**SY202 – Cyber Systems Engineering**

**Intro**

**CSEE**

**Due Date: 05 Feb 2019**

**LABORATORY INVESTIGATION #02: Modeling and Simulation of Cyber-Physical Systems**

**Objectives:**

Simulation can improve understanding of the effects of changes you realized across the attack surfaces identified. Knowing how to 1) create a function / mathematical relationship that expresses the way the systems behaves, 2) create a equation of motion, and 3) model a system and then simulate the dynamic behavior of that system MATLAB/SIMULINK is a vital skill for success. In this exercise you will learn to:

1. Build a mathematic model using simulation blocks that represents a cyber-physical system
2. Build/Run model in SIMULINK in order to solve a system of differential equations
3. Use MATLAB variables to configure simulation parameters
4. Plot results of model using MATLAB workspace variables

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.*

Simulink is a graphical extension to MATLAB for modeling and simulation of systems. One of the main advantages of Simulink is the ability to model a nonlinear system, which a transfer function is unable to do. Another advantage of Simulink is the ability to take on initial conditions. When a transfer function is built, the initial conditions are assumed to be zero.

**Part I: Modeling in Simulink (Review):**

**Change the current folder**

* 1. Change the current folder to “SY202\_MATLABExercise2” by clicking the open folder icon , creating a new folder “SY202\_MATLABExercise2” by right-clicking in the window/selecting New>Folder, and selecting this folder.

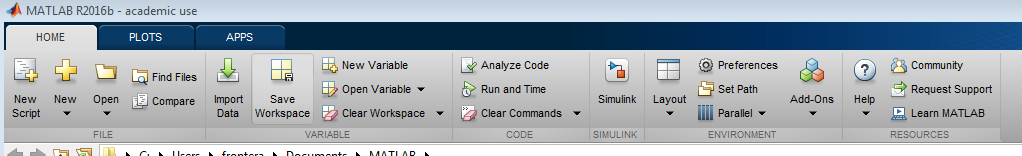
Recall in Simulink, systems are drawn on screen as block diagrams. Many elements of block diagrams are available, such as transfer functions, summing junctions, etc., as well as virtual input and output devices such as function generators and oscilloscopes.

**Starting Simulink**

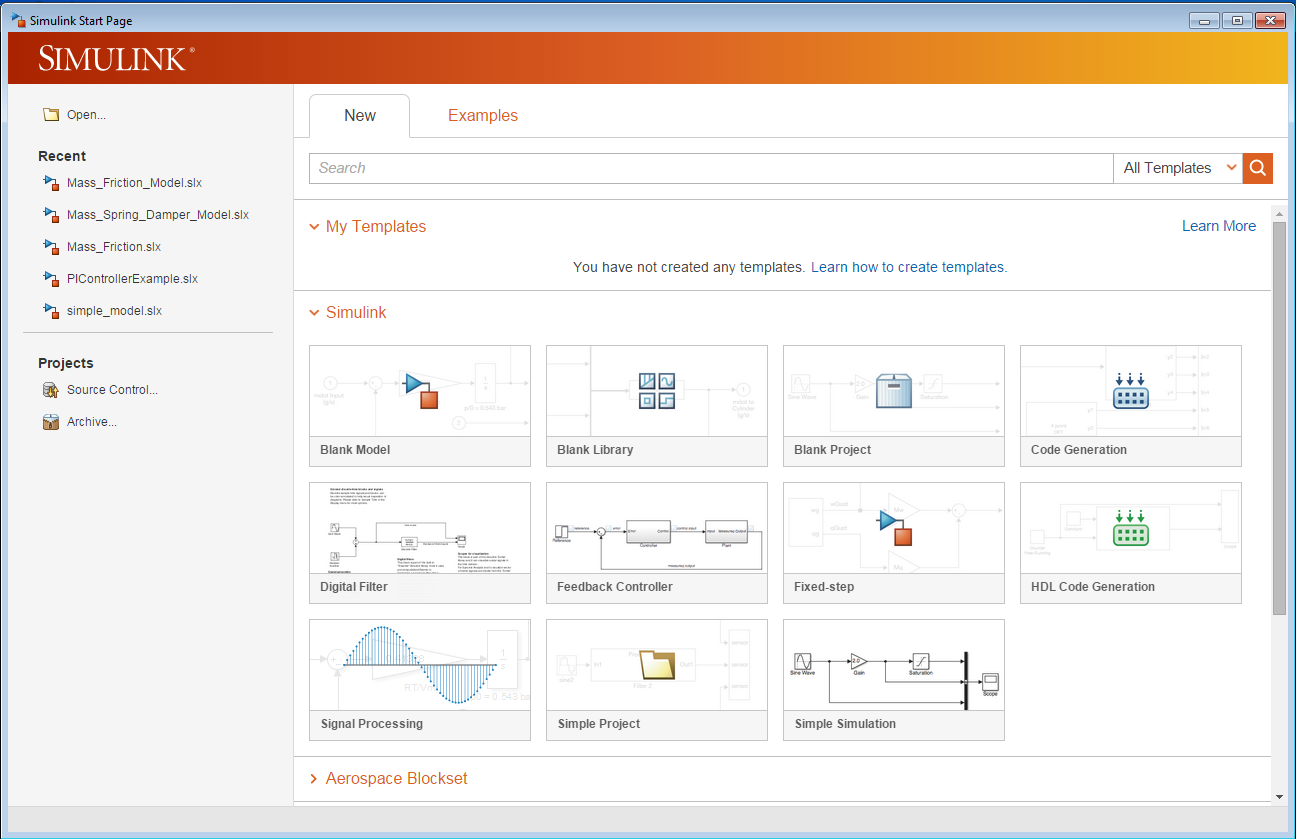
Simulink is started from the MATLAB command prompt by entering the following command:

>simulink

Alternatively, you can press the **Simulink** button at the top of the MATLAB window as shown here:



When it starts, Simulink brings up a single window, entitled **Simulink Start Page**, which can be seen here.



A new model can be created by selecting **Blank Model**  on the Simulink Start Page (or by hitting **Ctrl-N**).

**Cyber-Physical System**

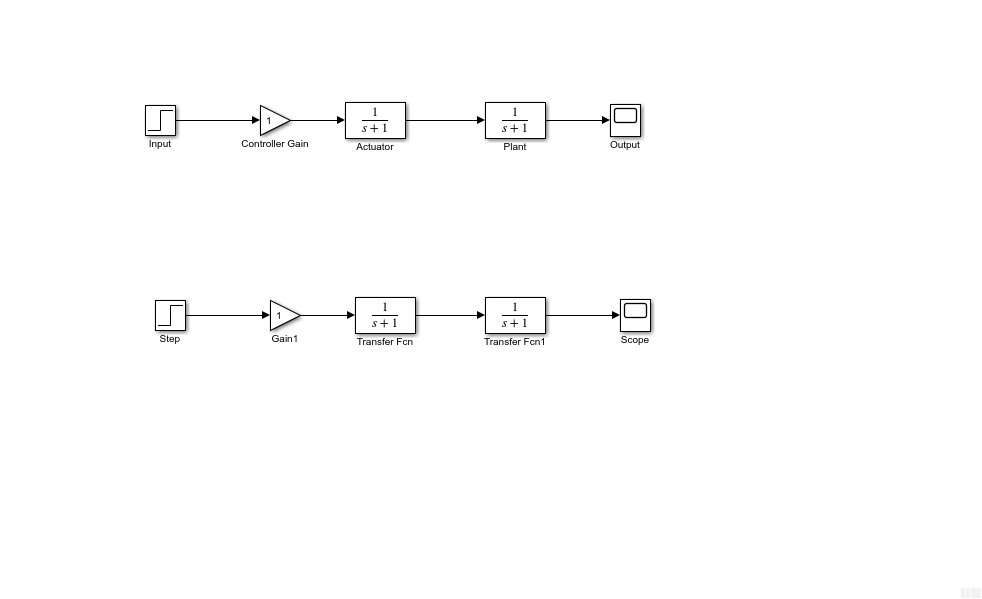
You will simulate an open-loop cyber-physical system. In particular, you will simulate the angular position control of an autonomous weapon, such as the Phalanx System used on US Navy ships and illustrated below.



**Basic Open-Loop System**

Herein, we will model an open-loop control system that regulates the angular orientation of a weapon.

This example provides a short review of conducting a system simulation using Simulink. In Simulink, a model is a collection of blocks, which, in general, represents a system. Build the following model using blocks from the Library Browser:

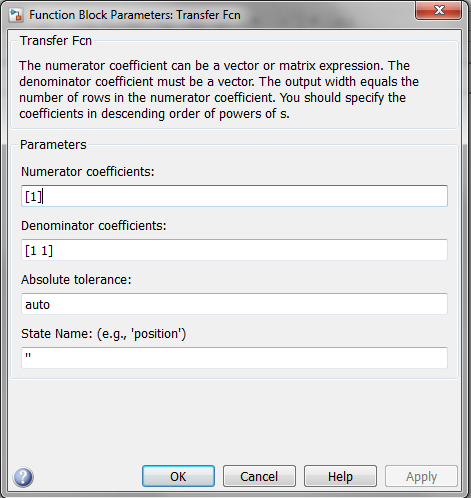


The basic model consists of five blocks: one *Step*, one *Gain*, two *Transfer Function*, and one *Scope*. The Step is a **Source** block from which a step input signal originates. This signal is transferred through the **line** in the direction indicated by the arrow to the *Gain* **Math Operation** block. The output of the *Gain* block connects to the *Transfer Function* **Continuous** block. The *Transfer Function* block modifies its input signal and outputs a new signal on a line to the next *Transfer Function* block, which output connects to the *Scope*. The *Scope* is a **Sink** block used to display a signal much like an oscilloscope.

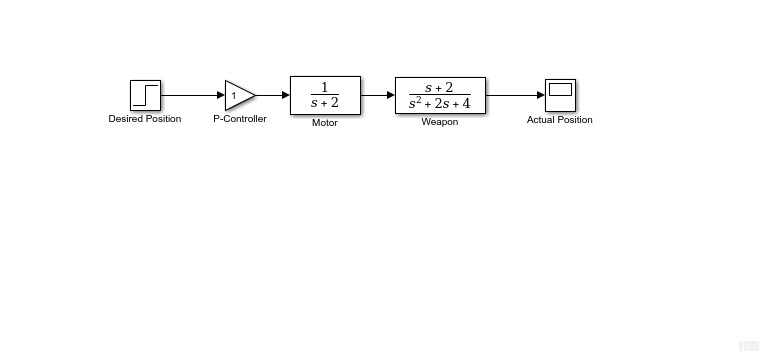
The first Transfer Function block represents the Actuator of our cyber-physical system, in this case, a Motor. Change the name of the first Transfer Function block to Motor by directly editing the displayed name. The second Transfer Function represents the plant. Change the name from Transfer Fcn1 to Weapon. The Gain block represents the controller. Changes its name to P-Controller. A Proportional (P) Controller is a type of control system where the control signal is proportional to the input. You will learn about this controller and other more advanced control systems later in the semester. Now, change the names of the Step block to Desired Position and the name of the Scope to Actual Position. We are building an open-loop controlled cyber-physical system. This model is ideal as it does not include disturbances. For better accuracy, one would like to include disturbances (such as the effect of wind, waves, etc.) into the model.

**Modifying Blocks**

A block can be modified by double-clicking on it. For example, if you double-click on the *Transfer Function* block in the Simple model, you will see the following dialog box.

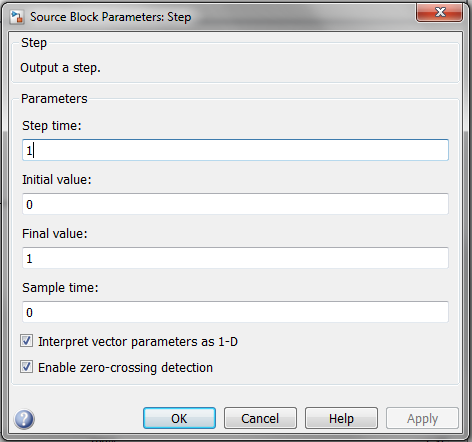


This dialog box contains fields for the numerator and the denominator of the block's transfer function. By entering a vector containing the coefficients of the desired numerator or denominator polynomial, the desired transfer function can be entered. For example, to change the second transfer function block (i.e., Plant or Weapon) to , enter the following sequence into the numerator field [1, 2] and in the denominator field [1, 2, 4] and press the apply button, the model window will change to the following, which reflects the change in the denominator of the transfer function. You can enlarge the size of the block to display the transfer function if the transfer function is too long. Repeat this step for the transfer function of the actuator (Motor) assuming it is .



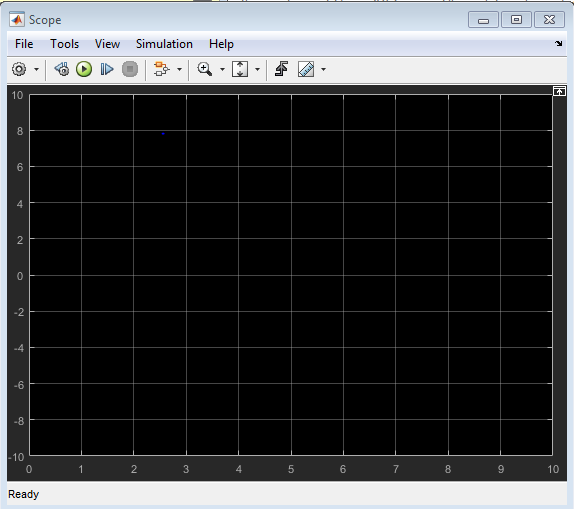
*Note: If the numerator or denominator of your transfer function is missing some powers, e.g., , you need to input zero in the place of the missing power. For instance, , which would require entering the polynomial as*

The *Step* block can also be double-clicked, bringing up the following dialog box.



The default parameters in this dialog box generate a step function occurring at time = 1 sec (*Step time*), from an initial level of zero (*Initial value*) to a level of 1 (*Final value*). In other words, a unit step that start t = 1. Each of these parameters can be changed as desired. Close this dialog before continuing.

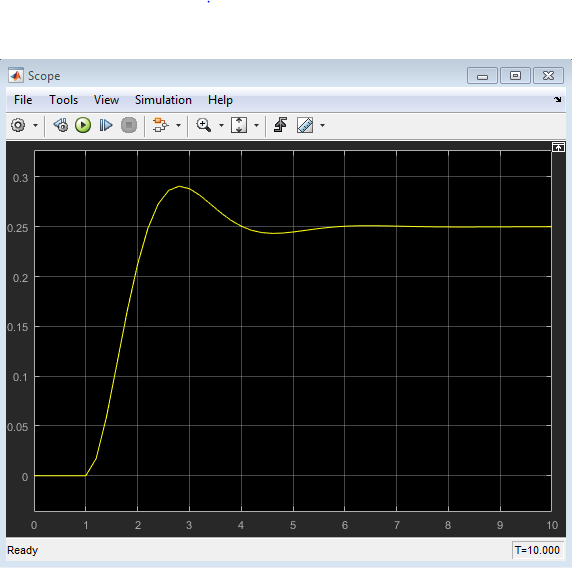
The most complicated of these three blocks in the *Scope* block. Double-clicking on this brings up a blank oscilloscope screen.



When a simulation is performed, the signal which feeds into the scope will be displayed in this window. Detailed operation of the scope will not be covered in this tutorial.

**Running Simulations**

To run a simulation, we will work with the model that you already built. Before running a simulation of this system, first open the scope window by double-clicking on the scope block. Then, to start the simulation, select **Run** from the **Simulation** menu, click the **Play** button at the top of the screen, or hit **Ctrl-T**. The simulation should run very quickly and the scope window will appear as shown below.

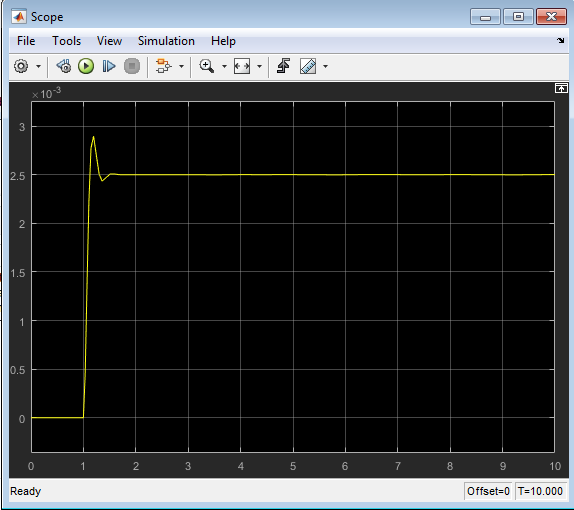
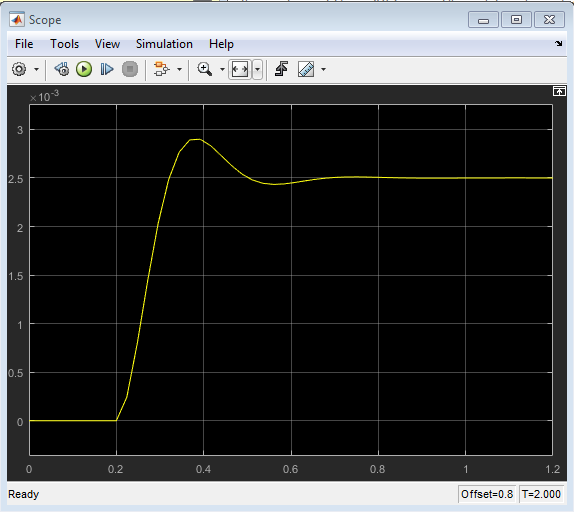


Note that the step response does not begin until t = 1. This can be changed by double-clicking on the *step* block. Now, we will change the parameters of the system and simulate the system again. Double-click on the *Transfer Function* block for the plant in the model window and change the denominator to: [1 20 400].

Re-run the simulation (hit **Ctrl-T**) and you should see what appears as a flat line in the scope window. Since the new transfer function has a very fast response, it compressed into a very narrow part of the scope window. This is not really a problem with the scope, but with the simulation itself. Simulink simulated the system for a full ten seconds even though the system had reached steady state shortly after one second.

To correct this, you need to change the parameters of the simulation itself. In the model window, select **Model Configuration Parameters** from the **Simulation** menu (top bar).

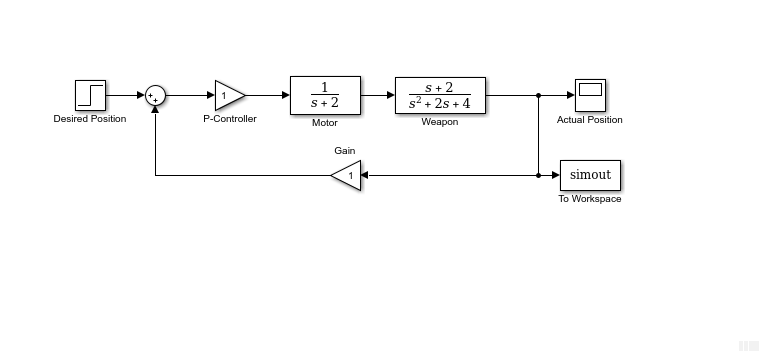
There are many simulation parameter options; we will only be concerned with the start and stop times, which tell Simulink over what time period to perform the simulation. Change **Start time** from 0.0 to 0.8 (since the step doesn't occur until t = 1.0). Change **Stop time** from 10.0 to 2.0, which should be only shortly after the system settles. Close the dialog box and rerun the simulation.

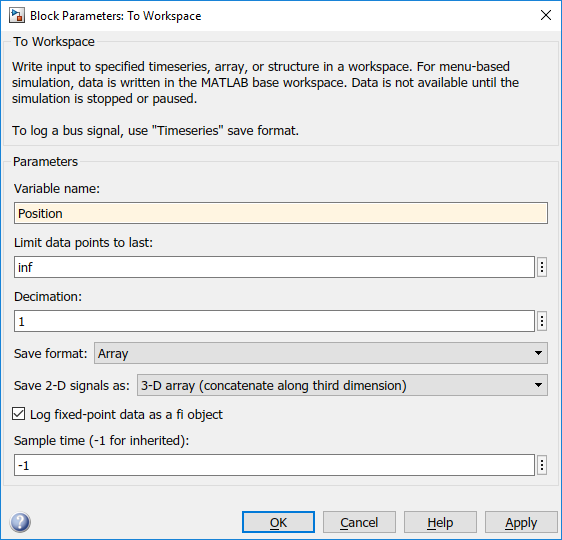
Change the Start time and the Stop time of the simulation back to 0.0 and 10.0, respectively. In addition, change the denominator of transfer function back to [1, 2, 4].

**Advanced Closed-Loop System**

Now that you are comfortable with SIMULINK, build the following system:

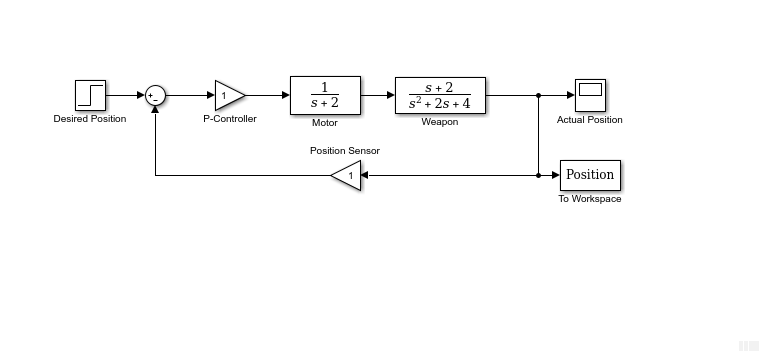


The *Sum* can be found in the *Math Operations* library, while the *To Workspace* can be found in the *Sinks* library. The To Workspace block saves the input of the block (in this case, the output of the system) to the workspace of MATLAB. Open the To Workspace block and assign a *Variable name* to the output, e.g., “Position”, and set the *Save Format* to *Array*.



The new Gain block in the feedback path represents the sensor. Change the name of the block to *Position Sensor* and leave the gain value as “1”. A gain value of “1” implies that the measured position is exactly equal to the actual position of the weapon (no error is introduced by the sensor).

Double-click on the *Sum* block. Since you will want the second input to be subtracted, enter +- into the list of signs field and close the dialog box.



Save your model, select **Save As** in the **File** menu and type in any desired model name.

**Simulation**

Now that the model is complete, you can simulate the model. Select **Start** from the **Simulation** menu to run the simulation. Double-click on the scope to view its output.

You should see in the workspace of MATLAB that some variables have been generated, including Position and tout. Position is an array containing the output of your simulation. tout is an array representing time. You can use the plot command, plot(tout,Position), to plot the data.

**DELIVERABLE #1:** Plot of Simulation in the step above. Do not forget to label the axes of your plot and include a title.Assume that the output is measured in radians and that the time is measured in seconds. Refer to Lab #1 for plotting examples.

**Taking Variables from MATLAB**

In some cases, parameters, such as gain, may be calculated in MATLAB to be used in a Simulink model. If this is the case, it is not necessary to enter the result of the MATLAB calculation directly into Simulink.

For example, suppose we calculated the gain in MATLAB in the variable *K*. Emulate this by entering the following command at the MATLAB command prompt.

>K = 1.0

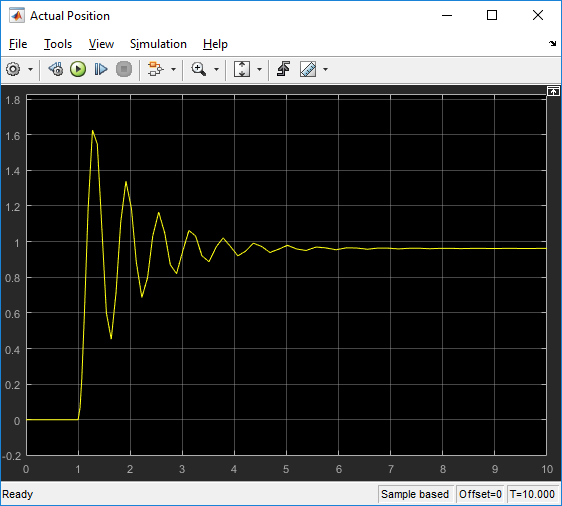
This variable can now be used in the Simulink *Gain* block (i.e., P-Controller). In your Simulink model, double-click on the *P-Controller* block and enter the following in the Gain field: K

Close this dialog box, then notice now that the *Gain* block in the Simulink model shows the variable K rather than a number. Now, you can re-run the simulation and view the output on the Scope. The result should be the same as before.

Now, if any calculations are done in MATLAB to change any of the variables used in the Simulink model, the simulation will use the new values the next time it is run. To try this, in MATLAB, change the gain, K, by entering the following at the command prompt.

>K = 100

Run again your simulation and bring up the Scope window. You should see a response similar to this.

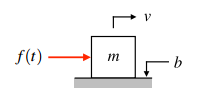


**DELIVERABLE #2:** Plot of Simulation in the step above. Do not forget to label the axes of your plot and include a title.Assume that the output is measured in radians and that the time is measured in seconds. Refer to Lab #1 for plotting examples. Compare the response of the system to the previous one (using a value of K=1).

**Instructor Led Intermission**

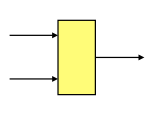
A block with mass is pushed over a horizontal surface by a force. The friction between the block and surface is viscous (proportional to the speed, ) with damping/friction coefficient, .

The equation of motion and free body diagram looks like this

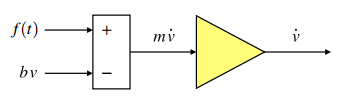


Let’s work through this simple example to illustrate the process for taking an equation of motion and expressing it in the form of a simulation diagram. The simulation diagram is another computer-based tool of solving differential equations (e.g., equations of motion).

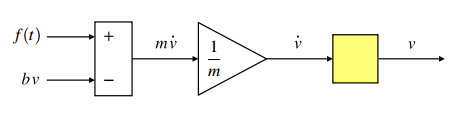
1. Solve for the term with the highest derivative
2. If the right-hand side has several terms, make them inputs to a summing junction and make the highest derivative term the output.



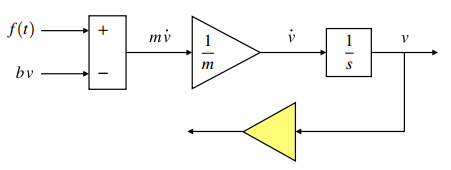
1. Isolate the highest derivative. If needed, use a gain.



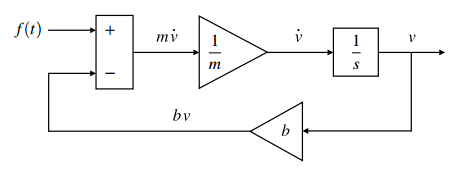
1. Insert integrators to solve for the lower derivatives



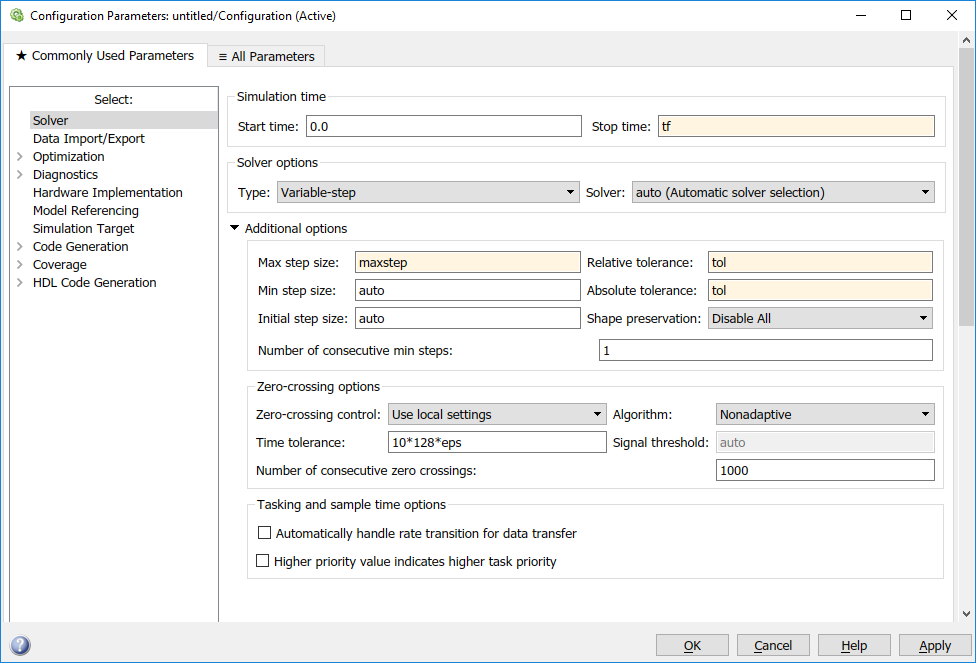
1. Create the signals that feedback to the summing junction, using gains as needed



1. Complete the simulation diagram by connecting the feedback signals to the summing junction



1. Build this model in a new SIMULINK file, send the velocity to a ‘Scope’ and to a ‘To Workspace’ block for future analysis. For the input, use a ‘Constant’ block which can be found under the Source Library. Double click on the constant block and set the ‘Constant value’ to ‘f’.
2. Change the value of the Gain block after the summation to ‘1/m’ and the value of the other gain block to ‘b’.
3. Make sure that Simulink exports the speed as an array (vector) to MATLAB. To set this, double-click on the “To Workspace” block and change the “Variable name” to ‘velocity’ and the “Save format” option to “Array”. It will now save the output of your simulation into a new variable called ‘velocity’.
4. Next, change the numerical simulation parameters by going to Simulation > Model Configuration Parameters… In the window that appears, change the default “Stop time” to the variable ‘tf’ so we can define our own stop time later. Also, replace auto in “Max step size” to ‘maxstep’, and set “Relative tolerance” and “Absolute tolerance” to ‘tol’.



1. When solving equations of motion, you need to specify initial conditions. In this case, the initial condition to specify is the initial velocity. Change the initial condition by double-clicking on the “Integrator” block and changing the default “Initial condition” to the variable name ‘v0’ (we’ll define this value later in MATLAB). If you have more than one “Integrator” block, you need to set the initial conditions in all blocks.
2. Save the model as a new file. For instance, Model\_Lab2.slx.

**Running the simulation with MATLAB**

Good practice is to create a MATLAB m-file to run the simulation instead of using the controls available in Simulink itself. The big advantage to using an m-file to run the simulation is that it’s very easy to change system and simulation parameters (like the mass m and the stop time tf) if we need to without having to modify the simulation diagram itself.

1. Start a new m-file in the same folder where your Simulink model is saved. It’s good practice to start with a clean slate by clearing all variables, closing all open figure windows, and clearing the command window, which you can do with the following code:

clear all

close all

clc

1. Assign the values for the system parameters. Remember the variable names used for m, b, f in the model

m = 1; %kg

b = 1; %Ns/m

1. Set the simulation parameters.

tf = 6; %stop time

maxstep = 0.01; %maximum step size

tol = 1e-6; %error tolerance

1. Call the model from the MATLAB script assuming that the initial conditions are zero and that the input to the system is 10 N.

%% Forced Response

f = 10; %N

v0 = 0; %initial condition (initial velocity) in m/s

sim(‘YourModelName’) %This command calls the simulation diagram, use %the name you assigned to your Simulink file

1. Run the script. If you are successful, you should see that in addition to the variables you created, you also have the arrays ‘tout’ and ‘velocity’ in MATLAB’s workspace. You should also be able to check the response by using the scope. The response of the system to nonzero input (f = 10) and zero initial condition (v(0)=0) is called **the forced response** of the system. It is how the system respond to an input.
2. If you change any of your simulation parameters and re-run the simulation, tout and velocity will be changed. Therefore, save the variables by assigning them to new variables. Do this after you have called the “sim” command.

tout\_forced = tout %use a descriptive name

velocity\_forced = velocity

1. Now, you will simulate the system when the input f(t)=0 N but the initial velocity is nonzero. This is called the **natural response** of the system. The natural response describes the system reaction to nonzero initial conditions with no input. For instance, imagine that the block starts in motion, with an initial velocity of 5 m/s. Under no other external force, except for friction, what do you expect will happen to the box? Friction will slow down the box and the velocity will eventually become zero.

%% Natural Response

f = 0; %N

v0 = 5; %initial condition (initial velocity) in m/s

sim(‘YourModelName’)

1. If you change any of your simulation parameters and re-run the simulation, tout and the velocity variable will change. Therefore, save the variables by assigning them to new variables after the sim command.

tout\_natural = tout %use a descriptive name

velocity\_natural = velocity

1. Finally, simulate the response of the system when both the input and the initial conditions are nonzero. This is the total response of the system.

%% Total Response

f = 10; %N

v0 = 5; %initial condition (initial velocity) in m/s

sim(‘YourModelName’)

tout\_total = tout %use a descriptive name

velocity\_total = velocity

1. In the same figure, plot the velocity vs time of all three scenarios. You can use ‘help plot’ to find more information of how to plot three sets of data together. Use different lines and do not forget about labels and legend. Compare results. You should see that the total response is the summation of the forced and natural responses ☺

**DELIVERABLE #3: A plot with the three different simulations. Do not forget to label the axes of your plot. Include units. Assume velocity is measured in meters per second while time is measured in seconds.**

**DELIVERABLE #4: Include your MATLAB script and Simulink model as enclosures.**

1. Change the value of the mass, damping coefficient, and input force and observe any changes on the output of your simulation. You might need to change the ‘tf’ to allow more time for the simulation to settle.
2. Discuss the results in general. Does it make sense (using your knowledge on physics) for the velocity to converge to a constant value? What happened as you increased the friction coefficient ‘b’? What happens when you increase the mass?

**Lab Report**

1. Submit a lab report using the template found in the course folder. Ensure that you follow the *SY202 – Lab Report General Guidelines* located in the GAfG Course Folder under Labs.