

**Signals & Systems****EEE-223****Lab # 07**

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

# **LAB # 07**

## **Analysis of Continuous Time LTI Systems using Convolution Integral**

## Lab 07- Analysis of Continuous Time LTI Systems using Convolution Integral

### Pre-Lab Tasks

#### 7.1 Impulse Response:

The meaning of impulse response of the system is easily derived if one considers the terms that it is named from. The terms, ‘response’, denotes the output of a system, while the term, ‘impulse’ denotes that the input signal applied to the system is the unit impulse or Dirac Delta function. Hence, the impulse response of a causal linear and time invariant continuous time system is the output when the Dirac Delta function is the input applied to the system. Mathematically, impulse response of the system is denoted by

$$h(t) = S\{\delta(t)\}$$

#### 7.2 Continuous-Time Convolution:

The impulse response of a linear time-invariant system completely specifies the system. More, specifically, if the impulse response of a system is known one can compute the system output for any input signal.

The response (or output) of a system to any input signal is computed by the convolution of the input signal with the impulse response of the system.

Suppose that  $y(t)$  denotes the output of the system,  $x(t)$  is the input signal, and  $h(t)$  is the impulse response of the system. The mathematical expression of the convolution relationship is

$$y(t) = x(t) * h(t)$$

##### 7.2.1 Computation of Convolution:

In order to calculate the convolution between two signals, a specific (and not exactly trivial) computational procedure has to be followed. The convolution computational procedure is introduced through an example.

#### Example

A linear time-invariant signal is described by the impulse response

$$h(t) = \begin{cases} 1-t & 0 \leq t \leq 1 \\ 0 & \text{elsewhere} \end{cases}$$

Calculate the response of the system to the input signal

$$x(t) = \begin{cases} 1 & 0 \leq t \leq 2 \\ 0 & \text{elsewhere} \end{cases}$$

In order to compute the convolution between  $x(t)$  and  $h(t)$  the computational procedure is implemented in the following steps.

Step 1: Time impulse response signals are plotted in the  $\tau$ -axis; that is,  $t$  is replaced with  $\tau$ , or in other words the signals

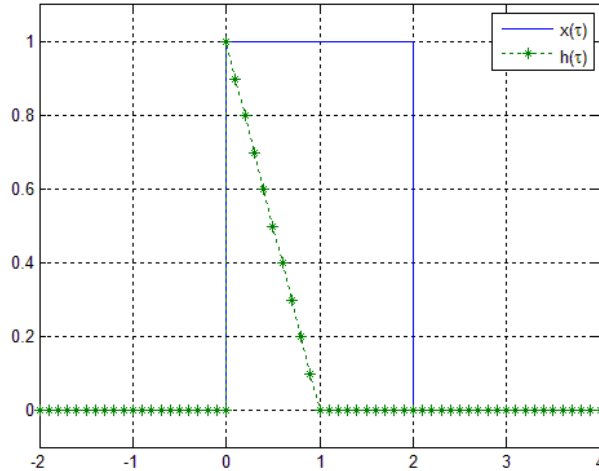
$$h(\tau) = \begin{cases} 1-\tau & 0 \leq \tau \leq 1 \\ 0 & \text{elsewhere} \end{cases} \text{ and } x(\tau) = \begin{cases} 1 & 0 \leq \tau \leq 2 \\ 0 & \text{elsewhere} \end{cases}$$

are defined and plotted.

Commands	Results	Comments
<pre>tx1=-2:0.1:0; tx2=0:0.1:2; tx3=2:0.1:4; tx=[tx1 tx2 tx3]; x1=zeros(size(tx1)); x2=ones(size(tx2)); x3=zeros(size(tx3)); x=[x1 x2 x3];</pre>		<p>The signals <math>x(\tau)</math> and <math>h(\tau)</math> are defined in the usual way of constructing two part functions. The input signal <math>x(\tau)</math> is plotted with solid line, while the impulse response signal <math>h(\tau)</math> is plotted with dotted line and asterisks. Both the signals are defined in the time interval <math>-2 \leq \tau \leq 4</math>. Notice that the signals are plotted in the <math>\tau</math>-axis and how Greek letters are</p>

inserted into the  
legend.

```
th1=-2:0.1:0;
th2=0:0.1:1;
th3=1:0.1:4;
th=[th1 th2 th3];
h1=zeros(size(th1));
h2=1-th2;
h3=zeros(size(th3));
h=[h1 h2 h3];
plot(tx,x,th,h,'*')
ylim([-0.1 1.1])
legend('x(\tau)', 'h(\tau)')
grid
```



Step2: The second step is known as reflection. One of the two signals is selected (in this example  $h(\tau)$  is chosen, but  $x(\tau)$  could have been also selected) and its symmetric with respect to the vertical axis, i.e.,  $h(-\tau)$  is plotted.

---

Commands

Results

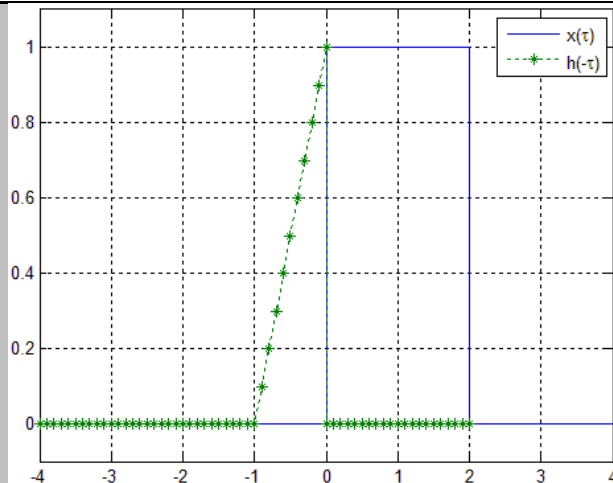
---

```
plot(tx,x,-th,h,':*)
```

```
ylim([-0.1 1.1])
```

```
legend('x(\tau)','h(-\tau)')
```

```
grid
```



Step 3: The third step is shifting. The signal  $h(-\tau)$  is shifted by  $t$ ; that is, the signal  $h(t-\tau)$  is plotted. Note that  $t$  is constant, as the variable of the defined signals is variable  $\tau$ .

**Commands**

**Results**

**Comments**

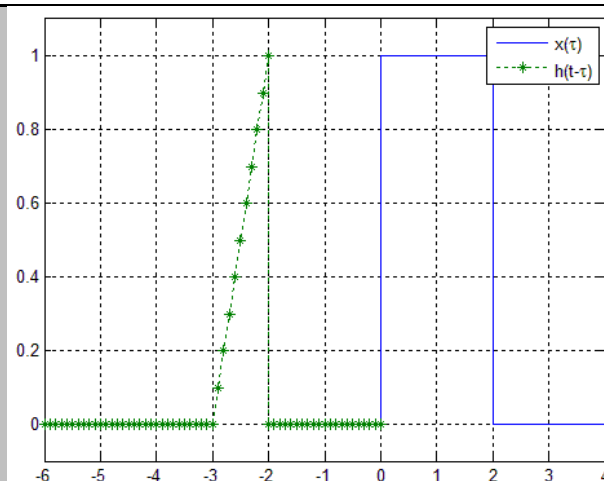
```
t=-2;
```

```
plot(tx,x,-th+t,h,':*)
```

```
ylim([-0.1 1.1])
```

```
legend('x(\tau)','h(t-\tau)')
```

```
grid
```

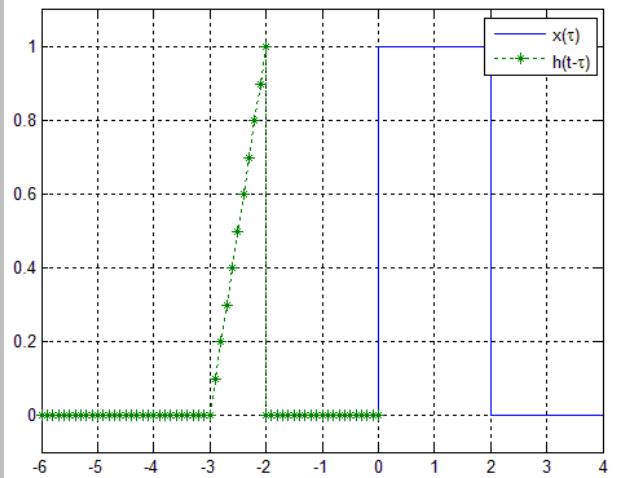


The signal  $h(t-\tau)$  is plotted for  $t = -2$ . Notice that  $h(t-\tau)$  is shifted by 2 units to the left compared to  $h(-\tau)$ .

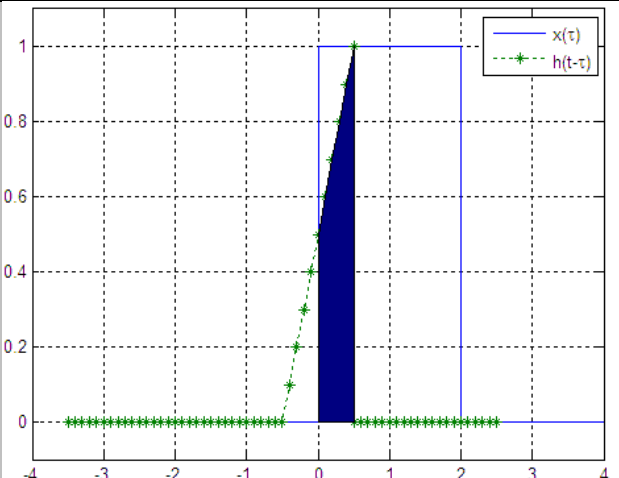
A very useful (for future computations) observation is the point in the  $\tau$ -axis where the right end of  $h(t-\tau)$  is. The right end  $h(t-\tau)$  for  $t = -2$ ; that is, the right end of  $h(-2-\tau)$  is at the point  $\tau = -2$ . In the previous graph, it has been observed that the right end of  $h(-\tau)$ , that is, the right end of  $h(0-\tau)$  is at  $\tau = 0$ . Hence, we conclude that in general case the right end of  $h(t-\tau)$  is the point  $\tau = t$ . On the other hand, the left end of  $h(t-\tau)$  is at  $\tau = t - T$ , where  $T$  is the duration of  $h(t-\tau)$ . In this example we have  $T = 1$ .

Step 4: The fourth step is sliding step. The value of the output signal  $y(t)$  at time  $t$ , that is, the value of convolution between the input signal and impulse response signal at time  $t$  depends on overlap between  $x(\tau)$  and  $h(t - \tau)$  at time  $t$ . In order to compute the convolution for all  $t$ , we have to slide the signal  $h(t - \tau)$  from  $-\infty$  to  $+\infty$  and to derive the kind (or stage) of overlap in reference to the value of  $t$ . Note that  $x(t)$  remains still.

- First Stage: Zero Overlap

Commands	Results	Comments
<pre> t=-2;  plot(tx,x,-th+t,h,'*')  ylim([-0.1 1.1])  legend('x(\tau)','h(t- \tau)')  grid                     </pre>		<p>For <math>t &lt; 0</math> (in this figure <math>t = -2</math>) the signal <math>h(t - \tau)</math> and <math>x(\tau)</math> do not overlap. Thus the output is <math>y(t) = 0</math>.</p>

- Second Stage: Partial Overlap

Commands	Results	Comments
<pre> t=0.5;  plot(tx,x,-th+t,h,'*'),grid on  ylim([-0.1 1.1])  legend('x(\tau)','h(t- \tau)')  % the following code % produce the shadowed % area plot                     </pre>		<p>For <math>0 &lt; t &lt; 1</math>, the signal <math>h(t - \tau)</math> is entering at <math>x(\tau)</math>. The two signals are partially overlapped. The overlapped part is shadowed.</p>

```
T=1;

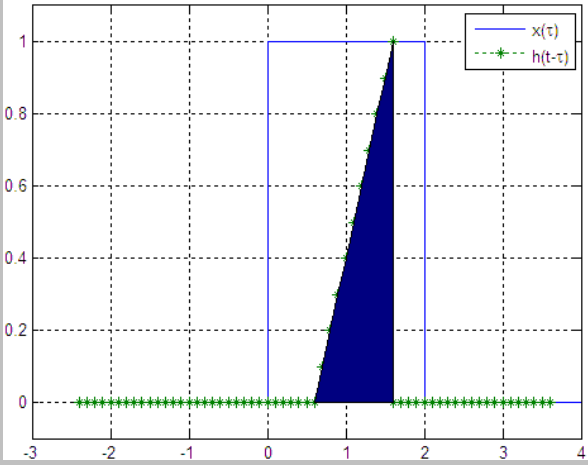
r=0:0.1:t;

a=1/T*r+1-t/T;

hold on; area(r,a);

hold off;
```

- Third Stage: Complete Overlap

Commands	Results	Comments
<pre>t=1.6;  plot(tx,x,-th+t,h,'*'),grid on  ylim([-0.1 1.1])  legend('x(\tau)','h(t-\tau)')  % the following code % produce the shadowed % area plot  T=1;  r=t-T:0.1:t;  a=1/T*r+1-t/T;  hold on; area(r,a);  hold off;</pre>		<p>For <math>1 &lt; t &lt; 2</math>, the signals <math>h(t - \tau)</math> and <math>x(\tau)</math> are completely overlapped.</p>

- Fourth Stage: Exit-partial overlap

Commands	Results	Comments
----------	---------	----------



```
t=2.4;

plot(tx,x,-th+t,h,'*'),grid
on

ylim([-0.1 1.1])

legend('x(\tau)','h(t-\tau)')

% the following code
% produce the shadowed
% area plot

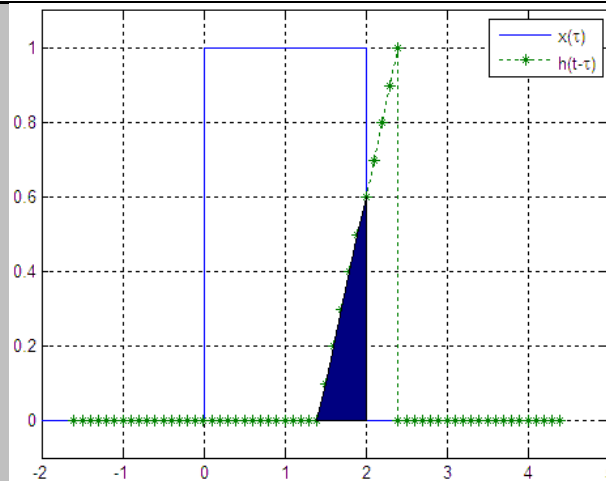
T=1;

r=t-T:0.1:2;

a=1/T*r+1-t/T;

hold on; area(r,a);

hold off;
```



For  $2 < t < 3$ , the signal  $h(t - \tau)$  exits  $x(\tau)$ . The two signals are partially overlapped.

- Fifth Stage: Zero Overlap

### Commands

### Results

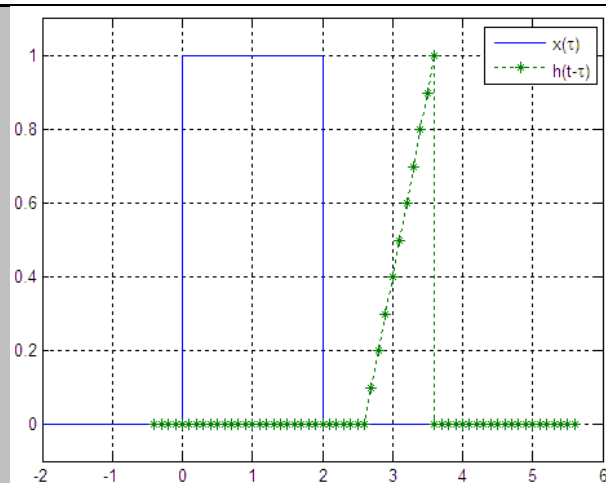
### Comments

```
t=3.6;

plot(tx,x,-th+t,h,'*'),grid
on

ylim([-0.1 1.1])

legend('x(\tau)','h(t-\tau)')
```



For  $t > 3$ , the signal  $h(t - \tau)$  does not overlap with  $x(\tau)$ . Thus the output is  $y(t) = 0$ .

Step 5: Specification of limits and calculation of the convolution integral. Having derived the time intervals for various stages of overlap, the convolution integral is computed separately for each stage. Before computing the integrals the integration limits have to be specified. Recall that the right end of  $h(t - \tau)$  is at point  $\tau = t$ , while the left end of  $h(t - \tau)$  is at point  $\tau = t - T = t - 1$ .

1. For  $t < 0$ , the two signals do not overlap (zero-overlap stage). Thus the response of the system is  $y(t) = 0$ .
2. For  $0 < t < 1$ , the two signals start to overlap (entry stage-partial overlap). The limits of the integral are the limits that specified the shadowed area. Hence  $y(t) = \int_0^t x(\tau)h(t-\tau)d\tau$ . The input signal  $x(\tau)$  is given by  $x(\tau) = 1$ , while the impulse response signal  $h(t - \tau)$  is given by  $h(t - \tau) = 1 - (t - \tau) = 1 - t + \tau$ . The expression of  $h(t - \tau)$  is derived by substituting  $t$  with  $t - \tau$  in  $h(t)$ . Thus the integral that has to be calculated is  $y(t) = \int_0^t 1(1-t+\tau)d\tau = \int_0^t (1-t+\tau)d\tau$

Commands	Results	Comments
<code>syms t r</code> <code>f=1-t+r;</code> <code>y=int(f,r,0,t)</code>	$y=t-1/2*t^2$	The integral is computed and the response of the system at the entry stage is $y(t) = t - t^2/2, 0 < t < 1$ .

3. For  $1 < t < 2$ , the two signals overlap completely (complete overlap stage). The only difference to the previous calculation is the integral limits. In this case, the output is given by  $y(t) = \int_{t-1}^t (1-t+\tau)d\tau$ .

Commands	Results	Comments
<code>y=int(f,r,t-1,t)</code> <code>simplify(y)</code>	$Y=1-t+1/2*t^2-1/2*(t-1)^2$ $ans=1/2$	The integral is computed and the result is simplified. The response of the system at complete overlap is $y(t) = 0.5, 1 < t < 2$ .

4. For  $2 < t < 3$ , the two signals overlap partially (exit stage). Thus, the output is given by  $y(t) = \int_{t-1}^2 (1-t+\tau)d\tau$ .

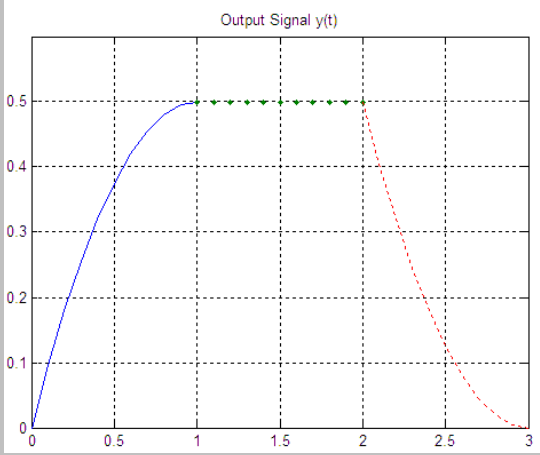
Commands	Results	Comments
<code>y=int(f,r,t-1,2)</code>	$Y=5-t-t*(3-t)-1/2*(t-1)^2$	The response of the system at exit stage is $y(t) = (3-t)^2/2, 2 < t < 3.$

5. For  $t > 3$ , there is no overlap; thus the output is  $y(t) = 0, t > 3$ .

Combining all the derived results, we conclude that the response of the system with impulse response  $h(t) = 1 - t, 0 \leq t \leq 1$  to the input signal  $x(t) = 1, 0 \leq t \leq 2$  is

$$y(t) = \begin{cases} t - t^2/2 & 0 \leq t \leq 1 \\ 1/2 & 1 \leq t \leq 2 \\ (3-t)^2/2 & 2 \leq t \leq 3 \\ 0 & t < 0 \text{ and } t > 3 \end{cases}$$

Finally, the output  $y(t)$  is plotted in MATLAB according to the technique of plotting multipart function.

Commands	Results
<pre> t1=0:0.1:1; t2=1:0.1:2; t3=2:0.1:3; y1=t1-(t1.^2)/2; y2=0.5*ones(size(t2)); y3=0.5*((3-t3).^2); plot(t1,y1,t2,y2,'t3,y3','t'),grid on ylim([0 0.6]) title('Output Signal y(t)') </pre>	

### 7.2.2 The Command conv:

The computational process followed in the previous subsection is the analytical way of deriving the convolution between two signals. In MATLAB, the command `conv` allows the direct computation of the convolution between two signals. In order to illustrate the `conv` command we consider the previously used example, namely the impulse response signal is given by  $h(t) = 1 - t, 0 \leq t \leq 1$  and an input signal given by  $x(t) = 1, 0 \leq t \leq 2$ . There are four 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  $a \leq t \leq b$ , where  $t = a$  is the first time instance that at least one of the signals  $x(t)$  or  $h(t)$  is not zero and  $t = b$  is the last time instance that at least one of the two signals is not zero. In our example, the time interval is  $0 \leq t \leq 2$ . Thus the input signal  $x(t) = 1, 0 \leq t \leq 2$  and the impulse response signal is expressed as

$$h(t) = \begin{cases} 1-t & 0 \leq t \leq 1 \\ 0 & 0 \leq t \leq 2 \end{cases}$$

- Second rule: When a signal consists of multiple parts, the time intervals in which each part is defined must not overlap. Therefore, the impulse response signal is defined as

$$h(t) = \begin{cases} 1-t & 0 \leq t \leq 1 \\ 0 & 1 < t \leq 2 \end{cases}$$

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

Commands	Comments
<code>step=0.01;</code>	The time step has to be quite small in order to approximate accurately the continuous time signal.
<code>t=0:step:2;</code>	The input signal $x(t) = 1, 0 \leq t \leq 2$ is defined.
<code>x=ones(size(t));</code>	
<code>t1=0:step:1;</code> <code>t2=1+step:step:2;</code>	The intervals of two parts of $h(t)$ are defined in such a way that they do not overlap. More specifically, vector $t2$ is defined from the instance $1+\text{step}$ , i.e., is defined from the time instance 1.01 (second rule). Also notice that the time step is used at the definition of the two signals must be same.

<b>h1=1-t1;</b>	The impulse response $h(t)$ is defined in the wider time interval, namely, in the same interval where the input $x(t)$ is defined (first rule).
<b>h2=zeros(size(t2));</b>	
<b>h=[h1 h2];</b>	

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  $y=\text{conv}(x,h)$ . However, there is still one detail that needs to be addressed and is described at the next rule.

- Third rule: The output of the conv command has to be multiplied with time step used in the definition of the input and impulse response signals, in order to correctly compute the output of the system. The rule emerges from the fact that the convolution integral is approximated by sum in MATLAB.

Commands	Comments
<b>y=conv(x,h)*step;</b>	The response $y(t)$ of the system is computed by convoluting the input signal $x(t)$ with the impulse response signal $h(t)$ and multiplying the result with the time step (third rule).

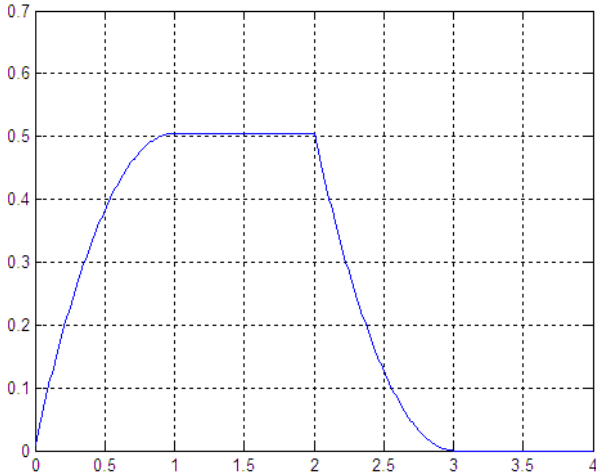
The response of the system is now computed and the only thing left to do is to plot it. However, the number of elements of the output vector  $y$  is not equal to the number of elements of vectors  $x$  or  $h$ .

The precise relationship is  $\text{length}(y)=\text{length}(x)+\text{length}(h)-1$

Commands	Results	Comments
<b>length(y)</b>	ans=401	Indeed the relationship $\text{length}(y)=\text{length}(x)+\text{length}(h)-1$ is true.
<b>length(x)</b>	ans=201	
<b>length(h)</b>	ans=201	

To overcome this, the fourth rule must be applied.

- Fourth 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.

Commands	Results	Comments
<code>ty=0:step:4;</code>		The time interval in which the output signal $y$ will be plotted is $0 \leq t \leq 4$ , namely, it is double from time interval where the vectors $x$ and $h$ are defined.
<code>plot(ty,y),grid on</code>		The response if the system $y(t)$ computed from the convolution between the input $x(t)$ and the impulse response $h(t)$ is plotted.

The output signal that was obtained through the MATLAB implementation (by using the command `conv`) is the same as the one derived with the analytical approach in section 7.2.1. In this point the necessity for applying the fourth rule must be already clear. In case if first and second rule was not followed it would not be possible to apply the fourth rule hardening the proper plotting of the output of the system.

### 7.2.3 Deconvolution:

Suppose that the impulse response of the system  $h(t)$  and output of the system is  $y(t)$  are available and we want to compute the input signal  $x(t)$  that was applied to the system in order to generate the output  $y(t)$ . The process called deconvolution and is implemented in MATLAB by using the command `deconv`. The syntax is `x=deconv(y,h)`, where  $x$  is the input vector,  $h$  is the impulse response vector, and  $y$  is the system response vector.

The deconvolution process is also useful for determining the impulse response of a system if the input and output signals are known. In this case the command is `h=deconv(y,x)`.

Remark

The commutativity property is not valid for `deconv` command; hence, the output signal must be the first input argument of the command `deconv`.

Example:

We will use the same signals used in the previous example. Therefore, the problem is to compute the impulse response  $h(t)$  of a system when the response of the system to the input  $x(t) = 1, 0 \leq t \leq 2$  is the signal  $y(t)$ , which is depicted in the previous figure.

The signals  $x(t)$  and  $y(t)$  and the time  $t$  are already defined in the previous example, so we can directly use them to compute the impulse response  $h(t)$ .

Commands	Results	Comments
<pre>hh=deconv(y,x)*(1/step); plot(t,hh),grid on ylim([-0.1 1.1]) legend('Impulse Response h(t)')</pre>		The impulse response $h(t)$ is computed by multiplying the outcome of the command <code>deconv</code> by the quantity $(1/\text{step})$ as deconvolution is the inverse operation of convolution.

## In-Lab Tasks

**Task 01:** Suppose that a LTI system is described by impulse response  $h(t) = e^{-t}u(t)$ . Compute the response of the system (by both methods) to the input signal

$$x(t) = \begin{cases} 0.6 & -1 \leq t \leq 0.5 \\ 0.3 & 0.5 \leq t \leq 3 \\ 0 & t < -1 \text{ and } t > 3 \end{cases}$$

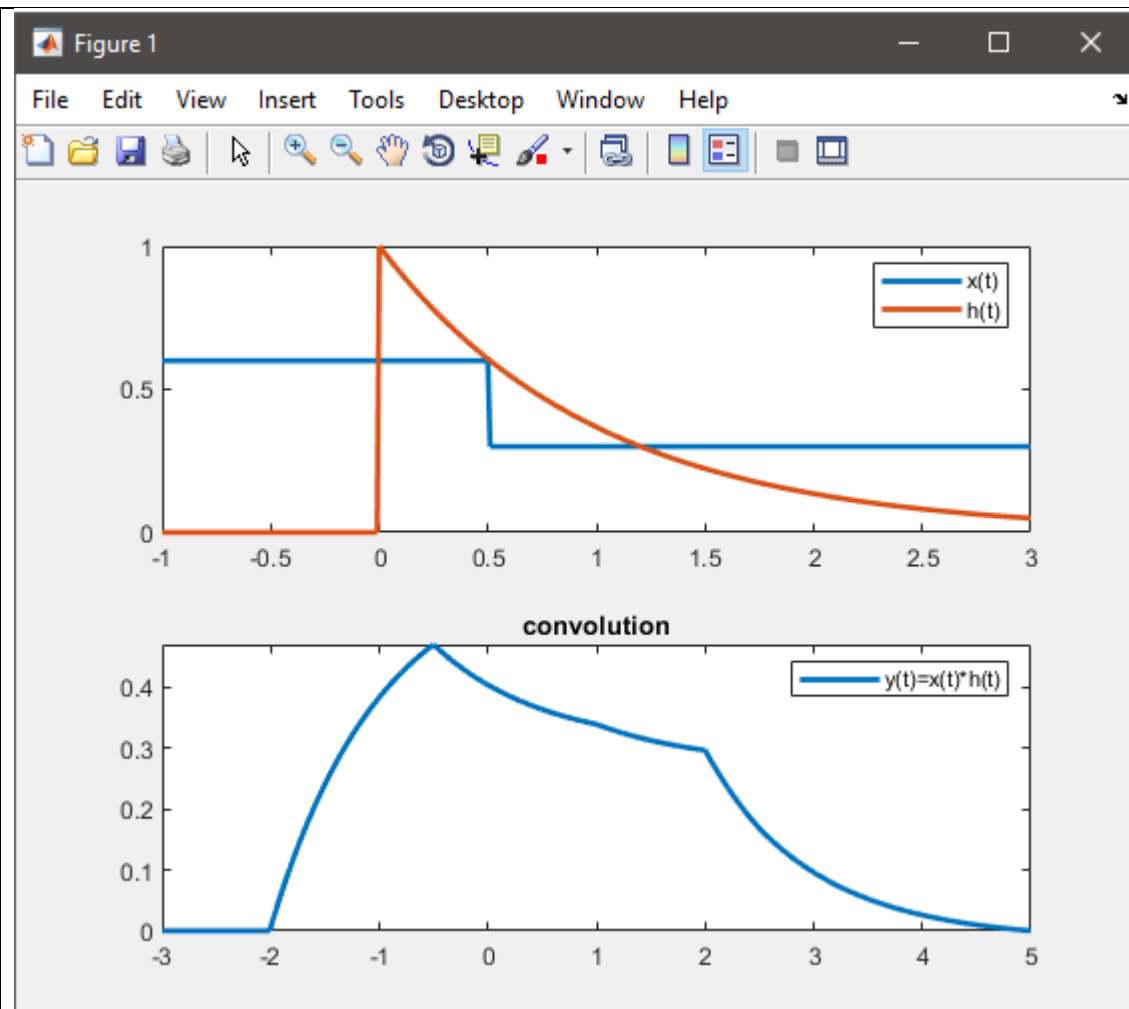
```
n = 0.01;
int1 = -1:n:0.5;
int2 = 0.5+n:n:3;
t = [int1 int2];

x1 = int1;
x1(:) = 1.*0.6;
x2 = int2;
x2(:) = 1.*0.3;
x = [x1 x2];

h = exp(-t).*(t>=0);
subplot(2,1,1);
plot(t,x,t,h,'LineWidth',2)
legend('x(t)', 'h(t)')

y = conv(x,h).*n;
z = -3:0.01:5;
subplot(2,1,2);
plot(z,y,'LineWidth',2)
legend('y(t)=x(t)*h(t)')
title convolution
```





### Other Method:

```
n = 0.01;
int1 = -1:n:0.5;
int2 = 0.5+n:n:3;
t = [int1 int2];

x1 = int1;
x1(:) = 1.*0.6;
x2 = int2;
x2(:) = 1.*0.3;
x = [x1 x2];

h = exp(-t).*(t>=0);
subplot(2,2,1) %n1
plot(t,x,t,h,'r')
xlabel('original');
```

```

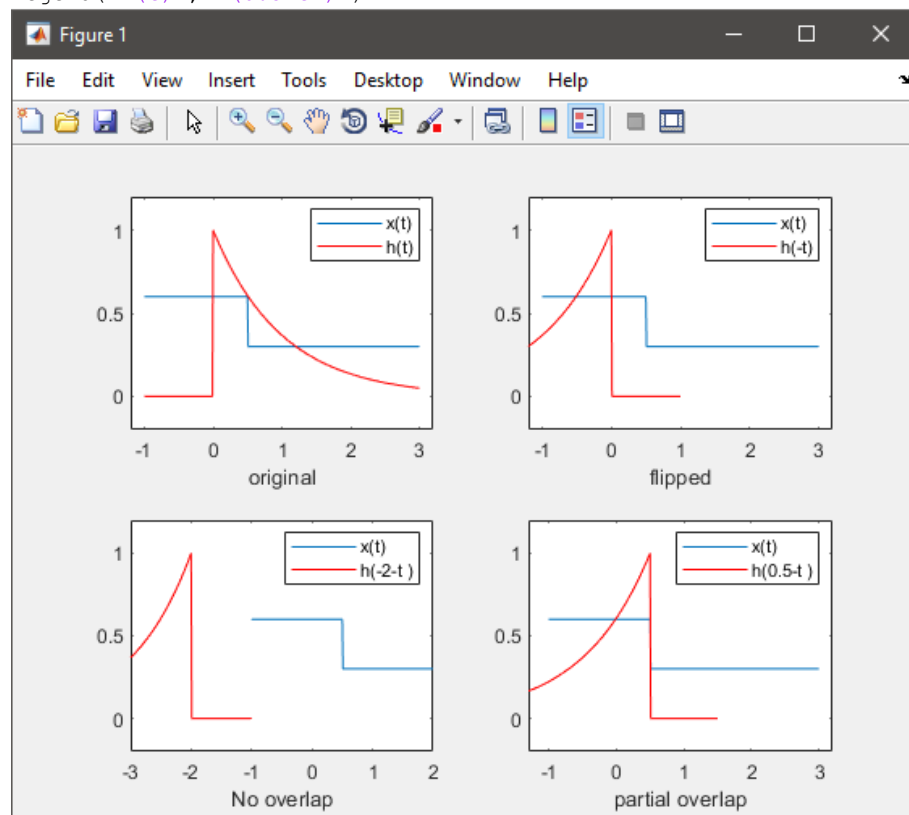
xlim([-1.2 3.2])
ylim([-0.2 1.2])
legend('x(t)', 'h(t)')

subplot(2,2,2)           %n2
plot(t,x,-t,h,'r')       %flipped
xlabel('flipped');
xlim([-1.2 3.2])
ylim([-0.2 1.2])
legend('x(t)', 'h(-t)')

subplot(2,2,3)           %n3
p = -2;                  %No Overlap
plot(t,x,-t+p,h,'r')
xlabel('No overlap');
xlim([-3 2])
ylim([-0.2 1.2])
legend('x(t)', 'h(-2-t)')

subplot(2,2,4)           %n4
p = 0.5;                 %Partial Overlap
plot(t,x,-t+p,h,'r')
xlabel('partial overlap');
xlim([-1.3 3.2])
ylim([-0.2 1.2])
legend('x(t)', 'h(0.5-t)')

```



```

subplot(2,2,1)           %Step5
p = 2;                   % Full Overlap
plot(t,x,-t+p,h,'r')
xlabel('Full overlap');
xlim([-1.3 3.2])

```

```

ylim([-0.2 1.2])
legend('x(\tau)', 'h(2.0-\tau)')

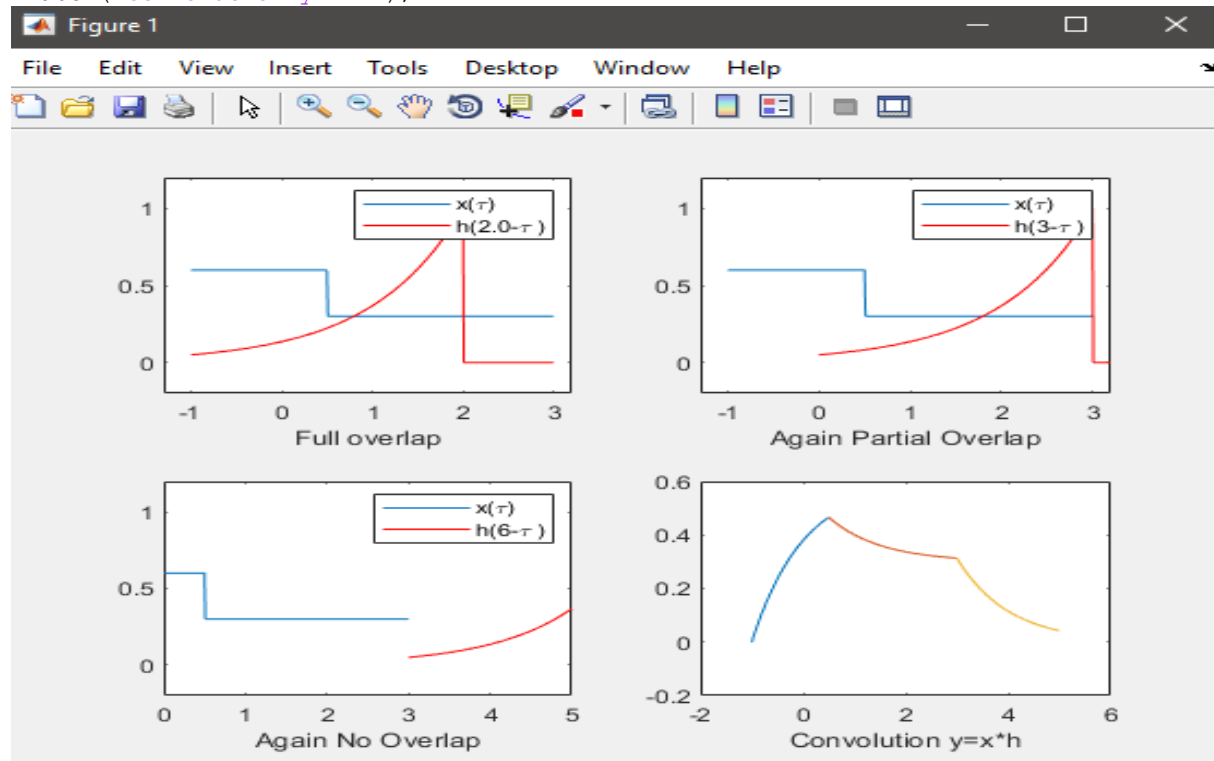
subplot(2,2,2)           %Step6
p = 3;                   %Exiting Overlap
plot(t,x,-t+p,h,'r')
xlabel('Again Partial Overlap');
xlim([-1.3 3.2])
ylim([-0.2 1.2])
legend('x(\tau)', 'h(3-\tau)')

subplot(2,2,3)           %Step7
p = 6;                   %No Overlap
plot(t,x,-t+p,h,'r')
xlabel('Again No Overlap');
xlim([0 5])
ylim([-0.2 1.2])
legend('x(\tau)', 'h(6-\tau)')

tt1 = -1:0.01:0.5;
tt2 = 0.5:0.01:3;
tt3 = 3:0.01:5;

yt1 = 3/5 - (3*exp(- tt1 - 1))/5;
yt2 = (3*exp(- tt2 - 1)*(exp(3/2) - 1))/5 - (3*exp(-tt2)*exp(1/2))/10 + 3/10;
yt3 = (3*exp(- tt3 - 1)*(exp(3/2) - 1))/5 + (3*exp(-tt3)*exp(1/2)*(exp(5/2) - 1))/10;
subplot(2,2,4);
plot(tt1,yt1,tt2,yt2,tt3,yt3);
ylim([-0.2 0.6]);
xlabel('Convolution y=x*h');

```



**Task 02: Compute (by both methods) and plot the response of the system**

$$x(t) = h(t) = \begin{cases} 0 & 0 < t < 1 \\ 1 & 1 \leq t \leq 2 \\ 0 & 2 \leq t \leq 10 \end{cases}$$

```

step = 0.01;
int1 = 0:step:1-step;
int2 = 1:step:2;
int3 = 2+step:step:10;
t_int = [int1 int2 int3];

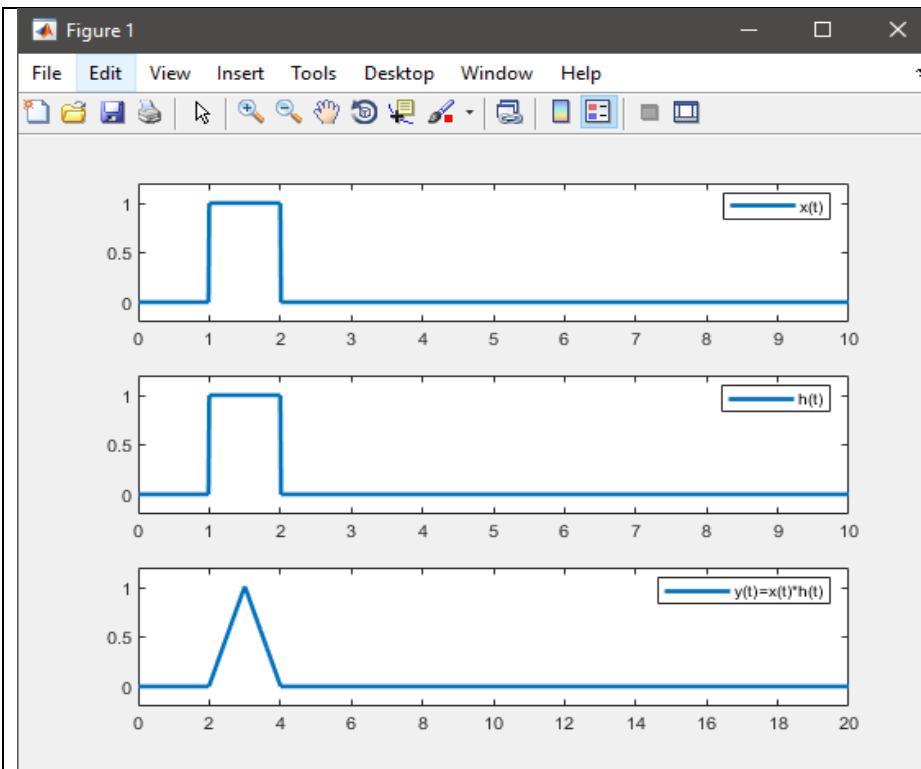
x1 = int1;
x1(:) = 0;
x2 = int2;
x2(:) = 1;
x3 = int3;
x3(:) = 0;
x = [x1 x2 x3];

h = x;
subplot(3,1,1)
plot(t_int,x,'LineWidth',2)
legend('x(t)')
ylim([-0.2 1.2])

subplot(3,1,2)
plot(t_int,h,'LineWidth',2);
legend('h(t)')
ylim([-0.2 1.2])

y = conv(x,h).*step;
z = 0:step:20;
subplot(3,1,3)
plot(z,y,'LineWidth',2);
legend('y(t)=x(t)*h(t)')
ylim([-0.2 1.2])

```



### Other Method:

```

step = 0.01;
int1 = 0:step:1-step;
int2 = 1:step:2;
int3 = 2+step:step:10;
t_int = [int1 int2 int3];

x1 = int1;
x1(:) = 0;
x2 = int2;
x2(:) = 1;
x3 = int3;
x3(:) = 0;
x = [x1 x2 x3];

h = x;

subplot(2,2,1)
plot(t_int,x,-t_int,h,'r','LineWidth',2)    %flipped
legend('x(t)','h(-t)')                    %No Overlap
xlabel('flipped & no overlap');
ylim([-0.1 1.1])
grid on

shift = 2.5;
subplot(2,2,2)
plot(t_int,x,'b',-t_int+shift,h,'r','LineWidth',2) %Partial Overlap
legend('x(t)','h(2.5-t)')
xlabel('Partial Overlap');

```

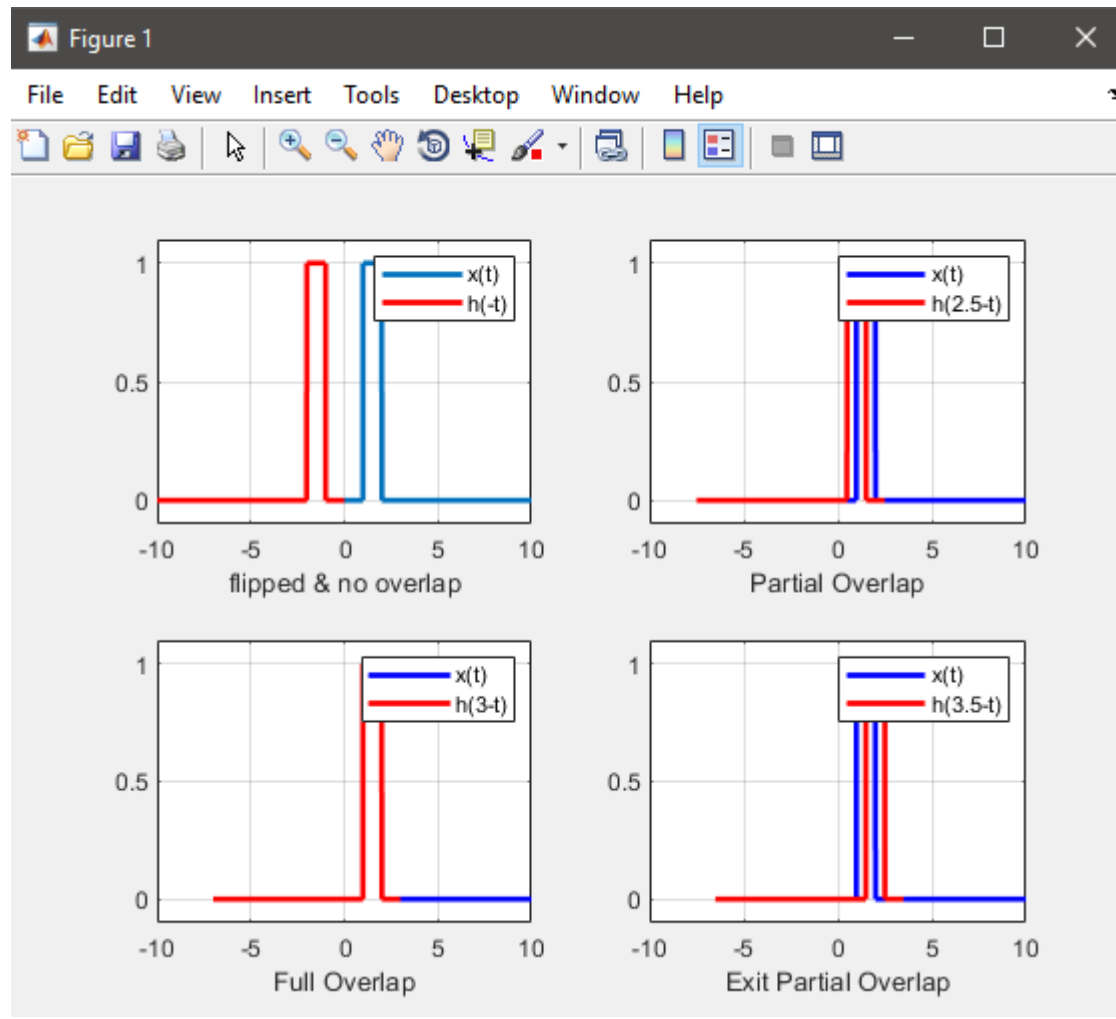
```

ylim([-0.1 1.1])
grid on

shift = 3;
subplot(2,2,3)
plot(t_int,x,'b',-t_int+shift,h,'r','LineWidth',2) %Full Overlap
legend('x(t)','h(3-t)')
xlabel('Full Overlap');
ylim([-0.1 1.1])
grid on

shift = 3.5;
subplot(2,2,4)
plot(t_int,x,'b',-t_int+shift,h,'r','LineWidth',2) %Partial Overlap
legend('x(t)','h(3.5-t)')
xlabel('Exit Partial Overlap');
ylim([-0.1 1.1])
grid on

```



```
shift = 3.5;
```

```

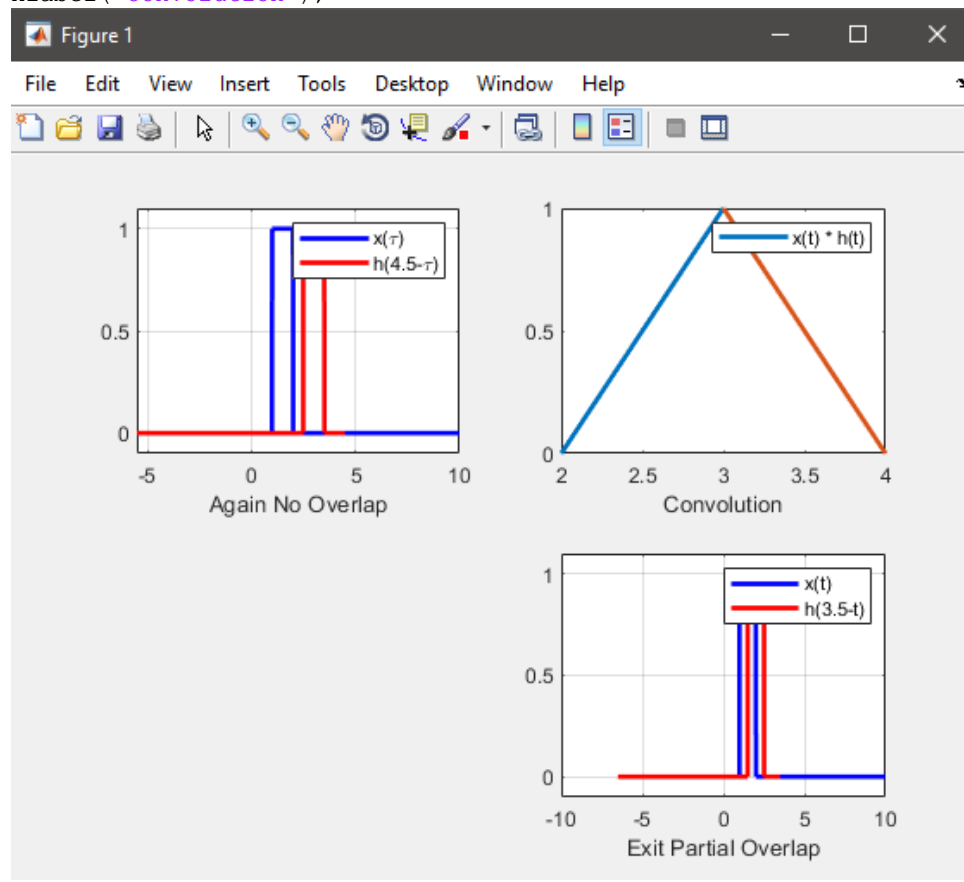
subplot(2,2,4)
plot(t_int,x,'b',-t_int+shift,h,'r','LineWidth',2) %Partial Overlap
legend('x(t)','h(3.5-t)')
xlabel('Exit Partial Overlap');
ylim([-0.1 1.1])
grid on

p = 4.5;
subplot(2,2,1)
plot(t_int,x,'b',-t_int+p,h,'r','LineWidth',2) %No Overlap
legend('x(\tau)','h(4.5-\tau)')
xlabel('Again No Overlap');
ylim([-0.1 1.1])
grid on

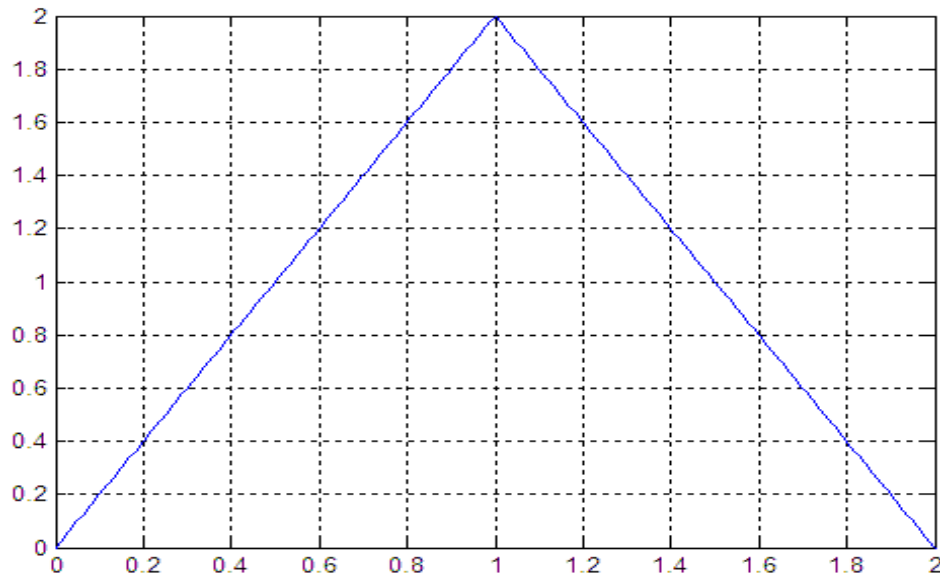
t1 = 2:0.01:3;
t2 = 3:0.01:4;

y1 = t1-2;
y2 = -t2+4;
subplot(2,2,2);
plot(t1,y1,t2,y2,'linewidth',2); %plotting the calculated equations
legend('x(t) * h(t)');
xlabel('Convolution');

```



**Task 03:** Suppose that a system is described by the impulse response  $h(t) = \cos(2\pi t)(u(t) - u(t-4))$ . Compute (by both methods) and plot the response of the system to the input shown in figure below



```

step = 0.01;
int1 = 0:step:1-step;
int2 = 1:step:2;
t_int = [int1 int2];

x1 = 2.*int1 ;
x2 = 4 - 2.*int2 ;
x = [x1 x2] ;

subplot(3,1,1)
plot(t_int,x,'LineWidth',2)
legend('x(t)')
xlim([0 8])
grid on

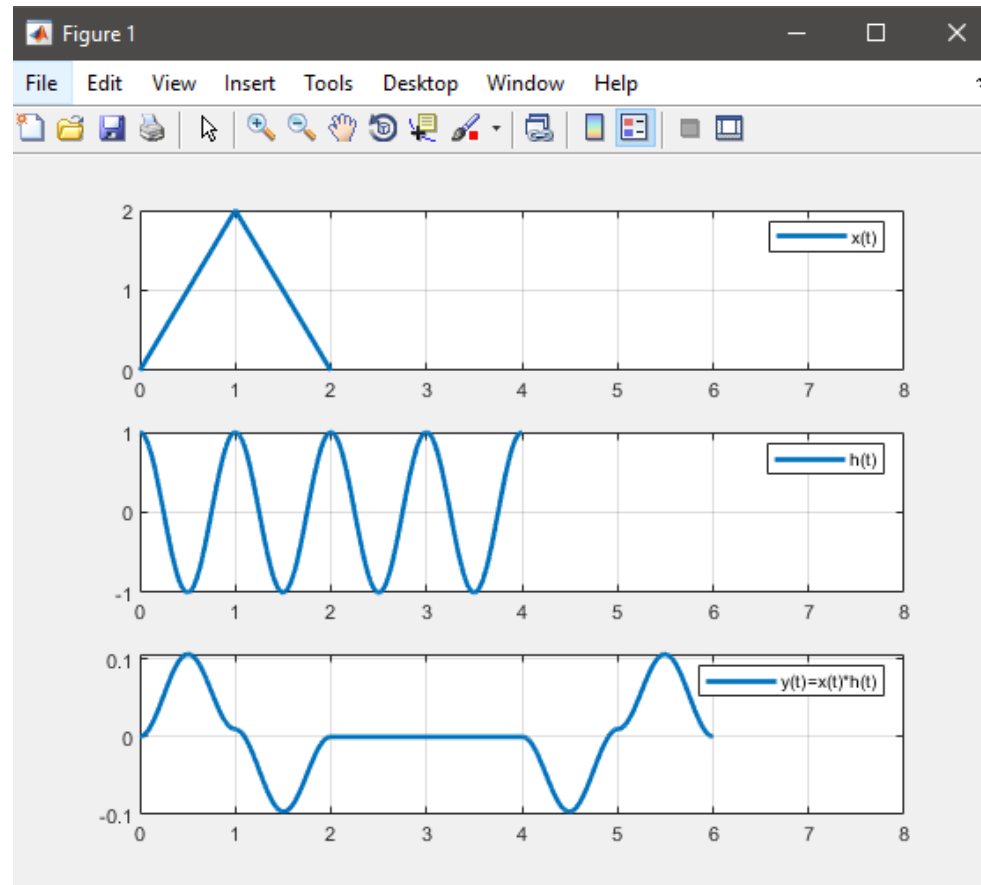
subplot(3,1,2)
h_int = 0:step:4;
h1 = h_int;
h1(:) = 1;
h2 = cos(2.*pi.*h_int);
h = h2.*h1;
plot(h_int,h,'LineWidth',2)
legend('h(t)')
xlim([0 8])
grid on

subplot(3,1,3)
y = conv(x,h).*step;
z = 0:step:6;
subplot(3,1,3)

```



```
plot(z,y,'LineWidth',2)  
legend('y(t)=x(t)*h(t)')  
xlim([0 8])  
grid on
```



**Other Method:**

```

step = 0.01;
int1 = 0:step:1-step;
int2 = 1:step:2;
t = [int1 int2];

x1 = 2.*int1 ;
x2 = 4 - 2.*int2 ;
x = [x1 x2] ;

h_int = 0:step:4;
h1 = h_int;
h1(:) = 1;
h2 = cos(2.*pi.*h_int);
h = h2.*h1;

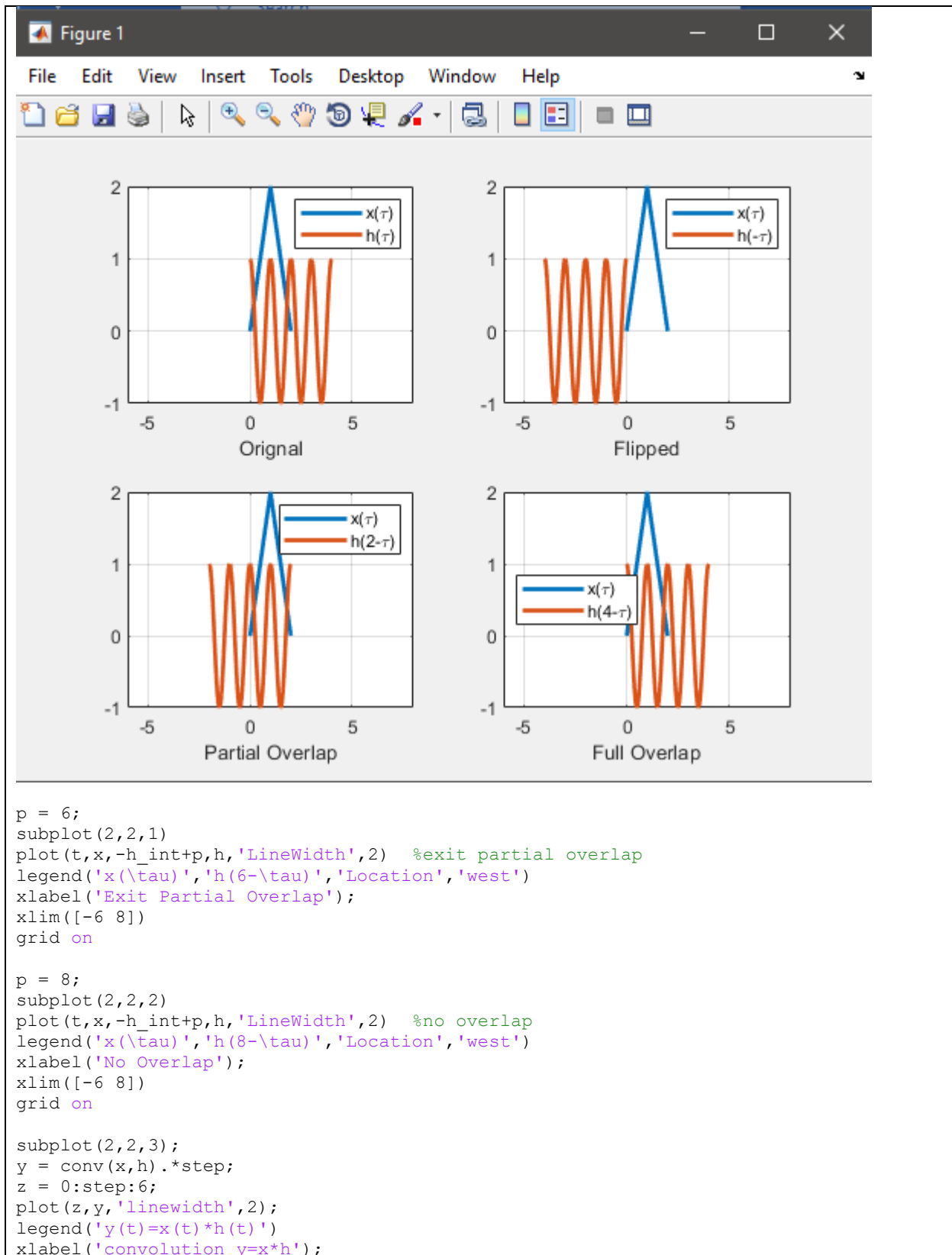
subplot(2,2,1)
plot(t,x,h_int,h,'LineWidth',2) %original
legend('x(\tau)', 'h(\tau)')
xlabel('Original');
xlim([-6 8])
grid on

subplot(2,2,2)
plot(t,x,-h_int,h,'LineWidth',2) %flipped
legend('x(\tau)', 'h(-\tau)')
xlabel('Flipped');
xlim([-6 8])
grid on

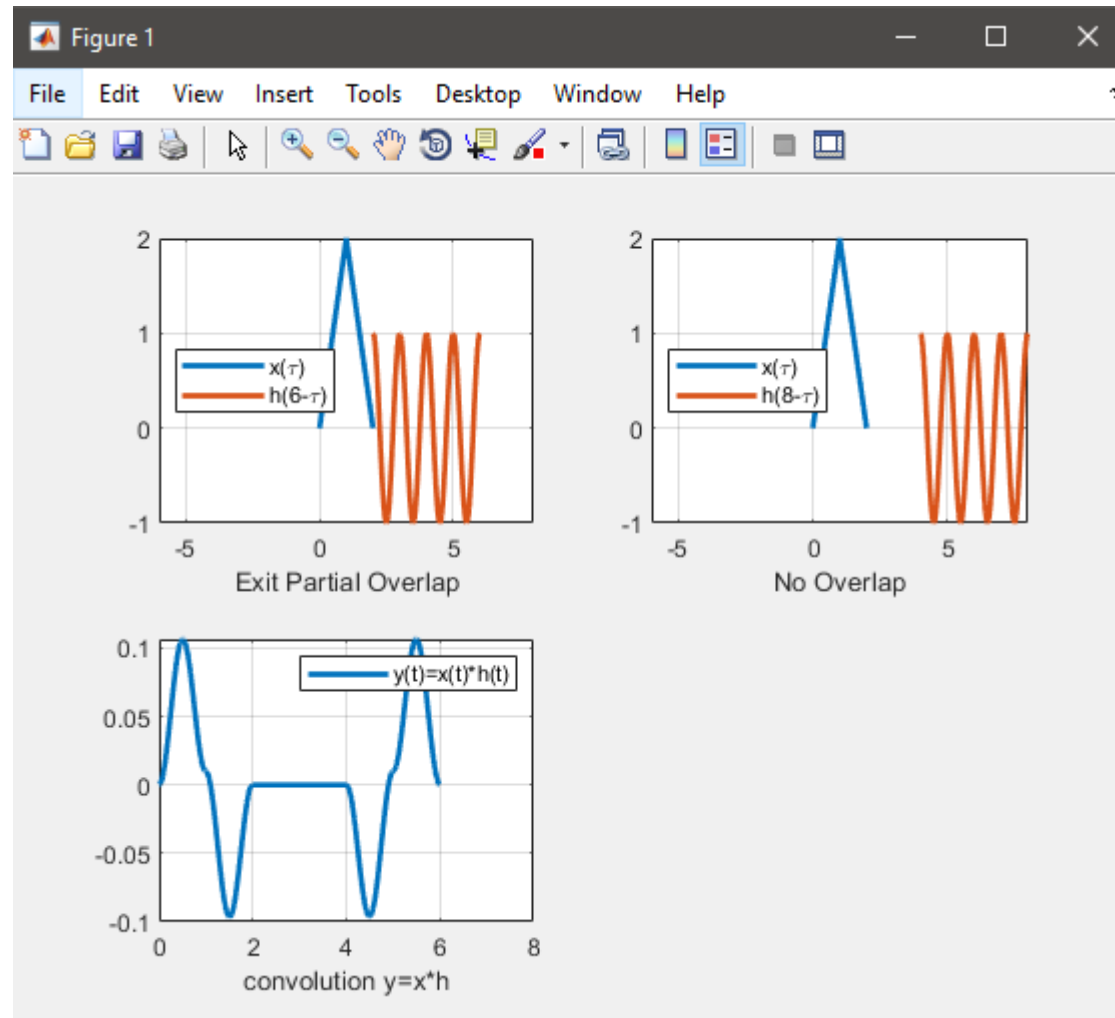
p = 2;
subplot(2,2,3)
plot(t,x,-h_int+p,h,'LineWidth',2) %partial overlap
legend('x(\tau)', 'h(2-\tau)')
xlabel('Partial Overlap');
xlim([-6 8])
grid on

p = 4;
subplot(2,2,4)
plot(t,x,-h_int+p,h,'LineWidth',2) %full overlap
legend('x(\tau)', 'h(4-\tau)', 'Location', 'west')
xlabel('Full Overlap');
xlim([-6 8])
grid on

```



```
xlim([0 8])
grid on
```



## Post-Lab Task

### Critical Analysis / Conclusion

In this lab we learnt how to perform convolution of continuous time signals in matlab. We learned two different methods of finding convolution as we did for the discrete time signals. In first method we used built-in conv() function of matlab to convolve an input signal and impulse response of the system. While dealing with continuous time signals, we should convolve the output of conv() function with step in order to obtain correct results. In the second method we used conventional method for finding convolution which includes the flipping of any of the two signals and then shifting and watching for the overlap of two signals.

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