

Hands-on Lab 4:

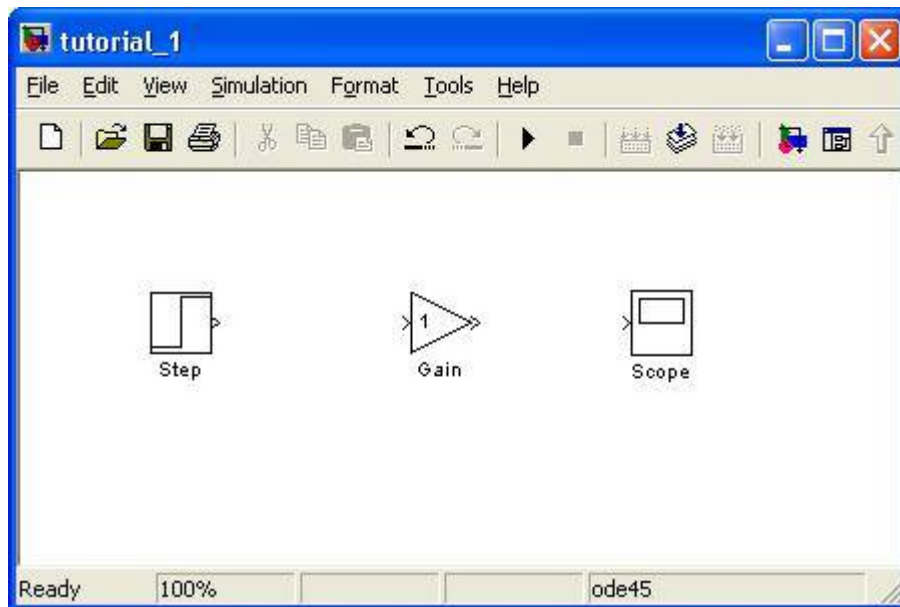
Simulink – Modeling Dynamic Systems

This lab introduces Simulink concepts necessary to model dynamic systems. Simulink is an extension of MATLAB that provides a graphical environment for the construction of a block diagram representation of a system. Simulink contains a number of libraries which allow the accurate modeling of a variety of systems (e.g. continuous, discrete, etc.).

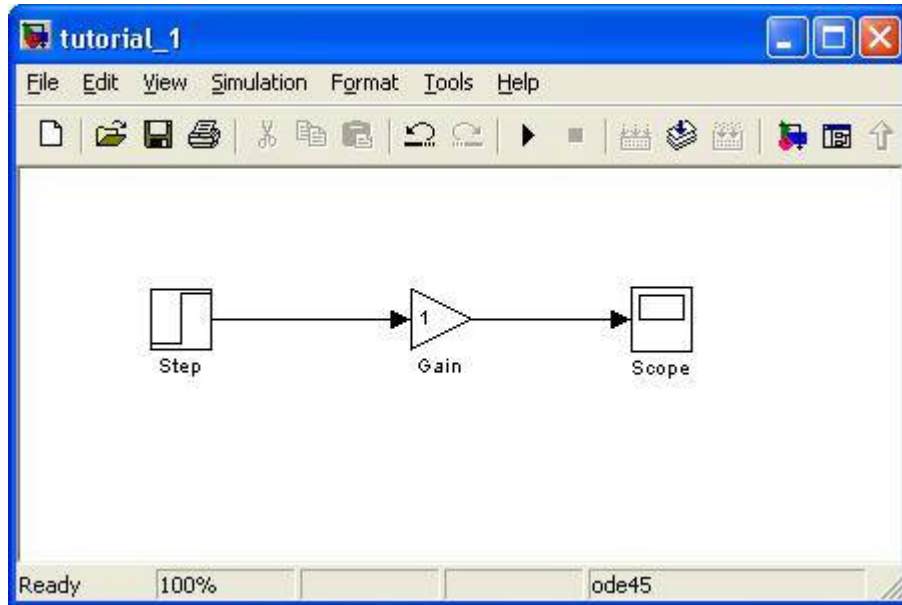
Concept 1: Getting to Know Simulink

Step 1: Open up MATLAB and click on the Simulink icon  to open up the Simulink Library Browser. Create a new block diagram by clicking File => New => Model.

Step 2: Create the model shown below by dragging the appropriate blocks from the library browser into the workspace (*Step* is contained in the *Sources* library; *Scope* is located in the *Sinks* library, and the *Gain* block can be found under *Math Operations*).

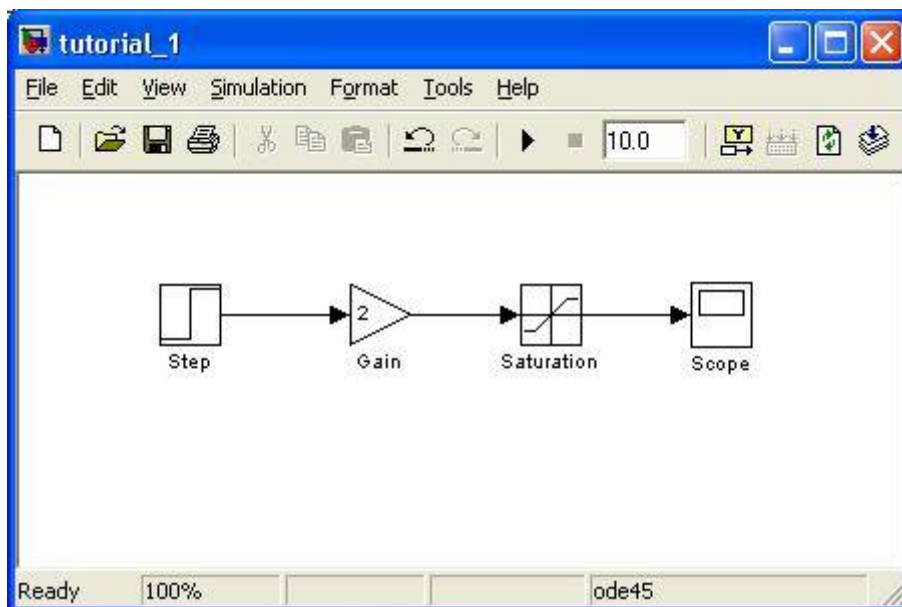


Step 3: Signals enable you to pass data between blocks. To connect the workspace above with signals, place your cursor over the output port of *Step* (notice that the pointer will change to a crosshair) and drag to the input port of the *Gain* block. Similarly connect the *Gain* block to *Scope*. **Note:** Simulink has an auto-connect feature. To see how this works, erase the newly added signal from the above step. Click on the *Gain* block to highlight it, hold down the control key and click on the *Scope* block.



The *Step* block has a default step time of 1 which denotes at what time in the simulation the step occurs. Change this value to 5 by double-clicking on it. The *Gain* block has a default value of 1 which has no affect on the signal. To change the value, double-click on it and type in 2 for the gain. Run the simulation by clicking the Play button. Double-click on the scope to view the signal.

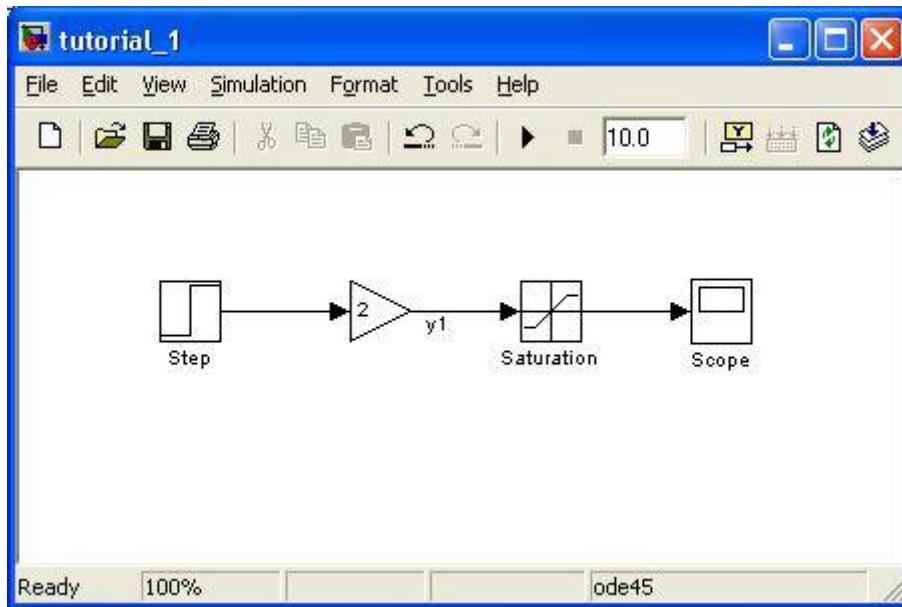
Step 4: If a block needs to be inserted in the middle of two blocks which are already connected, just drag the block over the signal connecting the two blocks. The inserted block automatically attaches itself to the signals. To test this out, try inserting a *Saturation* block (found under the *Discontinuities* tab) between the *Gain* and *Scope* blocks. The *Saturation* block is used to impose upper and lower bounds on a signal.



Step 5: Labels can increase the readability and usefulness of a model. This may not be evident for small models like the one created thus far. However, when dealing with larger models containing several input, output, and gain blocks, it will help the user if the blocks are labeled. Simulink has default names for every block. To rename a block, simply click on the label.

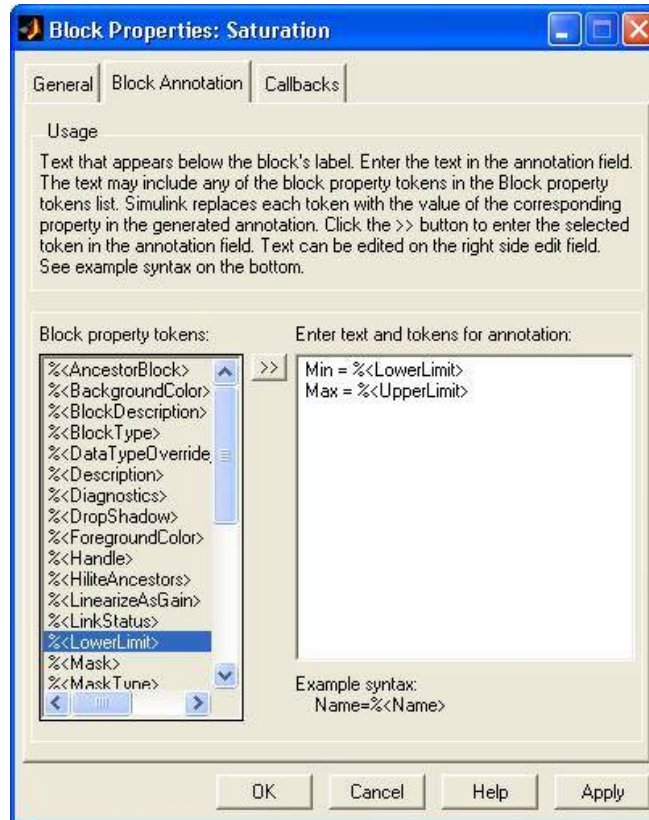
A block label which is unnecessary can also be hidden. For example, the *Gain* block is easily recognized and, thus, does not need to be labeled. To remove the label, right-click on the block and then go to Format => Hide Name.

Another feature to make a model more readable is signal labeling. To label a signal, double-click on it to bring up a text box. Label the signal coming out of the *Gain* block as *y1*. Your workspace should now look like the one below:

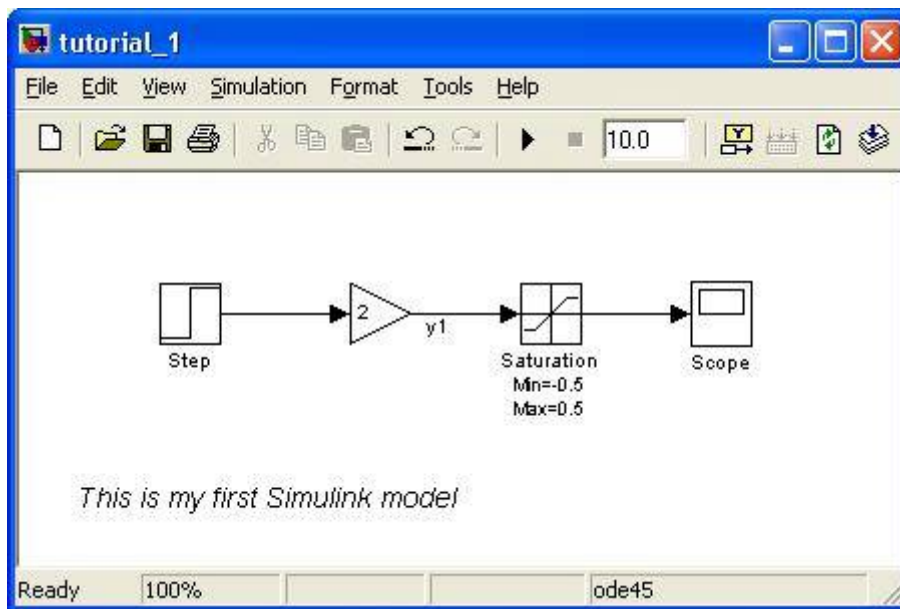


Step 6: Certain block attributes and/or parameters help make the block diagram more comprehensible. To display the block attributes for the *Saturation* block, right-click on it and selecting Block Properties. Select the Block Annotation tab and click LowerLimit and then the >> key to move it into the other window. Set the variable Min equal to the lower limit (see figure below). Likewise, set the variable Max equal to the upper limit.

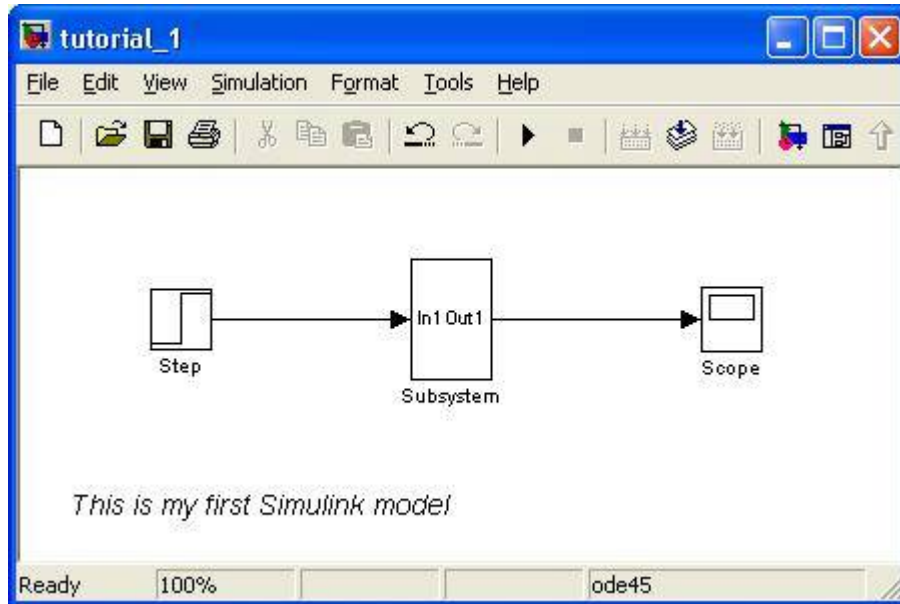
MEM 351 Dynamic Systems Lab 4



Step 7: Annotations help create a well documented Simulink model. To add an annotation, double-click anywhere on the background. Add the text, “This is my first Simulink model” in the box which pops up. Change it to 14pt and italicize the font by going to Format => Font menu.



Step 8: A subsystem allows you to combine multiple blocks and signals into a single block. Drag a box around both the *Gain* and *Saturation* blocks to highlight them and then go to Edit => Create Subsystem (or just hit Ctrl+G). Double-click on the subsystem in order to view its components.



Step 9: Click the Play button to run the model and double-click on the scope to view the data. Click the binoculars to fit the graphs in the window.

Exercise 1: In Simulink create a model for the following

1-1. Display a sine wave of amplitude 4 on a Scope using *Sin Wave* (Sources), *Constant* (Sources), and *Product* (Math Operations) blocks.

Concept 2: Modeling Continuous Systems

The generic building block of a continuous system is the *Integrator* block (found under the *Continuous* tab). The integrator block will, as the name implies, integrate the input signal:

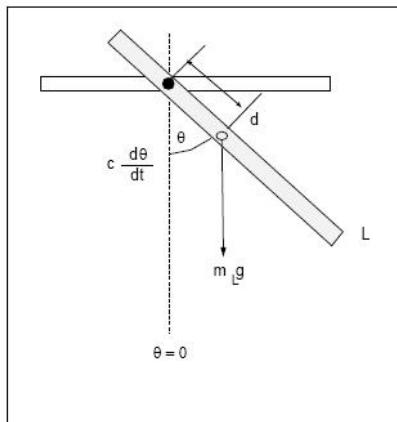
$$\dot{y}(t) \longrightarrow \boxed{\frac{1}{s}} \longrightarrow y(t)$$

Note: Derivative blocks are not commonly used to construct differential equations because they do not allow you to store any initial conditions.

When modeling the dynamics of a system (i.e. a differential equation), the following procedure can be used as a general guideline:

1. Arrange the equation such that the highest order derivative is on the left and all other terms are on the right.
2. Determine the number of *Integrator* blocks required (2^{nd} order requires 2) and drag them into the workspace.
3. Connect the Integrator blocks and label the input and output of each block.
4. Construct the equation, one differential equation at a time.

To get some practice modeling a continuous system and to further emphasize the guidelines above, let's look at the compound pendulum:



Where	L	Bar length	0.495	m
	d	Pivot to CG distance	0.023	m
	m_L	Mass of pendulum	0.43	kg

The parallel axis theorem yields moment of inertia:

$$J = J_c + m_L d^2 = \frac{1}{12} m_L L^2 + m_L d^2 = m_L \left(\frac{1}{12} L^2 + d^2 \right)$$

$$J = 0.43 \cdot \left(\frac{0.495^2}{12} + 0.023^2 \right) = 0.0090 \quad kg \cdot m^2$$

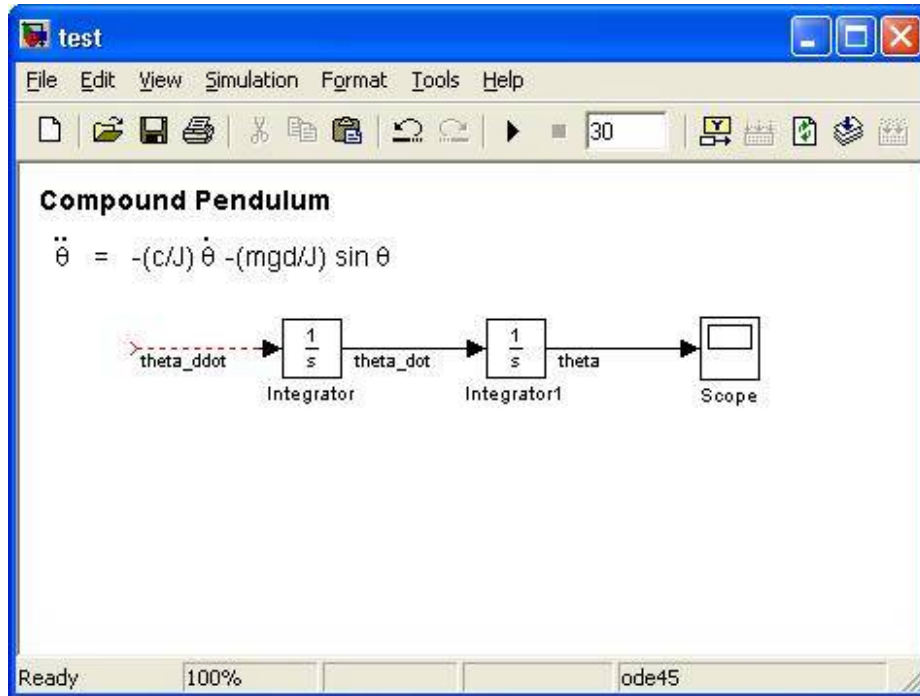
Step 1: The equation of motion of the above system is:

$$-mgd \sin \theta - c\dot{\theta} = J\ddot{\theta}$$

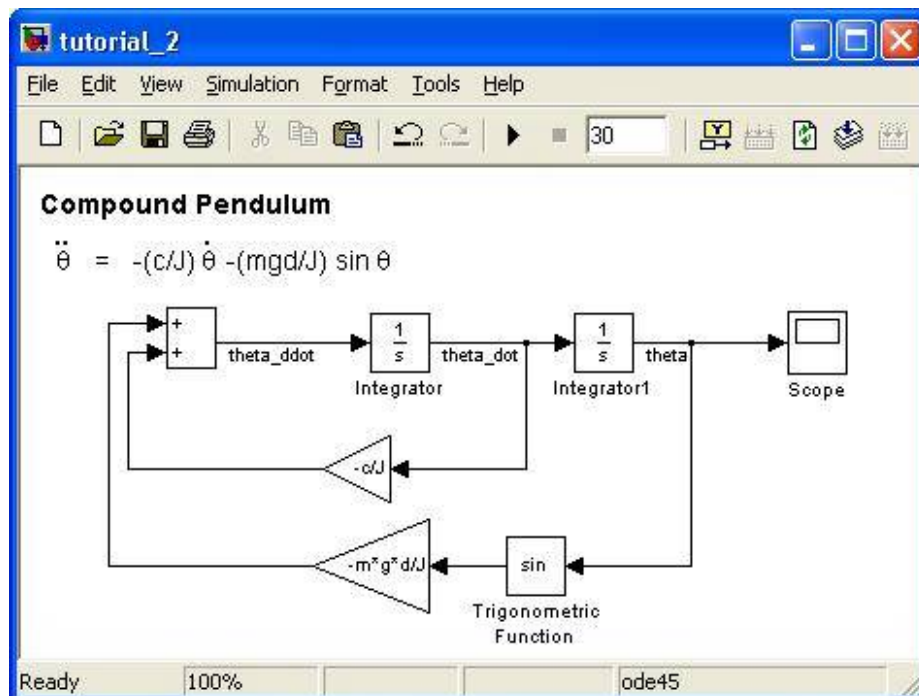
Step 2: Arranging the equation of motion such that the highest order derivative is on the one side and the rest of the terms are on the other, we have:

$$\ddot{\theta} = -\frac{c}{J} \dot{\theta} - \left(\frac{mgd}{J} \right) \sin \theta$$

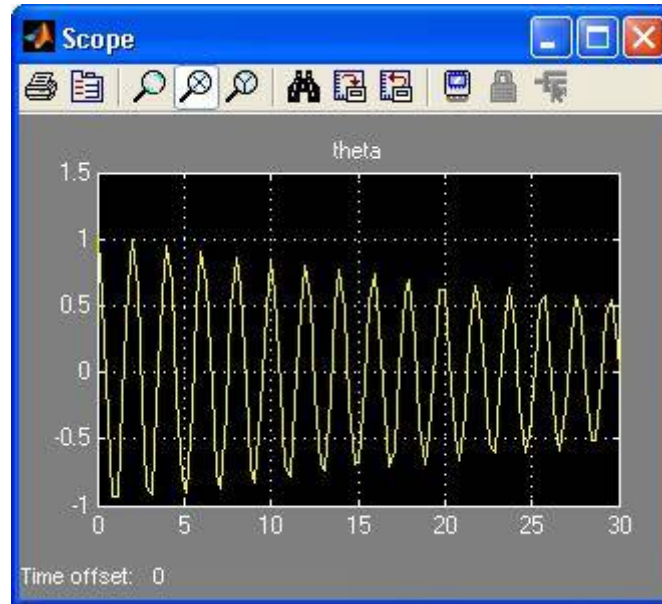
Step 3: Since we have a second order differential equation, we will need 2 *Integrator* blocks. Create a new model and drag them into the workspace. After connecting the integrator blocks and labeling the output signals, your model should look like the one below:



Step 4: Based on the differential equation, θ has to go into a sinusoid function and then $\sin \theta$ has to be multiplied by $(-mgd / J)$. Instead of grabbing *Product* blocks to achieve the above expressions, it is easier to use a *Gain* block. The $\sin \theta$ term can be achieved with a *Trigonometric Function* block (located under Math Operations). Highlight the *Trigonometric* and *Gain* blocks and hit Ctrl + I to get them to face to the left. Similarly, add another *Gain* block for the damping term, c .



Step 5: Finish the model by defining the initial conditions (double-click on the *Integrator1* block and enter in 3.14/3 in the initial condition box) and initializing the constants (go to File => Model Properties => Callbacks and type in $L=0.495$; $m=0.43$; $d=0.023$; $J=(1/12)*(m*L*L+m*d*d)$; $g=9.81$; $c=0.00035$). Hit play and double-click the scope to view your output. Note that theta is in radians.



You can see from the scope's output that the pendulum has a decaying oscillatory response starting at $\pi/3$ radians and eventually stabilizing at zero degrees.

Exercise 2: In Simulink create a program for the following

2-1. In the above model, theta is in radians. Edit/modify your model such that the scope displays theta in degrees.

Concept 3: State Space Form

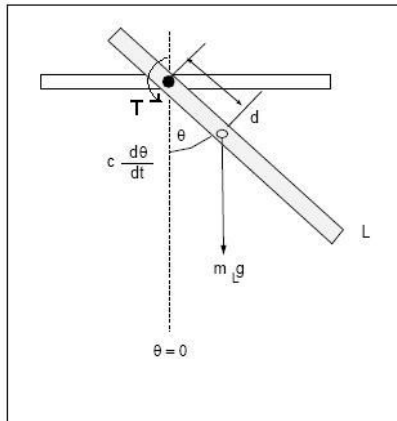
For a continuous time system, state space notation expresses its dynamics in the form of first-order differential equations. The general equation of state space representation is:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

where A, B, C, and D are matrices of size (# states)-by-(# states), (# states)-by-(# outputs), (# inputs)-by-(# states), and (# inputs)-by-(# outputs), respectively. Let's model the compound pendulum from above, but with an applied torque at the pivot point (see Figure below).

MEM 351 Dynamic Systems Lab 4



Where	L	Bar length	0.495	m
	d	Pivot to CG distance	0.023	m
	m_L	Mass of pendulum	0.43	kg

The parallel axis theorem yields moment of inertia:

$$J = J_c + m_L d^2 = \frac{1}{12} m_L L^2 + m_L d^2 = m_L \left(\frac{1}{12} L^2 + d^2 \right)$$

$$J = 0.43 \cdot \left(\frac{0.495^2}{12} + 0.023^2 \right) = 0.0090 \quad kg \cdot m^2$$

The equation of motion of the above system is:

$$\ddot{\theta} = -\frac{c}{J} \dot{\theta} - \left(\frac{mgd}{J} \right) \sin \theta + \frac{T}{J}$$

Formulating in state space first requires the linearization of the $\sin \theta$ term (i.e. $\sin \theta \sim \theta$). Therefore, in state space representation, we have:

$$x_1 = \theta \Rightarrow \dot{x}_1 = \dot{\theta}$$

$$x_2 = \dot{\theta} \Rightarrow \dot{x}_2 = \ddot{\theta}$$

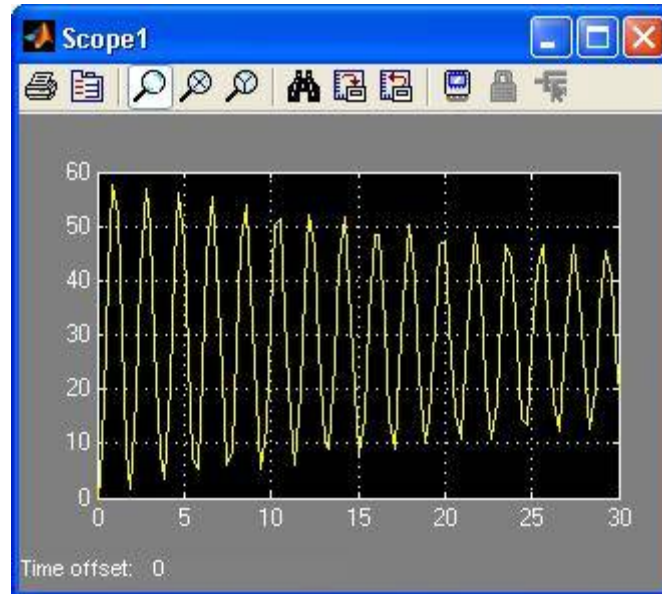
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{mgd}{J} & -\frac{c}{J} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} T$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u$$

Step 1: Create a new model and bring a *State-Space* block (located in the Continuous library) into the workspace. Double-click on it and enter in the appropriate values for the A, B, C, and D matrices.

Step 2: Let the input to the state space model be the torque and display the output (in degrees) on a scope.

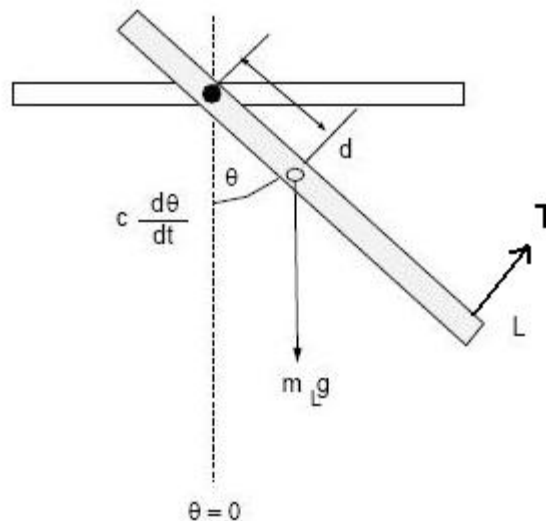
Step 3: Initialize the constants ($L=0.495$; $m=0.43$; $d=0.023$; $J=(1/12)*(m*L*L+m*d*d)$; $g=9.81$; $c=0.00035$) by going to File => Model Properties => Callbacks. Set the torque to be 0.05 [Nm] and hit play. The scope's output should like the figure below.



You can see from the scope's output that the pendulum has a decaying oscillatory response eventually stabilizing at around 30 degrees. **However, if a faster response time is required, control methodologies (pole placement, PID, etc.) MUST be implemented on the system's input, T.**

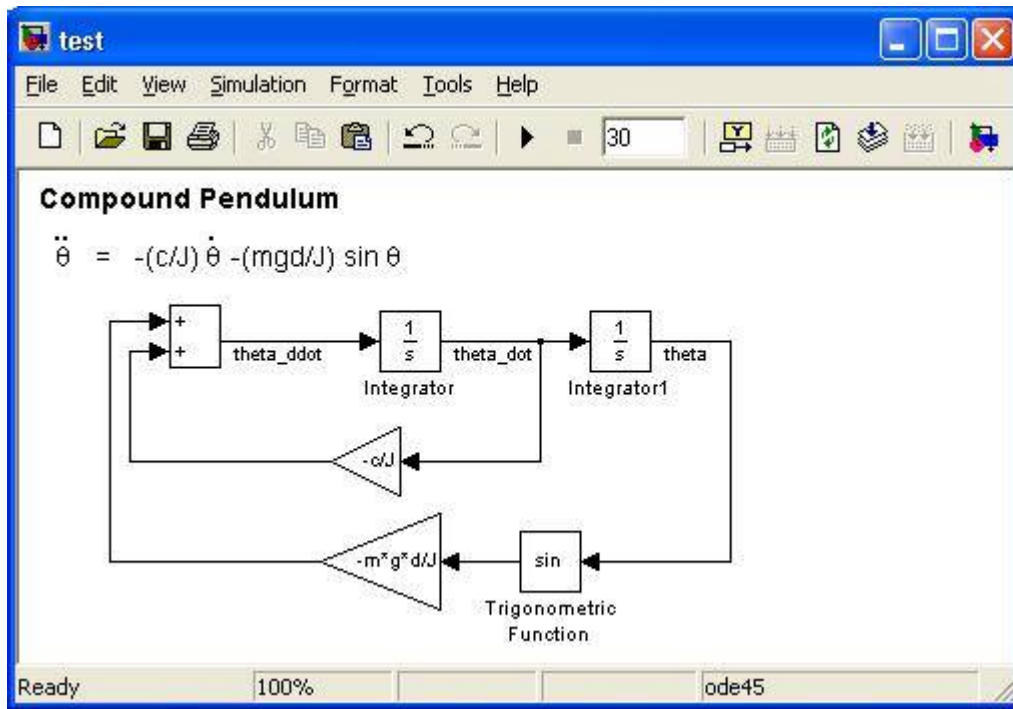
Exercise 3: In Simulink create a program for the following

3-1. Create a state space model to display theta in degrees when the applied torque is removed and a thrust vector T (\perp to the beam) is added at the end of the beam.



Concept 4: Utilizing Simulink's Pendulum Animation Function

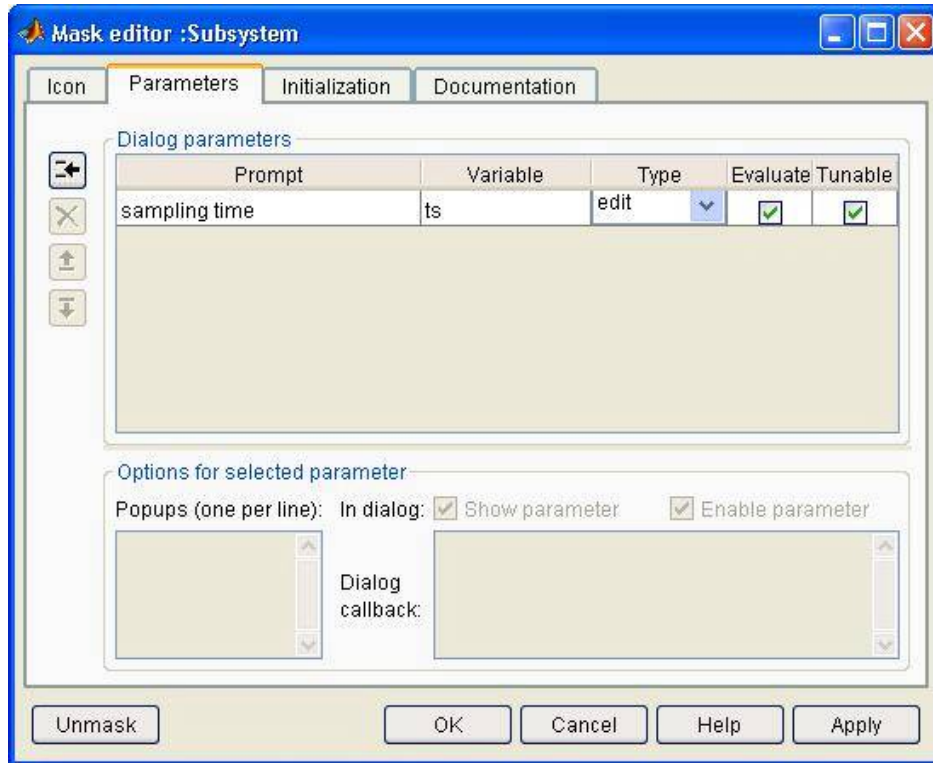
Step 1: Recreate the pendulum model in concept 2 but without the scope.



Step 2: Simulink comes with a pendulum animation function called `pndanim1.m`. *S-Function* blocks allow you to call a Matlab function from your model. So let's bring an *S-Function* block into our workspace (located under *User-Defined Functions*). Double-click on the block and enter in `pndanim1` in the *S-Function* Name box. Also enter in `ts` for the *S-Function* parameter (`ts`, or the sampling time, is the argument taken by the `pndanim1` function) and hit OK.

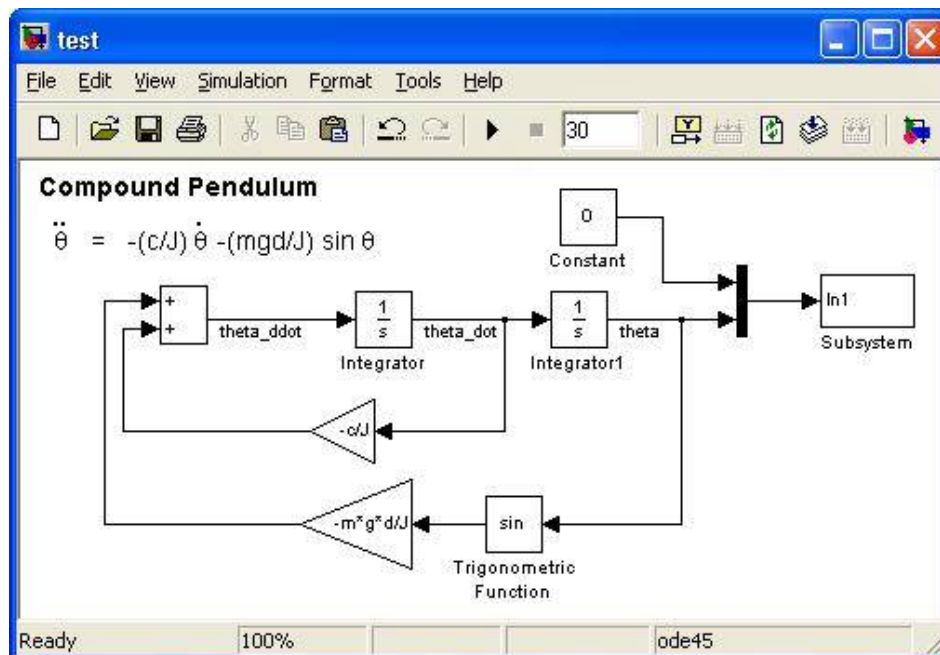
Step 3: Highlight the *S-Function* block and create a subsystem (`Ctrl + G`). Double-click on the subsystem and delete the *Out1* block since we will not need it.

Step 4: Close the subsystem window. Right-click on the *S-Function* block and select *Mask Subsystem*. Select the *Parameters* tab and click on the *Add* button in the top left. Enter in the parameters to match the window below and click OK.



Double-click on the subsystem block and enter in a sampling time of 0.005 seconds.

Step 5: The subsystem takes a vector input [horizontal beam position on screen, theta in radians]. To create a vector input, we need to Mux the two signals. Bring *Constant* and *Mux* (Signal Routing) blocks into the workspace. Route the signals so your model looks like the one below.



Step 6: Hit play and you should see an animation window pop up.

