Christopher Lum
lum@u.washington.edu

Simulink Tutorial

Introduction

This document is designed to act as a tutorial for an individual who has had no prior experience with Simulink. It is assumed that the reader has already read through the Beginner and Intermediate MATLAB Tutorials. For any questions or concerns, please contact

Christopher Lum
lum@u.washington.edu

Starting the Program

1. Start MATLAB.

2. Simulink is an extra toolbox that runs on top of MATLAB. To start this, type "simulink" in the Command Window or click on the Simulink icon.

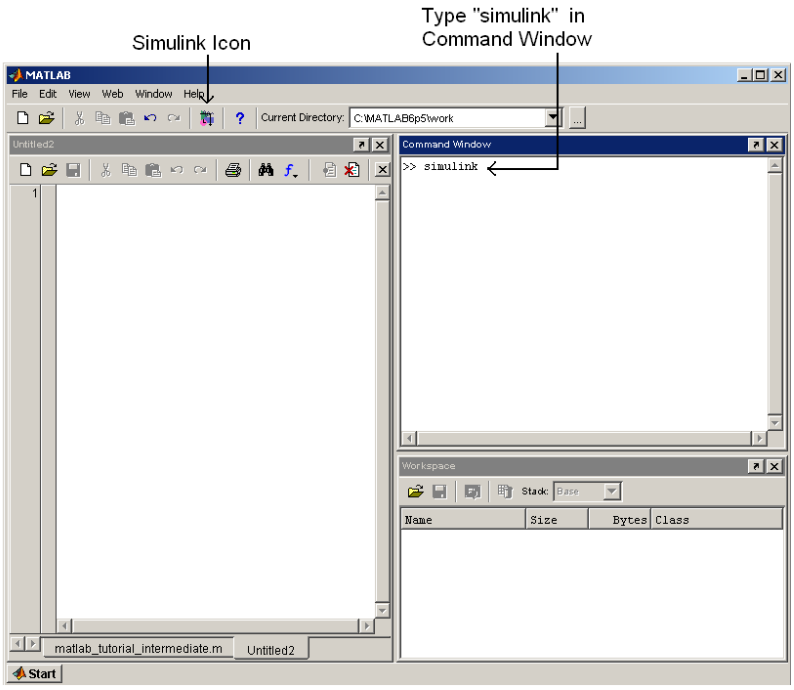


Figure 1: Starting Simulink using icon or Command Window

3. The Simulink interface should now appear as shown below in Figure 2.

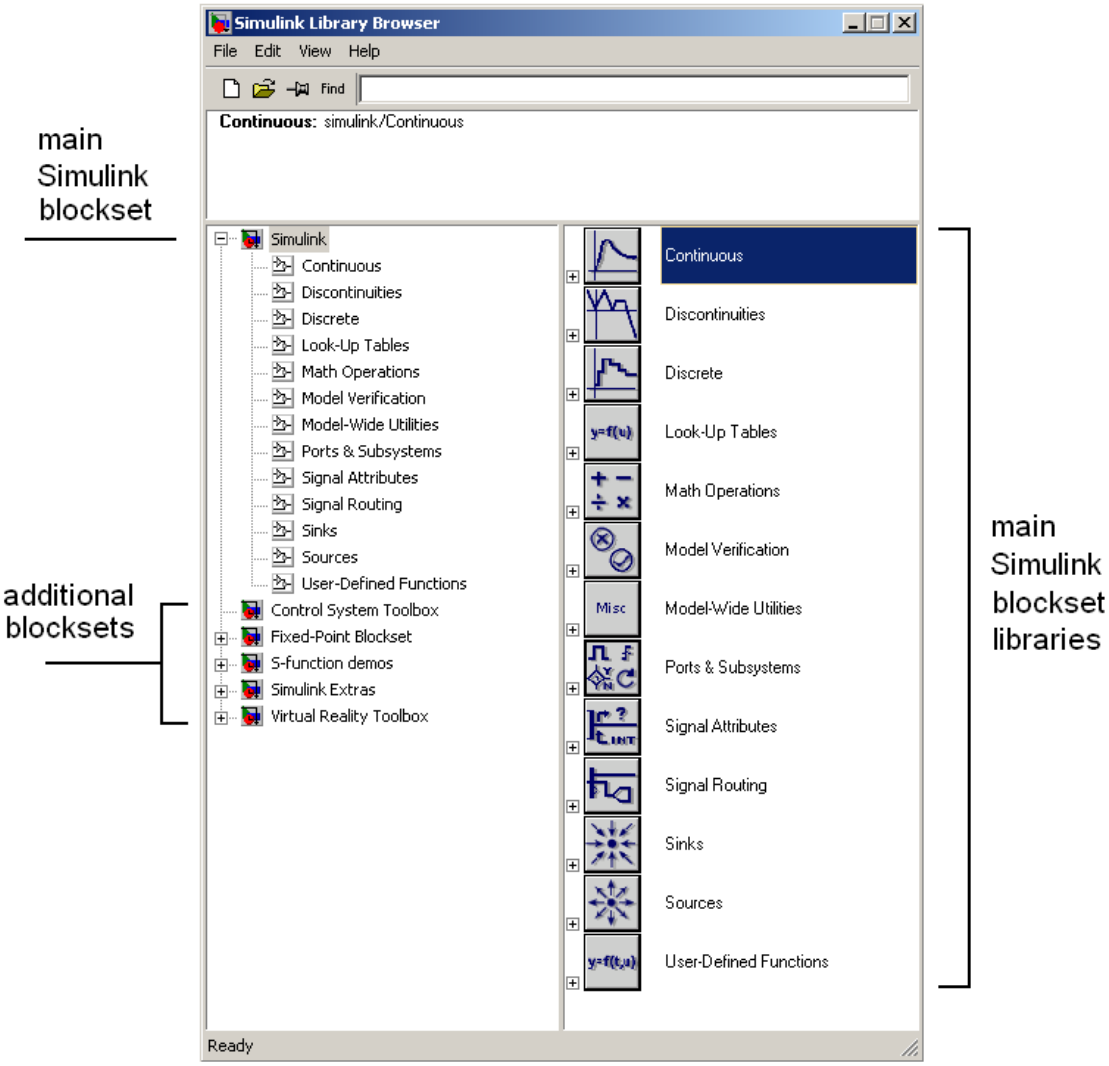


Figure 2: Simulink interface

4. Start a new Simulink model using File > New > Model

METHOD 1: 2nd Order Ordinary Differential Equation

5. Let's use Simulink to simulate the response of the Mass/Spring/Damper system described in Intermediate MATLAB Tutorial document. Recall that the second order differential equation which governs the system is given by

$$\ddot{z}(t) = \frac{1}{m}u(t) - \frac{k}{m}z(t) - \frac{c}{m}\dot{z}(t) \qquad \textbf{Equation 1}$$

Let us assume that we have signals which represent $u(t)$, $z(t)$, and $\dot{z}(t)$, we can multiply them by a product of constants, then sum then in a specific fashion to obtain the signal $\ddot{z}(t)$. Start a new model.

Double click on the "Continuous" library from the main Simulink Blockset. This opens library.

6. We will now construct the signal $\ddot{z}(t)$. The first thing we need is an integrator. Find this block in the 'Continuous' section and drag two of them into your blank model. You can rename them if you like. Connect them as shown in

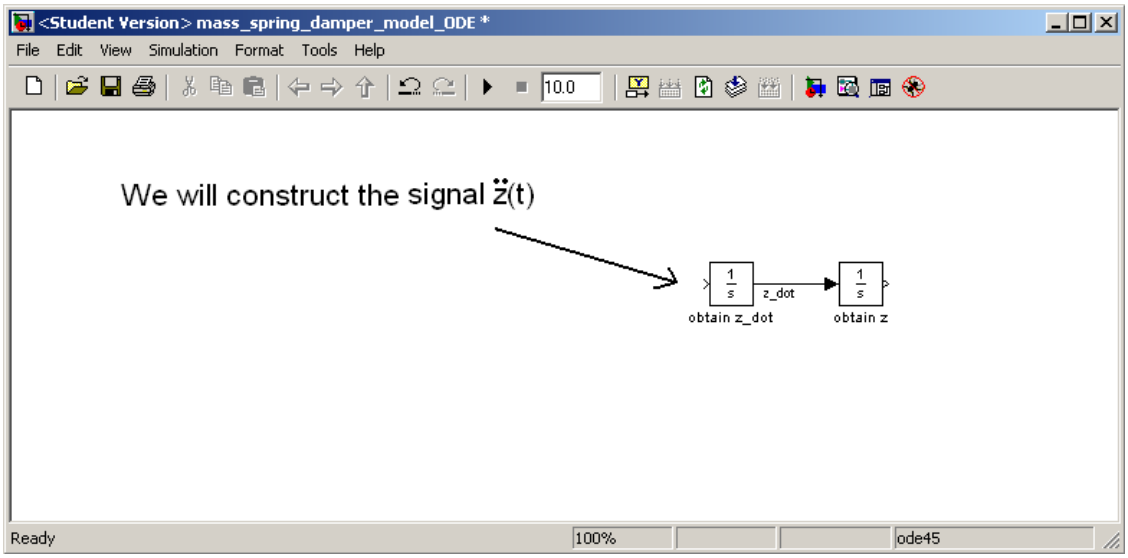


Figure 3: Model with two integrators

7. Looking at the ODE, we see that we also need the signal $u(t)$. This is an external input, or a source. Let's use a step function which can be found in the 'sources' group. Drag this into your model.

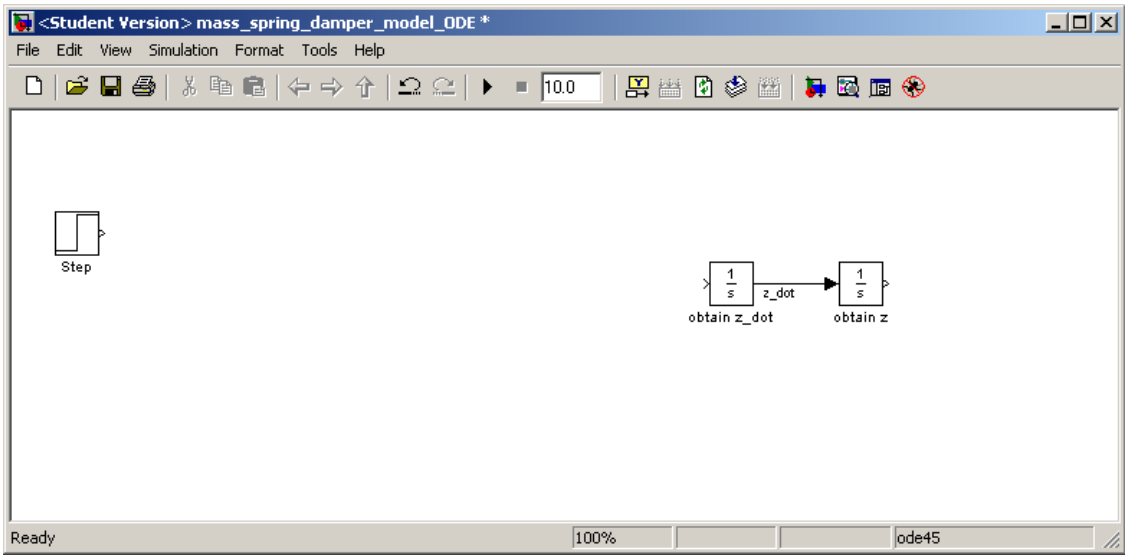


Figure 4: Model with two integrators and a step function

8. We now have the signals $u(t)$, $z(t)$, and $\dot{z}(t)$, we need to multiply these by the values $1/m$, k/m, and c/m, respectively. For this problem, let $k = 2$, $c = 0.5$, and $m = 1$. We can multiply the signals using a 'gain' block which is found in the 'Math Operations' group. Draw three 'gain' blocks into your model and connect them appropriately. (Hint: You can flip the orientation of the block by right click > Format > Flip Block).

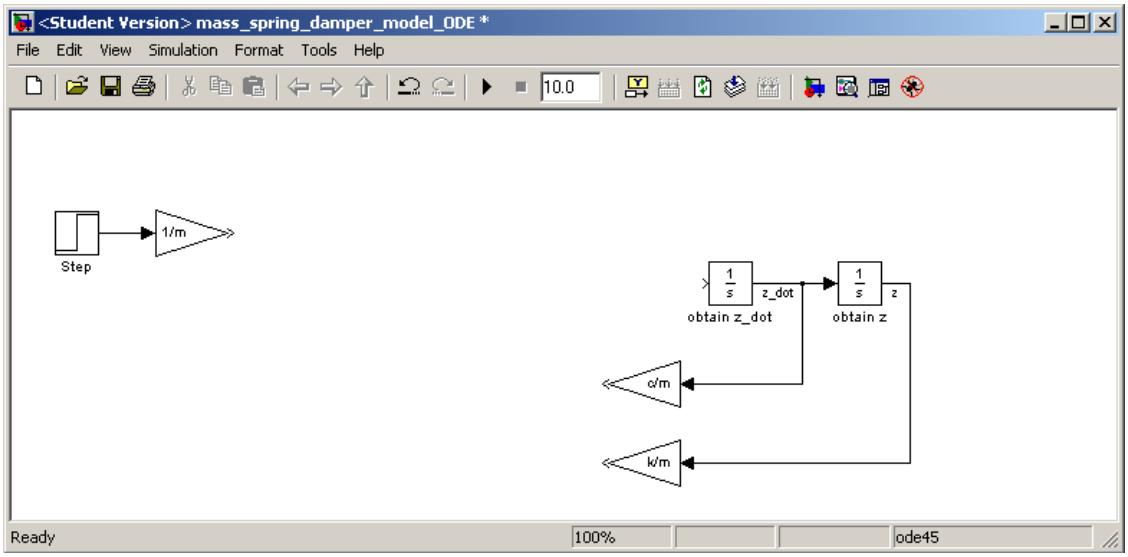


Figure 5: Model with integrators, step function, and gain blocks.

9. We can modify the multiplication factor of each 'Gain' block by double clicking and modify the parameters. Since this is scalar multiplication, ensure that the 'Multiplication' field is set to 'Element-wise (K.*u)' as shown in Figure 6.

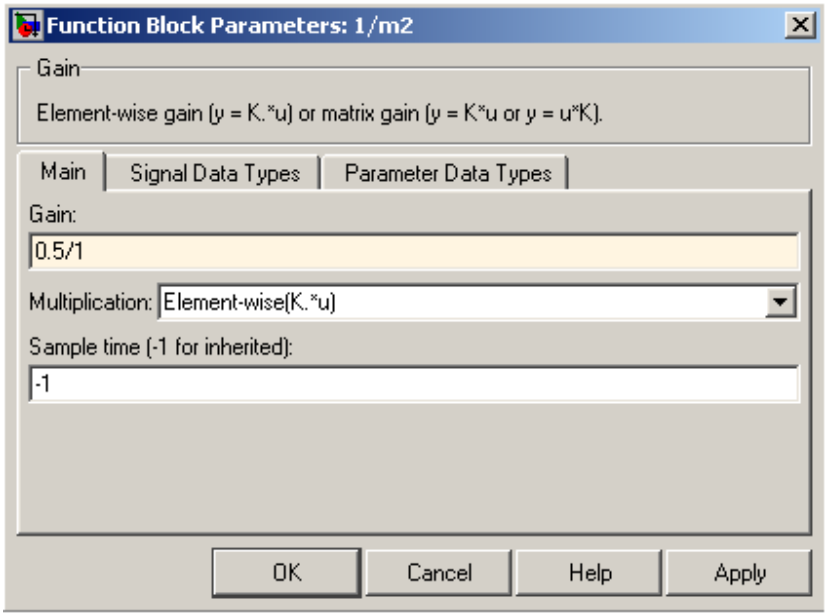


Figure 6: Parameters for the 'Gain' blocks.

10. The last thing we need to do is to add all the signals together using 'Sum' block. Drag one into your model and modify the parameters as shown in **Figure 7**

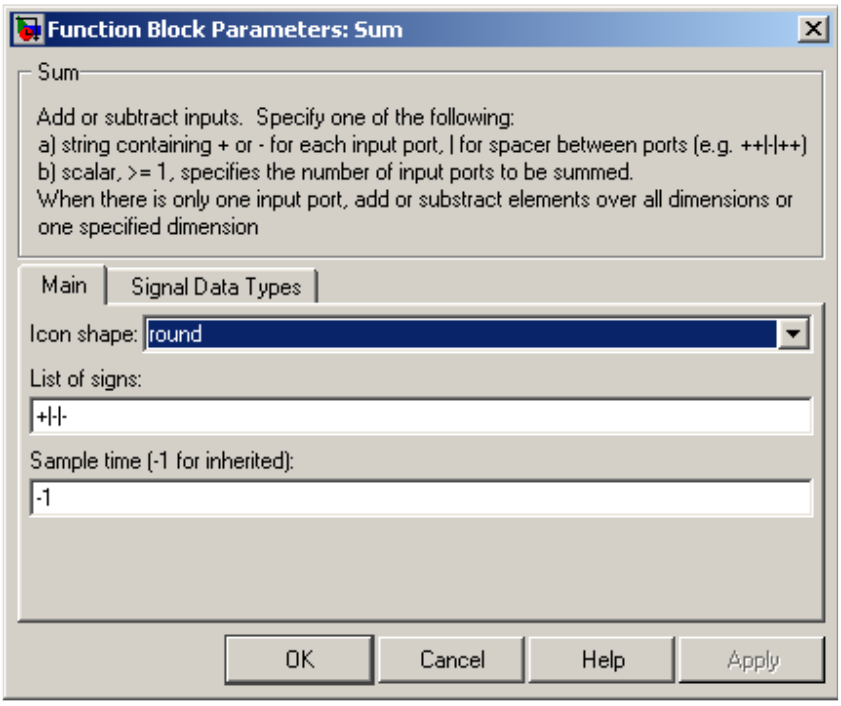


Figure 7: Parameters for the 'Sum' block

11. Connect the signals appropriately.

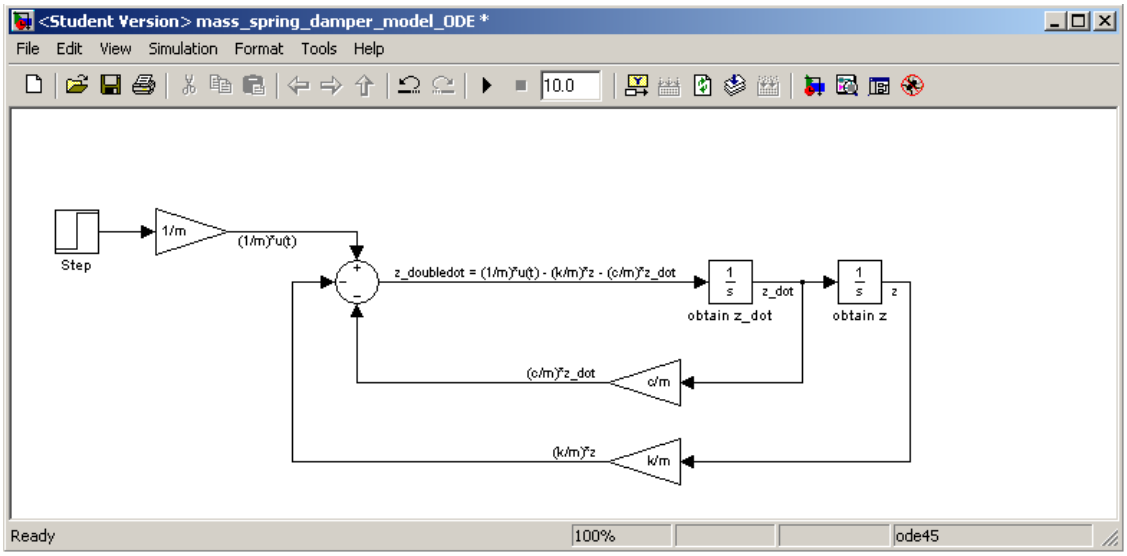


Figure 8: Model with signals connected.

12. At this point, the model accurately solves the ordinary differential equation. However, we would like to be able to see the inputs and outputs. In other words, we would like to monitor the signals $u(t)$ and $z(t)$. We can do this using a 'Scope' which is in the 'Sinks' group. Drag these into your model and connect them appropriately. (Hint: You can split a signal by holding down the right mouse button on the existing signal and dragging).

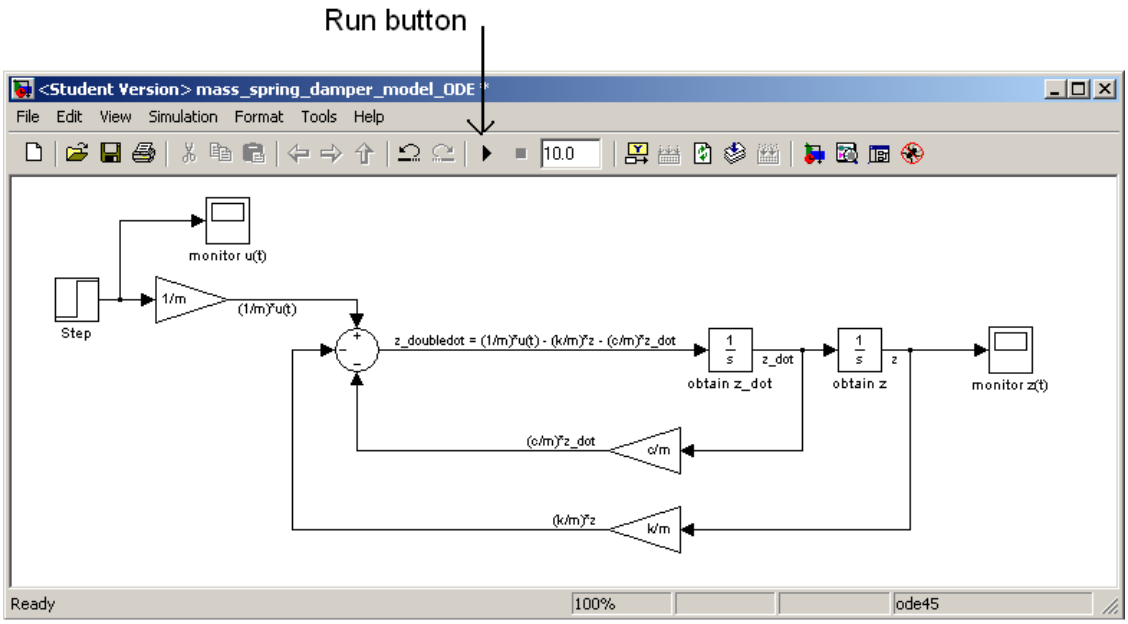


Figure 9: Final model.

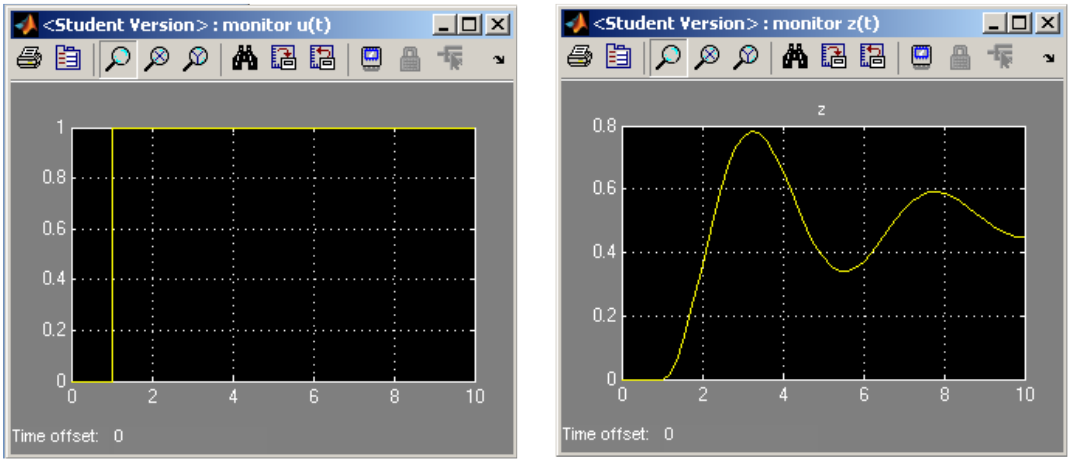
13. Save the model as 'mass_spring_damper_model_ODE.mdl".

14. We can now simulate the system. This can be done many ways as listed below

    a. Click on the "Run" button
    b. Go Simulation > Start
    c. Using the "sim" command from the Command window.

    For now, simply use option a or b, we will visit using option c later.

15. We would like to look at the response of the system using the scope. Double click on the scope block to open it up. Autoscale the plot so that you can see the response (the autoscale button looks like a pair of binoculars). You should see something similar to Figure 10.



a) u(t) signal

b) z(t) signal

Figure 10: Output from scope outputs

METHOD 2: State Space Representation

Recall that in addition to using a second order ODE to model the system, we can use a state space representation of this system of

$$\dot{\bar{x}} = A\bar{x} + Bu \qquad \textbf{Equation 2}$$

$$y = C\bar{x} + Du$$

$$\text{where} \quad A = \begin{pmatrix} 0 & 1 \\ -k/m & -c/m \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ 1/m \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad D = 0$$

Once again, let $k = 2$, $c = 0.5$, and $m = 1$.

16. Start a new, blank model. Click on the "State-Space" block and drag this into your blank model. Your model should now look like Figure 11.

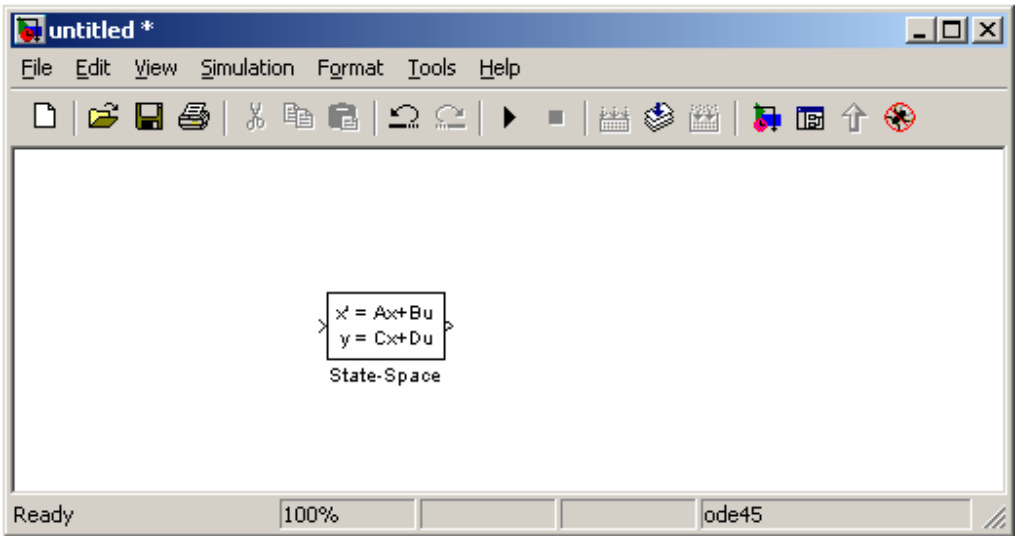


Figure 11: Simulink model with just state space block added

17. We now need to define the parameters of this block. Double click on the block to enter the parameters.

    Enter in the A, B, C, and D matrices. Leave the initial conditions as 0. (Note that this is the initial state vector, $\bar{x}(0)$ and since there are 2 states, 0 actually implies $\bar{x}(0) = (0 \quad 0)^T$)

18. Now, let's subject this system to a unit step input which occurs at t = 1 second. Click on "Sources" in the Simulink interface and find the "Step" block. Drag this into the model and connect the output of the step to the input of the state space model (this can be done by clicking on the Step then holding Ctrl and then clicking on the state-space block).

19. We would like to be able to view to output of the system so Click on "Sinks" in the Simulink interface and find the "Scope" block. Drag this into the model and connect the output of the state-space block to the input of the sink. Your simulink model should now look like

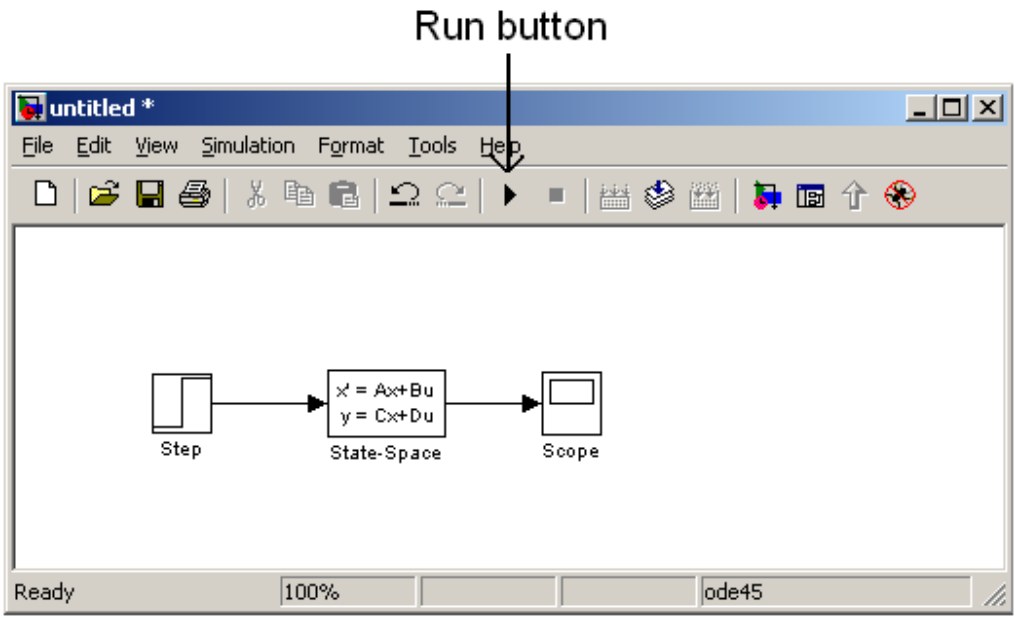


Figure 12: Simulink model with source and sink

20. Save the model as “mass_spring_damper_model.mdl”

21. We would like to look at the response of the system using the scope. Double click on the scope block to open it up. Autoscale the plot so that you can see the response (the autoscale button looks like a pair of binoculars). You should see something similar to Figure 13.

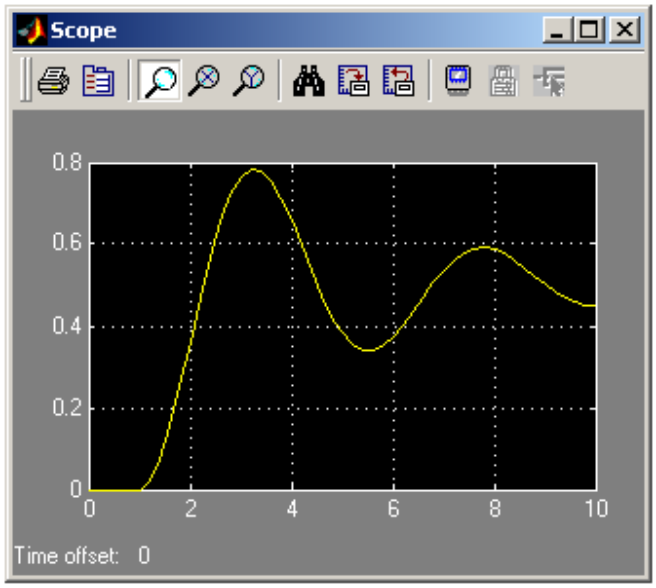


Figure 13: Simulated response of system from scope block

22. Although this is nice for simple analysis, we would like to interface this with Matlab so we can analyze the data using Matlab functions.

    Let's analyze how the system response changes if we use different damping coefficients of c = 0.5, 1.0 and 1.5.

    This would be very tedious if we had to change the A matrix each time by hand and then simulate the system and then look at the plot. Therefore, we will use the m-file to write a script which will do this for us.

23. Start a new m-file.

24. Let's first analyze the system response when c = 0.5. Define the A, B, C, and D matrices in the m-file. A sample code is shown below

```
%-------------------------Part 15--------------------------------
%Define the constants
k = 2;
m = 1;
c = 0.5;

%Enter the A, B, C, and D matrices
A = [0 1;
    -k/m -c/m];

B = [0;
     1/m];

C = [1 0];

D = [0];
```

25. We can actually use variables in all the simulink blocks provided that they are defined in the Workspace before the model is run. Now change the parameters of the State-space block to match the matrices that you defined in the m-file. The state space block should like similar to

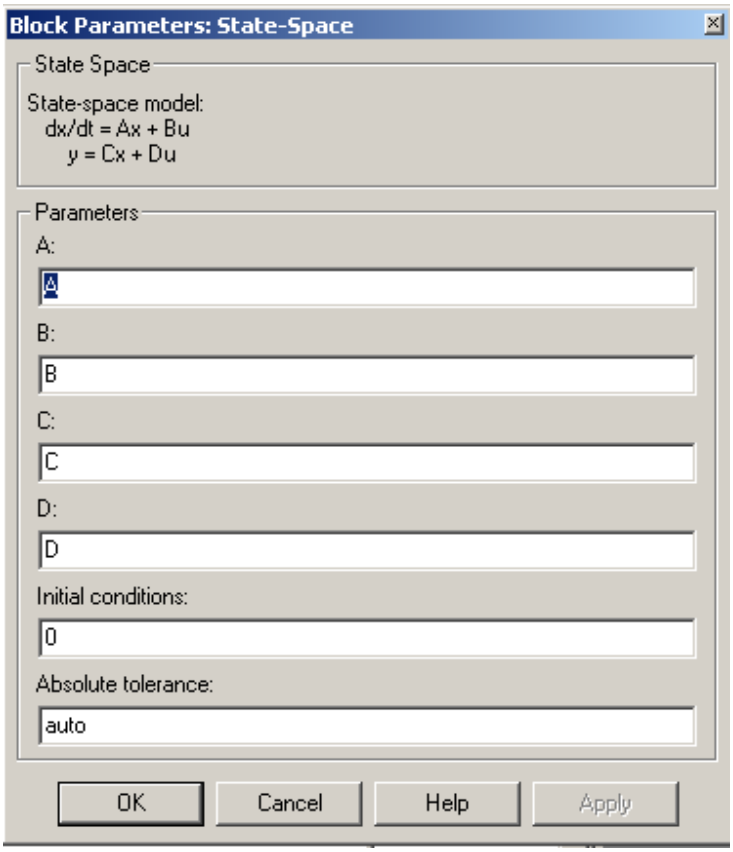


Figure 14: State space block parameters using variables as parameters

26. We need to export the data from simulink to Matlab so that we can plot it. Namely, we would like to see both the input and output of the system. To do this, we use the "To Workspace" block which can be found in the Sinks library. Drag 2 of these blocks into your model and connect them to the input and output (Note: to make a branch of a signal, right click on the signal and then drag to the second connection)

27. We need to modify the parameters of these two blocks slightly. The appropriate parameters are shown below

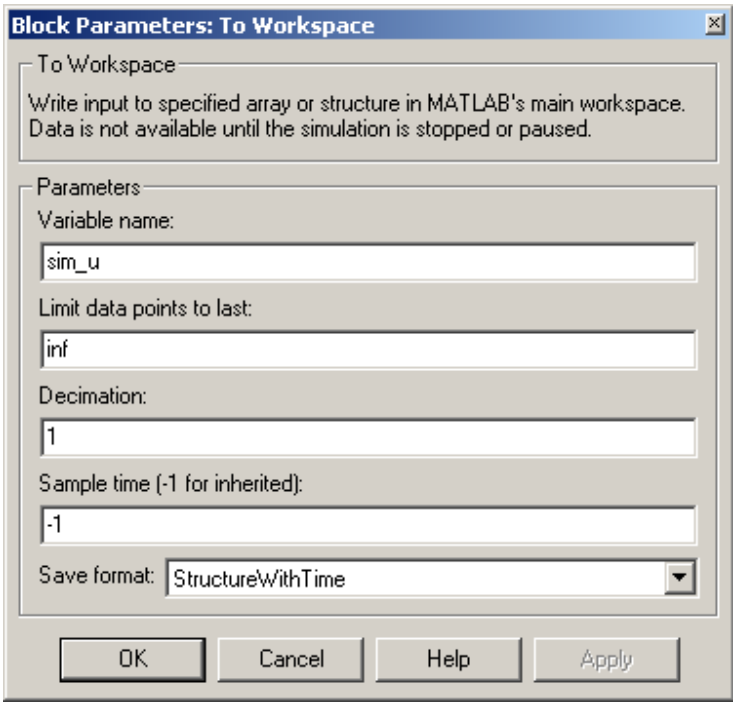


Figure 15: Parameters for the "To Workspace" block

Your simulink model should now look like

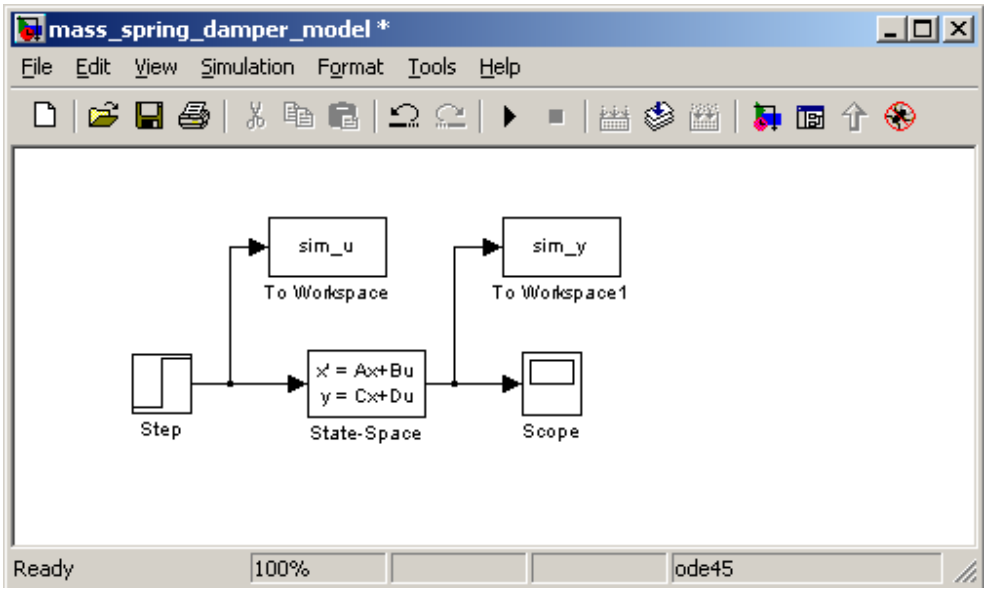


Figure 16: Simulink model with "To Workspace" blocks

28. The simulink model is now setup to export the data to the workspace. When the model is run, three variables will be created in the workspace. They are

Variable Name	Class	Comments
sim_u	struct array	Created by 1st "To Workspace" block
sim_y	struct array	Created by 2nd "To Workspace" block
tout	double array	Automatically created by Simulink when you run a model

29. Write code to run the model from the m-file using the "sim" command. Also, write code to extract the data (namely the input and output of the model). A sample code is shown below.

Functions:	*sim*
Note:	*For our purposes, use sim with only 1 argument, the name of the model which you are trying to run*

```
%Let's run the model from the m-file using the "sim" command
sim('mass_spring_damper_model');

%Now extract the data
t = sim_u.time;
u = sim_u.signals.values;
y = sim_y.signals.values;
```

30. Find the maximum response of the system. That is, find max(y). A sample code is shown below

Functions:	*max, find,*
Note:	*Here is an example of how to use these functions*

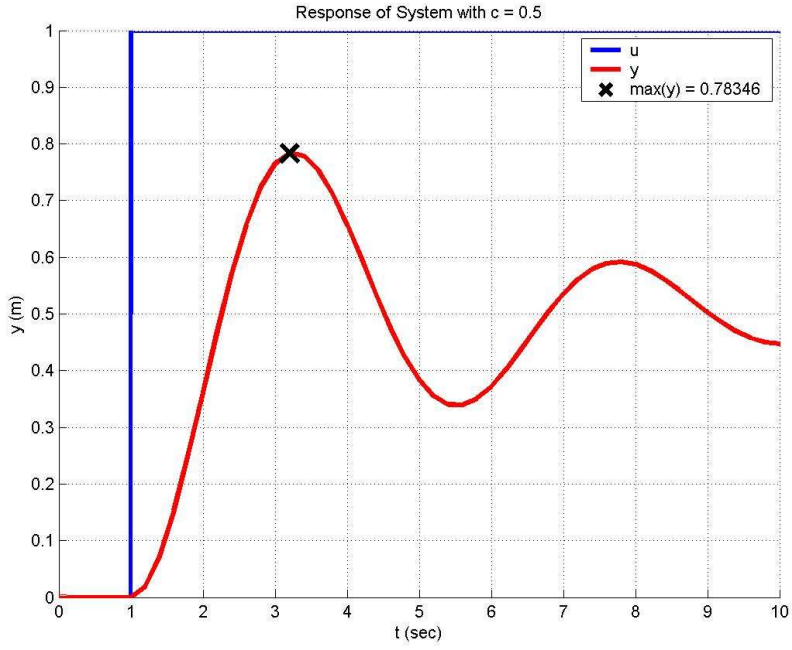
```
%Let's find the maximum response of the system
max_y = max(y);

%Now that we know max(y), we want to find out where in the vector this
%occurs.  This means that we want to find the index.
max_index = find(y==max_y);
```

31. Plot both the input and output of the system on the same graph. Plot the input as a thick blue line and plot the output as a thick red line. Mark the maximum response of the system with a large, thick, black x. Label the plot appropriately. Add a grid and a legend. In the legend, report what the maximum value of y(t) is. A sample code and the output (after being exported as a .jpg) is shown below

Functions:	*figure, hold on, plot, grid, xlabel, ylabel,legend, num2str*
Note:	*The num2str function is one of the most useful plotting functions. Be sure that you understand how to use it.*

```
%Now let's plot the data to verify that we have it correct
figure
hold on
plot(t,u,'b-','LineWidth',3)
plot(t,y,'r-','LineWidth',3)
plot(t(max_index),y(max_index),'kx','MarkerSize',15,'LineWidth',3)
grid
title('Response of System with c = 0.5')
xlabel('t (sec)')
ylabel('y (m)')
legend('u','y',['max(y) = ',num2str(max_y)])
hold off
```

The output of this code should appear as



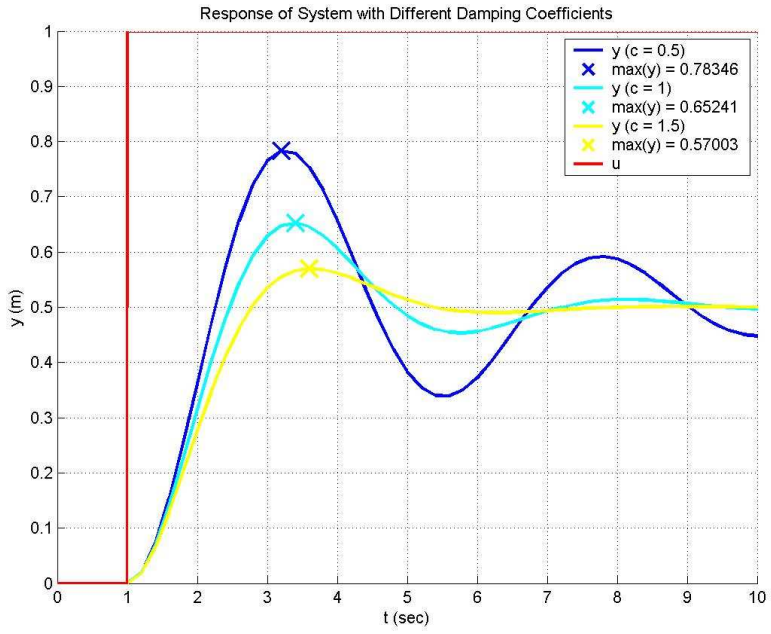
32. Now let's see how the response of the system changes with different damping coefficients. Simulate and plot the response of the system with three different damping coefficients of c = 0.5, 1.0, and 1.5. We can use a for loop to make the coding easier. A sample code is shown below.

Functions:	*for, length, …*
Note:	*The "…" symbol mean continue typing on the next line. This allows a very long line of code be broken up into two separate lines.*

```
%Now let's repeat this process for three different values of c.  We would
%like to use a for loop to make this easier.  First, we define a vector
%which contains the values of c which we would like to use
c_range = [0.5 1 1.5];

%Start a new figure to plot all the results in (we do this outside the for
%loop because we only want 1 figure, not 3).  Also turn the hold on
figure
hold on

%Define a color map which basically defines what different colors we use to
%plot the different responses
color_map = jet(length(c_range));

for n = 1:length(c_range)
    %Obtain the current value of c
    c = c_range(n);

    %Redefine the A, B, C, and D matrices with the new value of c
    A = [0 1;
        -k/m -c/m];

    B = [0;
         1/m];

    C = [1 0];

    D = [0];

    %Run the Simulink model with the new values of A, B, C, and D
    sim('mass_spring_damper_model');

    %Extract the data
    t = sim_u.time;
    u = sim_u.signals.values;
    y = sim_y.signals.values;

    %Find the max of y and the index where this occurs
```

```
    max_y = max(y);
    max_index = find(y==max_y);

    %Also we want to store this data so we can use it in the legend later
    max_y_data(n) = max_y;

    %Plot the new data
    plot(t,y,'LineWidth',2,'color',color_map(n,:))
    plot(t(max_index),y(max_index),'x','color',color_map(n,:),...
        'MarkerSize',15,'LineWidth',2)
end

%Plot the input only once
plot(t,u,'r-','LineWidth',2)

%Label the figure
title('Response of System with Different Damping Coefficients')
xlabel('t (sec)')
ylabel('y (m)')
legend(...
    ['y (c = ',num2str(c_range(1)),')'],['max(y) = ',num2str(max_y_data(1))],...
    ['y (c = ',num2str(c_range(2)),')'],['max(y) = ',num2str(max_y_data(2))],...
    ['y (c = ',num2str(c_range(3)),')'],['max(y) = ',num2str(max_y_data(3))],...
    'u')
grid
hold off
```

The output should look like



Version History:	09/28/04: Created:	
	11/23/05: Updated:	made this format match other to-do documents and removed references to AA547.
	12/01/05: Updated:	changed header to match how-to template
	12/09/05: Updated:	Made changes to layout and added footer.
	10/31/06: Updated:	Fixed typo of "Tout" to "tout"
	11/01/06: Updated:	Touched up some of the graphics.
	09/15/09: Updated:	Added 2nd order ODE method.
	09/25/10: Updated:	Minor changes