CARLETON UNIVERSITY
Department of Systems and Computer Engineering

SYSC 3600
Lab #1: Introduction to Simulation and SIMULINK[1]

The laboratories are intended to introduce students taking *SYSC 3600: Systems and Simulation* to some of the techniques used to simulate continuous-time linear systems; and, in conjunction with the lectures, to develop the students' understanding of the dynamic behaviour of linear systems.

For this lab, no lab report is required, but you will still be graded on the lab. To receive full marks, you will have to demonstrate to the TA or instructor that you know and understand the material in this lab for each of the **Instructor Verification** points, as provided at the end of this lab manual. Copies of the **Instructor Verification Sheet** will be made available in the lab.

---

1 Introduction to SIMULINK

SIMULINK is a software package developed by "The MathWorks Inc.". It is a powerful tool to simulate complex dynamic systems. The SIMULINK software is really just a module within the MATLAB software package, which is a powerful mathematical tool which operates on vectors and matrices and has become one of the standard mathematical software packages for research engineers and scientists.

The usefulness of SIMULINK is that one can draw out the *block diagram* or *simulation diagram* that describes a dynamic system and then simulate the system. SIMULINK will create the differential equations that represent the simulation diagram and one can then use the MATLAB tools to analyze the system. For example, one can find the transfer function, the state space equations, eigenvalues and eigenvectors, poles and zeros, plot Bode diagrams, etc. For the purpose of these laboratories you will only use a small portion of the capabilities of MATLAB/SIMULINK, and this tutorial will introduce you to the tool.

To start SIMULINK, you must first launch MATLAB which will bring up a window similar to that shown in Fig. 1. After launching MATLAB you will notice that within the MATLAB window there is a `Command Window` where MATLAB commands can be executed. From within this `Command Window`, type

```
simulink
```

to start the SIMULINK portion of the MATLAB software package. Doing so should bring up the SIMULINK Library Browser window similar to that shown in Fig. 2[2]. The SIMULINK Library Browser window is organized to show the major categories of simulation blocks as well as other installed sets of specialized simulation blocks. For the labs, you will require simulation blocks primarily from the following categories:

(a) *Sources*: Sources are blocks that act as the source for some form of input for your simulation. Within this course we will focus primarily on the step function, ramp function, sinusoids, constants, and noise through random number generators.

Special note: The continuous-time *impulse function* $\delta(t)$, or Dirac delta function, is not available as a source since it is a mathematical construct that is not physically realizable.

(b) *Sinks*: Sinks are blocks that gather results from your simulation to be displayed as plots, values, or stored to file or to the MATLAB variable workspace for later processing.

[1] This tutorial was prepared for SIMULINK® in MATLAB® 7.8.0 (R2009a) under Windows XP®.
[2] The appearance may differ for different versions of MATLAB as well as which additional toolboxes are installed.

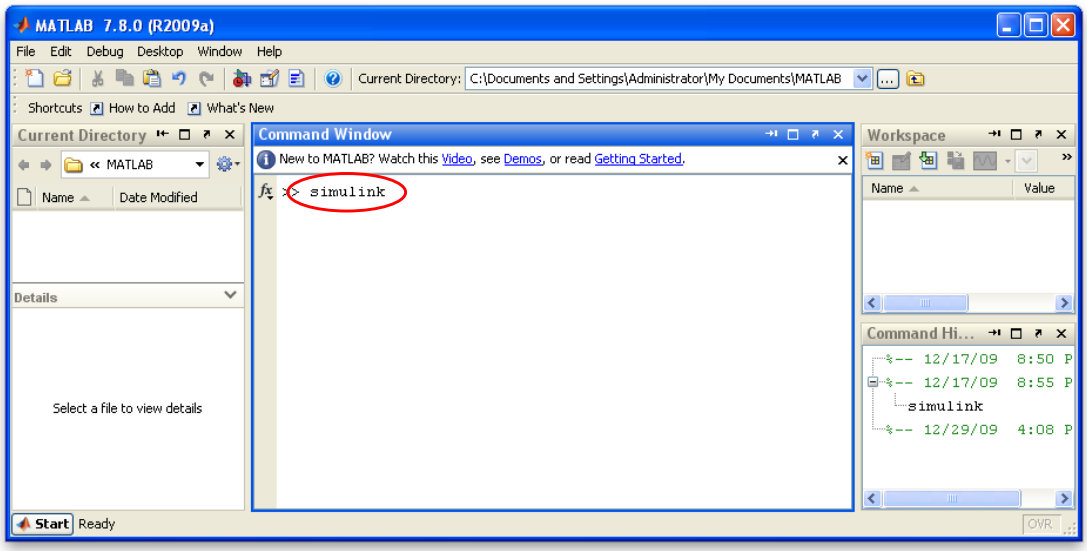

Figure 1: Launching SIMULINK from MATLAB.

(c) *Continuous*: Consists of some specialized continuous-time functions such as the integrator and the derivative, as well as transfer function blocks defined through the differential equation coefficients, poles/zeros, or state space representations we will learn about throughout the course.

(d) *Math Operations*: Consists of the mathematical "glue" that will allow us to connect blocks to implement differential equations. The blocks of primary interest are the sum unit for summing multiple links and the gain for scaling the amplitude of a particular link connection.

(e) *Ports & Subsystems*: Consists of blocks that will help manipulate and group other blocks as well as manipulate and group the signals. Some of these blocks can be used to make subsystems to modularize the look of our models

(f) *Signal Routing*: Consists of blocks that can help group, combine, and route signals for later uses such as plotting or passing to subsystems.

2 SIMULINK Tutorial: Converting Celsius to Fahrenheit

For this tutorial on SIMULINK, we will simply look at converting Celsius, °C, to Fahrenheit, °F. While converting from Celsius to Fahrenheit really doesn't warrant performing a *simulation*, we can at least learn some of the basics of using SIMULINK. Converting from Celsius to Fahrenheit can be expressed mathematically as

$$°\mathrm{F} = \frac{9}{5}°\mathrm{C} + 32 \tag{1}$$

From Eq. 1, let's build a simple SIMULINK model that will implement this conversion.

2.1 Building a SIMULINK model

An empty simulation model window must first be opened before we can setup our model to perform the Celsius to Fahrenheit temperature conversion. A new simulation model is opened by selecting

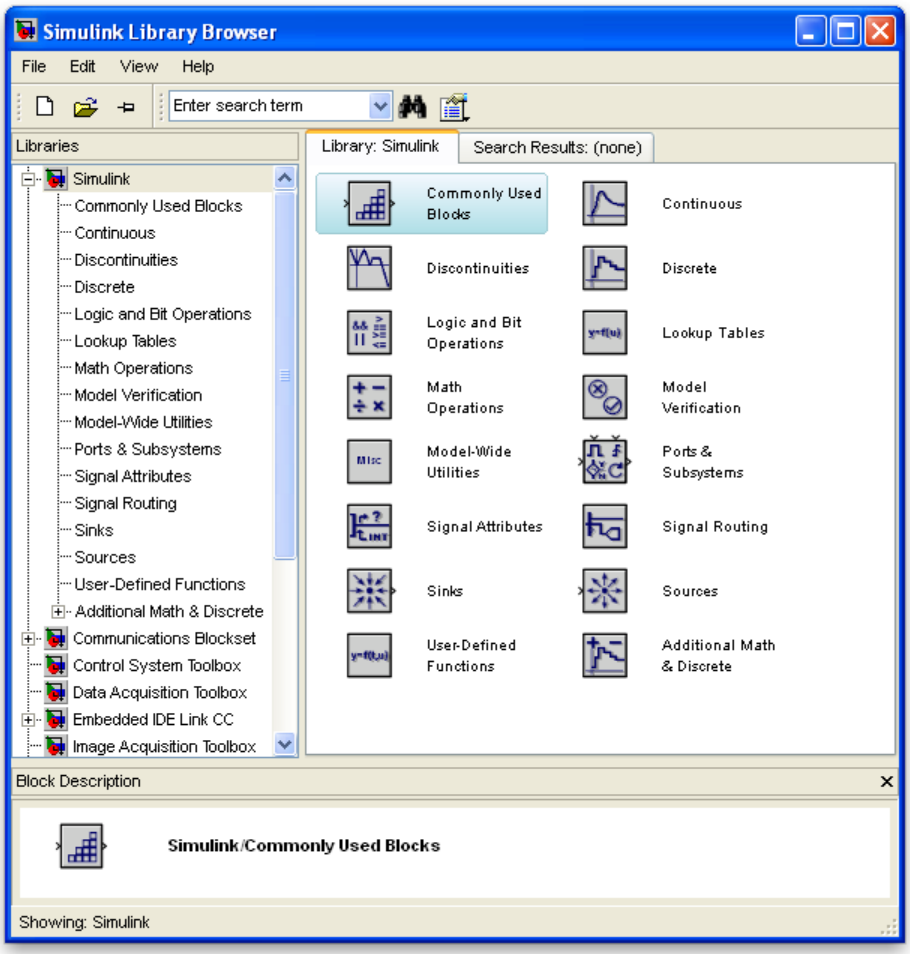

Figure 2: SIMULINK Library Browser window.

`File→New→Model`

in the SIMULINK Library Browser window (see Fig. 2). A new window, as shown in Fig. 3, will open for your new simulation model and will be labeled "untitled" (you may wish to make the window wider to see all of the icons along the top). It is within this window that you will construct your simulation model. We will start by setting up blocks to perform the mathematics of Eq. 1 as shown in Fig. 4.

You can drag-and-drop any of the blocks listed in the SIMULINK Library Browser into your model. Start by finding the `Gain` and `Sum` blocks under the `Math Operations` section of the SIMULINK Library Browser. Then, find the `Constant` block under the `Sources` section and drag-and-drop it into your model. One convenient icon, as pointed out in Fig. 3, is the `Library Browser` icon which when pressed will raise the SIMULINK Library Browser window if it is hidden under other windows.

Now that you have the primary blocks placed in your model, you must connect them appropriately. You can do this by left-clicking and holding on an output port of a unit and dragging a connection to the input port of another unit to make the desired connection. Another way of doing this is to left-click on the output source block and then hold CTRL and left-click on the destination block. Connect the three units in your model such that they implement Eq. 1 as shown in Fig. 4. Note that you can pick up and move the simulation blocks at any time and the connections will stretch as you move the block. Also, if you wish to delete a connection or delete a block, simply left-click on the connection/block and then press delete.

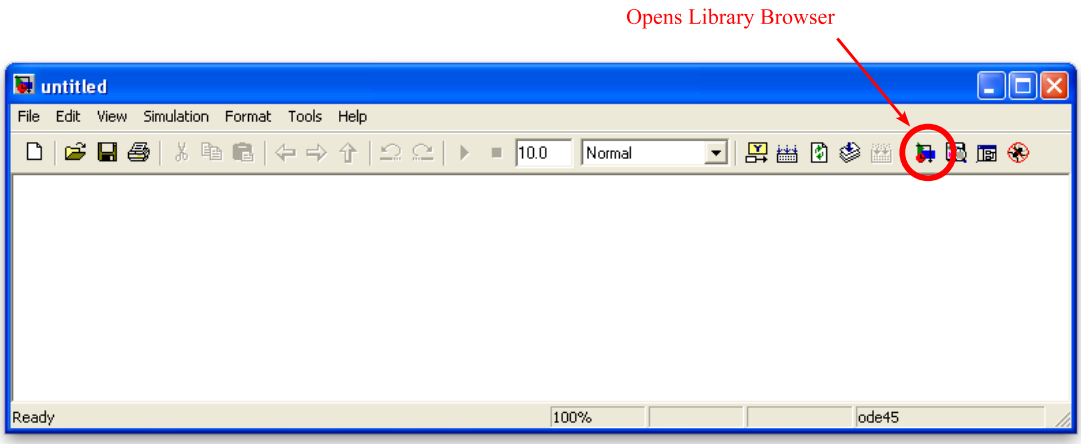

Figure 3: Empty window for the simulation model to be implemented.

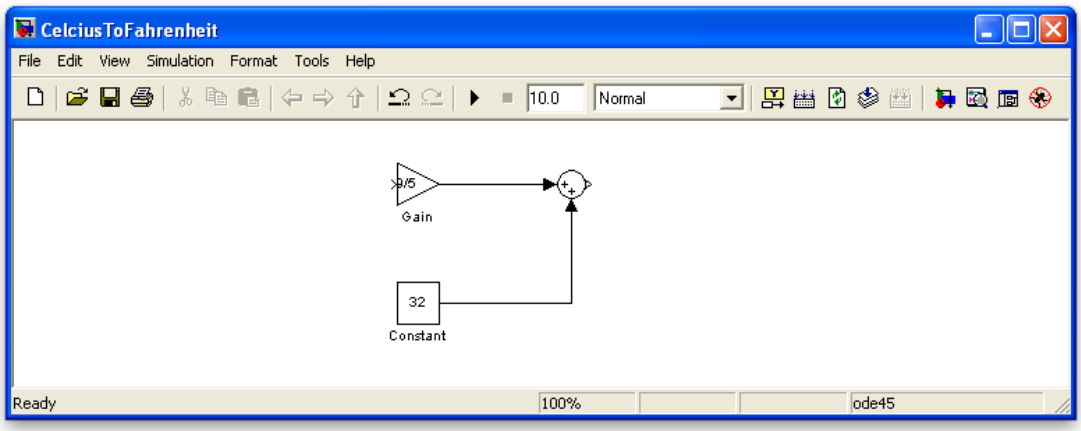

Figure 4: Math for implementing the Celsius to Fahrenheit converter.

Most blocks within SIMULINK can be configured through its block parameters. Let's start with setting the block parameters for the `Gain` block. If you double-click on the `Gain` block the Block Parameters window for the block will open as shown in Fig. 5. You can set the gain to a constant, or as shown in Fig. 5 to a simple expression. Set the gain of the block to $9/5$.

Now open the block parameter window for the `Constant` source block by double-clicking on it. As shown in Fig. 6, set the constant value to 32, which is the constant that must be added to implement Eq. 1. This block will now supply a value of 32 at its output.

The default setup for the `Sum` block is two summing inputs and one output, so we don't need to change it for implementing Eq. 1. If we needed to change the block parameters for the `Sum` block, you can similarly open its block parameters window by double-clicking on it which will open the window shown in Fig. 7. A sum unit always has a single output, but the number, type, and placement of the inputs can be changed. The default setting uses the string '|++' where each '+' indicates an input port and the '|' indicates a spacer. The type of port can be either a '+' (addition) or '-' (subtraction) port. The '|' indicates a spacer and has no function in the actual simulation, but is used to change the visual spacing of the ports around the sum unit. Using the spacer '|' in the appropriate spots may make your simulation model easier to read. For instance, a 4 input sum unit could be given as '++++' or similarly given as '++|++' where extra spacing is put between

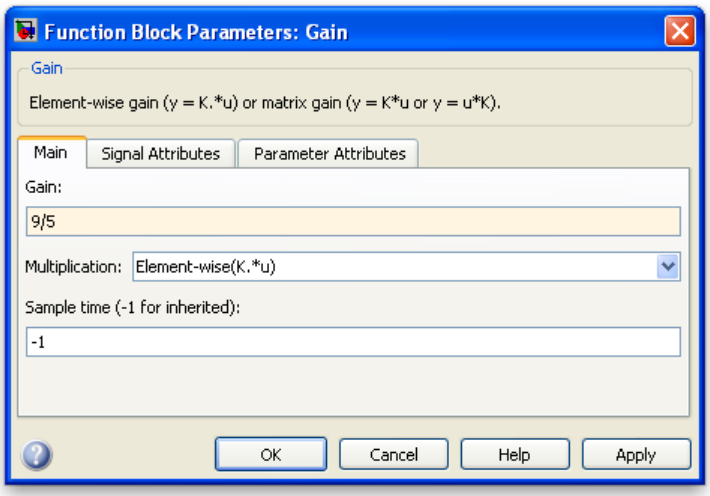

Figure 5: Block parameters for the `Gain` block.

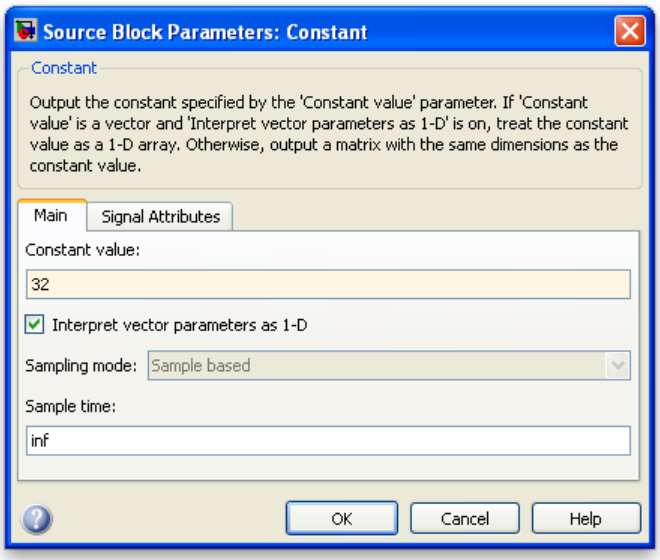

Figure 6: Block parameters for the `Constant` source block.

the two middle ports. Having a sum unit where two ports sum and one port subtracts could be given as '++-' with appropriate spacing using '|' inserted to adjust how the unit is displayed with the model. More complicated sum units could be formed such as '++|-|++', etc. Experiment with different configurations, but return the sum unit back to '|++' before proceeding.

The simulation model shown in Fig. 4 implements the math required for the Celsius to Fahrenheit conversion. What we need now to complete the simulation model is some input values to represent the temperatures in Celsius and then a way of observing the output of the model so that we can see the temperatures converted to Fahrenheit. One simple approach would be to supply a constant input using the `Constant` block. Create another `Constant` block by either selecting it from under the `Sources` section of the SIMULINK Library Browser or by copying and pasting the `Constant` block already available in your model. Connect this `Constant` block to the input of the system as shown in Fig. 8.

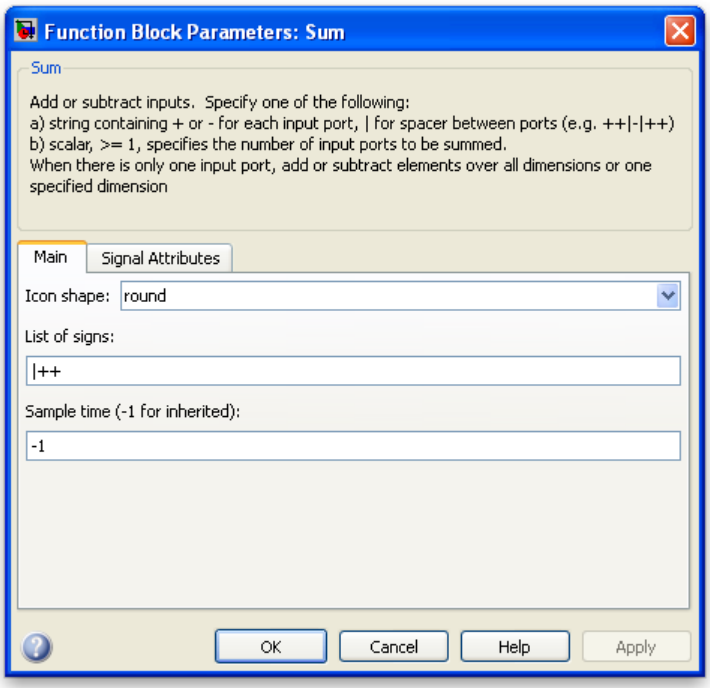

Figure 7: Block parameters for the `Sum` unit.

For monitoring the output, we require an appropriate sink for the output. In this example, our input is a constant so we will only obtain a constant for the output. The `Display` block under `Sinks` in the SIMULINK Library Browser is appropriate in this case. Place the `Display` block into your model and connect it to the output of the Celsius to Fahrenheit converter as shown in Fig. 8.

Now that a model is setup, we can simulate the model. This is done by either pressing the simulation button in the model toolbar (the play button circled in red in Fig. 8) or by selecting

`Simulation→Start`

from the menu. Run the simulation a few times for different values of the input Centigrade temperature in the `Constant` block and verify that the conversion takes place correctly. Before continuing to the next section, you may wish to save your simulation model using

`File→Save`

from the menu.

2.2 Subsystems within simulation models

Since simulation models can become quite complex, it is often convenient to group blocks in a model into subsystems. We will create a subsystem for the math portion of the Celsius to Fahrenheit converter to demonstrate how this is done. Following Fig. 9, select the three blocks that implement the mathematics of Eq. 1. With the blocks selected, choose

`Edit→Create Subsystem`

from the menu. This will group the selected blocks into a subsystem as shown in Fig. 10. If you double-click on the newly created subsystem, a subsystem model window will open as shown in Fig. 11.

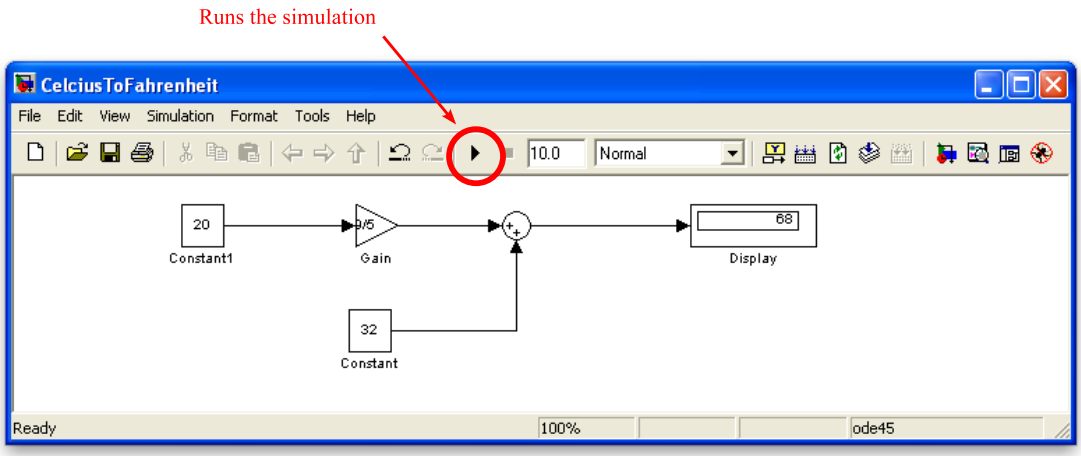

Figure 8: Celsius to Fahrenheit converter with constant input.

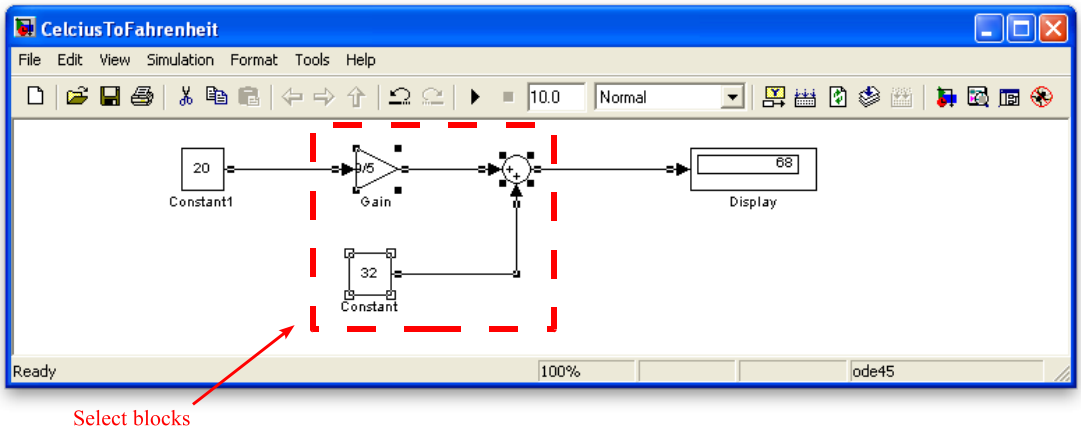

Figure 9: Selecting math portion of Celsius to Fahrenheit converter so that it can be made into a subsystem.

Notice that the math portion of the Celsius to Fahrenheit converter has simply been moved into the subsystem shown in Fig. 11. The only new elements are the input port and the output port required for passing information into and out of the subsystem. By left-clicking on the name of the input port or output port you can change the names of the ports. For instance, in Fig. 12 the input port has been renamed to `C` and the output port to `F`. Similarly, the label under any block can be changed by left-clicking on the label and editing the label. Notice that this relabeling was also done to Fig. 10 to obtain Fig. 13.

Verify again that your simulation model functions correctly and that Celsius is properly converted to Fahrenheit. You may wish to *save* your model again at this point.

Instructor Verification (separate page)

2.3 Simulations over time in SIMULINK

A simulation using only a constant input, such as in the last section, does not fully demonstrate the power of SIMULINK. Simulations are normally run with inputs that vary over time such that the response of the model can be observed over time.

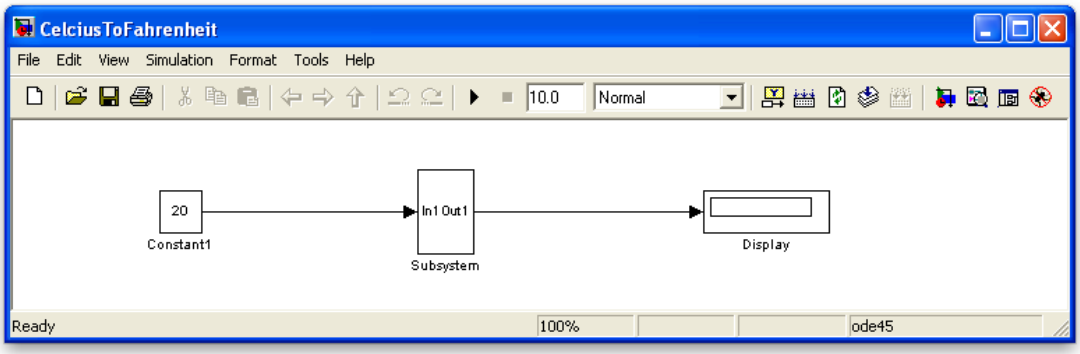

Figure 10: Celsius to Fahrenheit converter grouped as a subsystem.

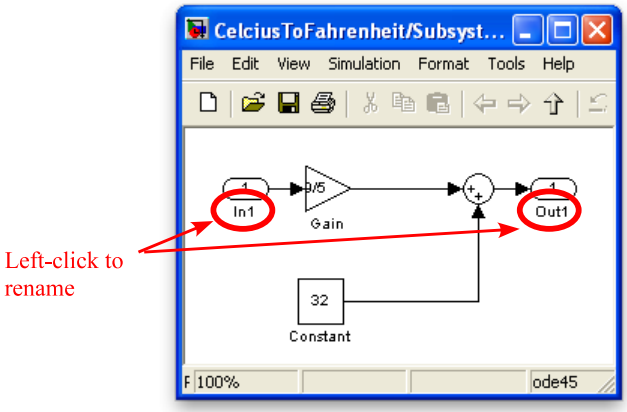

Figure 11: Celsius to Fahrenheit converter subsystem (before relabeling).

2.3.1 Inputting a signal and plotting the output

Let's change the input and output blocks for our model to see how simulations can be done over a certain time frame. First delete the `Constant` source block and the `Display` sink block from the model leaving only the Celsius to Fahrenheit converter subsystem. Next, drag-and-drop the `Signal Generator` block into your model from the `Sources` section of the SIMULINK Library Browser, and the `Scope` block from the `Sinks` section of the SIMULINK Library Browser. Connect these units to the Celsius to Fahrenheit converter as shown in Fig. 14.

The `Signal Generator` block is similar to a function generator you might find in a hardware lab and the `Scope` block is similar to an oscilloscope. Double-click on the `Signal Generator` block to open its block parameters window as shown in Fig. 15. Four different types of signal waveforms can be generated: 1) sine, 2) square, 3) sawtooth, and 4) random. For each signal you can set the amplitude and frequency of the signal. Set the signal generator to be a sine wave with an amplitude of 4 and a frequency of 0.5 Hz.

You can now simulate the model by clicking on the simulation button. The simulation will now use the `Signal Generator`'s sine wave as input for the Celsius to Fahrenheit converter. To view the output of the simulation, double-click on the `Scope` block to bring up a window such as in Fig. 16. For the output plotted for the `Scope` block, the x-axis is the time axis and the y-axis is the signal value feed into the scope at that point in time.

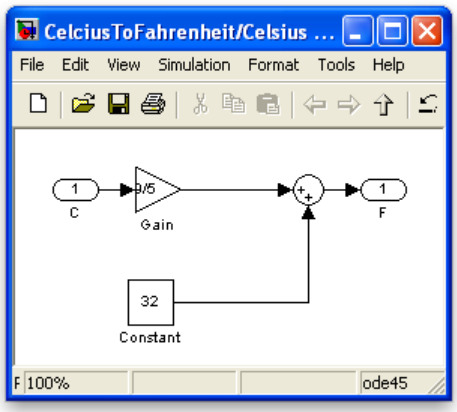

Figure 12: Celsius to Fahrenheit converter subsystem.

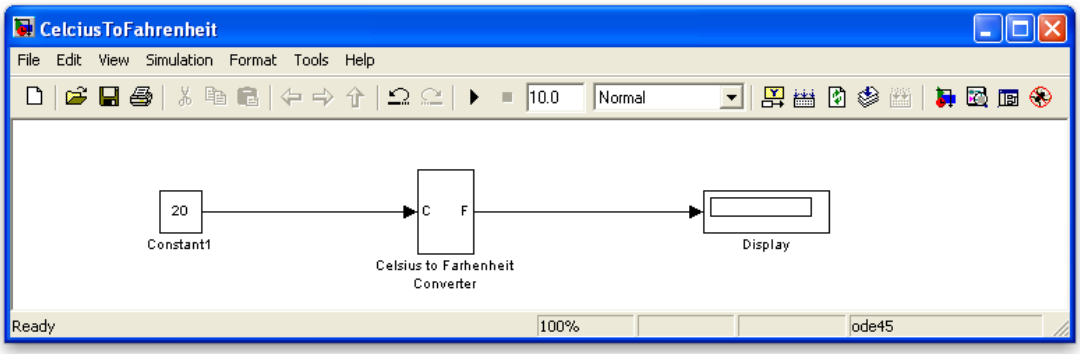

Figure 13: Celsius to Fahrenheit converter with constant input.

2.3.2 Zooming on the `Scope`'s plot and controlling the axes

Often the plot given for the `Scope` is not properly scaled. If you left-click on the binoculars icon, the plot will autoscale in the x- and y-axis to view the entire signal plotted. The plot in Fig. 16 has been autoscaled to show the entire signal. If you need to zoom in on the plotted signal, you can use the magnifying glass icon and then select a boxed zoom region in the plot. You can also use either the x-axis or y-axis magnifying glass icons to zoom in on a portion of the output but only in the x- or y-axis directions, respectively.

Another useful aspect of the `Scope` block's plot is the `Save current axis settings` and `Restore saved axis settings` icons. These are useful if you need to zoom into a particular portion of the plot and wish to come back to that zoomed setting later on. Try zooming into a portion of your plot and then click the `Save current axis settings` icon. Then, click the `Autoscale` icon (the binoculars icon) to zoom back out. Afterwards, click the `Restore saved axis settings` icon which will cause the plot to zoom back to the axis settings the last time you saved them. This is useful when you simulate your system with different settings and wish to monitor a certain zoomed portion of the results.

Finally, by default for the `Scope` window is to only save the last 5000 data points of the simulation. We can turn this limitation off by clicking on the scope parameters button as shown in Fig. 16, and under the data history tab, deselect the limit on the data history as shown in Fig. 17.

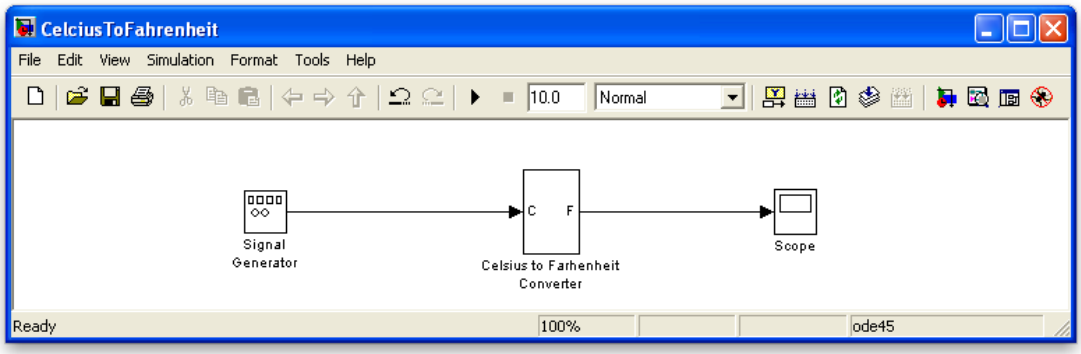

Figure 14: Celsius to Fahrenheit converter with signal generator input.

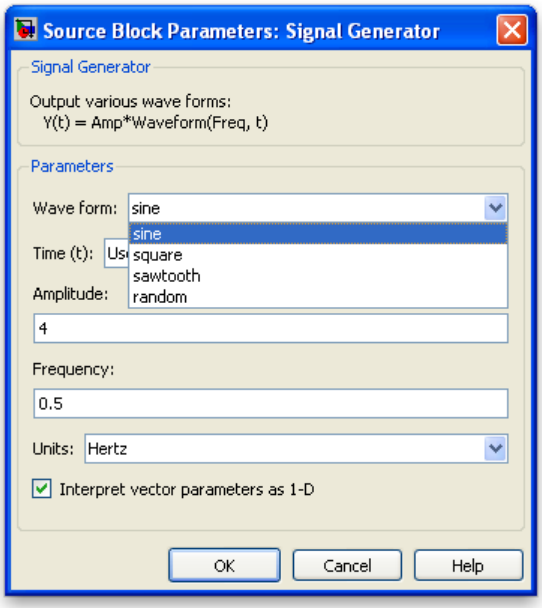

Figure 15: Block parameters for the `Signal Generator` source block.

2.3.3 Changing the start time and stop time of the simulation

The default configuration for the simulation is to simulate from time $t = 0$ seconds to $t = 10$ seconds. To set the start and stop times for the simulation, select

`Simulation→Configuration Parameters...→Solver`

from the menu which will give you the window shown in Fig. 18.

As shown in Fig. 18, you can control the start time and the stop time of your simulation. Set the start time to -1.0 seconds and the stop time to 5.0 seconds and then run the simulation. Verify that the simulation results in the `Scope` show that the simulation time is now only from -1 to 5 seconds. You might have to press the binoculars icon again to autoscale the plot.

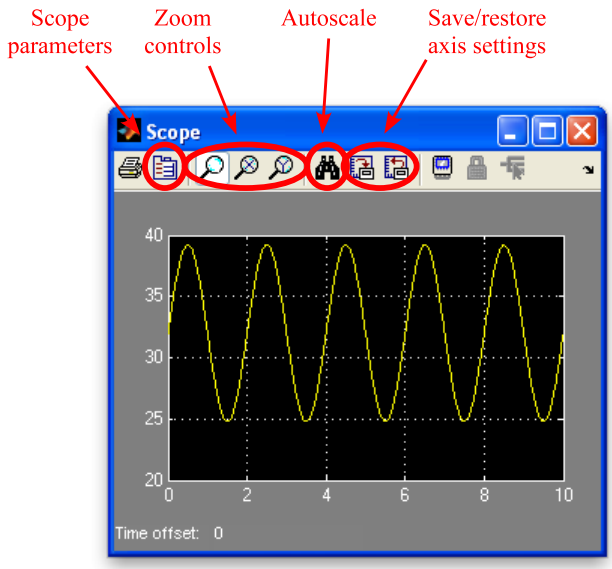

Figure 16: Scope output window.

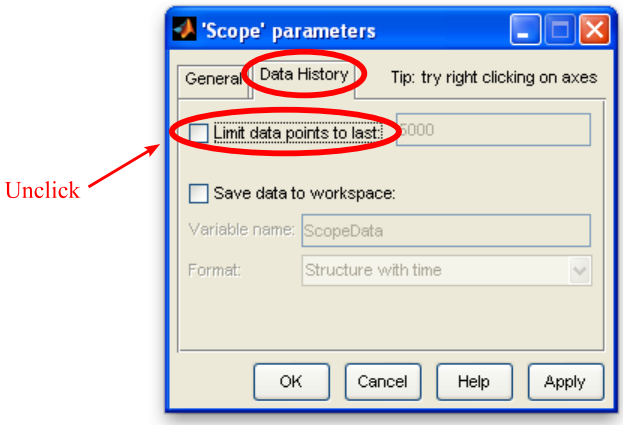

Figure 17: Removing data history limits on the `Scope` block.

2.3.4 Step size and refinement of sampling interval

After a simulation, you may notice that your resulting plots are not as fine detailed as perhaps expected. For instance, even though your input to the Celsius to Fahrenheit converter is a sine wave, the output appears more jagged than a sine wave. When simulating with continuous-time signals, SIMULINK can only calculate the output at a finite set of time instants. As such, simulations with continuous-time signals that move quickly (such as a sine wave) may produce results that don't appear smooth if SIMULINK chooses unsuitable time intervals between calculations.

The simulation parameters shown in Fig. 18 allow some control over how SIMULINK chooses the time intervals. By default, the `Solver option type` is set to `Variable-step` using the `ode45 (Dormand-Prince)` solver. The `ode45` solver is an *ordinary differential equation* (ODE) solver based on the Runge-Kutta algorithm and is appropriate for most simulations. These numerical ODE solver algorithms is past the scope of this course, but effectively the algorithm attempts to base the time interval step

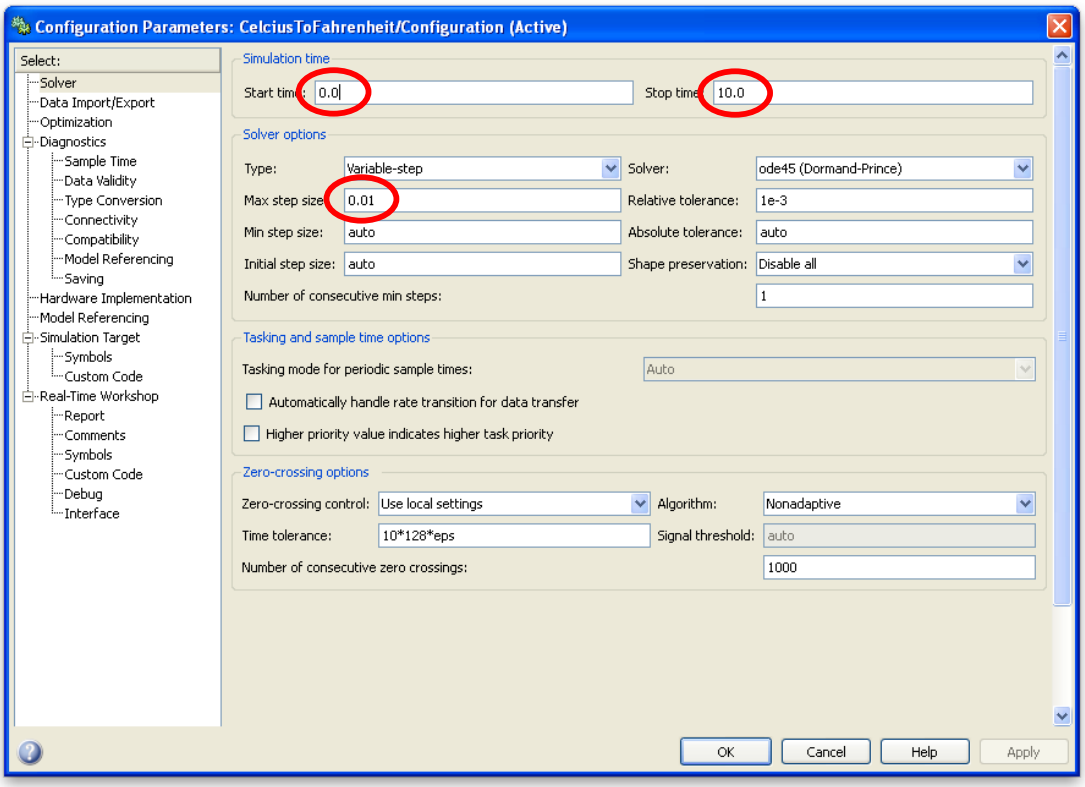

Figure 18: Configuration parameters window.

size according to the rate of change expected by the system.

In general, SIMULINK does not have *a priori* knowledge of the input source into the model (such as the `Signal Generator` block in this case with a sine wave) so some adjustments by the user might be necessary. One useful parameter to set is the `Max step size`. By default the `Max step size` is set to `auto` but you can set this to an appropriately small value to limit the maximum time interval between simulation calculations. Adjusting the `Max step size` appropriately will produce an output of sufficient detail.

A preferred approach of refining the detail in the output is to increase the `Refine factor` as shown in Fig. 19. By default the `Refine factor` is set to 1 and uses SIMULINK's best guess at the required time interval. The value given for the `Refine factor` will increase the level of refinement (decrease the time interval) by that factor. For instance, if you wish to double the refinement factor from SIMULINK's best guess, you can enter 2 for the `Refine factor` parameter. Adjust the `Refine factor` until you obtain a simulation result that appears reasonably smooth such as in Fig. 16.

Instructor Verification (separate page)

2.4 Viewing multiple signals from your simulations

It is often useful to view a number of signals so that they can be compared. For instance, you might wish to monitor the input signal as well as the output signal. One approach is to have multiple `Scope` blocks within your simulation model as shown in Fig. 20. Using this approach, you can independently monitor various signals throughout your simulation model. You can even choose to monitor intermediary points within your

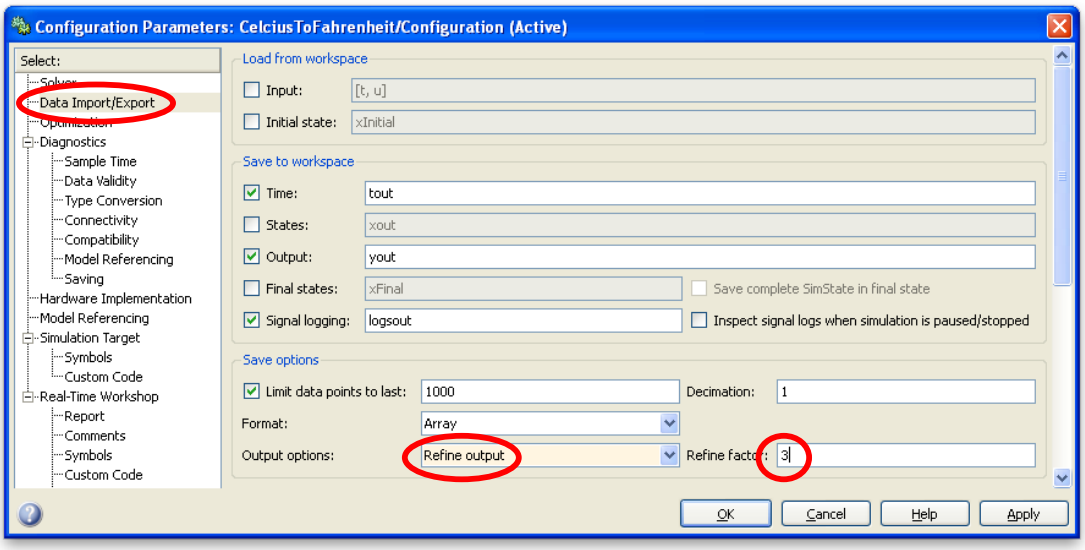

Figure 19: Configuration parameters window — controlling output refinement.

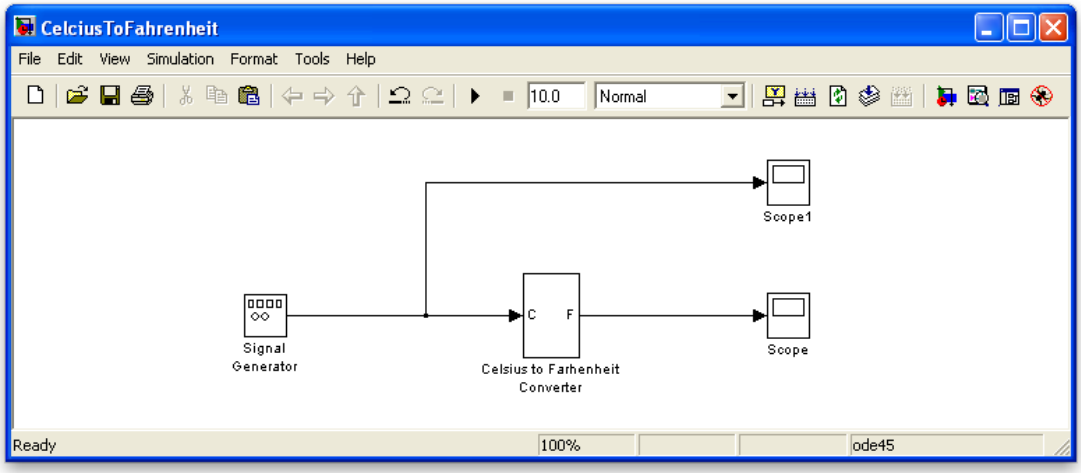

Figure 20: Using multiple `Scope` blocks to view both input and output signals.

model with a few `Scope` blocks.

It is sometimes useful to have the signals on the same plot for the purpose of comparison. For the `Scope` block, multiple signals can be combined together using the `Mux` block which multiplexes the signals into one structure that the `Scope` block will interpret as multiple signals. Each signal is then plotted within the `Scope`'s output window. The `Mux` block is located in the `Signal Routing` section of the SIMULINK Library Browser. Drag-and-drop the `Mux` into your model and double-click on it to open its block parameters, as shown in Fig. 21. From within the block parameters window for the `Mux` block you can set the number of inputs for the `Mux`. In this case we wish to monitor the input and the output, so set the number of inputs to two.

Now connect the model's input and output to the input of the `Mux` block and connect the output of the `Mux` block to the `Scope` block as shown in Fig. 22. When you now simulate the model, both the input and output will appear on the `Scope`'s output window.

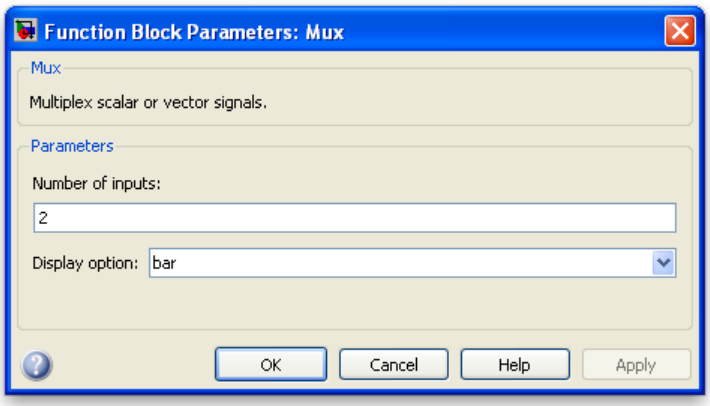

Figure 21: Block parameters for the `Mux` block.

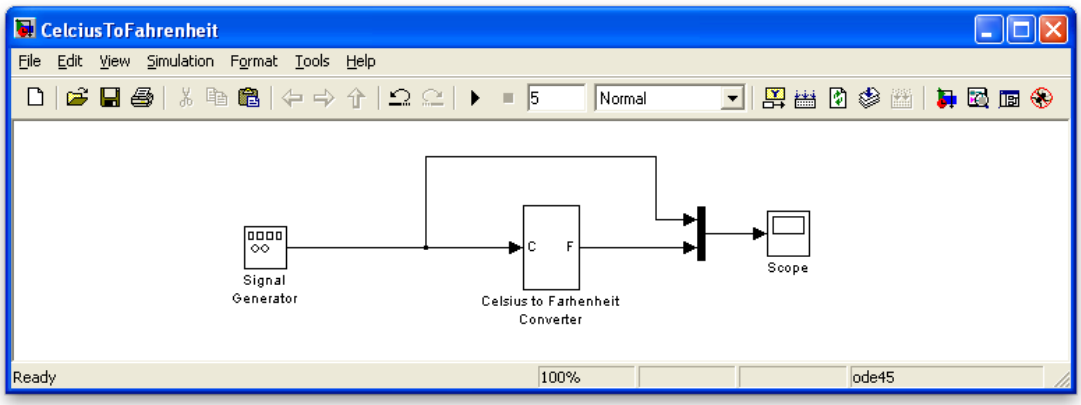

Figure 22: Using `Mux` to view both input and output signals on the same scope.

Rerun your simulation and monitor the input and output on the scope. You should be able to see how the input temperature in Celsius compares to the converted output temperature in Fahrenheit. Make sure you can obtain a smooth plot like that shown in Fig. 23.

2.5 Sharing data with MATLAB

It is often useful to share data/variables between SIMULINK and MATLAB. This sharing of data/variables might be used for configuring settings within your simulation model, to input other types of signals into your SIMULINK model, or to save output signals for further processing/plotting with MATLAB's more versatile plotting functions.

2.5.1 Using variables from MATLAB within SIMULINK

A useful feature of SIMULINK is that expressions can often be used for some block parameters instead of simply a constant value. We saw in the Celsius to Fahrenheit converter subsystem in Fig. 12 that the constant 32 and the expression $9/5$ were used when realizing Eq. 1.

Another approach of realizing Eq. 1 would be to define some variables with MATLAB that could then be used within the Celsius to Fahrenheit subsystem. Using the MATLAB `Command Window` (see Fig. 1),

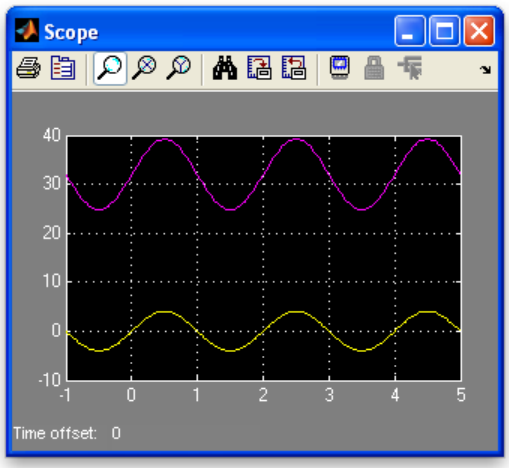

Figure 23: Scope output window with multiple traces.

enter in the following commands.

```
scale=9/5
offset=32
```

These two commands will create the variables `scale` and `offset` for the Celsius to Fahrenheit converter. To see which variables are currently defined within MATLAB, type

```
whos
```

Note that there may be other variables already defined. You can remove all of the variables in MATLAB by typing

```
clear
```

but make sure you redefine the `scale` and `offset` variables before continuing this subsection.

With the `scale` and `offset` variables defined in MATLAB, you can use these variables within blocks such as the `Gain` block and the `Constant` block. Open the block parameters for the `Gain` block and replace the 9/5 expression with the `scale` variable name, and similarly change the 32 value in the `Constant` block to the `offset` variable. You should have a subsystem that looks like Fig. 24. Note that if the block is too small then the variable name may not show. You can always click on the block and resize it by dragging the corner anchors to make it large enough for the variable name to display.

An advantage of using MATLAB variables in this manner is that you can easily change the MATLAB variable to change the values used within the SIMULINK model. For instance, if we wished to convert from Celsius to Kelvin instead of to Farhenheit, we would need to realize the following equation.

$$°\mathrm{K} = °\mathrm{C} + 273.15 \tag{2}$$

We could use the subsystem given in Fig. 24 to calculate Eq. 2 if we set the MATLAB variables to

```
scale=1
offset=273.15
```

Using MATLAB variables can allow the design of SIMULINK models with more control over its configuration.

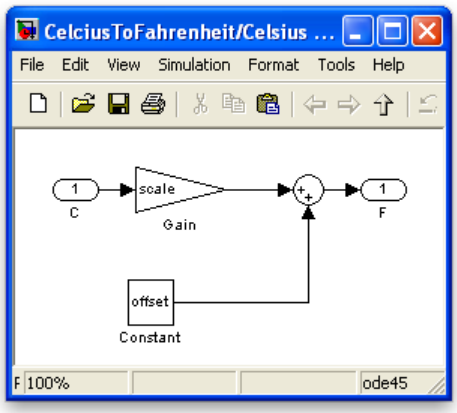

Figure 24: Celsius to Fahrenheit converter subsystem using variables from MATLAB.

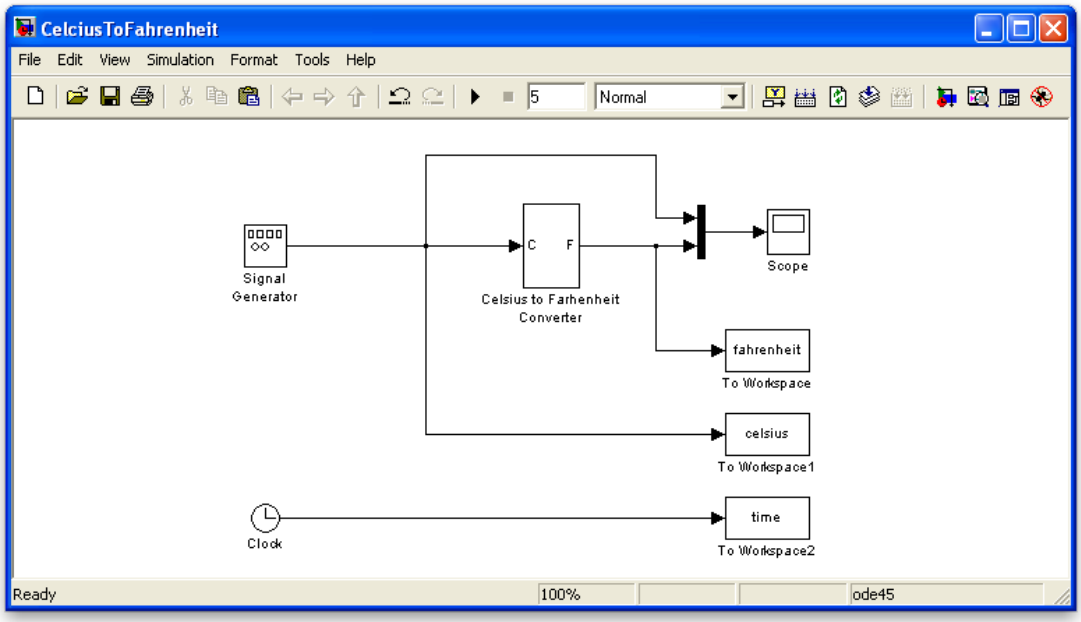

Figure 25: Simulation model with signals sent to the MATLAB workspace.

2.5.2 Outputting data to MATLAB

To export signals to use in MATLAB's variable workspace environment, use the `To Workspace` block from the `Sinks` section of the SIMULINK Library Browser. For instance, you can use the `To Workspace` block as shown in Fig. 25 to share the signal from the `Signal Generator` (*i.e.*, the input temperature in degrees Celsius) and the output of the Celsius to Fahrenheit converter.

Drag-and-drop two `To Workspace` blocks into your model and connect them to the Celsius and Fahrenheit signals as shown in Fig. 25. To configure the `To Workspace` blocks, double-click on them which will open a window similar to that shown in Fig. 26. Within the `To Workspace` block parameters window you can set the variable name that will be assigned within MATLAB for the simulation data passed to the `To Workspace` block. For instance, the temperature signal in Celsius is set to the variable `celsius` and the temperature signal converted to Fahrenheit is set to the variable `fahrenheit` in Fig. 25.

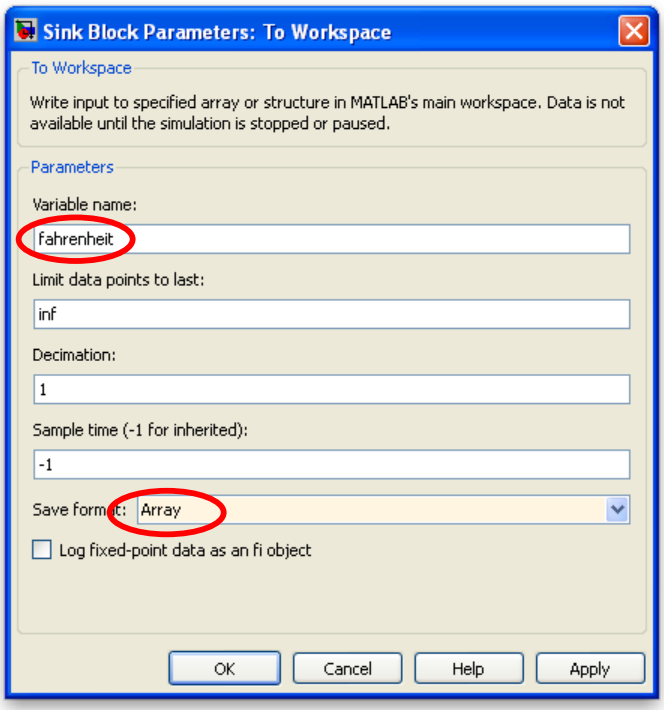

Figure 26: Block parameters for the `To Workspace` block.

The default `Save format` for the `To Workspace` block parameters is as a `Structure`. While this setting can be used, it requires using the more complicated structure syntax in MATLAB. For our purpose, set the `Save format` to the straightforward `Array` format. A signal then input to that `To Workspace` block will be passed to MATLAB as a simple array of numbers.

Notice that the simulation model in Fig. 25 also uses the `Clock` block from the `Sources` section of the SIMULINK Library Browser and that the output of the `Clock` block is being passed to MATLAB through another `To Workspace` block. The `Clock` block outputs the time steps used in the SIMULINK simulation run. It is important to pass this information to MATLAB, since otherwise data such as the temperatures in Celsius or Fahrenheit passed to MATLAB have no corresponding time frame of when the data samples occurred. Since the time step interval between data samples might not be uniform (recall the `Variable-step` setting in Fig. 18), this time information must also be passed along with any simulation results. Setup the `Clock` block and connecting `To Workspace` block in your simulation model.

Now you can run your simulation and have the signals passed to MATLAB. Before running the simulation, be sure that all of your `To Workspace` blocks have the `Save format` set to `Array`. After running the simulation, you can now use commands from within MATLAB to process/plot your data. Using the MATLAB `Command Window` (see Fig. 1), enter in the following command.

```
whos
```

The `whos` command lists the current variables within the MATLAB variable environment. You should see variables for `celsius`, `fahrenheit`, and `time` as passed to MATLAB through the `To Workspace` blocks once you have run your simulation.

With the temperature and time variables passed to MATLAB, you can run the following commands to plot the data.

```
plot(time,celsius);
hold on;
plot(time,fahrenheit);
hold off;
title('Plot of temperature over time');
xlabel('Time');
ylabel('Temperature')
```

If you wish further help on any of the MATLAB commands, you can type

```
help command
```

to get details on the use of `command`. For instance, typing

```
help plot
```

will give you help on the `plot()` function in MATLAB.

2.5.3 Inputting data into SIMULINK

In a similar way that data can be passed from SIMULINK to MATLAB, data can also be passed from MATLAB into SIMULINK. For the Celsius to Fahrenheit converter, we could generate any type of input. The input data with MATLAB must be setup as an $L \times 2$ matrix. The first column of the matrix must be an $L \times 1$ vector containing the time reference of when the data sample was taken. The second column of the matrix is an $L \times 1$ vector containing the data samples at the corresponding sample times.

For instance, we could generate an $L \times 2$ matrix of temperature data as follows. This temperature data is just a sine wave that varies the temperature from -1 °C to $+1$ °C.

```
sindata(:,1) = 0:0.01:10;          % Times for each sample from
                                   % 0 to 10 in increments of 0.01
sindata(:,2) = sin(sindata(:,1)); % Temperature at time values
plot(sindata(:,1), sindata(:,2))
```

To make the data a little more interesting, let's load temperature data as measured from Day 2 to Day 7 on the Mars Pathfinder mission. Download the file `mpf.dat` and place it in your current MATLAB working directory (check the top of your MATLAB window). Then, type

```
load mpf.mat
```

in MATLAB's `Command Window`, which will load the variable `mpfdata` stored in the file. You will note that `mpfdata` is a 234×2 matrix. We can use this data as input into our Celsius to Fahrenheit converter. To check the variables created in the MATLAB environment, type

```
whos
```

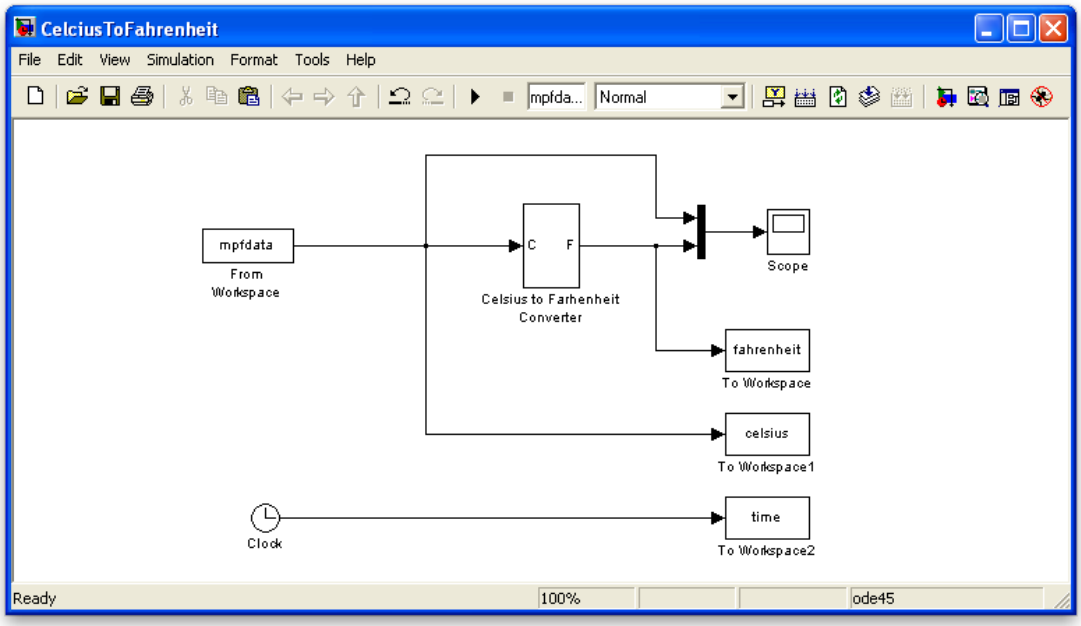

Figure 27: Simulation model with the input taken from the MATLAB workspace variable `mpfdata`.

With the temperature data setup in MATLAB, you can now setup your SIMULINK model to use this data as input data for the Celsius to Fahrenheit converter. In your model, delete the `Signal Generator` block and replace it with the `From Workspace` block as shown in Fig. 27 from the `Sources` section of the SIMULINK Library Browser.

Before simulating, you must first configure the `From Workspace` block to use the data from the `mpfdata` matrix. Double-click on the `From Workspace` block to open its block parameters window as shown in Fig. 28. For the `Data` parameter, enter the MATLAB variable name `mpfdata` and click `Ok`.

Next, we should note that `mpfdata(1,1)` gives the time for the first data sample and `mpfdata(234,1)`, or similarly `mpfdata(end,1)`, gives the time for the last data sample. Set the start and stop times for the simulation through

`Simulation→Configuration Parameters...→Solver`

as shown in Fig. 29. You can check these start and end time values (in days) in MATLAB's `Command Window` using

```
mpfdata(1,1)
mpfdata(234,1)
```

With the data source set, you can now simulate the model and see the temperature at the Mars Pathfinder landing site in both degrees Celsius and Fahrenheit. You should obtain results as shown in Fig. 30.

Instructor Verification (separate page)

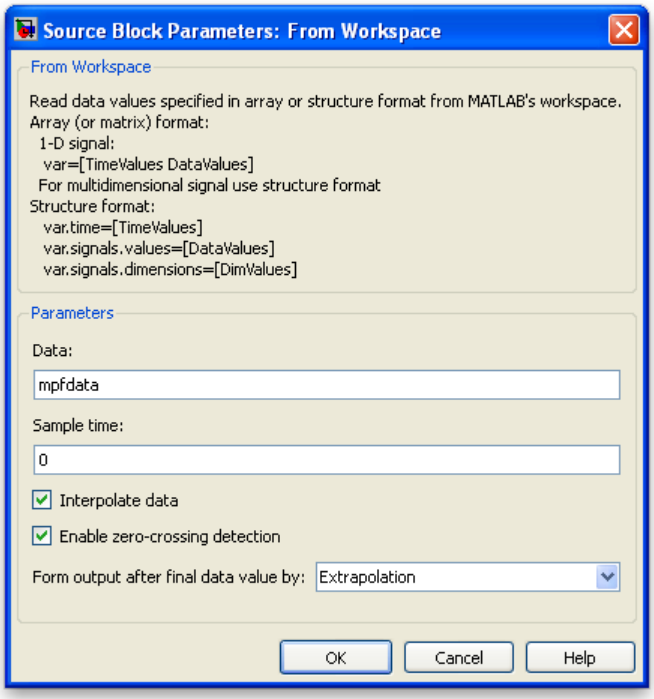

Figure 28: Block parameters for the `From Workspace` block.

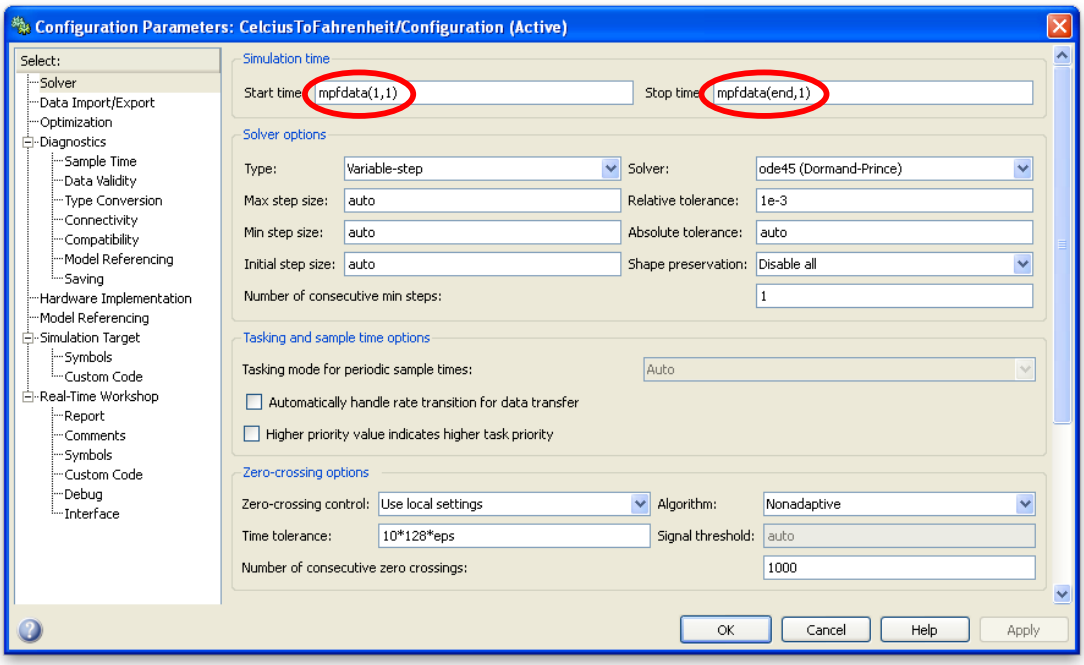

Figure 29: Configuration parameters window — setting start/end times for Mars PathFinder data.

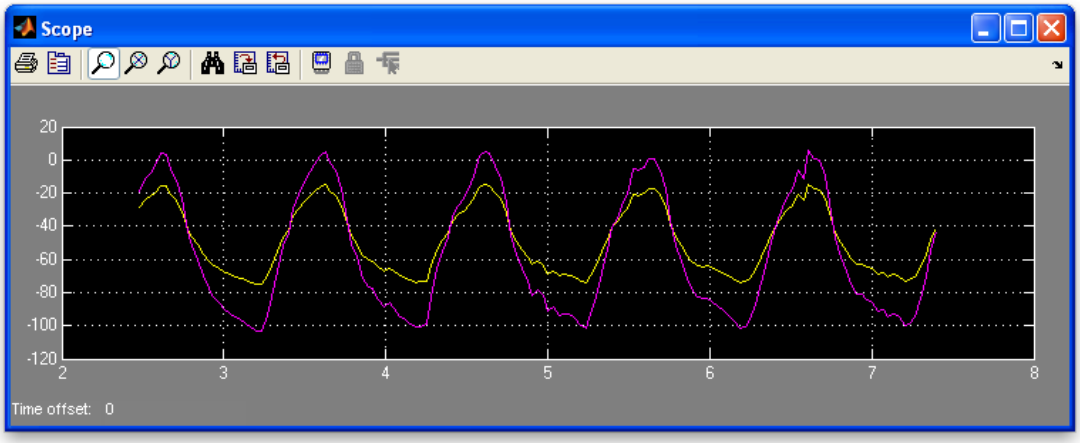

Figure 30: Mars Pathfinder temperature readings converted from Celsius to Fahrenheit.

Lab #1
SYSC 3600
Instructor Verification Sheet

Ideally, you should be able to complete this verification during the lab session. If circumstances prevent completion, this page must be submitted to the lab instructor by the *beginning* of your next lab period.

Name: _____ Student ID: _____

Sec. 2.3: Demonstrate a simulation over time in SIMULINK of the Celsius to Fahrenheit converter with a *refined* output:

Verified:_____ Date/Time:_____

Sec. 2.5.3: Demonstrate simulating the Celsius to Fahrenheit SIMULINK model with the Mars Pathfinder temperature data, showing both the Celsius and Fahrenheit signals in one plot.

Verified:_____ Date/Time:_____