**SY202 – Cyber Systems Engineering**

**Intro**

**CSE**

**Due Date: 29 Jan 2019 (Start of Lab)**

**LABORATORY INVESTIGATION #01: MATLAB Scripts, Functions & SIMULINK**

**Objectives:**

Knowing how to 1) appropriately plot and label, 2) automate the process of loading data, and 3) model a system and then simulate the dynamic behavior of that system using MATLAB/SIMULINK is necessary for developing understanding of cyber physical systems. In this exercise you will learn to:

1. change the working directory to a desired folder
2. create plots and graphs with appropriate labels
3. create functions that perform mathematical operations on inputs to produce desired outputs
4. create scripts to automate the process of loading and manipulating data
5. create simple models, using four Simulink blocks

The table below provides an overview of the terminology and MATLAB functions used throughout this exercise. During the exercise keep in mind that **if you are confused about a MATLAB command you can type ‘help’ or ‘doc’ followed by the name of the function** you’d like more information about to see a document explaining the function in question. *Google is also an extremely helpful reference.*

|  |  |
| --- | --- |
| MATLAB Terminology and Function Overview | |
| Terminology | |
| Current Folder (see reference 1.) | The current folder is the hard drive location where MATLAB looks for data and functions that one may attempt to access. |
| Workspace (see reference 2.) | The MATLAB® workspace consists of the variables you create and store in memory during a MATLAB session. You add variables to the workspace by using functions, running MATLAB code, and loading saved data files. |
| Function [often written f(x)]  (see reference 3.) | A mathematical expression relating an independent *input* variable (or variables) to a dependent *output* variable (or vairables). In MATLAB a function is a file that accepts input quantities, performs operations on the inputs, and returns output quantities. |
| Script  (see reference 3.) | When you invoke a ***script***, MATLAB simply executes the commands found in the file. Scripts can operate on existing data in the workspace, or they can create new data on which to operate. Although scripts do not return output arguments, any variables that they create remain in the workspace, to be used in subsequent computations. In addition, scripts can produce graphical output using functions like plot. |
| Section (see reference 4.) | A portion of code in a script following the use of two percent symbols “%%”. MATLAB understand double percent symbols as demarcations between successive sections of code. A section of code can be run separate from the remainder of the script. |
| Functions and Programming Syntax | |
| cd(‘folderpath/foldername’) | Change current folder to the folder named “foldername” that is specified by the path “folderpath”. |
| figure(fig\_number) | Creates a figure with the specified figure number ‘fig\_number’ |
| clf | Clears the current selected figure |
| plot(x\_data,y\_data,formatting) | Plots the data arrays ‘x\_data’ and ‘y\_data’ with specified formatting |
| subplot(n\_rows,n\_cols,plot\_num) | Creates a subplot in the current figure such that the number of rows of figures are ‘n\_rows’, the number of columns of figures are ‘n\_cols’ and subsequent plot commands are drawn in the figure specified by the figure number ‘plot\_num’ |
| title(‘Figure Title’) | Places a title on the current figure with label ‘Figure Title’ |
| xlabel(‘x-axis label’) | Labels the horizontal axis of the current figure with the label ‘x-axis label’ |
| ylabel(‘y-axis label’) | Labels the vertical axis of the current figure with the label ‘y-axis label’ |
| legend(‘data 1’,’data 2’) | Creates a legend within the figure and labels the first two plotted entries ‘data 1’ and ‘data 2’, respectively |
| axis([xmin xmax ymin ymax]) | Sets the axes of the figure to window specified by xmin, xmax, ymin, and ymax |
| get(plot\_handle) | Displays the properties of the figure or plot handle defined by ‘plot\_handle’ |
| set(plot\_handle,’property’,property\_value) | Changes the property ‘property’ of the plot handle ‘plot\_handle’ to the value ‘property\_value’ |

**Part I: Changing the current folder and creating a script:**

I have provided two files called “Lab01\_3\_2DPlotting.m” and “Lab01\_4\_experimental\_results.m” in the GAfG (Google Apps for Government) folder.

* 1. Create a new folder on your desktop labeled “SY202\_MATLABExercise1”.
  2. Locate the “Lab01\_3\_2DPlotting.m” and “Lab01\_4\_experimental\_results.m” files and download from GAfG. Move the file to the “SY202\_MATLABExercise1” folder on your desktop.

In order for MATLAB to access the data file, the folder where the data file lies must be in MATLAB’s working directory. In practice it is a good idea to set MATLAB’s current folder to the location where the data file is placed. In MATLAB the current folder is displayed just above the command window as shown below:



* 1. Open MATLAB and identify the default current folder. Is the current folder the same as where your data file is located?

You should notice that by default MATLAB chooses its own folder as the current folder. As such, the current folder window to the left of the command window does not show your data file. We must change the current folder to access the data file.

* 1. Change the current folder to “SY202\_MATLABExercise1” by clicking the open folder icon  and selecting the folder.

After selecting the folder you should notice that the current folder window to the left of the command window now shows the two files downloaded from GAfG.

Note: If a path or file name contains a space, the input string must be enclosed in single quotes. Example: ‘c:\users\wtdoor\SY202 Files\SY202 MATLABExercise1 Data.mat’

**Part II: 2-D Plotting Exercise:**

Before creating a plot in MATLAB it is important to understand how data must be formatted. The inputs to the plot command must be arrays (or vectors) of the same size. Simply put, this means that every data point must have both an x and y value in order to plot correctly.

* 1. Open the file “Lab01\_3\_2DPlotting.m”.
  2. Complete in-class exercise with instructor

**Key points for making an effective plot**

* **Always label axes** and include units if appropriate
* If there are multiple plots on a graph, **include a legend**
* **Always include a title on the figure**
* **Use an appropriate font size on all figure labels**

**Part III: Creating a script in MATLAB :**

We will use a MATLAB function (Lab01\_4\_experiment\_result.m) to generate data for analysis. Next, we will make a script to import the data into MATLAB. A script is an ordered list of commands that MATLAB executes in succession. MATLAB executes commands in the order they are written.

* 1. Open a new script in MATLAB. Save the script as “Lab01\_3\_script.m”. Insert the following commands.

%% Section 1

a = 2;

b = 3;

c = a + b;

fprintf(‘c = %f’, c);

%% Section 2

clear all % clears all variables in the workspace

clc % clears command window

Lab01\_4\_experiment\_results; % Generates theoretical and “experimental” signals in MATLAB workspace

* 1. Highlight “Section 1” and then from the EDITOR tab execute the highlighted portion of code by clicking the “Run Section” button in the top right of the MATLAB window. Look at your MATLAB workspace to the right of your command window. You will notice variables named a, b, and c. Double click any variable to edit, the array with only one element (1x1 double). You can edit the values right in the workspace if you wish.

Now run both the whole script (all sections) by clicking on the “Run” button. You will notice a variable called ‘time’ that is an array with one row and 201 columns. Each element of the array represents a point in time. For example the fifteenth element of the array, time(15) =1.4 seconds, represents 1.4 seconds. The two other arrays “theory\_signal” and “noisy\_signal” represent the theoretically predicted signal and the experimentally measured signal, respectively. The fifteenth element of the theory\_signal variable, theory\_signal(15)= 8.3099, represents the theoretical data value at time 1.4 seconds. The figure populated when you call the “Lab01\_4\_experiment\_results” function graphically displays the data.

Note in your script that by including two percent symbols “%%” the section of code is highlighted. This highlighted portion of code is called a “section.” By partitioning your code using sections you can isolate portions of the code for further development or debugging. You can run only a highlighted section of code by clicking the “Run Section” button on the top right of your script window.

**Part IV: Creating a function in MATLAB :**

Throughout this course you will be taking data from sensors and comparing your measurements to those predicted by a theoretical model. This exercise will develop a script and a function to compare theoretical and experimental data.

In this exercise you will quantitatively compare the theoretical and experimental data. To do so, you will calculate the root mean square error and normalized root mean square error between the theoretical and experimental signals

The root-mean-square error and the normalized root-mean-square error are commonly used to measure the differences between a value predicted by a model and those actually measured. Basically, the root-mean-square error measures how accurately the theoretical model represents the observed data, or vice-versa. The root-mean-square error formula is

The experimental data is represented by . The expression represents the kth element of the array . The variable represents the theoretically predicted signal. The variable represents the number of elements in the theoretical and experimental signals. *Notice that the number of elements in each signal must be equal in order to calculate the RMSE.* The normalized root-mean-square error is simply the root-mean-square error divided by the range of observed values in the theoretical signal, represented as a percentage, i.e.

* 1. Create a new function in MATLAB by clicking NewFunction in the top left corner. Modify the function name, inputs, and outputs as follows:

function [RMSE,NRMSE] = RMSE\_Calc(y\_theory,y\_exp,N)

% Function to calculate the root-mean-square error and normalized

% root-mean-square error.

function [y1,...,yN] = myfun(x1,...,xM) declares a function named myfun that accepts inputs x1,...,xM and returns outputs y1,...,yN. This declaration statement must be the first executable line of the function.

* 1. Save the file as “RMSE\_Calc.m”. The name of the file must have the same name as the function.
  2. Within the environment of the function, enter the formula to calculate the RMSE:

numerator = (y\_exp-y\_theory).^2;

RMSE = sqrt(sum(numerator)/N);

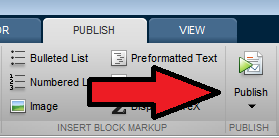
Notice that instead of using a for loop, you can calculate the difference between two variables element by element over the whole array. To square each element of the result, use the syntax “.^” instead of “^” (the dot in front of the hat implies that the operation is element wise).

* 1. Enter (type) the formula to calculate the NRMSE and save the file.

**Part V: Calling a function from a script:**

5.1) Return to your script file: Lab01\_3\_script.m.

5.2) Compute the RMSE and NRMSE of the experimental and theoretically predicted signals in a new section of your script. Evaluate the section.

5.3) Show your instructor the result. Select “Publish” from MATLAB’s top menu and hit the “Publish” button. 

**Key points**

* **Scripts** are a series of commands MATLAB executes in order
* **Scripts can be divided into sections or “cells”** using the “%%” symbol to partition each section
* **Functions** perform mathematical operations on input variables to produce output variables.
* The **root-mean-square error and normalized root-mean-square error** are two ways of measuring how accurately experimental data agrees with a theoretical model.

**Part VI: Modelling in Simulink:**

You can use Simulink to model a system and then simulate the dynamic behavior of that system. The basic techniques you use to create a simple model in this tutorial are the same techniques that you use for more complex models.

To create this simple model, you need four Simulink blocks:

* Sine Wave — Generates an input signal for the model.
* Integrator — Processes (Integrates) the input signal.
* Bus Creator — Combines the input signal and processed signal into one signal.
* Scope — Visualizes the signals.

Simulating this model integrates a sine wave signal to a cosine signal and then displays the result, along with the original signal, in a scope window.

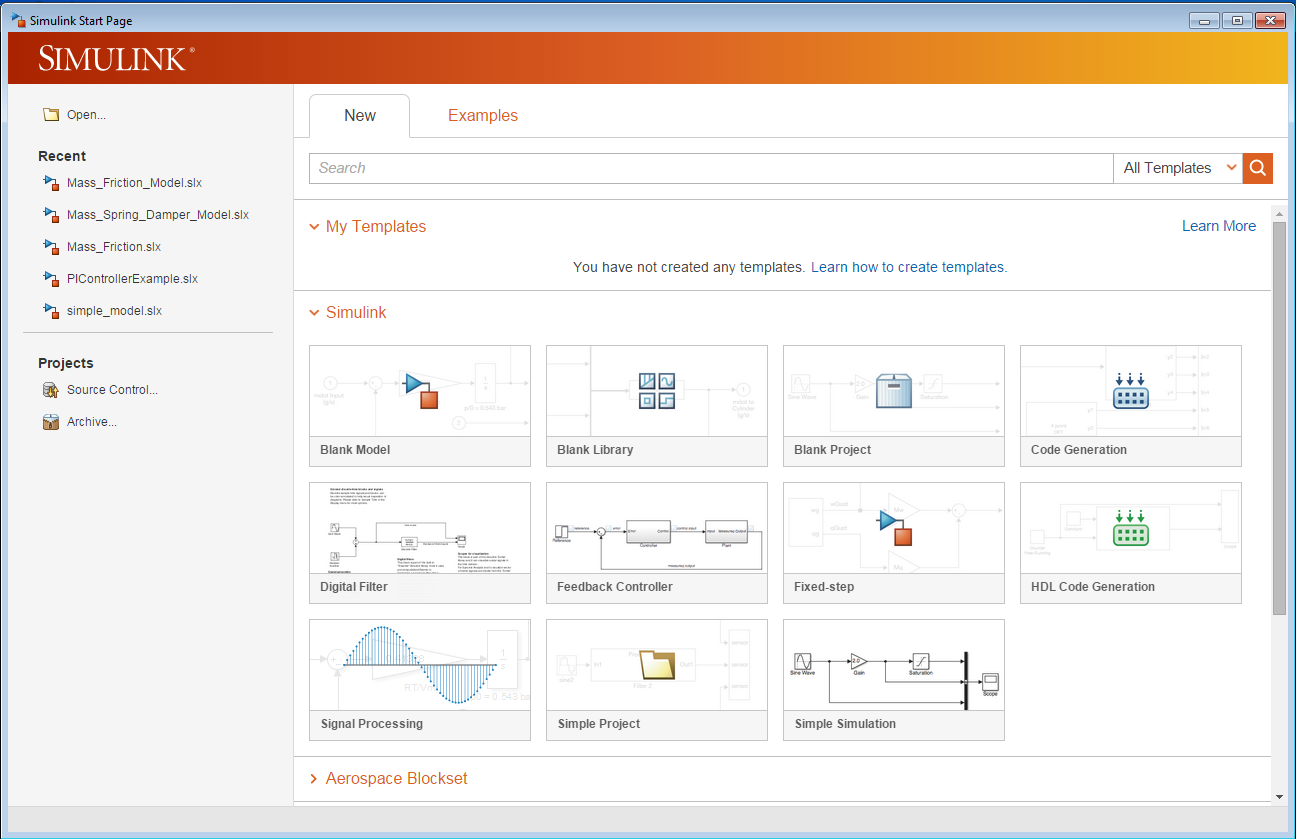
**Open the Simulink Start Page**

From the Simulink Start Page, you can create a new model (block diagram) where you can build and simulate models of your system using blocks. You need MATLAB® running before you can open the Simulink Start Page.

1. From the MATLAB Home Toolstrip, click the **Simulink** button , or in the **Command Window**, enter

simulink

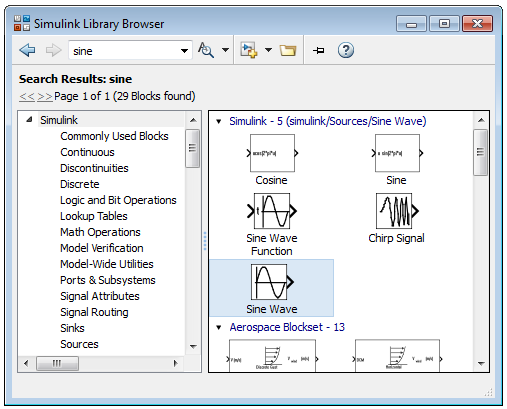
A short delay occurs the first time you open Simulink.



1. Select a “Blank Model”. This will open a Simulink Editor window where the model will be created and simulated.
2. Select File > Save as. In the File name text box, enter a name for your model. For example, enter simple\_model, and then click Save. Simulink saves your model with the file name simple\_model.slx.

### Browse or Search for Specific Blocks

1. Select the Simulink Library Browser button  or select View > Library Browser to use Simulink Blocks. Set the Library Browser to stay on top of the other desktop windows. On the Library Browser toolbar, select the **Stay on top** buttonhttp://www.mathworks.com/help/simulink/gs/simulik_library_browser_pin_icon.png.
2. To browse through the block libraries, select a MathWorks® product and a functional area in the left pane. Alternatively, you can search all of the available block libraries at once.
3. Search for a Sine Wave block. In the search box on the browser toolbar, enter sine, and then press the Enter key. Simulink searches the libraries for blocks with sine in their name, and then displays the blocks.



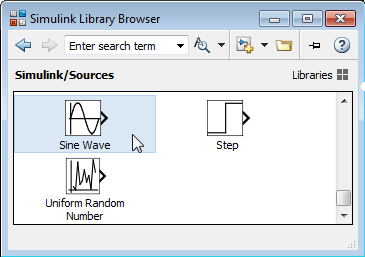
1. Get detailed information about a block. Right-click a block, and then select **Help for the <block name>**. The Help browser opens with the reference page for the block.
2. View block parameters. Right-click a block, and then select **Block Parameters**. The block parameters dialog box opens.

**System Modeling**

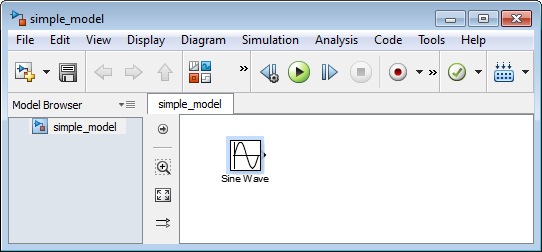
### Add Blocks to a Model

To build a model, begin by copying blocks from the Simulink Library Browser to the Simulink Editor.

1. In the Simulink Library Browser, select the Sources library. The Library Browser displays the source blocks in the library.
2. Select the Sine Wave block.



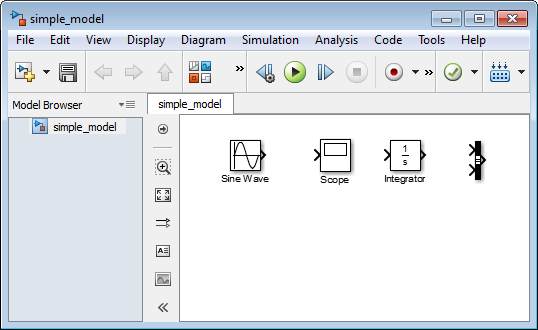
1. Drag the Sine Wave block to the Simulink Editor. A copy of the Sine Wave block appears in your model.



1. Add the following blocks to your model, using the same approach that you used to add the Sine Wave block.

| **Library** | **Block** |
| --- | --- |
| Sinks | Scope |
| Continuous | Integrator |
| Signal Routing | Bus Creator |

1. Your model now has the blocks you need for the simple model.

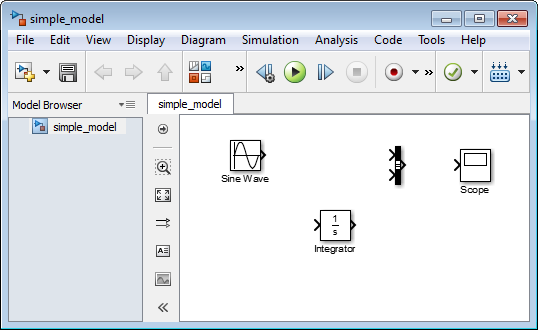


### Move and Resize Blocks

Before you connect the blocks in your model, arrange them logically to make the signal connections as straightforward as possible.

1. Move the Scope block after the Bus block output. You can either:
   * Click and drag a block.
   * Select the block, and then press the arrow keys on your keyboard.
2. Move the Sine Wave and Integrator blocks before the Bus block.

Move the blocks until your model looks similar to the following figure.

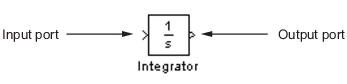


### Simulink Block Connections

After you add blocks to your model, connect them with lines. The connecting lines represent the signals within your model.

Most blocks have angle brackets on one or both sides. These angle brackets represent input and output ports:

* The > symbol pointing into a block is an input port.
* The > symbol pointing out of a block is an output port.



### Draw Signal Lines Between Blocks

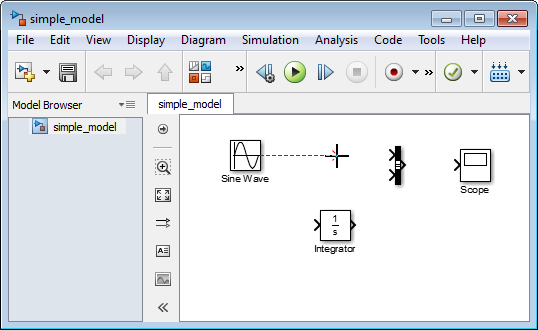
Connect the blocks by drawing lines between output ports and input ports.

1. Position the cursor over the output port on the right side of the Sine Wave block.

The pointer changes to a cross hair (+) while over the port.

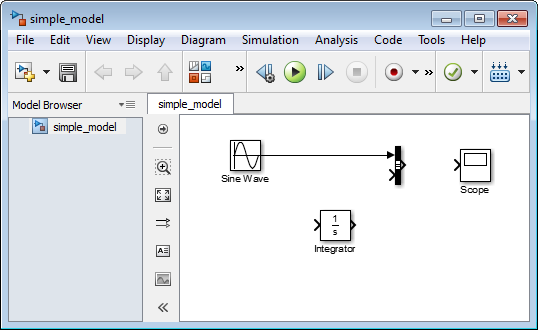
1. Click, and then drag a line from the output port to the top input port of the Bus block.

While you are holding down the mouse button, the connecting line appears as a red dotted arrow.



1. Release the mouse button when the pointer is over the output port.

Simulink connects the blocks with a line and an arrow indicating the direction of signal flow.

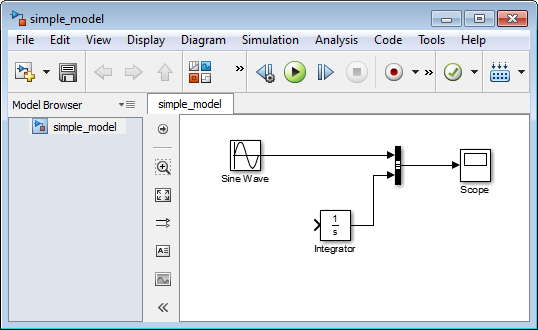


1. Connect the output port of the Integrator block to the bottom input port on the Bus Connector block using this alternative procedure:
   1. Select the Integrator block.
   2. Press and hold the **Ctrl** key.
   3. Click the Bus Connector block.

The Integrator block connects to the Bus Connector block with a signal line.

|  |
| --- |
| **Note:**   The **Ctrl+click** shortcut is useful when you are connecting widely separated blocks or when working with complex models. |

1. Connect the Bus Connector block to the Scope block by aligning ports:
   1. Click and drag the Scope block until its input port is aligned with the Bus Connector output port. A light blue line appears between the ports.
   2. Release the mouse button, and then click the blue line to create a black signal line.

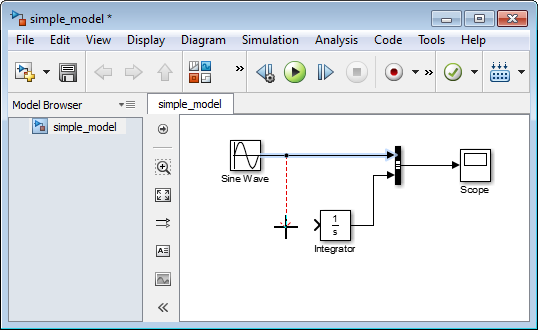


### Draw Branched Signal Lines

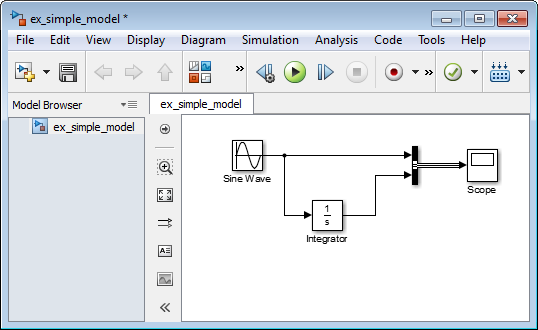
Your simple model is almost complete. To finish the model, connect the Sine Wave block to the Integrator block.

This connection is different from the other connections, which all connect output ports to input ports. Because the output port of the Sine Wave block is already connected, you must connect this existing line to the input port of the Integrator block. The new line, called a branch line, carries the same signal that passes from the Sine Wave block to the Bus block.

1. Hold down the **Ctrl** key.
2. Position the cursor where you want to start a branch line. Click, and then drag the cursor away from the line to form a dotted-red line segment.



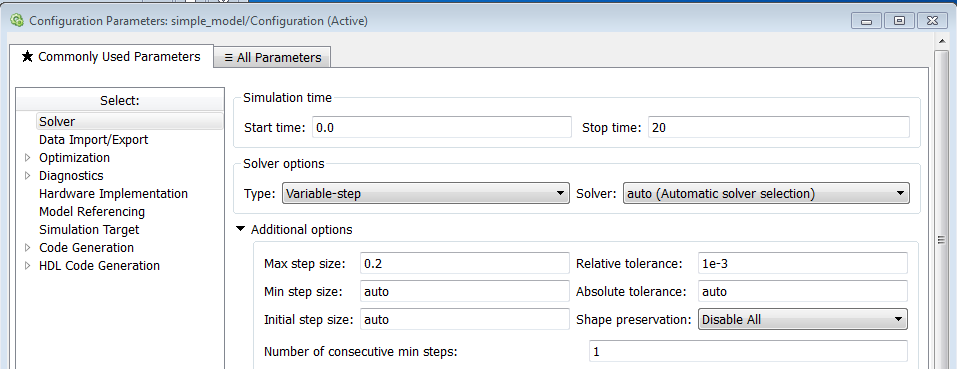
1. Drag the cursor to the Integrator input port, and then release the mouse button.
2. Drag line segments to straighten and align with blocks. Your model is now complete.



### Define Simulation Parameters

Before you simulate the behavior of a model, define the simulation parameters. Simulation parameters include the type of numerical solver, start and stop times, and maximum step size.

1. From the Simulink Editor menu, select **Simulation** > **Model Configuration Parameters**. The Configuration Parameters dialog box opens to the **Solver** pane.
2. In the **Stop time** field, enter 20. Select **Additional options.** In the **Max step size** field, enter 0.2.



1. Click **OK**.

### Run Simulation

After you define Model Configuration Parameters, you are ready to simulate your model.

1. From the Simulink Editor menu bar, select **Simulation** > **Run**.

The simulation runs, and then stops when it reaches the stop time specified in the Model Configuration Parameters dialog box.

|  |
| --- |
| **Tip:**   Alternatively, you can control a simulation by clicking the **Run** simulation button http://www.mathworks.com/help/simulink/gs/start_button_ue.png and **Pause** simulation button http://www.mathworks.com/help/simulink/gs/pause_button_ue.png on the Simulink Editor toolbar. |

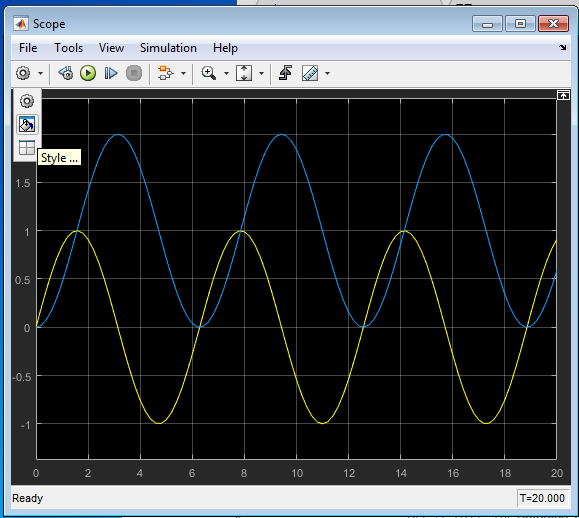
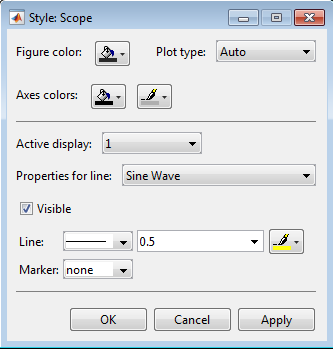
### Observe Simulation Results

After simulating a model you can view the simulation results in a Scope window.

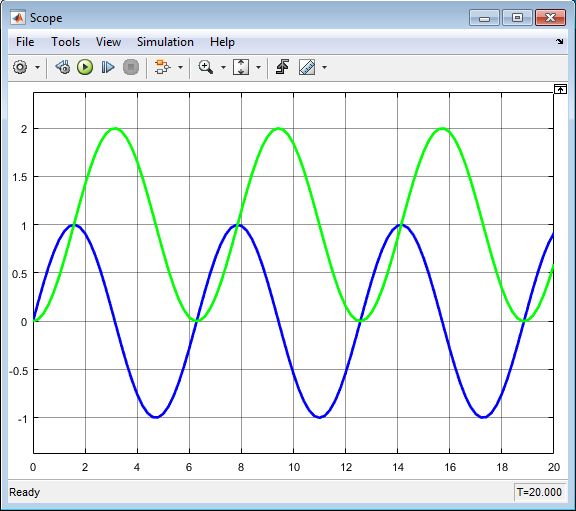
1. Double-click the Scope block.

The Scope window opens and displays the simulation results. The plot shows a sine wave signal with the resulting cosine wave signal.

1. From the Scope block toolbar, click the tab next to the **Parameters** button . Select the **Style** tab (paint bucket) shown in the picture below. The Scope Parameters dialog box displays figure editing options.

1. Change the appearance of the figure. For example, select white for the Figure color and Axes background color (icons with a paint bucket).
2. Select black for the Ticks, labels, and grid colors (icon with a paint brush).
3. Change the signal line colors for signal 1 (Sine Wave) to blue and for signal 2 (Integrator) to green. Change the line weight for both signals to 2.0. To see your changes, click **OK**.



**Deliverables:**

1. Submit a hardcopy published document of “Lab01\_3\_script.m”
2. Provide a screenshot of Simulink Model.

**References:**

Mathworks Website:

1. Current Folder: [http://www.mathworks.com/help/MATLAB/MATLAB\_env/files-and-folders-that-MATLAB-accesses.html](http://www.mathworks.com/help/matlab/matlab_env/files-and-folders-that-matlab-accesses.html)
2. Workspace: [http://www.mathworks.com/help/MATLAB/MATLAB\_env/what-is-the-MATLAB-workspace.html](http://www.mathworks.com/help/matlab/matlab_env/what-is-the-matlab-workspace.html)
3. Functions and Scripts: [http://www.mathworks.com/help/MATLAB/learn\_MATLAB/scripts-and-functions.html](http://www.mathworks.com/help/matlab/learn_matlab/scripts-and-functions.html)
4. Sections and Markups: [http://www.mathworks.com/help/MATLAB/MATLAB\_prog/marking-up-MATLAB-comments-for-publishing.html](http://www.mathworks.com/help/matlab/matlab_prog/marking-up-matlab-comments-for-publishing.html)
5. Root-mean-square error: <http://en.wikipedia.org/wiki/Root-mean-square_deviation>