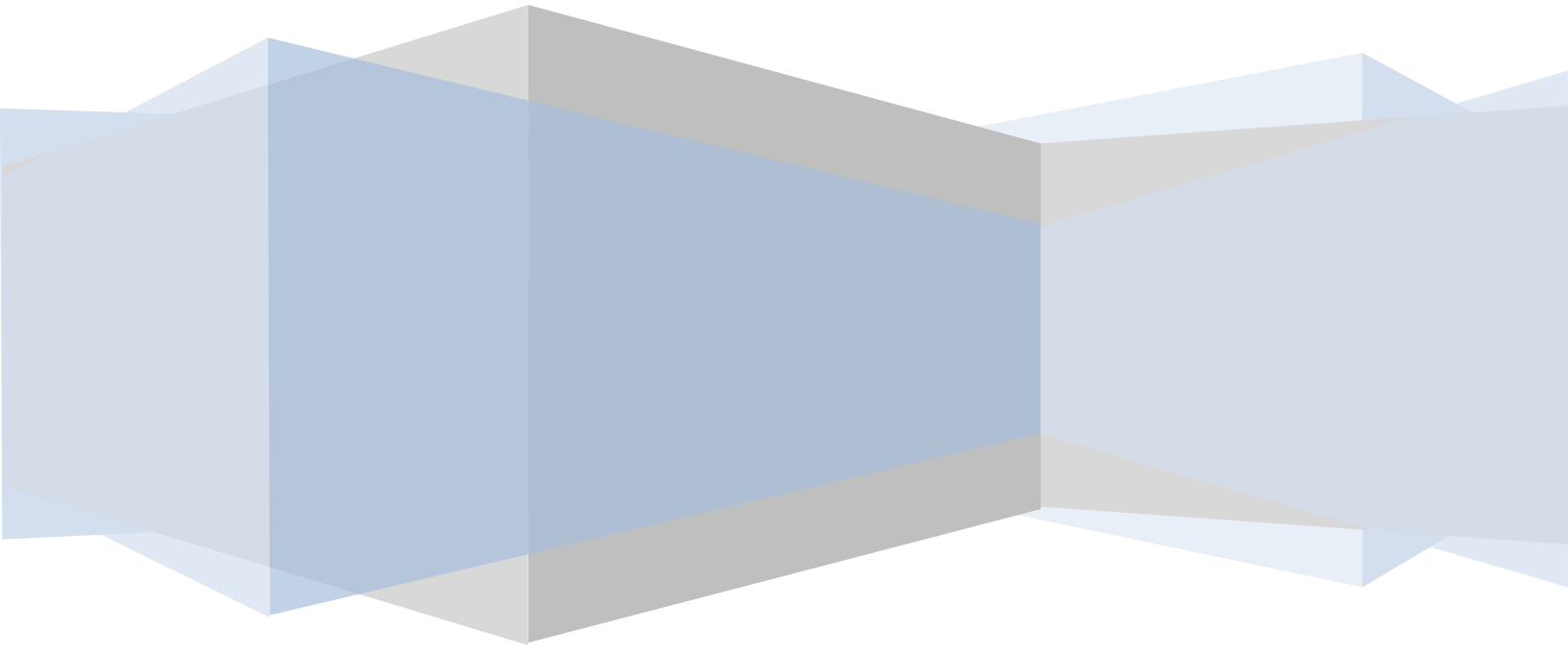
Colorado State University
Department Of Mechanical Engineering

MECH 417
Laboratory Exercise #2
Introduction to Simulink



Purpose

This laboratory exercise provides a tutorial introduction to Simulink. Simulink is a Matlab toolbox for simulating dynamic systems, and it will be used throughout the remainder of the course. All the exercises in this assignment should be done entirely in Matlab/Simulink.

1) Running, Plotting, Printing

In order to see a demonstration Simulink model, type *sldemo_househeat* at the Matlab prompt. Open the *scope* block, labeled "PlotResults" by double clicking, and then run the simulation using the green *Run* button or the *Simulation* menu. Print the plot of the simulation output (scope block) and print the simulation model itself. FYI, **an alternative to "printing" throughout this Lab is to copy the model window or use the Windows Snipping tool to copy a Scope window and past them into a Word document** so you can add any pertinent notes or comments and so you can format the document later for final printing.

2) Model Building

Figure 1 shows a Simulink model that represents a servomotor system, with a PID controller implemented in the feedback loop. Launch the Simulink library browser from within Matlab by using the *Simulink Library* icon or typing in: *simulink*. Then open a new model (using the *New Model* button or the *File* menu), and build the model in Figure 1. This is achieved by dragging components from the *Simulink Library Browser* window to the model window and connecting them using the mouse. Double clicking a box allows you to edit the component properties, for example entering polynomial coefficient values for the transfer function as shown. For the PID block, set the proportional gain to 0.05, and the integral and derivative gains to zero. **Don't change the filter coefficient (N=100), and don't use the "Tune" button in this Lab.**

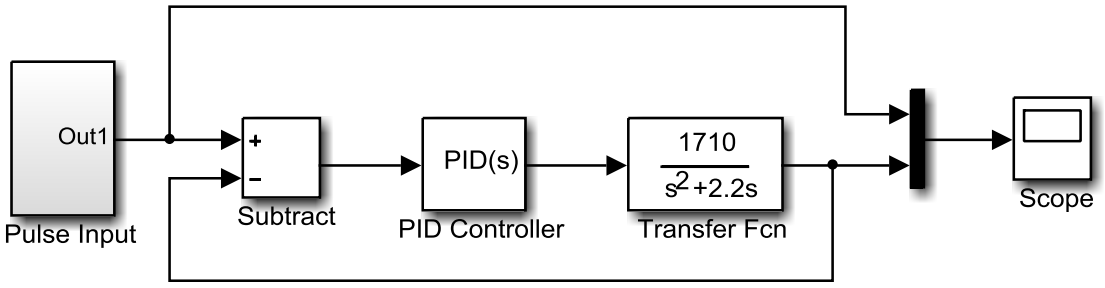


Figure 1: Simulink model of servomotor gear drive system

Hint: Standard building blocks are located in the *Simulink Library Browser* window. For example, the PID controller can be found in the *Continuous* library. Alternatively, you can search for things by entering keywords in the search box (e.g., "PID").

Browse through the many available blocks in Simulink. For this Lab and future work, you will typically use blocks in *Sources, Sinks, Continuous, and Math Operations* areas.

Note that there is no standard block for "Pulse Input," but one can be constructed from the basic components shown in Figure 2 by group-selecting the components (excluding the "Out1" component that is created for you), right clicking on the selection, and picking *Create Subsystem From Selection* in the menu. The components will be collapsed into a subsystem block which can be viewed or edited in the future by double-clicking on the block. The pulse should start at t=0 and last 4 seconds with an amplitude of 4000. Think about how to set the values on the two step-input blocks so they will create the desired pulse when they are added.

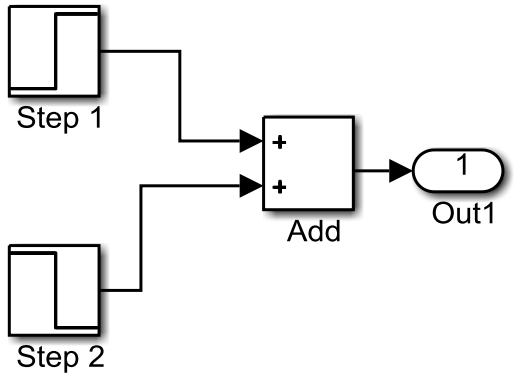


Figure 2: Simulink pulse input subsystem

Note – An alternative to building a pulse block as shown in Figure 2 is to simulate a pulse (starting at t = 0) with a single step input using a positive initial value and zero for a final value, but for practice, we want you to build and configure the pulse input subsystem.

After you complete the model, save it with the name "mymotor" (it will actually be saved as *mymotor.slx*). You can then launch this model later from Matlab simply by typing *mymotor* at the command line or by opening it from within Simulink. Before running the model, be sure to set the simulation stop time to 8 seconds under *Simulation - Model Configuration Parameters*. Run the simulation and print the results from the scope block. You should get a plot similar to Figure 3 which shows the commanded response and the actual system response. Note – You might need to use the auto scale button in the scope window to see the entire plot. Also, you can change the colors used in the Scope window by clicking on the *Parameters* gear icon. Also, you can get a smoother trace by setting the "Max step size" in the *Simulation - Model Configuration Parameters* dialog box to a small value (e.g., 0.01) and then re-running the model.

Having completed this exercise, you should have a model plot and a simulation run that essentially reproduce the figures shown in this document. Now try varying the parameters of the PID controller and see how they affect the closed-loop control system. (Note that you can enter variable names in Simulink Blocks if you like. Simulink will read the values from the Matlab workspace, where you can change the variables easily between your runs.) You do not need to generate large numbers of plots but plot a few of the most interesting results and discuss how the different controller parameters (Proportional, Integral, and Derivative) and values affect the closed loop performance. Start by changing the P parameter, and then add different amounts of D and I, starting with small values and increasing them, adjusting the P parameter as necessary also. Then see if you can manually "tune" the controller (by adjusting the parameters) to get what you consider a "good" step response, while attempting to keep the parameters as small as possible. **Do not use the "Tune" button in the PID block.** We want you to experiment with the PID parameters manually to get a feel for how the values and combinations change the response. Later in the

semester, we will learn how to determine PID parameters analytically to create the desired type of response.

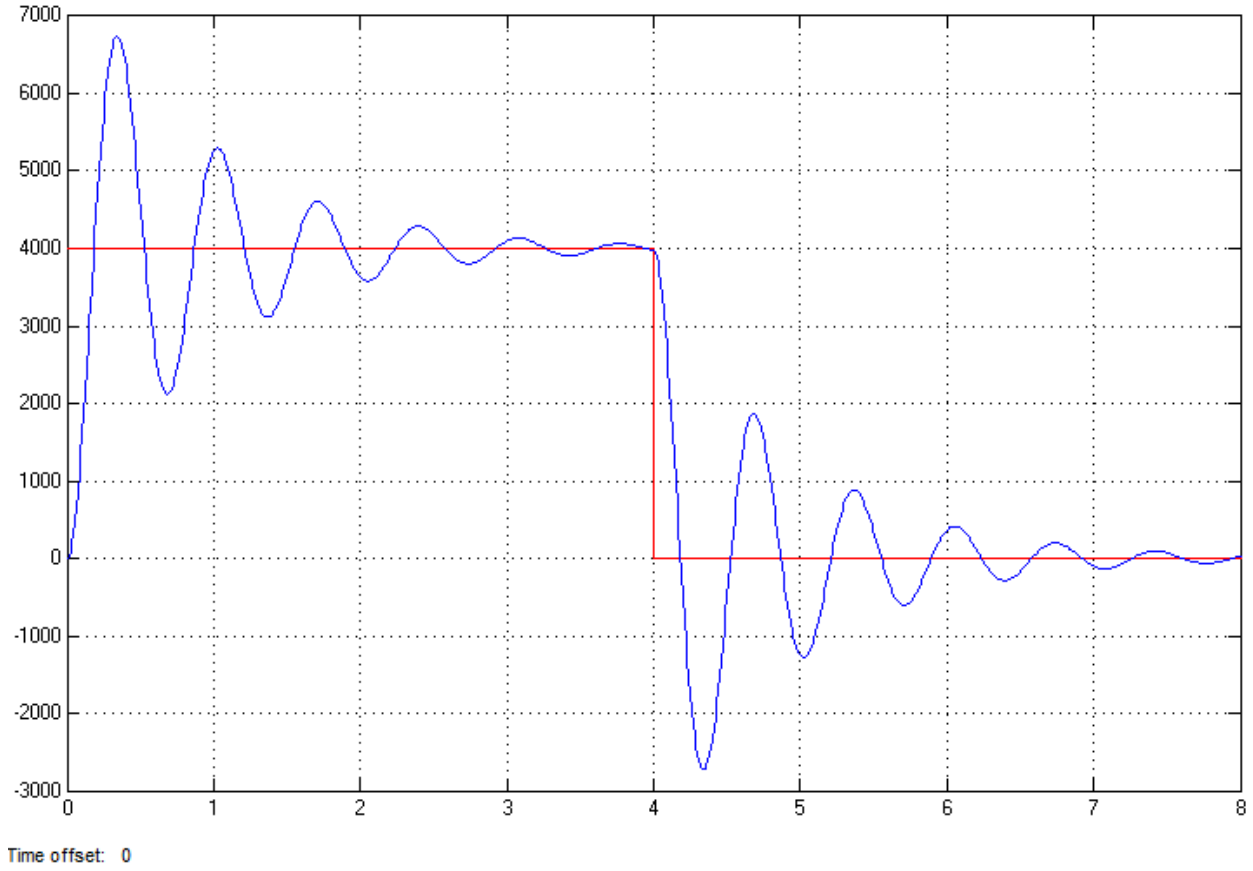


Figure 3: Simulation Scope block output

Summary of Documentation Requirements

Part 1

- Printout of the simulation model.
- Plot of the simulation results.
- Brief description of the model and results.

Part 2

- Block diagram.
- Brief description of how your pulse input block was created.
- 3 or more representative plots showing how the response changes with controller parameter values.
- Plot of a good step response with the final parameters you determined after manual tuning (while also keeping the parameters as small as possible).
- Explanation of why your tuned response is "good."
- Discussion of how changes in the controller parameters affect the closed-loop performance.