EXPERIMENT-I

Introduction to Matlab, Simulink and solving ODE

Objectives: The objective of this exercise is to familiarize students with Matlab and Simulink, followed by their application in solving Ordinary Differential Equations (ODEs).

List of Equipment/Software: MATLAB/SIMULINK.

Introduction to Matlab

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming, developed by MathWorks.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and Fortran; analyze data; develop algorithms; and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots and performing numerical methods.

Getting started:-

To start MATLAB:

START ⇨ PROGRAMS ⇨ MATLAB 13a

Or shortcut creation/activation on the desktop shown in figure 1 .

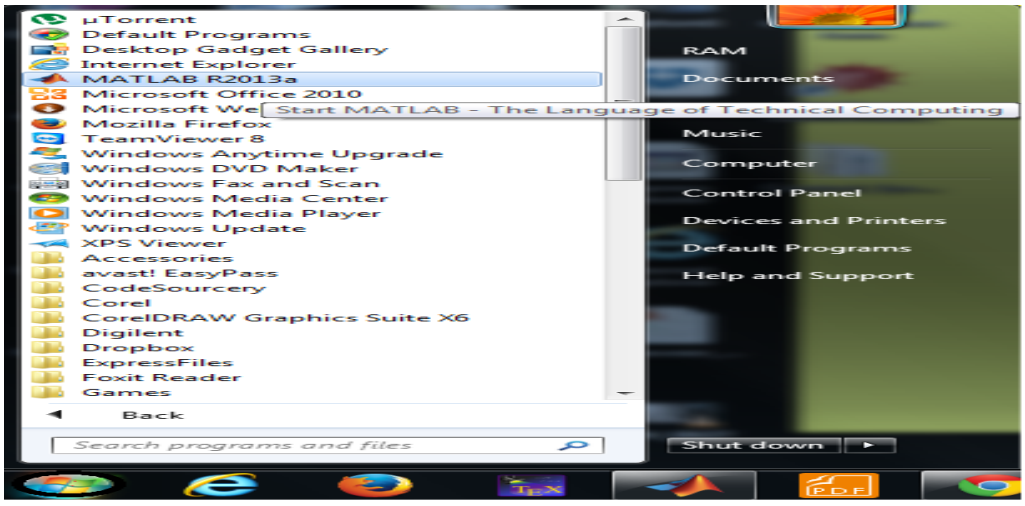

Figure 1: Basic started Matlab

Overview:

The Matlab window should roughly look as in Figure 2. If it does not, click on ”View” in the menu bar and select ”Desktop layout”, ”Default”. Four windows appear in default ‘Desktop layout’ :

- ”Current folder” : displays the content of the current work-directory,

- ”Workspace” : lists the variables assigned by the users,
- ”Command history” : shows the last commands used.
- ”Command window” : where the commands are typed and the outputs displayed.
- “Editor”: To write a program in .mat file.

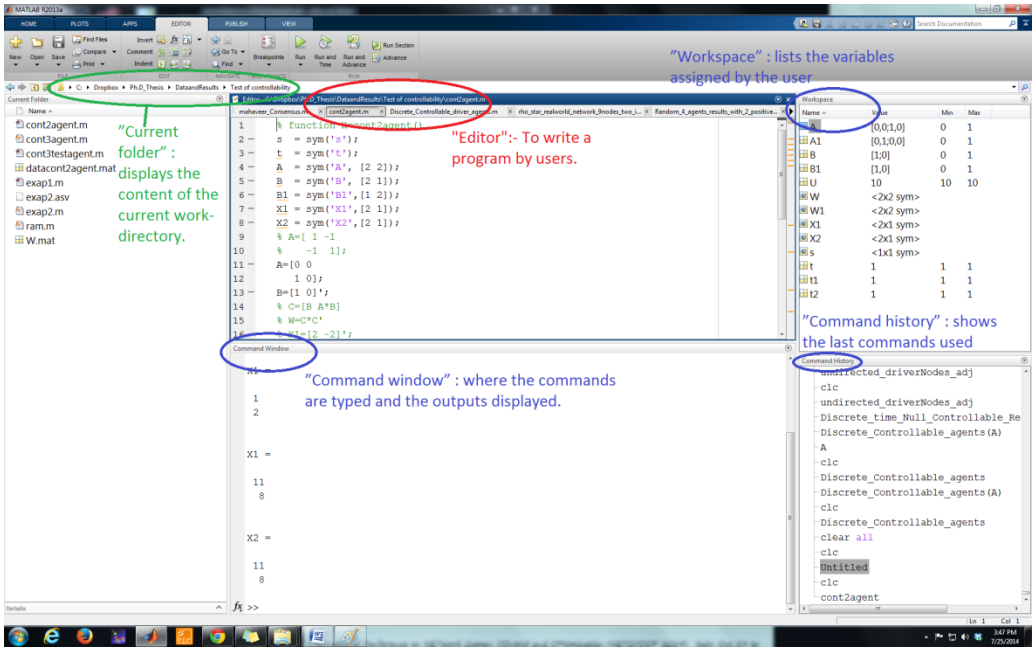

Figure 2: General view of the Matlab window.

Getting Help:

Type one of following commands in the command window:

- help – lists all the help topic
- help topic – provides help for the specified topic
- help command – provides help for the specified command
- help help – provides information for use of the help command
- helpwin – opens a separate help window for navigation
- lookfor keyword – Search all M-files for keyword.

Plotting:

Type help graph2d

- Plotting a point:

>> plot(variablename, ‘symbol’)

- Plotting Curves:

>> plot (x,y) – generates a linear plot of the values of x (horizontal axis) and y (vertical axis).

- Multiple Curves:

>> plot (x, y, w, z) – multiple curves can be plotted on the same graph by using multiple arguments in a plot command. The variables x, y, w, and z are vectors. Two curves will be plotted: y vs. x, and z vs. w.

legend (‘string1’, ‘string2’,…) – used to distinguish between plots on the

same graph.

- Multiple Figures:

figure (n) – used in creation of multiple plot windows. place this command before the plot() command, and the corresponding figure will be labeled as "Figure n"

close – closes the figure n window.

close all – closes all the figure windows.

- Subplots:

subplot (m, n, p) – m by n grid of windows, with p specifying the current plot as the pth window

Example: (polynomial function)

Plot the polynomial using linear/linear scale, log/linear scale, linear/log scale, & log/log scale:

```
y = 2*x^2 + 7*x + 9
% Generate the polynomial:
x = linspace (0, 10, 100);
y = 2*x.^2 + 7*x + 9;
% plotting the polynomial:
figure (1);
subplot (2,2,1), plot (x,y);
title ('Polynomial, linear/linear scale');
ylabel ('y'), grid;
subplot (2,2,2), semilogx (x,y);
title ('Polynomial, log/linear scale');
ylabel ('y'), grid;
subplot (2,2,3), semilogy (x,y);
title ('Polynomial, linear/log scale');
xlabel('x'), ylabel ('y'), grid;
subplot (2,2,4), loglog (x,y);
title ('Polynomial, log/log scale');
xlabel('x'), ylabel ('y'), grid;
```

Some useful commands to design the LTI control systems using Matlab command window.

Command	**Syntax**	**Description**
laplace	laplace(F,t,s)	Computes Laplace transform of F on time t and returns it as a function of complex variable s (time domain to frequency domain conversion).
ilaplace	ilaplace(F,s,t)	computes Inverse Laplace transform of F on the complex variable s and returns it as a function of the time, t.
tf	sys = tf(num, den)	Get transfer function for systems.
zp2tf	[NUM,DEN] = zp2tf(Z,P,K)	Zero-pole to transfer function conversion.
zero	[Z,G] = zero(sys)	Compute the zeros and gain of a SISO system.
pole	P = pole(sys)	Computes the poles of the systems.
series	[num, den] = series(num1, den1, num2, den2) sys = series(sys1, sys2)	Forms the series combination of 2 blocks to produce the equivalent transfer function.

parallel	[num, den] = parallel(num1, den1, num2, den2) sys = parallel(sys1, sys2)	Give Parallel combination of two blocks to produce the equivalent transfer function.
feedback	[num, den] = feedback(num1, den1, num2, den2, −1) sys = feedback(sys1, sys2, −1)	Give closed-loop system that has block 1 in the forward path and block 2 in the feedback path.“-1” is for negative Feedback.
cloop	[num, den] = cloop(num1, den1, −1)	Give the closed-loop system with negative unity feedback.
damp	[Wn,Z] = damp(sys)	Natural frequency and damping of linear system.
eig	[V,D] = eig(A)	produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors.
rlocus	rlocus(num, den) rlocus(sys)	Computes and plots the root locus for a system.
rlocfind	[K, poles] = rlocf ind(num, den) [K, poles] = rlocf ind(sys);	Computes the gain and the corresponding closed-loop pole locations for a specified point on the root locus.
	[K, poles] = rlocf ind(num, den, P) [K, poles] = rlocf ind(sys, P);	Instead of graphically selecting a point, one or more desired closed-loop pole locations can be input to the function. The gains and closed-loop poles corresponding to those specified pole locations are returned.
bode	[mag, ph] = bode(num, den, w) [mag, ph] = bode(sys, w);	Computes the magnitude and phase in degrees of a system .
nichols	nichols(sys) nichols(sys,{wmin,wmax})	Nichols frequency response of dynamic systems.
logspace	w = logspace(N1, N2, 1 + N3 ∗ (N2 − N1))	Generates a row array of elements that are spaced logarithmically, that is, the ratio of two consecutive elements is equal throughout the array. This is useful for generating a frequency vector for use with the bode function. For example, to generate frequencies from 0.001 r/s to 10 r/s with 100 points per decade (the value that I use), the command would be w = logspace(−3, 1, 401);
semilogx	semilogy(X1,Y1,...)	semilogy plots data with logarithmic scale for the *y*-axis.
margin	[Gm, Pm, W cg, W cp] = margin(num, den); [Gm, Pm, W cg, W cp] = margin(sys); [Gm, Pm, W cg, W cp] = margin(mag, ph, w);	Compute gain margin (Gm), phase margin (Pm), phase Crossover frequency (Wcg), gain crossover frequency (Wcp).
nyquist	[re, im] = nyquist(num, den, w) [re, im] = nyquist(sys, w)	Computes the real and imaginary components of the frequency response of a system .A frequency vector is an optional input argument.
step	c = step(num, den, t) c = step(sys, t)	Computes the unit step response for a system. A time vector is an optional input argument.
pzmap	pzmap(sys1,'r',sys2,'y',sys3,'g')	Pole-zero map of dynamic systems.
ltiview	ltiview ltiview(sys1,sys2,...,sysn) ltiview(plottype,sys)	LTI Viewer for LTI system response analysis

	ltiview({'step';'nyquist'},sys)	
zp2ss	[A,B,C,D] = zp2ss(Z,P,K)	Zero-pole to state-space conversion.
ss2zp	[Z,P,K] = ss2zp(A,B,C,D,IU)	State-space to zero-pole conversion
tf2ss	[A,B,C,D] = tf2ss(NUM,DEN)	Transfer function to state-space conversion.
ss2tf	[NUM,DEN] = ss2tf(A,B,C,D,iu)	State-space to transfer function conversion.

Basics of Simulink Tool:

It allows user to develop his/her mathematical model using Graphical user interface (GUI) as Simulink model.

1. Open the Simulink Library Browser:

I. Run the MATLAB before you can open the Simulink Library Browser.

II. In the MATLAB Command Window, enter Simulink.

The Simulink Library Browser opens.

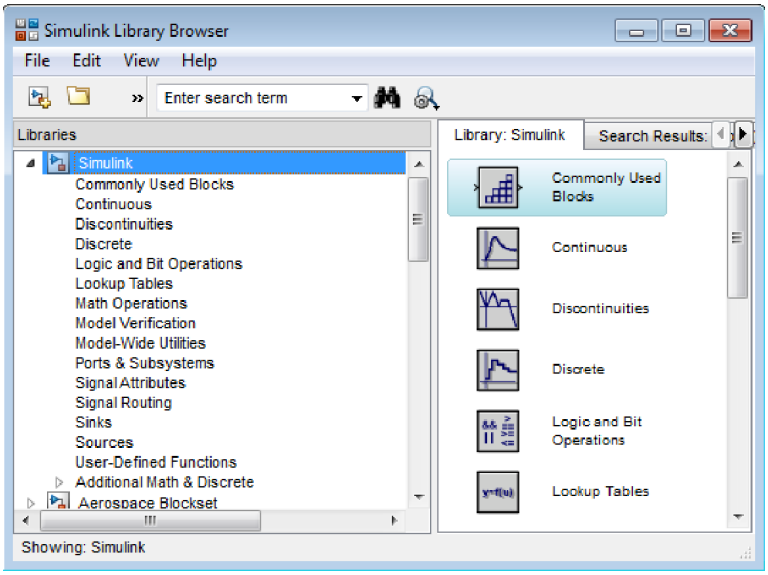

Figure 3: Simulink library browser

III. You can also open the Simulink Library Browser from the MATLAB Toolstrip, by clicking the **Simulink Library** button**.**

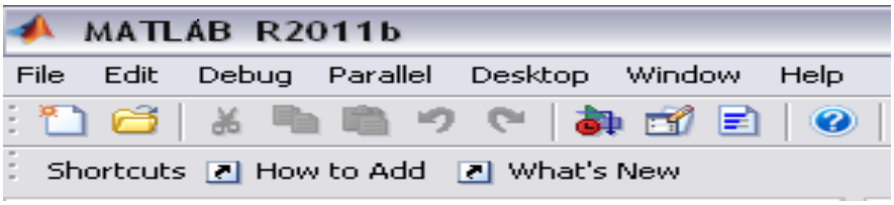

Figure 4: Shortcut to Simulink library browser

If you have not already loaded Simulink, a short delay occurs while it loads.

The Library Browser opens.

To keep the Library Browser above all other windows on your desktop, in the Library Browser, select **View > Stay on Top**.

2. Simulink Library Browser

The Simulink Library Browser displays the block libraries installed on your computer. You start to build models by copying blocks from a library into a Simulink Editor Model window.

For example, in the Library Browser below:

- In the **Libraries** pane on the left, the Sources library is the selected library.
- The Sine Wave block is selected.

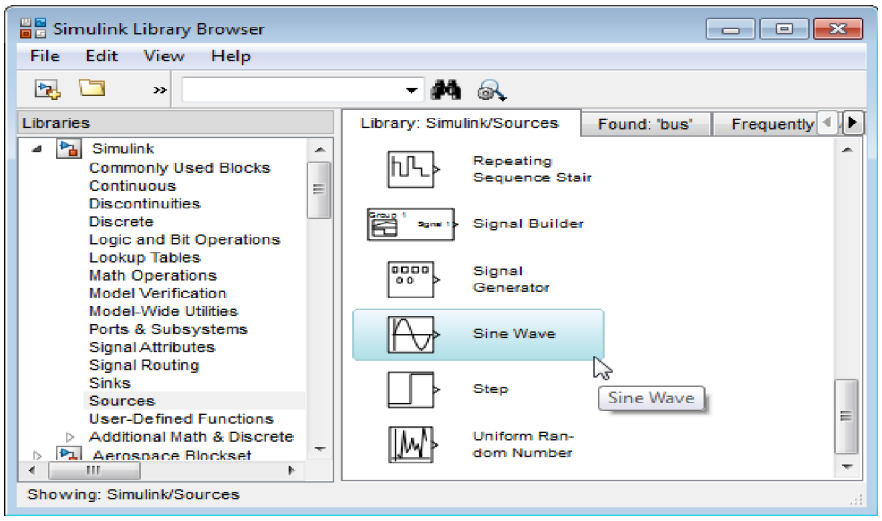

Figure 5: Selection of block (sine wave)

3. Create a New Simulink Model

Create a new Simulink model from the Simulink Library Browser.

I. From the Simulink Library Browser menu, select **File> New> Model.**

An empty model opens in the Simulink Editor.

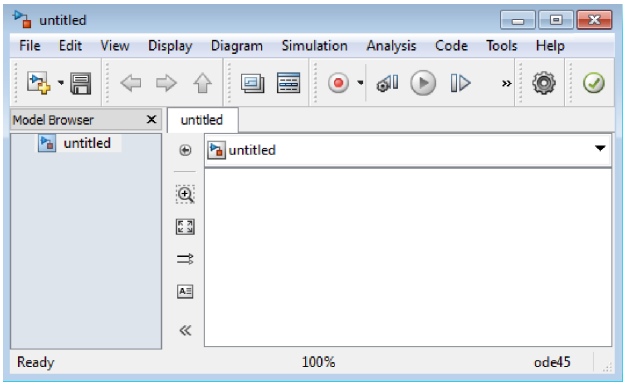

Figure 6: New model window

II. In the Simulink Editor, select **File > Save.**

III. In the Save As dialog box, enter a name for your model, and then click **Save**.

Simulink saves your model.

IV. Expand the Sources subnode, move the pointer and click the block labeled **Constant**, and while keeping the mouse button pressed down, drag the block and drop it inside the Simulation Window; then release the mouse button or right click on the Block and select the option Add to Untitled(Name of Model).

V. Right clicking on the block (Model window) will provide various options to users from which one can cut, copy, delete, format (submenu provides facilities for rotation of the block, flipping, changing the font of block name,...), etc...

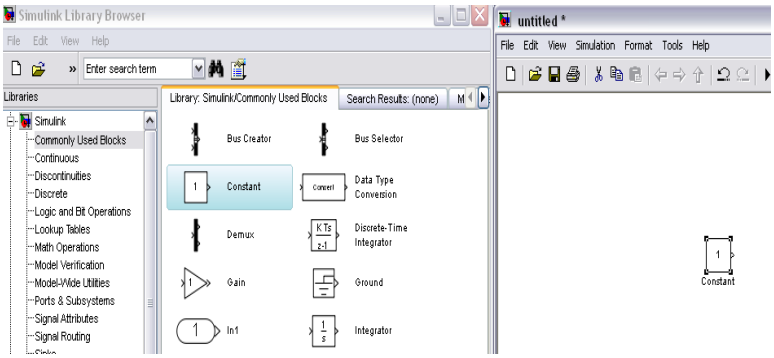

Figure 7: Drag and drop model from Simulink library to new model window.

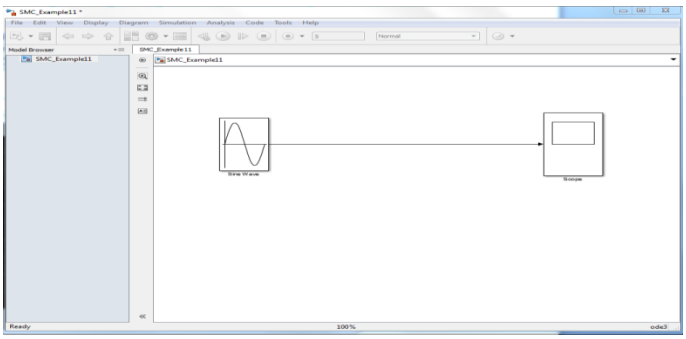

Figure 8: Connection between blocks

VI. Connect the blocks as per the system model and modify all the block parameters as per the system parameters.

VII. Select the **Simulation>Run** from Simulink editor menu bar. The simulation runs, and then stops when it reaches the stop time specified in the Model Configuration Parameters dialog box (simulation can be controlled by clicking the **Run** or **Pause** button on toolbar).

4. Simulink Editor

The Simulink Editor contains a block diagram of your model. You build models by dragging blocks from the Simulink Library Browser window to the Simulink Editor model window. In the model window, you build a block diagram by arranging the blocks logically, connecting the blocks with signal lines, and setting the parameters for each block.

Use Simulink Editor to set configuration parameters for the model, including the start and stop time, type of solver to use, and data import/export settings.

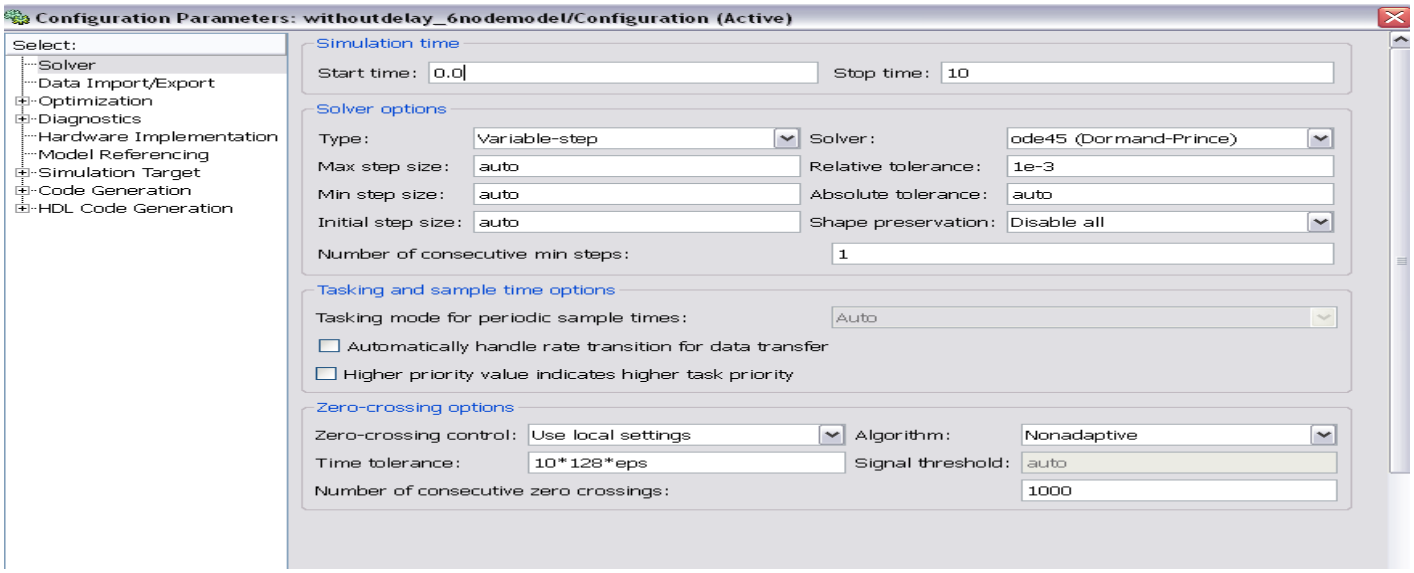

Figure 9: Configure parameters

5. Open an Existing Model

Open an existing Simulink model from the Simulink Library Browser.

I. From the Simulink Library Browser menu, select **File > Open**.

II. In the Open dialog box, select the model file that you want to open, and then click **Open**.

The selected model opens in the Simulink Editor.

EXERCISES

6. **Plot the graph of a polynomial equation in MATLAB**

   For the following equation, plot the values of y in MATLAB for a range of values in x,

   y = 2x^2+7x+9

   Include the code, plot and the corresponding description.

7. **Solve an ODE in MATLAB and Simulink, and plot the results obtained**

   a. Write a MATLAB code to solve the following ODE

   $$\frac{d^2}{dx^2}(y) = Cos(2x) - y$$

   Plot the results obtained and attach the code. Also describe the Toolboxes required for the code to work, if any.

   b. Model the following using Simulink

   $$\frac{d^2}{dt^2}(y) = -2\,\frac{d(y)}{dt} - 5y + 1$$

   Describe the blocks used and how each one of them contributes in obtaining the solution of the above equation. Use Scope to observe the output and attach the corresponding plot.