

Experiments #6

Convolution and Linear Time Invariant Systems

1) Introduction:

In this lab we will explain how to use computer programs to perform a convolution operation on continuous time systems and know how we can use it to make an analysis into it and get output related to its system and the input.

2) Convolution and LTI systems:

2.1) Convolution:

The output $y(t)$ of a continuous-time linear time-invariant (LTI) system is related to its input $x(t)$ and the system impulse response $h(t)$ through the convolution integral expressed as:

$$y(t) = \int_{-\infty}^{\infty} x(\tau) h(t-\tau) d\tau$$

Equation (1)

But as we said before all computer programs operate in a discrete fashion, so to perform the above continuous-time convolution integral we need a numerical approximation for this integral noting that computer programs operate in a discrete not continuous fashion.

One way to approximate the continuous functions in the Equation (1) integral is to use piecewise constant functions.

These piecewise constant functions define $\delta_{\Delta}(t)$ to be a rectangular pulse of width Δ and amplitude 1, centered at $t = 0$, expressed as:

$$\delta_{\Delta}(t) = \begin{cases} 1 & , \quad -\frac{\Delta}{2} \leq t \leq \frac{\Delta}{2} \\ 0 & , \quad \text{otherwise} \end{cases}$$

Equation (2)

Now, if we approximate a continuous signal $x(t)$ with a piecewise constant function $x_{\Delta}(t)$ as a sequence of pulses spaced every Δ seconds in time with amplitude $x(k\Delta)$:

$$x_{\Delta}(t) = \sum_{k=-\infty}^{\infty} x(k\Delta)\delta_{\Delta}(t - k\Delta)$$

Discretized signal $x(t)$

It can be shown as continuous time as $\Delta \rightarrow 0$, $x_{\Delta}(t) \rightarrow x(t)$.

Similarly $h(t)$ can be approximated by:

$$h_{\Delta}(t) = \sum_{k=-\infty}^{\infty} h(k\Delta)\delta_{\Delta}(t - k\Delta)$$

Discretized impulse response $h(t)$

As an example, Figure 6.1 shows the approximation of a decaying exponential function $x(t) = e^{(-\frac{1}{2}t)}$ with amplitude starting from 1 and goes to 0 at $t \rightarrow \infty$ using $\Delta = 1$ second.

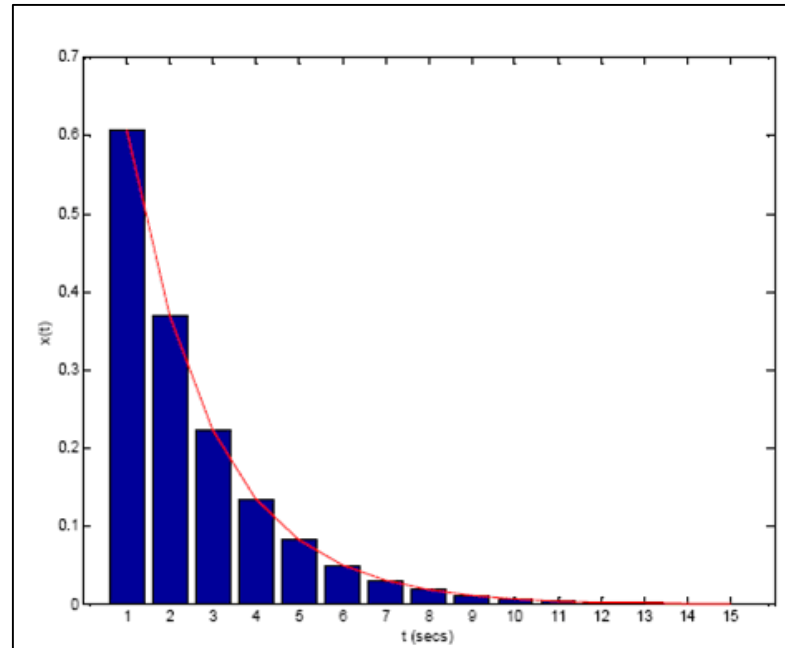


Figure 6.1: Approximation of a Decaying Exponential with Rectangular pulse of width 1sec

One can thus approximate the convolution integral by convolving the two piecewise constant signals as follows:

$$y_{\Delta}(t) = \int_{-\infty}^{\infty} x_{\Delta}(\tau) h_{\Delta}(t-\tau) d\tau$$

Equation (3)

Notice that $y_{\Delta}(t)$ is not necessarily a piecewise constant, for computer representation purposes, discrete output values are needed, which can be obtained by further approximating the convolution integral as indicated below:

$$y_{\Delta}(t) = \Delta \sum_{k=-\infty}^{\infty} x(k\Delta) h(n\Delta - k\Delta)$$

Equation (4)

If one represents the signals $h_{\Delta}(t)$ and $x_{\Delta}(t)$ in a M-file by vectors containing the values of the signals at $t = n\Delta$, then Equation (4) can be used to compute an approximation to the convolution of $x(t)$ and $h(t)$.

Compute the discrete convolution summation $\sum_{k=-\infty}^{\infty} x(k\Delta)h(n\Delta - k\Delta)$ with the built in LabView MathScript command (**conv**).

Then, multiply this summation by Δ to get an estimate of $y(t)$ at $t = n\Delta$, note that as Δ is made smaller, one gets a closer approximation to $y(t)$.

To compute the mean squared error (MSE) between the true and approximated output values using the following equation:

$$MSE = \frac{1}{N} \sum_{n=1}^N (y(n\Delta) - y_{\Delta}(n\Delta))^2$$

Equation (5)

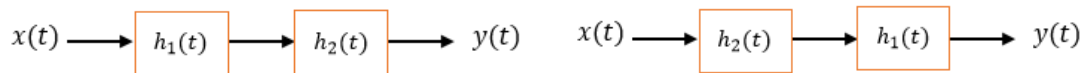
Notice that $N = \text{floor}\left(\frac{T}{\Delta}\right)$, T is an adjustable time duration expressed in seconds.

Conv. Function computes the convolution between two finite duration sequences and it's assume that the two sequences starting at $n = 0$.

2.2) convolution properties:

The main three properties for a convolution are:

- **Commutative:**



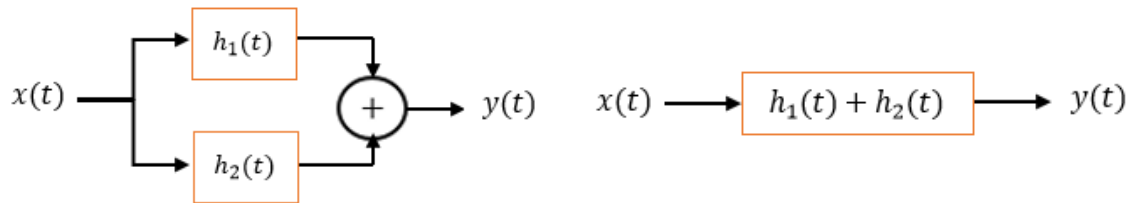
$$x(t) * h(t) = h(t) * x(t)$$

- **Associative:**



$$x(t) * h_1(t) * h_2(t) = x(t) * (h_1(t) * h_2(t))$$

- **Distributive:**



$$x(t) * (h_1(t) + h_2(t)) = x(t) * h_1(t) + x(t) * h_2(t)$$

2.3) Implement convolution:

As we know there are two types for Linear Time Invariant systems which are continuous time and discrete time systems, so I need to know how I can implement the convolution operation on these types.

a) Discrete Convolution:

In this example, use the function **conv** to compute the convolution of the signals $x[n] = \{-1, 3, -1, -2\}$ and $h[n] = \{-2, 2, 0, -1, 1\}$ to find $y[n]$ (**using MatLab and LabView**)

- **By MatLab program:**

Open new script file and save it in specific location then write the M-File to compute the discrete convolution.

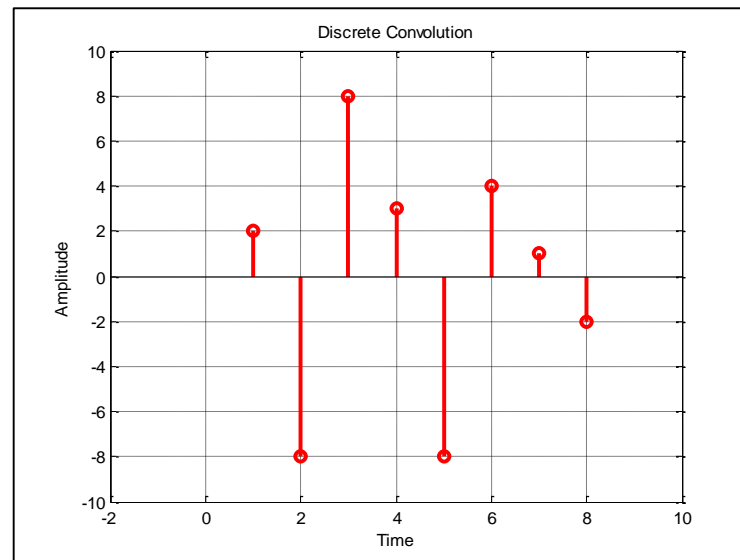
First, enter the two signals ($x[n]$ & $h[n]$) as row vectors and use the function **conv** to compute the discrete convolution as shown in figure 6.2.

```

1
2     %***** Discrete Convolution %*****
3
4     x=[-1 3 -1 -2];
5     h=[-2 2 0 -1 1];
6     y=conv(x,h);
7     stem(y,'r','linewidth',2.5)
8     axis([-2 10 -10 10])
9     grid on
10    xlabel('Time')
11    ylabel('Amplitude')
12    title('Discrete Convolution')

```

Figure 6.2: M-File textual code

Figure 6.3: The output signal $y[n]$

Now, we face an important problem which is how I can know the starting and the ending for finite output sequence $y[n]$ and its length.

To fix this problem we need to use some rules that help us to know these variables for output signal from the input signal $x[n]$ and impulse response $h[n]$.

$$x[n] \text{ defined } \forall n_{xs} \leq n \leq n_{xe} \text{ \& } h[n] \text{ defined } \forall n_{hs} \leq n \leq n_{he}$$

So I can compute the length of output sequence and determine the starting and ending points

- Length $y[n] = N + M - 1$, where N : length of $x[n]$ and M : length of $h[n]$.
- Starting point of $y[n] = n_{xs} + n_{hs}$.
- End point of $y[n] = n_{xe} + n_{he}$

Now we need to write new function that return the output sequence and its index, as shown in figure 6.4.

```

1      %***** Modified Convolution Function %*****
2
3      function [y,ny]=convolution(x,nx,h,nh)
4      y=conv(x,h);
5      nys=nx(1)+nh(1);
6      nye=nx(length(x))+nh(length(h));
7      ny=[nys:nye];

```

Figure 6.4: Defined a Modified Discrete Convolution function

The Input arguments for this function are:

- $X[n]$: Discrete input signal.
- $h[n]$: Impulse response for discrete system.
- nx : is a row vector contain the starting and ending points for input signal.
- nh : is a row vector contain the starting and ending points for impulse response.
- ny : is a row vector contain the starting and ending points for output signal.
- *length*: A function that return the length of signal use it to indicate the last element in row vector of signal.

```

15      %***** Modified Discrete Convolution %*****
16
17      x=[-1 3 -1 -2];
18      nx=[-1:2];
19      h=[-2 2 0 -1 1];
20      nh=[-1:3];
21      [y,ny]=convolution(x,nx,h,nh)
22      stem(ny,y,'b','linewidth',2.5)
23      axis([-4 10 -10 10])
24      grid on
25      xlabel('Time')
26      ylabel('Amplitude')
27      title('Discrete Convolution')

```

Figure 6.5: M-File textual code for modified convolution function

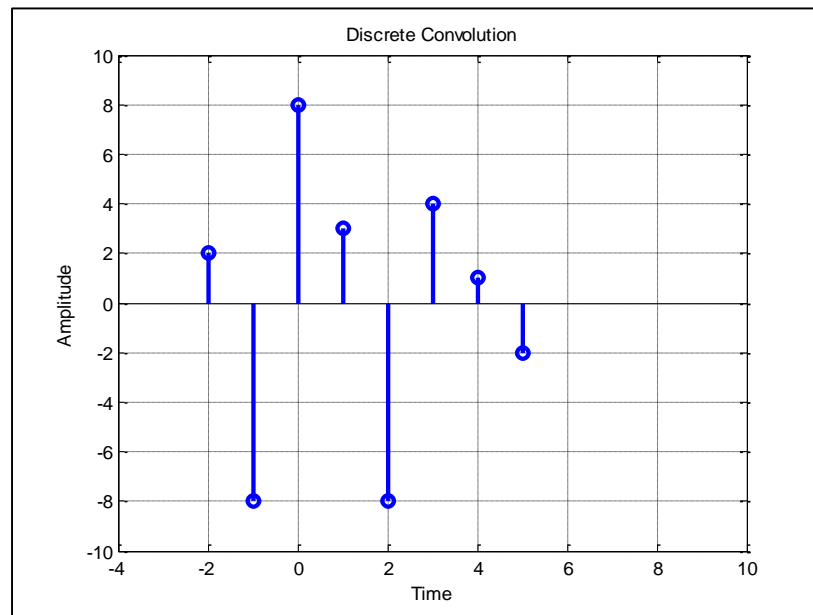


Figure 6.6: The output signal $y[n]$ with respect to true value of n

So the output sequence $y[n] = \{2, -8, 8, 3, -8, 4, 1, -2\}$.

- **By LabView program:**

Now, we will implement the previous discrete convolution by using LabView program via (using MathScript window or MathScript node) the same, but here we need to build hybrid program using M-File textual code into block diagram and user interface into front panel.

First, enter the two discrete signals ($x[n]$ & $h[n]$) as row vectors in MathScript work space and use a function **conv** to compute the discrete convolution.

Using MathScript node right clicking (Function → programming → structures → MathScript Node) add it in blank white area as while loop, and write the M-File textual code inside it as shown in figure 6.7.

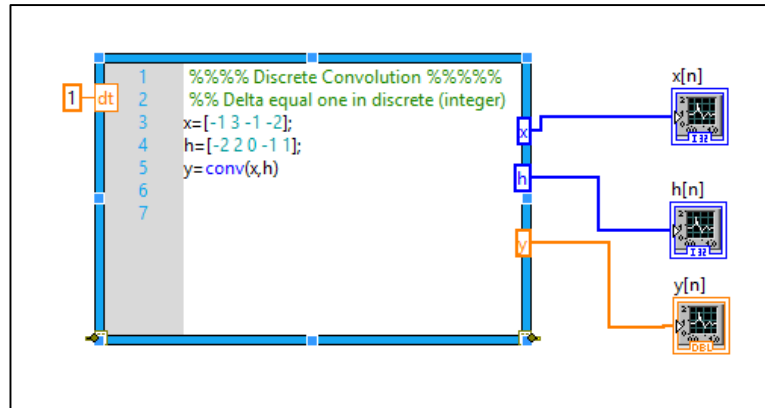


Figure 6.7: MathScript node consist the textual M-File code for discrete convolution

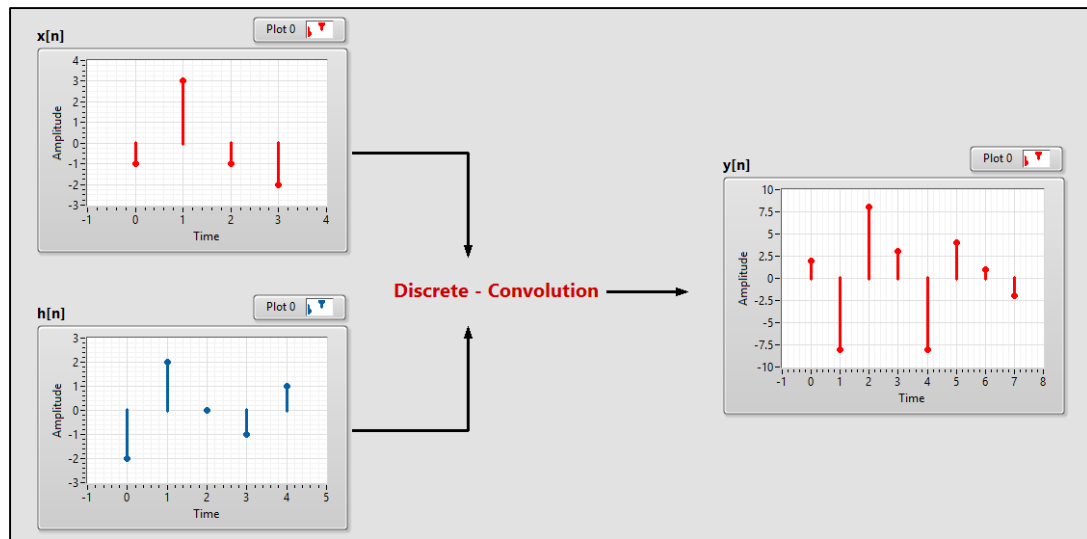


Figure 6.8: The front panel of hybrid program that compute the discrete convolution

We face the same problem which is I can't know the starting and the ending point for a finite output sequence $y[n]$ and its length so all answer will be have the same starting point which is $n=0$, that is not correct.

To fix this problem we need to use some rules that help us to know these variables for output signal from the input signal $x[n]$ and impulse response $h[n]$ as shown before.

The Modified discrete convolution function (**convolution**) which defined before in MatLab, I will recall it in LabView and use it to compute the discrete convolution.

Choose (Tools → MathScript window → new script), then write the definition of a modified function or copy the last code and use it in LabView as shown in figure 6.9.

```

1 %***** Modified Convolution Function %*****
2
3 function [y,ny]=convolution(x,nx,h,nh)
4 y=conv(x,h);
5 nys=nx(1)+nh(1);
6 nye=nx(length(x))+nh(length(h));
7 ny=[nys:nye];
8 end
    
```

Figure 6.9: The M-file for Modified convolution function

Then insert MathScript node into block diagram and add the output signal (x, h, y) to display it in front panel via XY-waveform graph to sketch the relation between index and amplitude of the signals (x, h, y) as shown in figure 6.10 and 6.11.

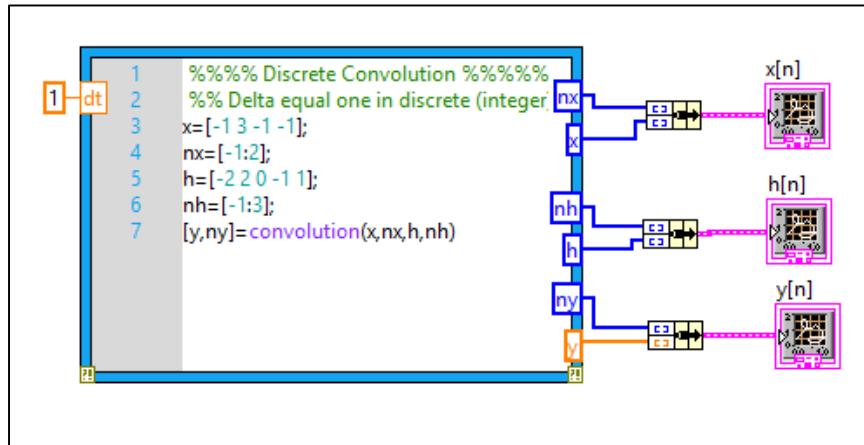


Figure 6.10: The block diagram and MathScript node create to compute discrete convolution

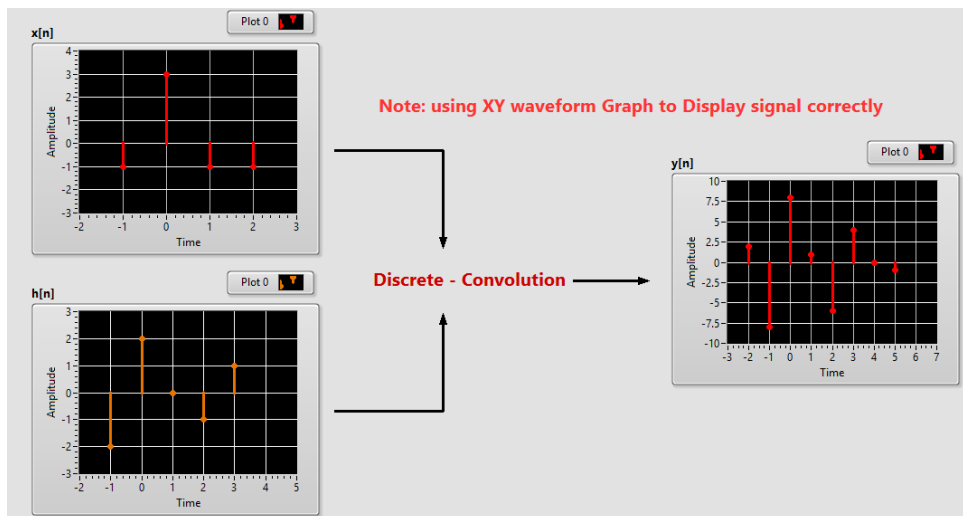
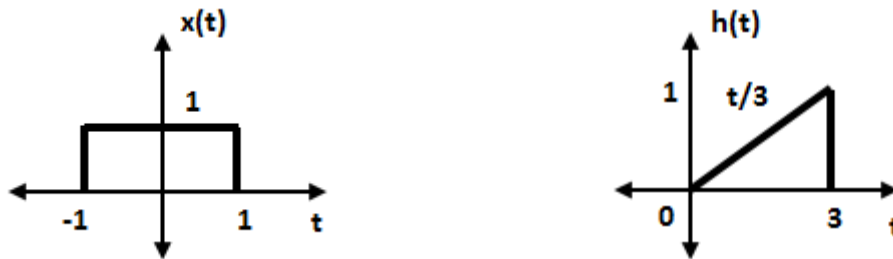


Figure 6.11: The front panel for hybrid program the computer the discrete convolution

b) Continuous Convolution:

In this example, use the function **conv** to compute the convolution of the continuous signals $(x(t), h(t))$ with interval of time between $-1 \leq t \leq 4$, where the both signal are shown below:



- **By MatLab program:**

To apply the continuous convolution by using MatLab, we need first to find the approximation for to convolution integral using time interval delta, and we will assume it equal small number as $\Delta = 0.001 \text{ sec}$, and using discrete convolution function **conv** and multiply the approximation answer by Delta to get an estimate of $y(t)$.

Now, we need to write the expression for $x(t)$ and $h(t)$ to write it in M-File:

- $x(t) = u(t + 1) - u(t - 1)$.
- $h(t) = \left(\frac{1}{3}\right) t[u(t) - u(t - 3)]$.

Then insert it into M-File and compute the continuous convolution and plot the output signal using plot function as shown in figure 6.12.

Note: you need to defined to time interval (t & T) the first on to sketch a given signals and the second to sketch the output signal to avoid the problem of the different length and execute the code without error.

```

##### Continuous Convolution #####
t=-1:0.001:4;    %% Delta=0.001
T=-2:0.001:8;
x=u(t+1)-u(t-1);    %% input signal x(t)
h=(1/3)*t.*(u(t)-u(t-3)); %% impulse response h(t)
y=(conv(x,h)).*0.001; %% multiply by Delta to get an estimate of y(t)
length(y)
subplot(2,2,1)    %% plot the input signal
plot(t,x,'r','linewidth',2.5)
xlabel('Time')
ylabel('Amplitude')
title('Input Signal')
grid on
axis([-2 3 -0.5 2])
subplot(2,2,2)    %% plot the impulse response
plot(t,h,'b','linewidth',2.5)
xlabel('Time')
ylabel('Amplitude')
title('Impulse Response Signal')
grid on
axis([-1 4 -0.5 2])
subplot(2,1,2)    %% plot the output signal
plot(T,y,'k','linewidth',2.5)
xlabel('Time')
ylabel('Amplitude')
title('Output Signal')
grid on
axis([-2 5 -0.5 2])

```

Figure 6.12: The textual M-File code for the previous example

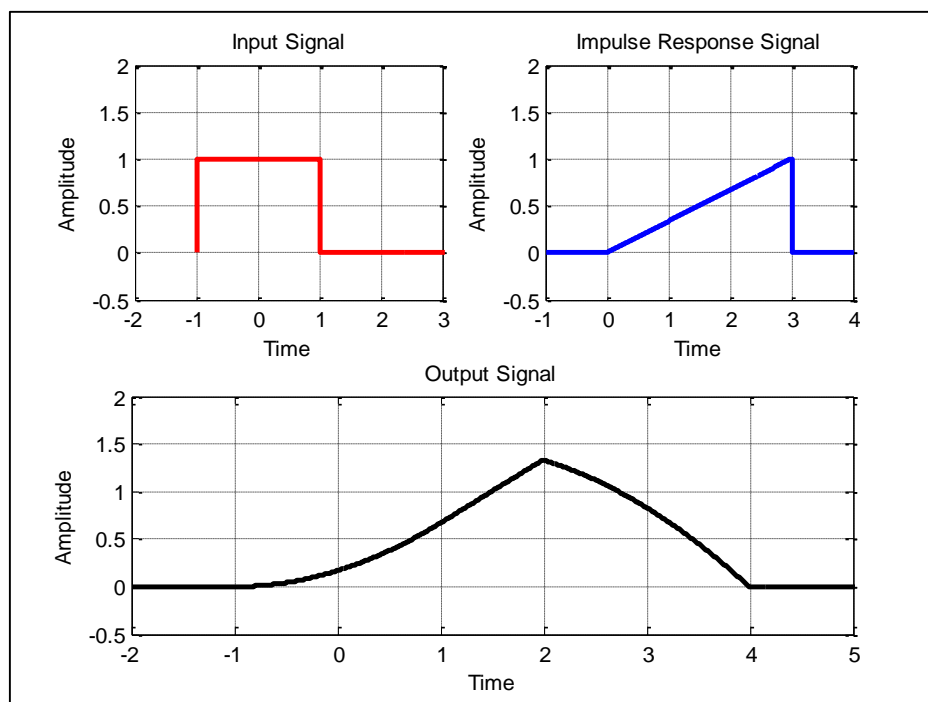


Figure 6.13: this figure display all signals need it in convolution operation

There are graphical user interface into MatLab for compute the convolution of LTI system by using GUI Tools, you can download this library via google search for this names (cconvdemo, dconvdemo) one for continuous and another for discrete convolution download it and add it to path of MatLab to be able to recall at any time.

Use the MatLab tool **CCONVDEMO** that helps visualize the process of continuous-time convolution by running it and:

- Choose pulse wave for $x(t)$ and exponential wave for $h(t)$ with options have seen in the given signal.
- Make the Flip $h(t)$ option is selected.
- Move the mouse on the flipping signal to move it and see the convolution region.
- Note the multiplication region in multiplication graph.
- Make sure the analytically and simulation results are typical.

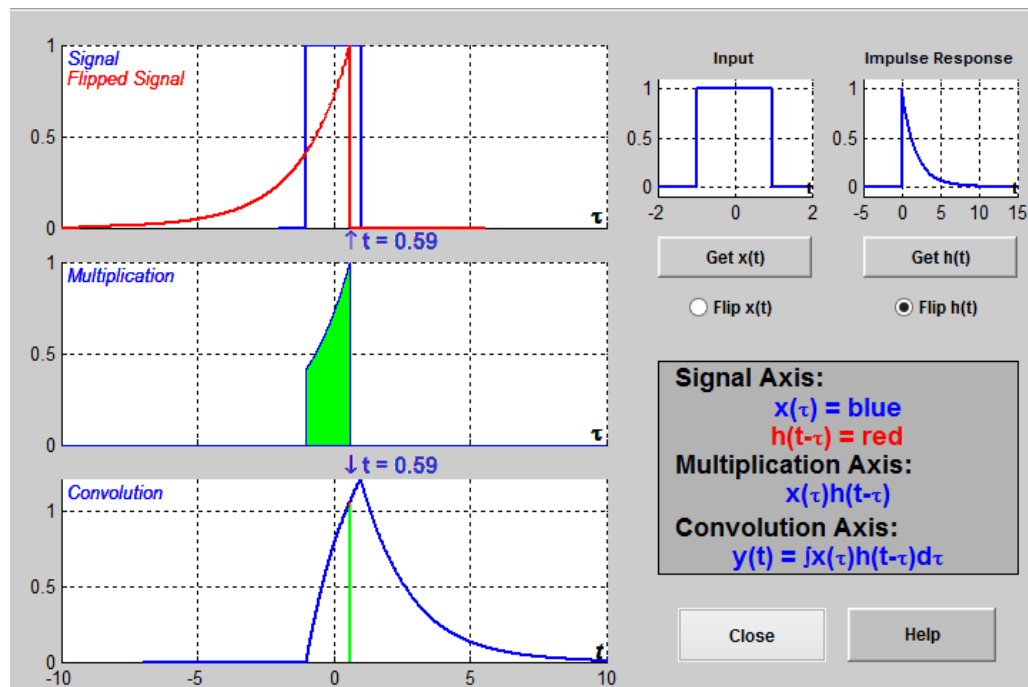


Figure 6.13: cconvdemo GUI to compute continuous convolution

Use the MatLab tool **DCONVDEMO** that helps visualize the process of discrete -time convolution.

- **By LabView program:**
 - **Example one:**

Let us apply the LabView MathScript function **conv** to compute the convolution of two signals, now we need to choose various values of the time interval Δ to compute the numerical approximations to the convolution integral as shown in equation (4).

In the previous example, we need to use a function **conv** to compute the continuous convolution of the signals $(x(t), h(t))$ for $0 \leq t \leq 8$.

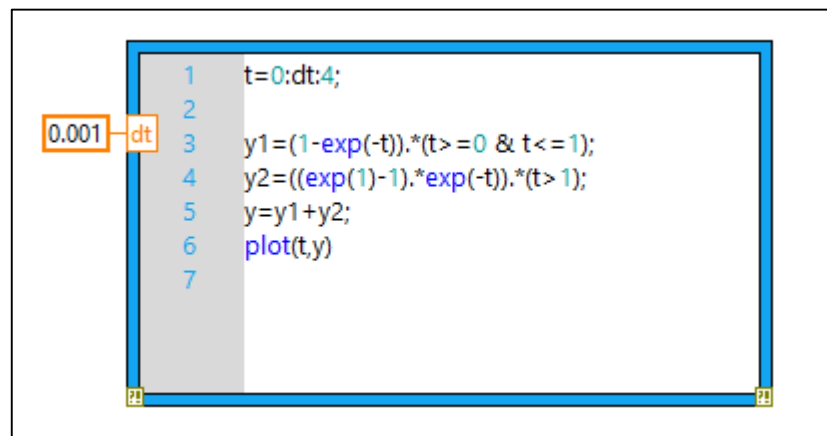
Consider the following values of the approximation pulse width or delta:

$$\Delta = 0.5; 0.1; 0.05; 0.01; 0.005; 0.001.$$

Mathematically, the convolution of $h(t)$ and $x(t)$ is given by: (from discussion notes)

$$y(t) = \begin{cases} 1 - e^{-t} & , \quad 0 \leq t \leq 1 \\ [e^1 - 1]e^{-t} & , \quad t > 1 \end{cases}$$

To define this signal with this expression you need to use control flow function and useful function as shown in M-File in figure 6.14.



```

1 t=0:dt:4;
2
3 y1=(1-exp(-t)).*(t>=0 & t<=1);
4 y2=((exp(1)-1).*exp(-t)).*(t>1);
5 y=y1+y2;
6 plot(t,y)
7

```

Figure 6.14: The textual M-File code to plot actual output signal using LabView

Compare the approximation $y_{\Delta}(t)$ obtained via the function **conv** with the theoretical value $y(t)$ given by Equation (1).

To better see the difference between the approximated $y_{\Delta}(t)$ and the true $y(t)$ values, display both signals in the same graph.

Compute the mean squared error (MSE) between the true and approximated values using equation (5).

The inputs to this program consist of an approximation pulse width Δ , and a desired time duration T.

An important consideration is the selection of the output data type, set the outputs to consist of MSE, actual or true convolution output $y(t)$ and approximated convolution output $y_{\Delta}(t)$.

The first output is a scalar quantity while the other two are one-dimensional vectors. The output data types should be specified by right-clicking on the outputs and selecting the Choose Data Type option as shown in figure 6.15

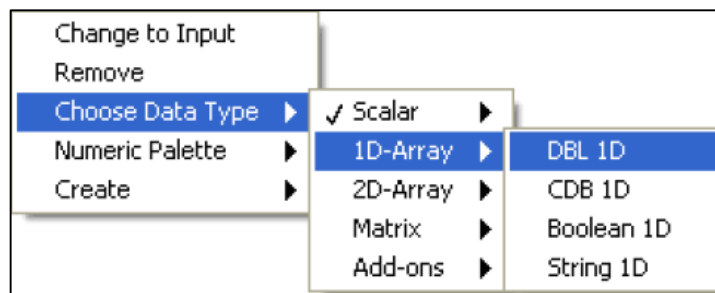


Figure 6.15: Selecting the data type for output signal

Use a waveform graph to show the waveforms, with the function **Build Waveform** (Functions → Programming → Waveforms → Build Waveforms), connect the time interval Delta to the input dt of this function to display the waveforms along the time axis (in seconds).

Merge together and display the true and approximated outputs in the same graph using the function **Merge Signal** (Functions → Express → Signal Manipulation → Merge Signals) as shown in figure 6.16, 17.

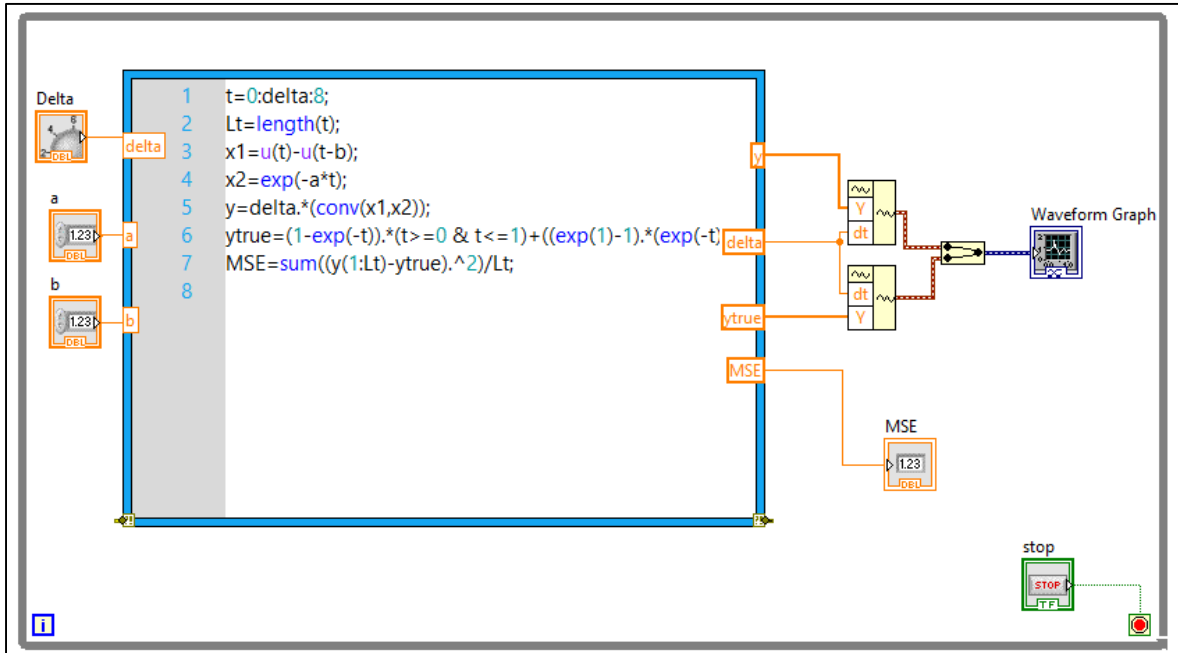


Figure 6.16: The block diagram for the previous example

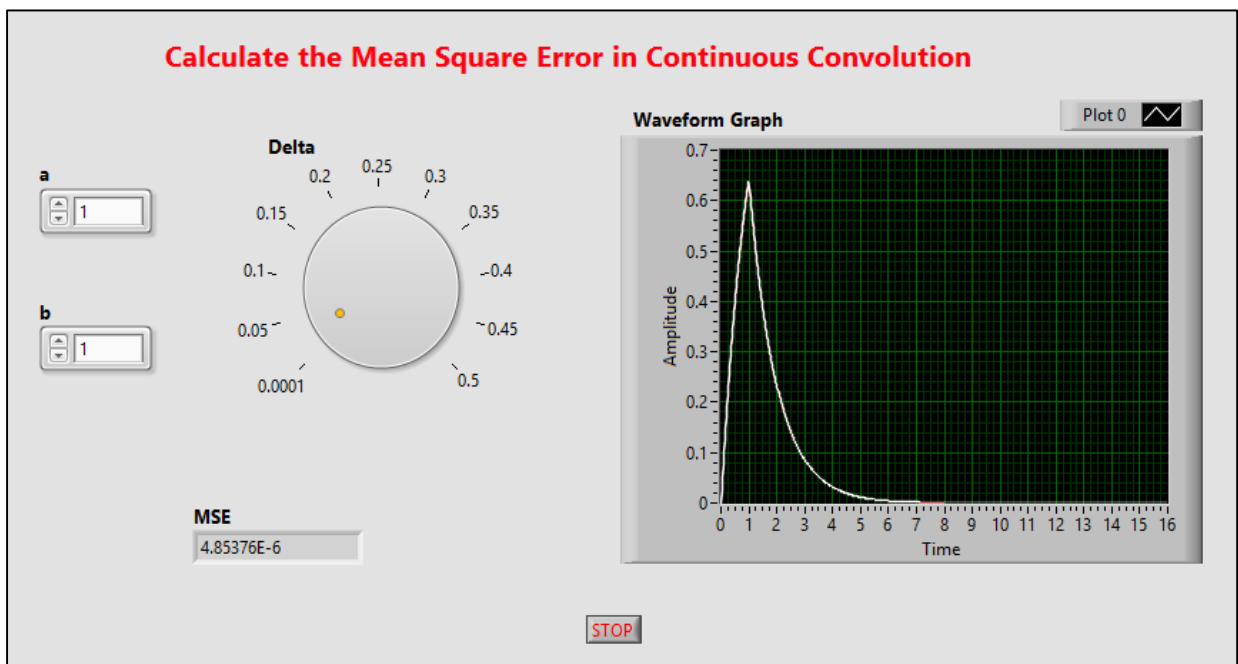


Figure 6.17: The front panel for the previous example

○ Example two:

Compute the convolution for the following signals using LabView MathScript node:

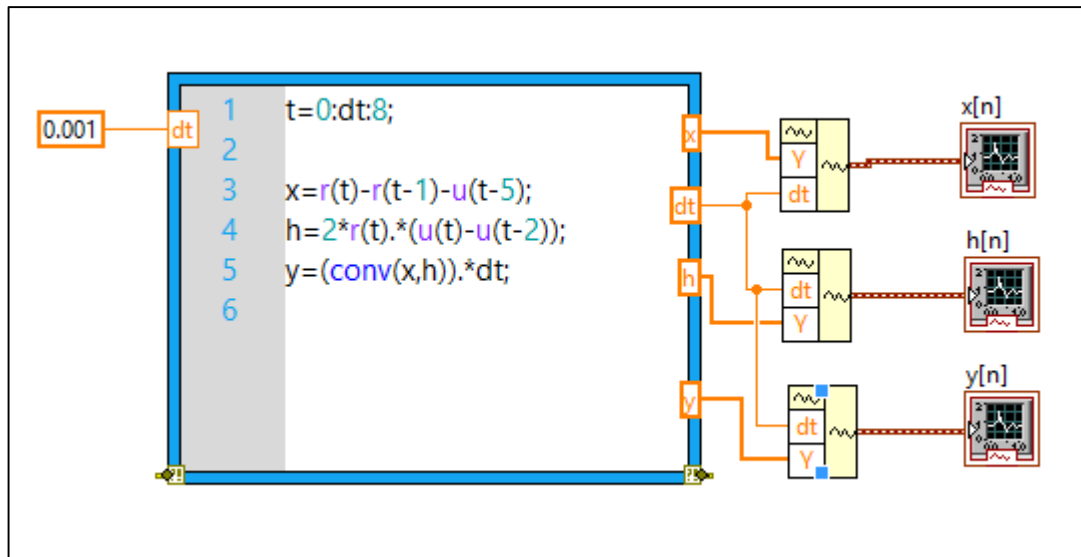
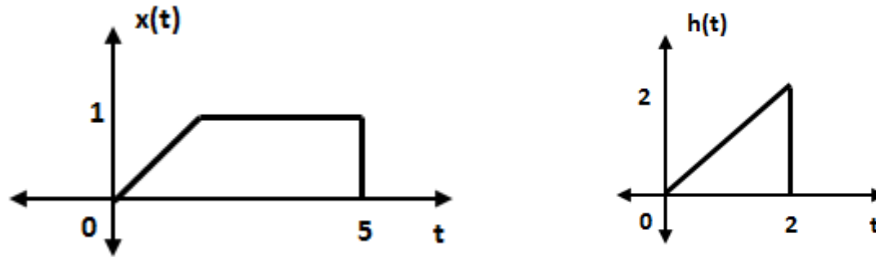


Figure 6.18: The Block Diagram for the Convolution of Two Signals

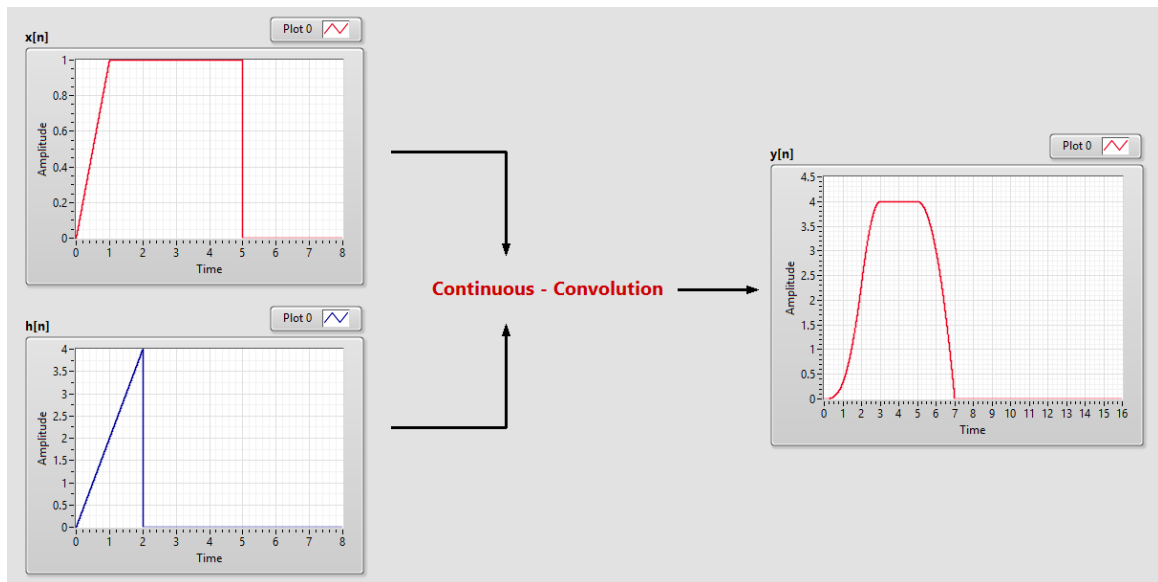


Figure 6.19: The Front panel for the Continuous Convolution of Two Signals

- **Convolution Properties:**

In this part, we will be check the properties of convolution as shown in a Figure 6.20 shows the block diagram to check the properties.

Both sides of equations are plotted in this front panel to verify the convolution properties, to display different convolution properties within a limited screen area, use a Tab Control (Controls → Modern→ Containers → Tab Control) in the front panel as shown in figure 6.21, 22, 23, 24.

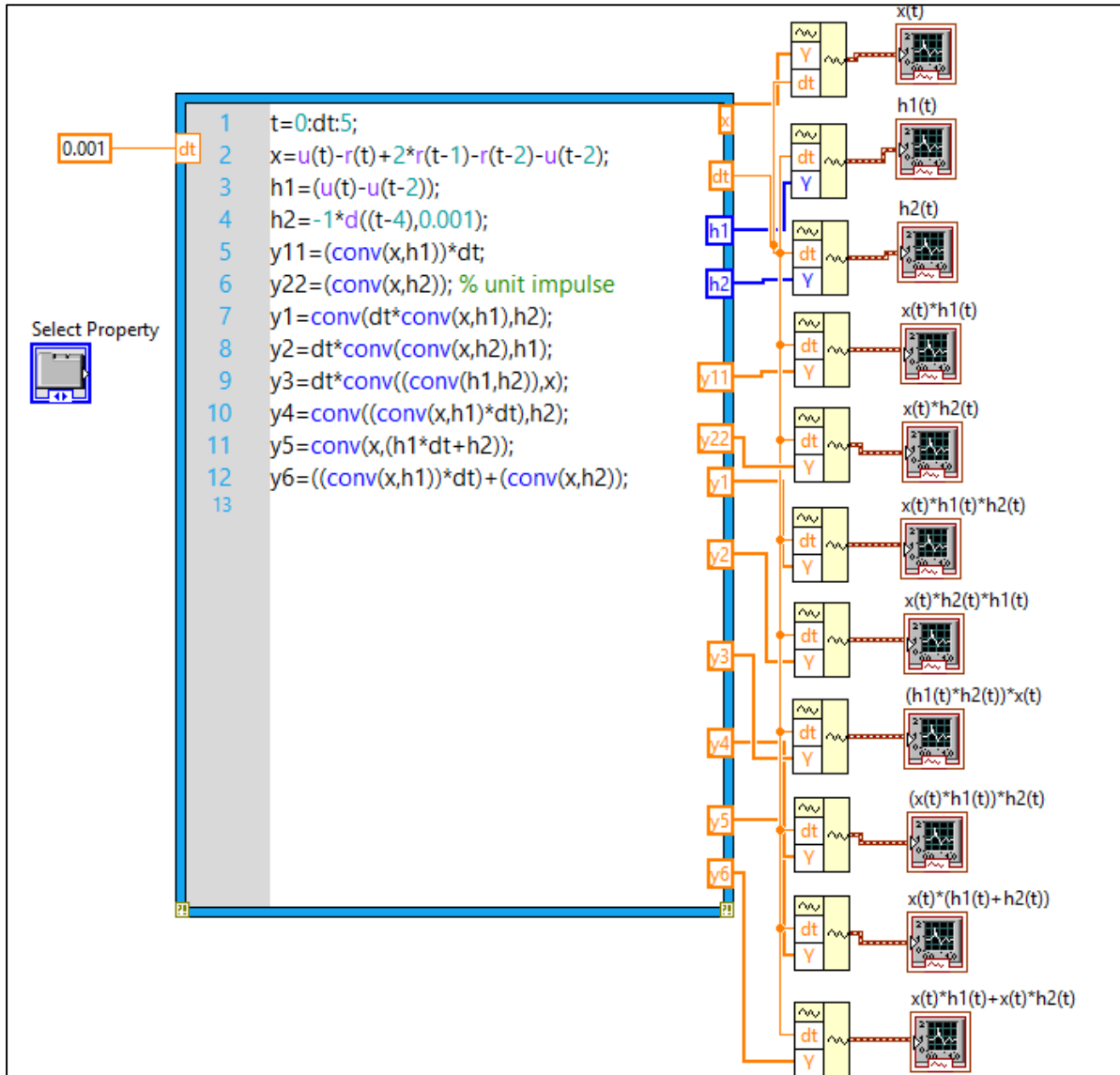


Figure 6.20: The block diagram to check the convolution properties

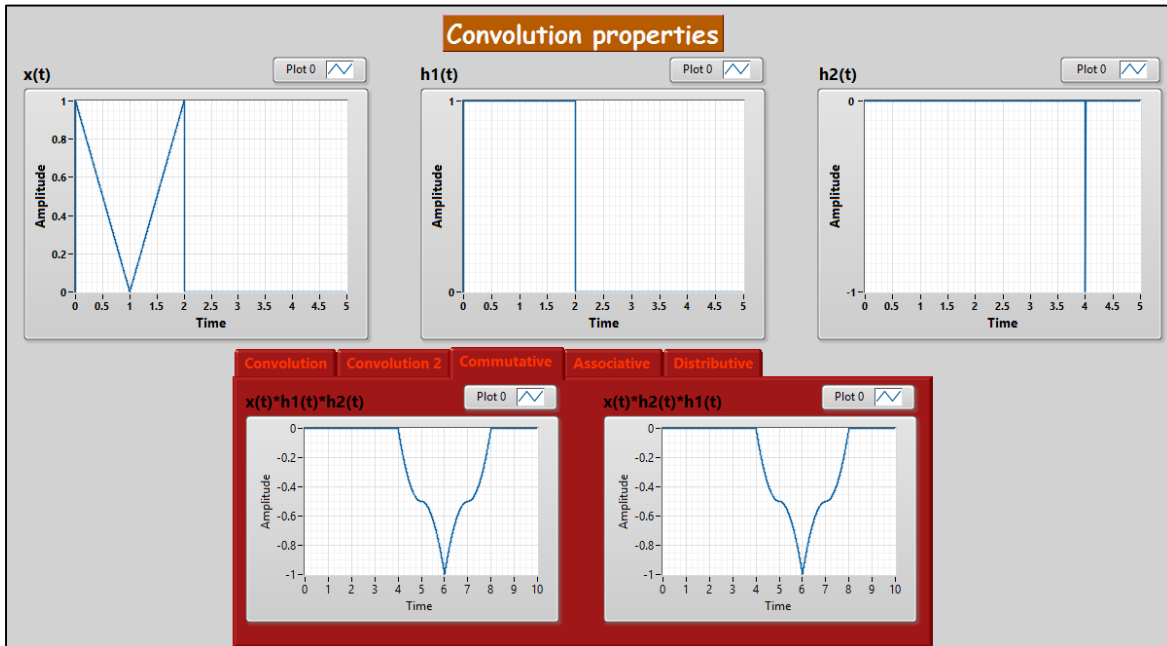


Figure 6.21: The front panel of convolution properties (Commutative property)

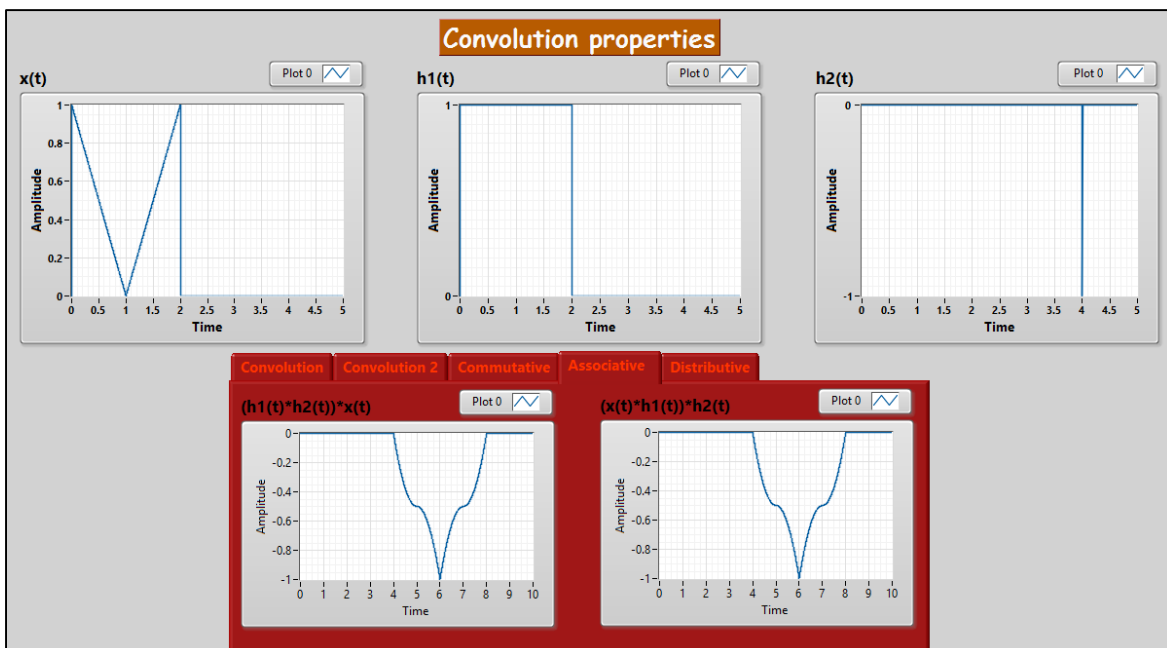


Figure 6.22: The front panel of convolution properties (Associative property)

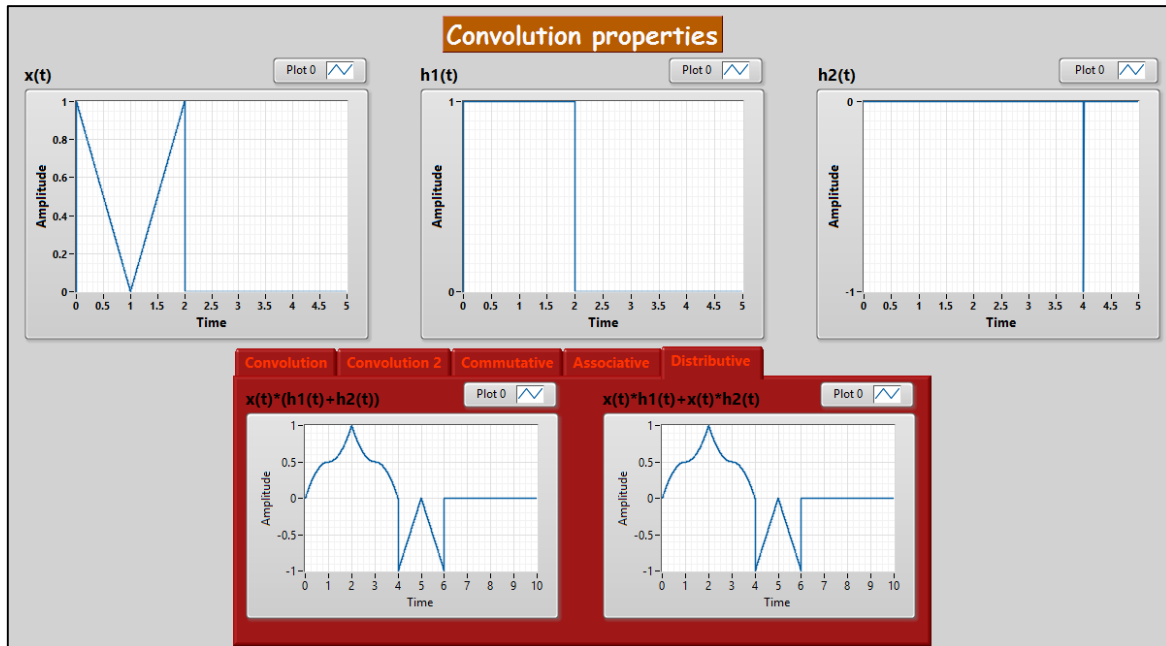


Figure 6.23: The front panel of convolution properties (Distributive property)

• Exercises:

- 1) By using LabView program, find the output signal $y(t)$ and display it with respect to time on waveform graph by apply the continuous convolution between $x(t)$ & $h(t)$ which is shown below:



Hint: using convolution properties and display all signal in front panel

- 2) By using MatLab program, find the output signal $y(t)$ and display it with respect to time t by apply the continuous convolution between $x(t)$ & $h(t)$.

$$X(t) = 2\sin(t) \quad , \quad h(t) = \text{rect}\left(\frac{t-2}{2}\right)$$

Hint: take one period for sinusoidal function 2π and sampling interval $\Delta = 0.01\text{sec}$

- 3) By using LabView program, apply the discrete convolution to find the output signal $y[n]$, when the $x[n]$ and $h[n]$ are given below:

$$x[n] = \{1, 4, 8, 2\} \quad \& \quad h[n] = \{0, 1, 2, 3, 4\}$$

Hint: use modified convolution function

- 4) By using MatLab program, apply the discrete convolution to find the output signal $y[n]$, when the $x[n]$ and $h[n]$ are shown below:

