CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING
CHITTAGONG-4349, BANGLADESH.

Course Title –Signals and System Sessional
Course No: ETE 204

Experiment Name: Convolution of two signals using MATLAB

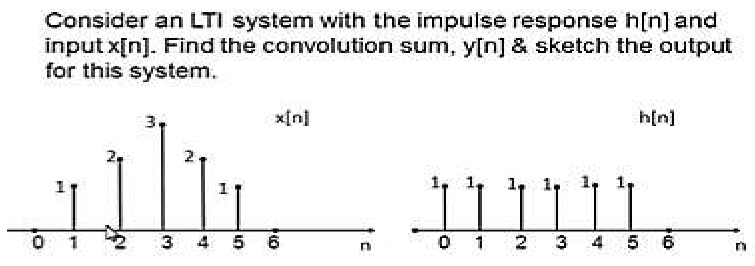
OBJECTIVE

- To review the concept of convolution.
- To learn how to convolve signals in MATLAB

THEORY

Matlab has a function called conv(x,h) that you can use to convolve two signals x(n) and h(n). It assumes that the time increment is the same for both signals. The input signals are finite-length, so the result of the convolution should have a length equal to the sum of the lengths of the inputs– which turns out to be length(x) + length(h) - 1. As you have learned in class, a linear time-invariant (LTI) system is completely described by its impulse response. When working in matlab, this impulse response must also be discrete. Work through the following example.

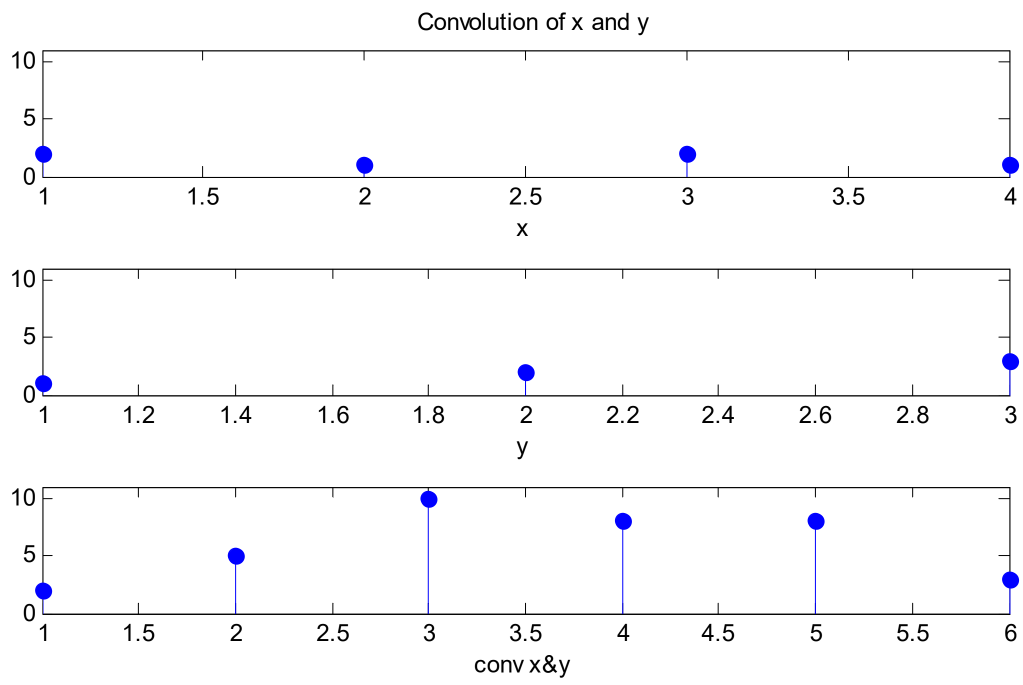
MATLAB Code:


Consider an LTI system with the impulse response h[n] and input x[n]. Find the convolution sum, y[n] & sketch the output for this system.

```
clc;
clear all;
close all
x = [2 1 2 1];
y = [1 2 3];
```

```
clin = conv(x,y);
subplot(3,1,1)
stem(x,'filled')
ylim([0 11])
subplot(3,1,2)
stem(y,'filled')
ylim([0 11])
subplot(3,1,3)
stem(clin,'filled')
ylim([0 11])
title('Convolution of x and y')
```

After writing this code save it and run it you will get output as shown below

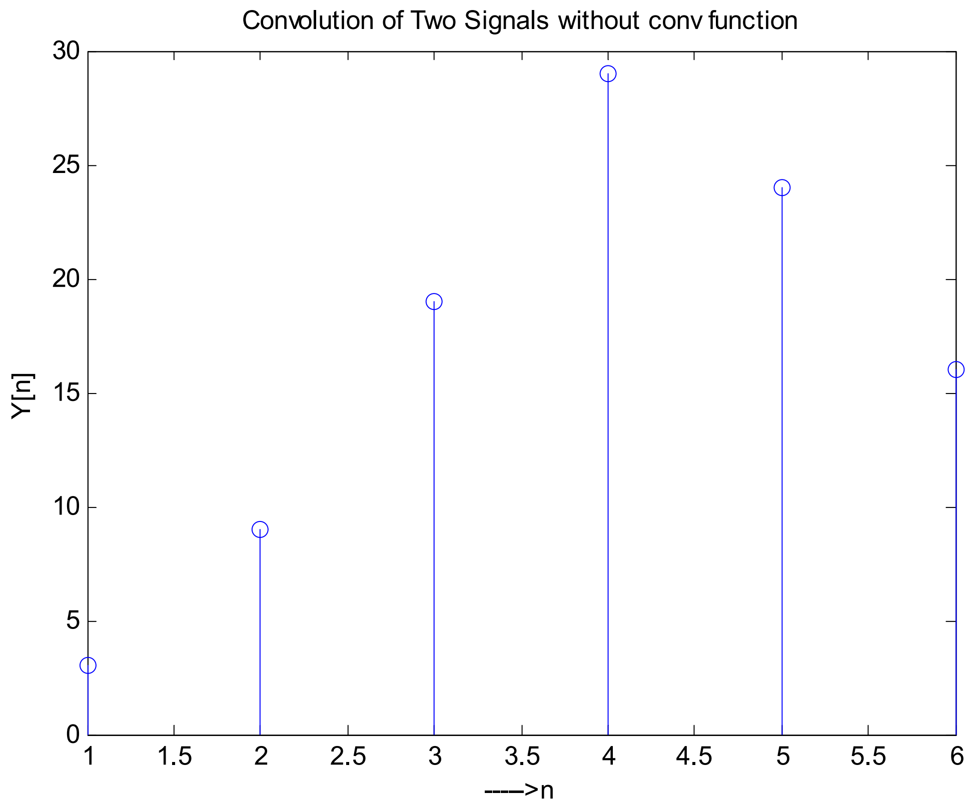


%Without CONV

```
clc;
close all;
clear all;
x=input('Enter x: ');
h=input('Enter h: ');
m=length(x);
n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)];
for i=1:n+m-1
    Y(i)=0;
for j=1:m
if(i-j+1>0)
        Y(i)=Y(i)+X(j)*H(i-j+1);
else
end
end
```

```
end
Y
stem(Y);
ylabel('Y[n]');
xlabel('----->n');
title('Convolution of Two Signals without conv function');
```

Enter x: [1 2 3 4]
Enter h: [3 3 4]



CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING
CHITTAGONG-4349, BANGLADESH.

COURSE NO.: ETE 204

COURSE TITLE: SIGNALS AND SYSTEMS SESSIONAL

Name of Experiment: Studying Time Frequency Characteristics using Continuous Time Fourier Series

OBJECTIVE:

1. Generate a square wave in time domain with certain time period and pulse width.
2. Apply CTFS equation to compute spectral coefficients.
3. Observe the plot to verify the properties mentioned above.
4. Modify time-periods and pulse widths to assess the corresponding change in frequency domain.

THEORY:

In this session, we will explore the relation between time domain properties of the signal (Time period and Pulse Width) and frequency domain properties (Spectral density and Lobe width) – using CTFS equations.

The Analysis equation for CTFS is given as: $C_k = 1/T_p \int_{-T_p/2}^{T_p/2} x(t) e^{-j2\pi F_o t} dt$; where F_o is the Fundamental Frequency and is the inverse of the Time period of the signal.

- The spectral spacing of the spectrum is equal to the inverse of the time period of the signal.
- The Lobe width of the spectrum is equal to the inverse of the pulse width of the signal.

PROCEDURE:

1. Make a folder at desktop and name it as your current directory within MATLAB.

2. Open M-file editor and write the following code:

```
clear all;
clc;
close all;
Ts = 0.001; %sampling period - so small to approximatecont time
% Time Vector - again approx. continuous
t = [-10:Ts:9.999 ];
Tp = 1; % Time Period of the signal
tau = 0.5;  % Pulse Width - Duty Cycle
x = (1+square(t*2*pi/Tp,tau*100))/2;    % Generating Square wave
figure,
plot(t,x);
xlabel('secs');
ylim([-1.2 1.2]);   % Press any key
pause;
Fo = 1/Tp;  % Fundamental Frequency
% Extracting a portion of the signal equal to its period.
I_period = find(round(t*1000)/1000 == -Tp/2) :find(round(t*1000)/1000 ==
Tp/2);
xport = x( I_period );
figure,
plot(t(I_period),xport);
xlabel('secs') ylim([-1.2 1.2]) pause
% Computing CTFS Coefficients
for k = 1:20
B   = exp(-j*2*pi*(k-1)*Fo.*[-Tp/2:Ts:Tp/2]);
C(k)  = sum(xport.*B)/(length(xport)*Tp);
end
kFo = Fo*[0:k-1]; % Frequency Scale in Hz.
figure,
stem(kFo,abs(C));
title('CTFT Coefficients');
xlabel('Hz');
```

RESULT: Explain (write) in your own words the cause and effects of what you just saw.

EXERCISE:

Now make the following modifications one by one and observe the effects.

a) Change 'Tp' to 2 secs and then to 0.5 secs.
b) Change 'tau' to 0.2.
c) Change 'tau' to 0.001 and see the spectra.
d) Change 'tau' to 1 and observe the spectra.

Make notes about all observations and learn to play around with the code.

Note: Do not change the sampling time, as this is an artificially made continuous-time signal. Also, do not increase Tp than 10 secs and reduce below 0.05 secs.

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING
CHITTAGONG-4349, BANGLADESH.

Course Title –Signals and System Sessional

Course No: ETE 212

Experiment Name: MATLAB implementation of Continuous Time Fourier Transform (CTFT)

OBJECTIVE

- To understand and implement the CTFT using MATLAB.

THEORY

As stated that, the Fourier transform expresses a signal (or function) $x(t)$ in the (cyclic) frequency domain; that is, the signal is described by a function $X(\omega)$. The Fourier transform is denoted by the symbol $F\{.\}$; that is one can write

$$X(\omega) = F\{x(t)\}$$

In other words, the Fourier transform of a signal $x(t)$ is a signal $X(\omega)$. An alternative way of writing is

$$x(t) \xrightarrow{F} X(\omega)$$

The mathematical expression of Fourier transform is

$$X(\omega) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t).e^{-j\omega t}dt$$

It is clear that $X(\omega)$ is a complex function of ω. In case of the Fourier transform of $x(t)$ has to be express in the frequency domain f, then substituting ωby $2\pi f$ yields

$$X(f) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t).e^{-j2\pi ft}\,dt$$

In order to return from frequency domain to back time domain the *inverse Fourier transform* is applied. The inverse Fourier transform is denoted by the symbol$F^{-1}\{.\}$; that is, one can write

$$x(t) = F^{-1}\{X(\omega)\}$$

or alternatively

$$X(\omega) \xrightarrow{F^{-1}} x(t)$$

The mathematical expression of the inverse Fourier transform is

$$x(t) = F^{-1}\{X(\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega).\, e^{j\omega t} d\omega$$

Using f instead of ω,

$$x(t) = F^{-1}\{X(f)\} = \int_{-\infty}^{\infty} X(f)\,.\, e^{j2\pi f t} df$$

The cyclic frequency is measured in rad/s, while f is measured in Hertz. The Fourier transform of a signal is called (frequency) *spectrum.*

The Command fourier and ifourier

The computation of the integral is not always a trivial matter. However, in MATLAB® there is a possibility to compute is directly the Fourier transform $X(\omega)$ of a signal $x(t)$ by using command `fourier.` Correspondingly, the inverse Fourier transform is computed by using the command `ifourier.` Before executing these two commands, time t and frequency ω must be declared as symbolic variables. Symbolic variable is declared by using the command `syms`.

MATLAB Code:

Compute the Fourier transform of this expression with respect to the variable `x` at the evaluation point `y`:

```
syms x y
f = exp(-x^2);
fourier(f, x, y)

ans =
pi^(1/2)*exp(-y^2/4)
```

Compute the Fourier transform of this expression calling the `fourier` function with one argument. If you do not specify the transformation variable, it is determined by `symvar`. For this expression, `symvar` chooses `x` as the transformation variable.

```
syms x t y
f = exp(-x^2)*exp(-t^2);
fourier(f, y)

ans =
pi^(1/2)*exp(-t^2)*exp(-y^2/4)
```

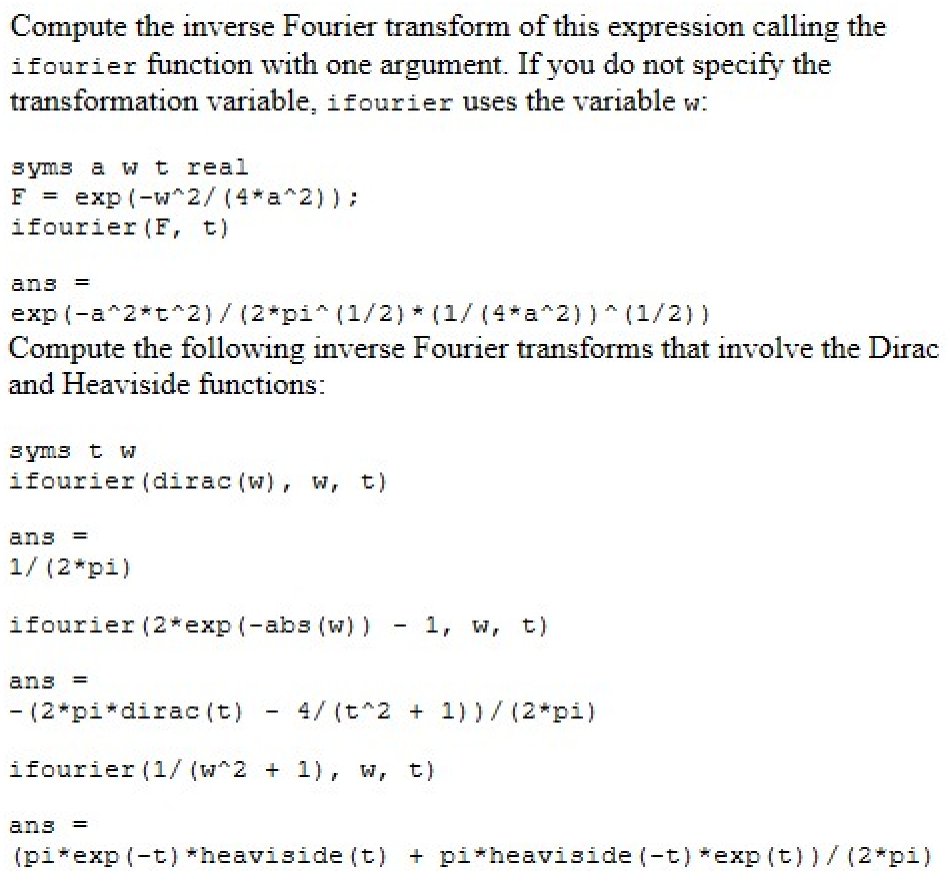
Compute the following Fourier transforms that involve the Dirac, Heaviside, and piecewise functions:

```
syms t w
fourier(t^3, t, w)

ans =
-pi*dirac(w, 3)*2*i

syms t0
fourier(heaviside(t - t0), t, w)

ans =
exp(-t0*w*i)*(pi*dirac(w) - i/w)

assume(x,'real')
f = exp(-x^2*abs(t))*sin(t)/t;
fourier(f, t, w)

ans =
piecewise([x ~= 0, atan((w + 1)/x^2) - atan((w - 1)/x^2)])
```

Inverse Fourier Transform

Compute the inverse Fourier transform of this expression with respect to the variable `y` at the evaluation point `x`:

```
syms x y
F = sqrt(sym(pi))*exp(-y^2/4);
ifourier(F, y, x)

ans =
exp(-x^2)
```



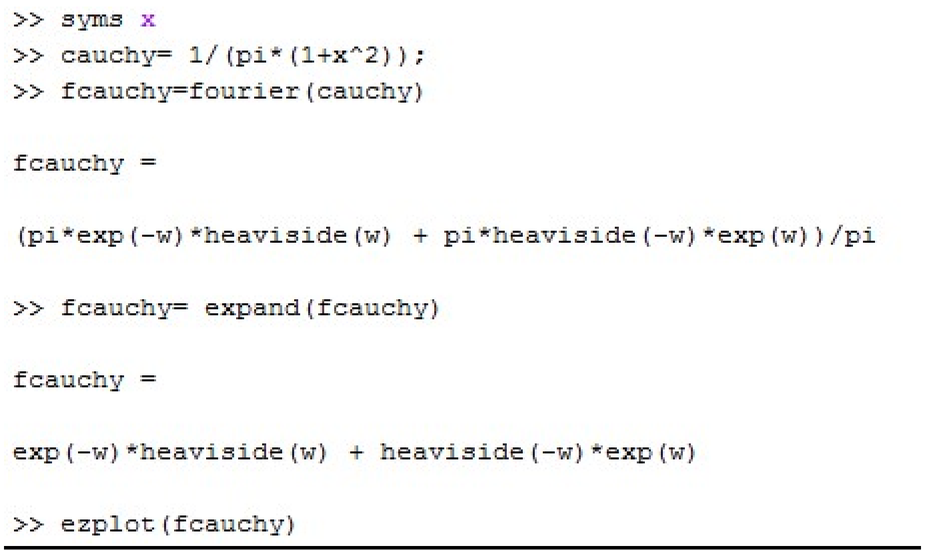
Compute the inverse Fourier transform of this expression calling the `ifourier` function with one argument. If you do not specify the transformation variable, `ifourier` uses the variable `w`:

```
syms a w t real
F = exp(-w^2/(4*a^2));
ifourier(F, t)

ans =
exp(-a^2*t^2)/(2*pi^(1/2)*(1/(4*a^2))^(1/2))
```

Compute the following inverse Fourier transforms that involve the Dirac and Heaviside functions:

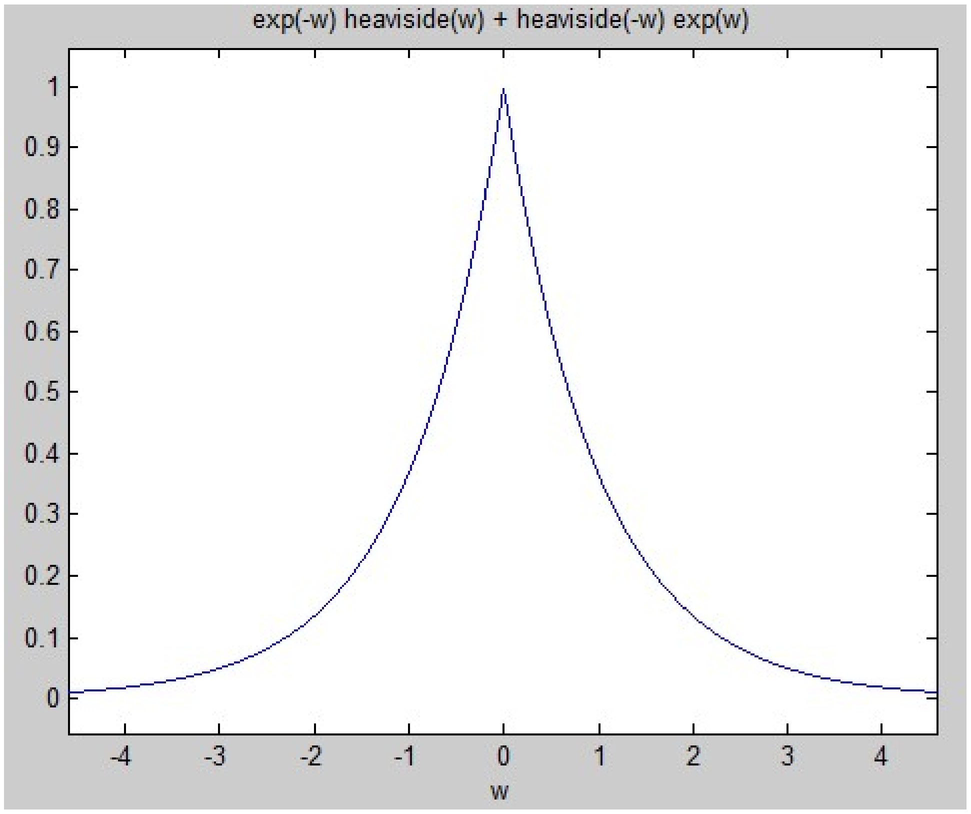
```
syms t w
ifourier(dirac(w), w, t)

ans =
1/(2*pi)

ifourier(2*exp(-abs(w)) - 1, w, t)

ans =
-(2*pi*dirac(t) - 4/(t^2 + 1))/(2*pi)

ifourier(1/(w^2 + 1), w, t)

ans =
(pi*exp(-t)*heaviside(t) + pi*heaviside(-t)*exp(t))/(2*pi)
```

<u>Another example of Fourier Transform:</u>



```
>> syms x
>> cauchy= 1/(pi*(1+x^2));
>> fcauchy=fourier(cauchy)

fcauchy =

(pi*exp(-w)*heaviside(w) + pi*heaviside(-w)*exp(w))/pi

>> fcauchy= expand(fcauchy)

fcauchy =

exp(-w)*heaviside(w) + heaviside(-w)*exp(w)

>> ezplot(fcauchy)
```

After writing this code save it and run it you will get output as shown below



To recover the Cauchy density function from the Fourier transform, call `ifourier`:

```
finvfcauchy = ifourier(fcauchy)
```

```
finvfcauchy =
-(1/(x*i - 1) - 1/(x*i + 1))/(2*pi)
```

```
simplify(finvfcauchy)
```

```
ans =
1/(pi*(x^2 + 1))
```

Lab Task:

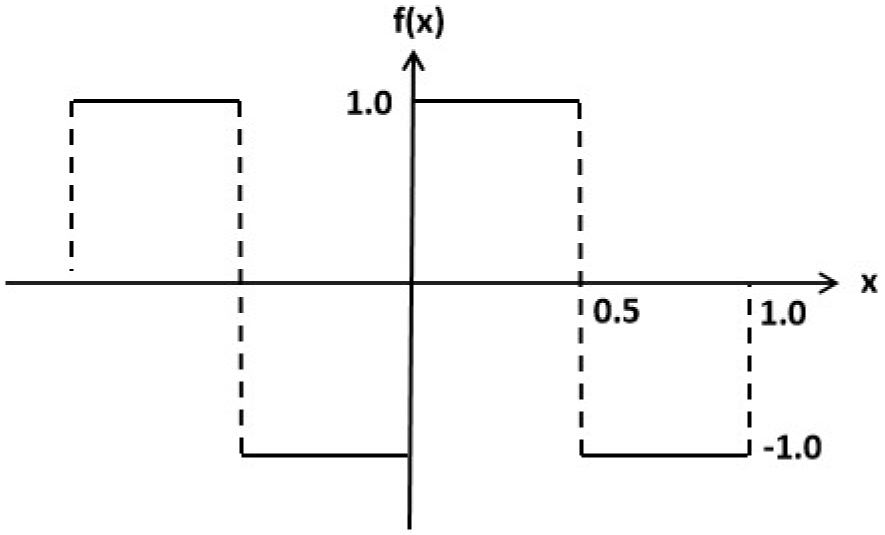
Compute and plot the Fourier transform of the following signals:

a. $x_1(t) = e^{5t}u(-t)$
b. $x_2(t) = te^{-5t}u(t)$

Consider the following square wave function defined by the relation

$$f(x) = \begin{cases} 1\,, & 0 < x < 0.5 \\ -1\,, & 0.5 < x < 1 \end{cases}$$

This function is shown below.



CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING
CHITTAGONG-4349, BANGLADESH.

COURSE NO.: ETE 204

COURSE TITLE: SIGNALS AND SYSTEMS SESSIONAL

Name of Experiment: Studying Laplace Transformation using MATLAB

OBJECTIVE

- To implement Laplace transformation using MATLAB.

THEORY

A continuous-time signal described in the time domain t is a signal given by a function $f(t)$. As previously mentioned, the Laplace transform expresses a signal in the complex frequency domain s(or s domain); that is, a signal is described by a function $F(s)$. Laplace transform is denoted by the symbol $L\{.\}$; that is, one can write

$$F(s) = L\{f(t)\}.$$

In other words, the Laplace transform of a function $f(t)$ is a function $F(s)$. An alternative way of writing is

$$f(t) \xrightarrow{L} F(s).$$

There are two available forms of Laplace transform. The first is the two-sided (or bilateral) Laplace transform where the Laplace transform F(s) of a function $f(t)$ is given by:

$$F(s) = L\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-st}\,dt.$$

Variable s is a complex-valued number, and thus can be written as $s = \sigma + j\omega$. This relationship reveals the association between the Laplace transform and the Fourier transform. More specially, substituting s by $j\omega$ yields the Fourier transform $X(\omega)$ of $x(t)$. Setting the lower limit of the integral to zero yields the one-sided (or unilateral) Laplace transform, which is described by the relationship

$$F(s) = L\{f(t)\} = \int_{0}^{\infty} f(t)e^{-st}\,dt.$$

Variable s is a complex-valued number, and thus can be written as $s = \sigma + j\omega$. This relationship reveals the association between the Laplace transform and the Fourier transform. More specially, substituting s by $j\omega$ yields the Fourier transform $X(\omega)$ of $x(t)$. Setting the lower limit of the integral to zero yields the one-sided (or unilateral) Laplace transform, which is described by the relationship

$$F(s) = L\{f(t)\} = \int_0^\infty f(t)e^{-st}\,dt.$$

The use of the one-sided Laplace transform is better suited for the purpose of this lab, as the signals considered are usually casual signals. In order to return from the s-domain back to the time domain, the inverse Laplace transform is applied. The inverse Laplace transform is denoted by the symbol $L^{-1}\{.\}$; that is, one can write

$$f(t) = L^{-1}\{F(s)\},$$

Or alternatively

$$F(s) \xrightarrow{L^{-1}} f(t).$$

Command Laplace and ilaplace

The computation of the integrals can sometimes be quite hard. This is the reason that already computed Laplace transform pairs are used to compute the Laplace transform and the inverse Laplace transform of a function (or signal). However, in MATLAB®, the Laplace transform $F(s)$ of a function $f(t)$ is easily computed by executing the command `laplace. Moreover,` the inverse Laplace transform of a function $F(s)$ is obtained by executing the command `ilaplace`. Before using these two commands, the declaration of complex frequency s and of time t as symbolic variables is necessary. Recall that in order to define a symbolic variable the command `syms` is used. Finally, note that the command `laplace` computes the unilateral Laplace transform of a function.

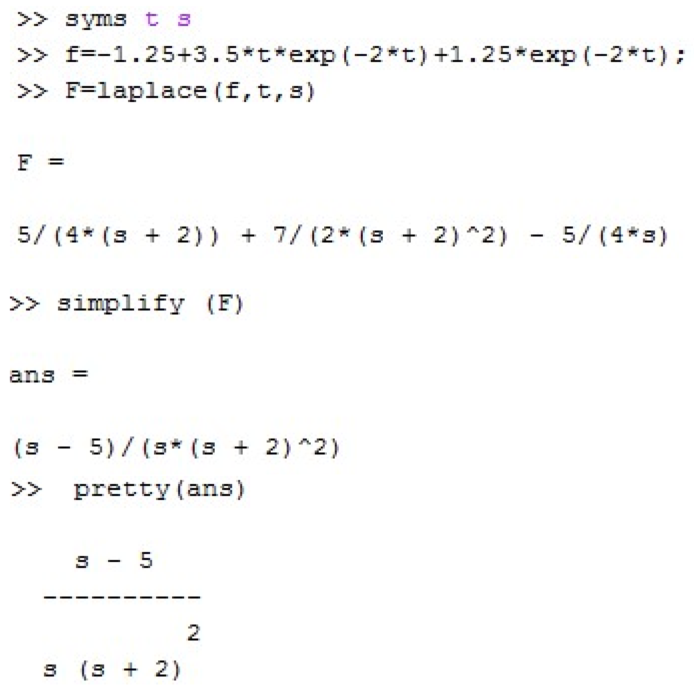
Functions $f(t) = e^{-t}$and $F(s) = 1/(1+s)$ are a Laplace transform pair. In other words, the (unilateral) Laplace transform of $f(t) = e^{-t}$ is $F(s) = 1/(1+s)$, while the inverse Laplace transform of $F(s) = 1/(1+s)$ is $f(t) = e^{-t}$. A Laplace transform pair is denoted by $f(t) \rightarrow F(s)$. In our case,
[illegible]

pairs are presented. At the moment, let us introduce alternative syntaxes of the `laplace` and `ilaplace` commands. The most effective and in the same time simple syntax of the command `laplace` is `ilaplace(f,s)`. In this way, the Laplace transform of the signal $f(t)$ is optimal and makes possible the computation of the Laplace transform in every case. Such a case is when the Laplace transform of a constant function has to be computed.

Finally an alternative available syntax for the Laplace transform computation is `laplace(f,t,s)`;that is, the function f is transferred from t to s, while the inverse Laplace transform is computed using the command `ilaplace(F,s,t)`; that is, the function `F` is transferred from s to t.

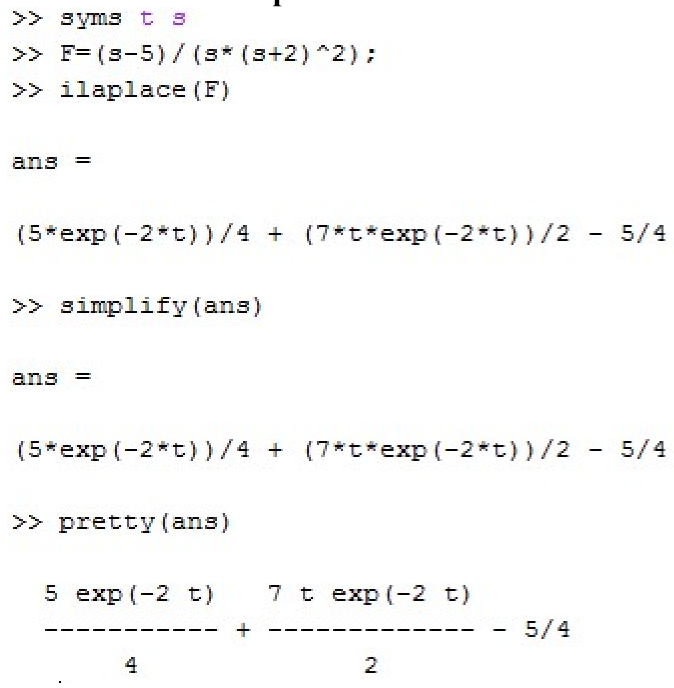
MATLAB Code:

Calculate the Laplace transform for the function f(t) where $f(t) = -1.25 + 3.5te^{-2t} + 1.25e^{-2t}$



```
>> syms t s
>> f=-1.25+3.5*t*exp(-2*t)+1.25*exp(-2*t);
>> F=laplace(f,t,s)

F =

5/(4*(s + 2)) + 7/(2*(s + 2)^2) - 5/(4*s)

>> simplify (F)

ans =

(s - 5)/(s*(s + 2)^2)
>>  pretty(ans)

    s - 5
  ----------
           2
  s (s + 2)
```

Find the inverse Laplace for the above function



```
>> syms t s
>> F=(s-5)/(s*(s+2)^2);
>> ilaplace(F)

ans =

(5*exp(-2*t))/4 + (7*t*exp(-2*t))/2 - 5/4

>> simplify(ans)

ans =

(5*exp(-2*t))/4 + (7*t*exp(-2*t))/2 - 5/4

>> pretty(ans)

  5 exp(-2 t)   7 t exp(-2 t)
  ----------- + ------------- - 5/4
       4              2
```

Lab Task:
Calculate the inverse Laplace transformation for the below function using MATLAB:

$$F(s) = \frac{10(s+2)}{s(s^2+4s+5)}$$

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING
CHITTAGONG-4349, BANGLADESH.

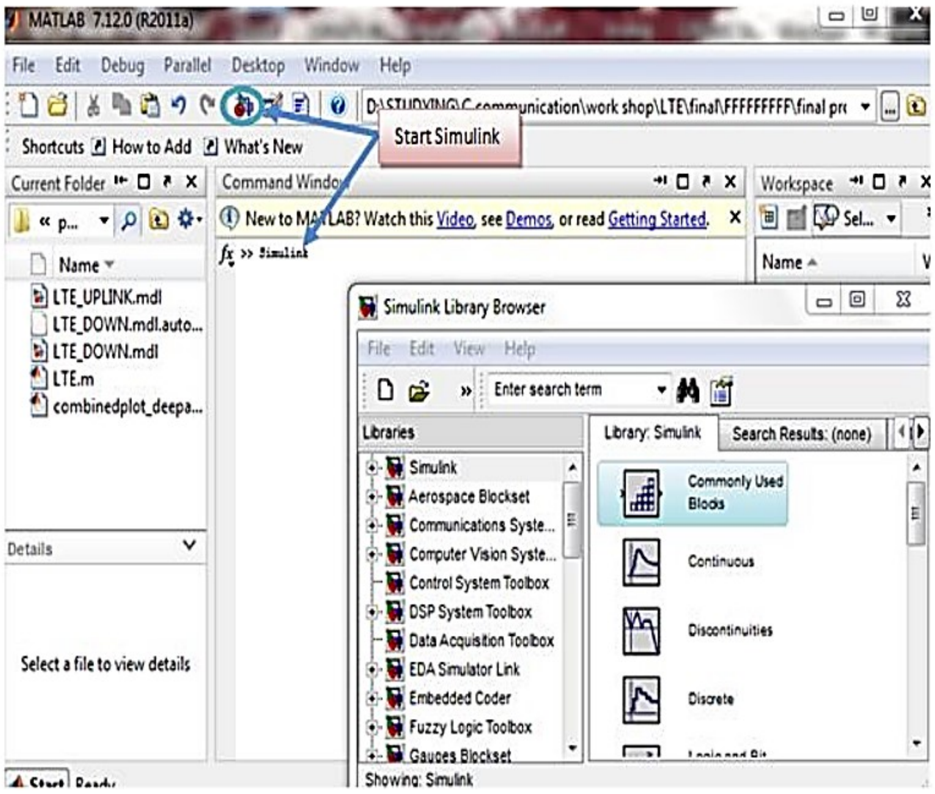
COURSE NO.: ETE 204

COURSE TITLE: SIGNALS AND SYSTEMS SESSIONAL

Name of Experiment: Getting start with Simulink

1.1 Starting Simulink:

1. Start up Matlab.
2. Start up Simulink by clicking the Simulink icon or by typing >>**Simulink** at the Matlab command window.
3. You should see the Simulink Block Library window as shown in figure 1.

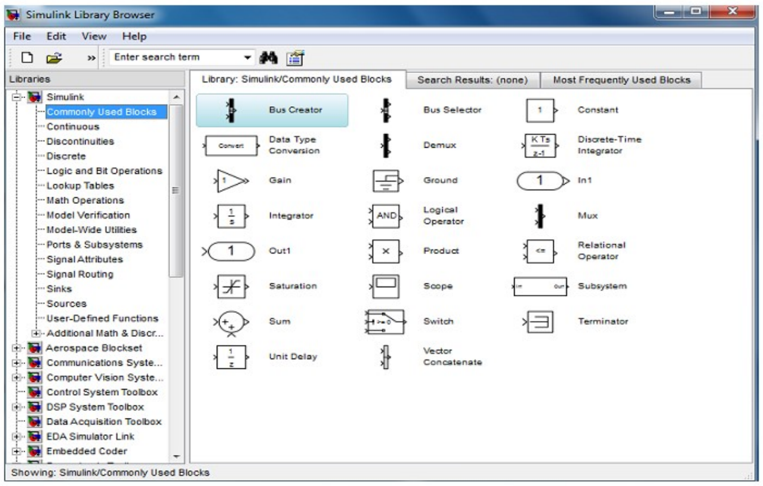


```
>> Simulink
```

1.3.1 Common Block Libraries:

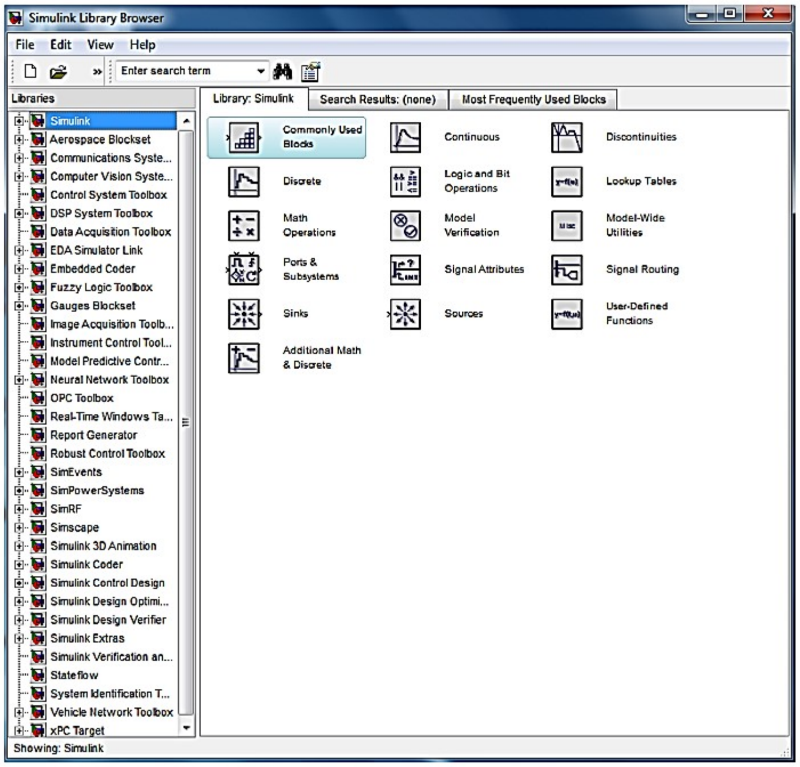
In this section we will see the most common used block libraries in communication system models.

1. **Commonly Used Block**



1.3 The Simulink Library

The Simulink Library Browser is the library where you find all the blocks you may use in Simulink. Simulink software includes an extensive library of functions commonly used in modeling a system. These include:



1.6 Building a Simple Model

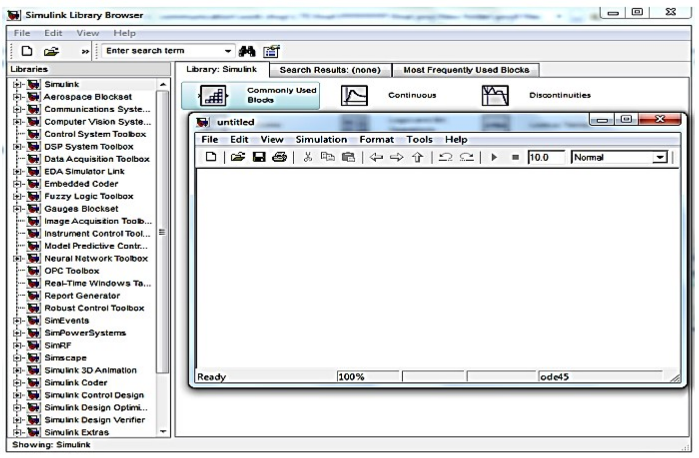
The Basic Steps

This section describes the basic steps in building a model. It explains how to:

- Open a new model window
- Open block libraries
- Move blocks into a model window
- Connect the blocks
- Set block parameters
- Set simulation parameters
- Run the model

1. Opening a New Model Window

The first step in building a model is to open a new model window. To do so, by clicking the new form icon or select New from the **File** menu, and then select **Model**. This opens an empty model window, as shown in the following figure.



2. Opening Block Libraries

The next step is to select the blocks for the model. These blocks are contained in libraries. To view the libraries for the products you have installed, type Simulink at the MATLAB prompt (or, on Microsoft Windows, click the Simulink button on the MATLAB toolbar).

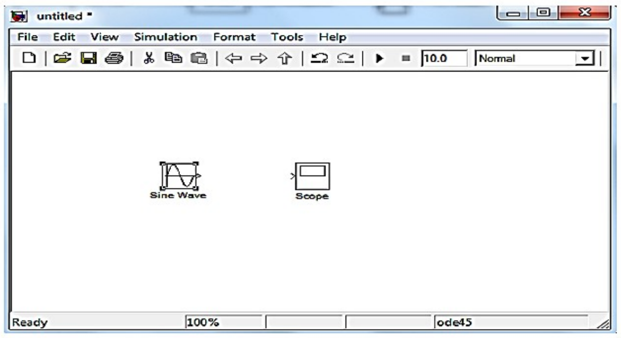
3. Simulink Library Browser

The left pane displays the installed products, each of which has its own library of blocks. To open a library, click the plus sign **(+)** next to the name of the blockset in the left pane. This displays the contents of the library in the right-hand pane. You can find the blocks you need to build models of communication systems in the libraries of the Communications System Toolbox, the DSP Toolbox Blockset, and Simulink.

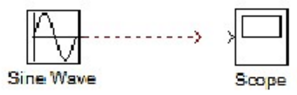
4. Moving Blocks into the Model Window

The next step in building the model is to move blocks from the Simulink Library Browser into the model window. To do so,

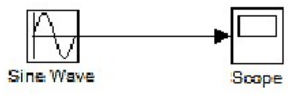
1. Click the + sign next to **Simulink** in the left pane of the Library Browser. This displays a list of the Signal Processing Blockset libraries.
2. Click **Sources** in the left pane. This displays a list of the Sources library blocks in the right pane.
3. **C**lick the Sine Wave block and **Drag** and **Drop** it into the model window.
4. Click **Sinks** in the left pane of the Library Browser. This displays a list of the Sinks library blocks in the right pane.
5. **C**lick the Scope Block and **drag** and **Drop** it into the model window.



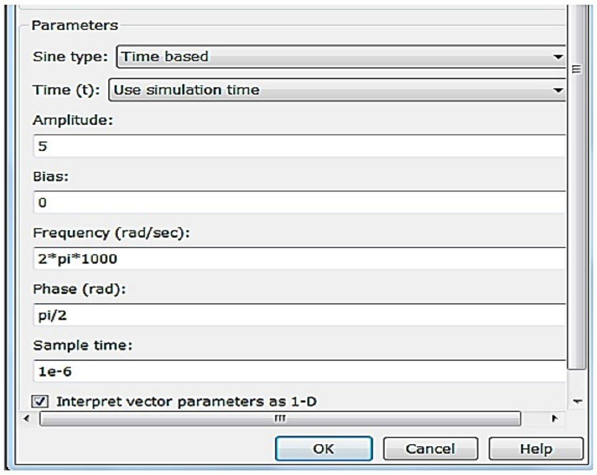
5. Connecting Blocks
The small arrowhead pointing outward from the right side of the Sine Wave block is an output port for the data the block generates. The arrowhead pointing inward on the Scope block is an input port.
To connect the two blocks, click the output port of the SineWave block and drag the mouse toward the input port of the Vector Scope block, as shown in the following figure.



When the pointer is on the input port of the Vector Scope block, release the mouse button. You should see a solid arrow appear, as in the following figure.



6. Setting Block Parameters
To set parameters for the Sine Wave block, double-click the block to open its dialog, as shown in the following figure. Change the following parameters by clicking in the field next to the parameter, deleting the default setting, and entering the new setting in its place:



Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 5

Bias: 0

Frequency (rad/sec): 2*pi*1000

Phase (rad): pi/2

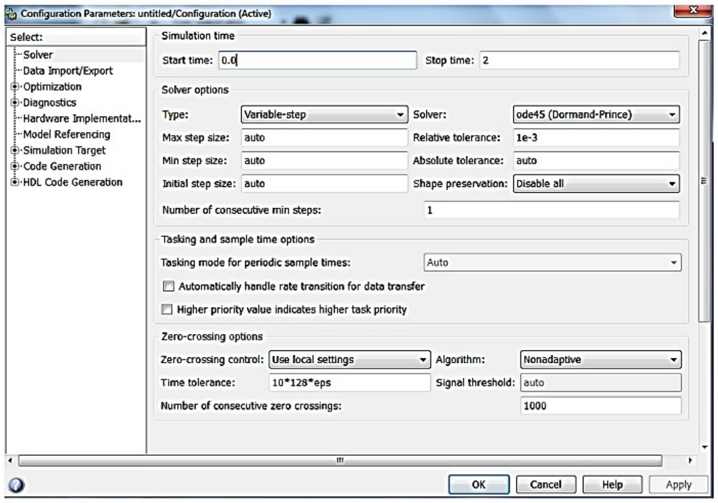
Sample time: 1e-6

☑ Interpret vector parameters as 1-D

7. Setting Simulation Parameters

Besides individual block parameters, the model also has overall simulation parameters. To view the current settings,

1. Select the **Simulation** menu at the top of the model window.
2. Select **Configuration parameters** to open the **Configuration Parameters** dialog box, as shown in the following figure.



Configuration Parameters: untitled/Configuration (Active)

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementat...
- Model Referencing
- Simulation Target
- Code Generation
- HDL Code Generation

Simulation time

Start time: 0.0 Stop time: 2

Solver options

Type: Variable-step Solver: ode45 (Dormand-Prince)

Max step size: auto Relative tolerance: 1e-3

Min step size: auto Absolute tolerance: auto

Initial step size: auto Shape preservation: Disable all

Number of consecutive min steps: 1

Tasking and sample time options

Tasking mode for periodic sample times: Auto

☐ Automatically handle rate transition for data transfer

☐ Higher priority value indicates higher task priority

Zero-crossing options

Zero-crossing control: Use local settings Algorithm: Nonadaptive

Time tolerance: 10*128*eps Signal threshold: auto

Number of consecutive zero crossings: 1000

The **Stop time** determines the time at which the simulation ends. Setting **Stop time** to inf causes the simulation to run indefinitely, until you stop it by selecting **Stop** from the **Simulation** menu.

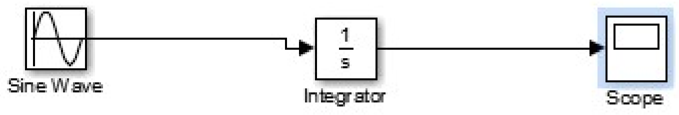
The **Stop time** is not the actual time it takes to run a simulation. The actual run-time for a simulation depends on factors such as the model's complexity and your computer's clock speed. The settings in the **Configuration Parameters** dialog box affect only the parameters of the current model.

8. Running the Model

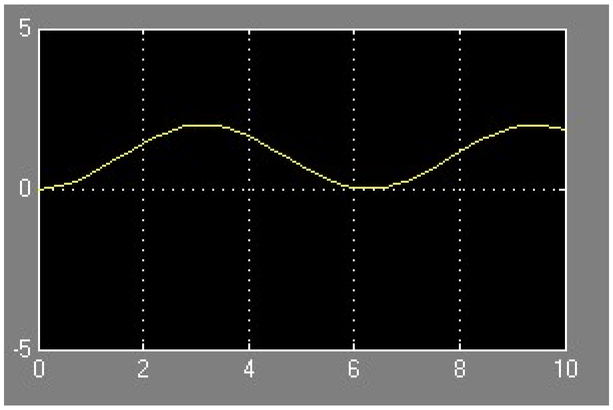
Run the model by clicking start simulation icon ▶ or selecting **Start** from the **Simulation** menu. When you do so, a scope window appears, displaying a sine wave as shown in the following figure.

Example:

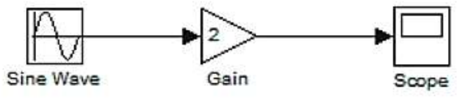
- Integrating a sine wave

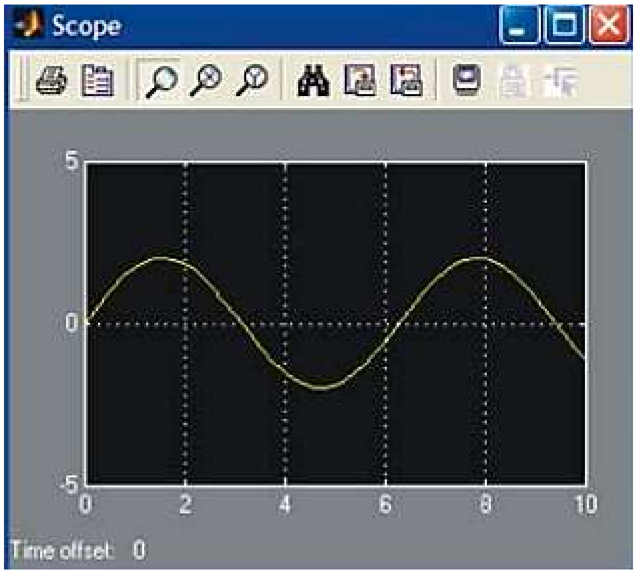


Output:

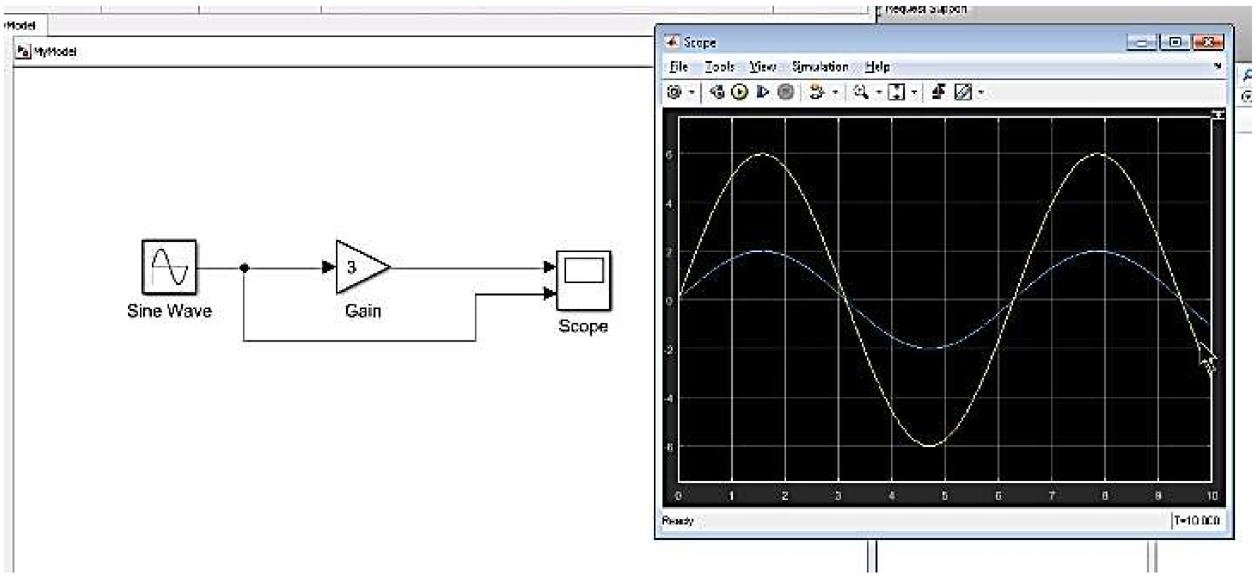


EXAMPLE: Given an input signal u(t)=sin(t).It is fed into anamplifier with gain 2. Simulate the output of the amplifier y(t). Theanswer is y(t) 2sin(t).




Scope
5
0
-5
0
2
4
6
8
10
Time offset: 0

Example:


MyModel
Sine Wave
3
Gain
Scope
File Tools View Simulation Help
Ready
T=10.000