stem(-k+n,h,'LineWidth',2)
title('h[6-k]')
xlim([-6 12])

n=7;         %delay
subplot(6,2,7)
stem(-k+n,h,'LineWidth',2)
title('h[7-k]')
xlim([-6 12])

n=8;         %delay
subplot(6,2,8)
stem(-k+n,h,'LineWidth',2)
title('h[8-k]')
xlim([-6 12])

n=9;         %delay
subplot(6,2,9)
stem(-k+n,h,'LineWidth',2)
title('h[9-k]')
xlim([-6 12])

n=10;         %delay
subplot(6,2,10)
stem(-k+n,h,'LineWidth',2)
title('h[10-k]')
xlim([-6 12])
```

```
n=11;           %delay
subplot(6,2,11)
stem(-k+n,h,'LineWidth',2)
title('h[11-k]')
xlim([-6 12])
```



```
%Convolution
y = [0 0 0 0 3 2.2 1.8 1.2 1 0.8 0];
yn = 0:10;
stem(yn,y,'LineWidth',2),grid on
legend('y[n]')
title('Convolution')
```



Post-Lab Tasks

Critical Analysis / Conclusion

In this lab we learnt how to perform convolution in Matlab in two different ways. Firstly, we used matlab built-in function conv() which takes two argument i.e. the input signal and Impulse response of the signal .

Secondly, we use the conventional method to convolve the two signals in which we flip one of the signals in first step then we shift the flipped signal until there is an overlap between the input and impulse response.

Lab Assessment		
Pre-Lab	**/1**	**/10**
In-Lab	**/5**	
Critical Analysis	**/4**	
Instructor Signature and Comments		