ECE-320 Lab 6: PID, I, and PD Control with Dynamic Prefilters

Overview

In this lab you will be controlling the one degree of freedom systems you previously modeled using PID, I, and PD controllers with and without dynamic prefilters. If you only have one type of system, you will use two different configurations of that system. If you have two kinds of systems (both a torsional and a rotational system, you will use one model of each).

You will need your Simulink model and the ***closedloop_driver.m*** *file from your homework for this lab.*

Design Specifications: *For each of your systems, you should try and adjust your parameters until you have achieved the following:*

Torsional Systems (Model 205)

- Settling time less than 1.0 seconds.
- Steady state error less than 2 degrees for a 15 degree step, and less than 1 degree for a 10 degree step (*the input to the Model 205 must be in radians!*)
- Percent Overshoot less than 25%

Rectilinear Systems (Model 210)

- Settling time less than 1.0 seconds.
- Steady state error less than 0.1 cm for a 1 cm step, and less than 0.05 cm for a 0.5 cm step
- Percent Overshoot less than 25%

As a start, you should initially limit your gains as follows:

$$\begin{aligned} k_p &\leq 0.5 \\ k_i &\leq 5 \\ k_d &\leq 0.01 \end{aligned}$$

Your memo should include eight graphs for each of the 1 dof systems you used (two different PID controllers, two PD controllers, two I controllers with and without dynamic prefilters.) Be sure to include the values of k_p, k_i, *and* k_d *and whether the PID controller had real zeros or complex conjugate zeros in the captions for each figure. Your memo should compare the difference between the predicted response (from the model) and the real response (from the real system) for each of the systems. How does the use of a dynamic prefilter change the response? Attach your Matlab driver file* ***closedloop_driver.m***

For each of your two 1 dof systems, you will need to go through the following steps:

Step 1: Set up the 1 dof system exactly the way it was when you determined its model parameters.

Step 2: Modify **closedloop_driver.m** to read in the correct model file. You may have to copy this model file to the current folder.

Step 3: Modify **closedloop_driver.m** to use the correct *saturation_level* for the system you are using.

Step 4: PID Control (complex conjugate zeros)

- Design a PID controller with complex conjugate zeros using sisotool to meet the design specs (you may have already done this in the homework). Use a **constant prefilter** (i.e., a number, most likely the number 1)

- Implement the correct gains into **closedloop_model.m**

- Simulate the system for 1.5 seconds. *Be sure to use radians for the Model 205 system!* If the design constrains are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)

- Compile the correct closed loop ECP Simulink driver, connect to the system, and run the simulation.

- Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on one nice, neatly labeled graph. The results for the torsional systems must be displayed in degrees. You need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*

- Change the prefilter to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation, recompile the ECP system, run the ECP system, and compare the predicted with the measured response. You also need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*

Step 5: PID Control (real zeros)

- Design a PID controller with real zeros using sisotool to meet the design specs (you may have already done this in the homework). Use a **constant prefilter** (i.e., a number, most likely the number 1)

- Implement the correct gains into **closedloop_model.m**

- Simulate the system for 1.5 seconds. *Be sure to use radians for the Model 205 system!* If the design constrains are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)

- Compile the correct closed loop ECP Simulink driver, connect to the system, and run the simulation.

- Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on one nice, neatly labeled graph. The results for the torsional systems must be displayed in degrees. You need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*

- Change the prefilter to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation, recompile the ECP system, run the ECP system, and compare the predicted with the measured response. You also need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*

Step 6: PD Control

- Design a PD controller with real zeros using sisotool to meet the design specs. (It may be difficult to meet the PO constraint, do the best you can.) Use a **constant prefilter** (i.e., a number). However, the number will probably not be 1! You will probably need to use *sisotool* to get reasonably close, but then use the prefilter to get the steady state error correct.

- Implement the correct gains into **closedloop_model.m**

- Simulate the system for 1.5 seconds. *Be sure to use radians for the Model 205 system!* If the design constrains are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)

- Compile the correct closed loop ECP Simulink driver, connect to the system, and run the simulation.

- Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on one nice, neatly labeled graph. The results for the torsional systems must be displayed in degrees. You need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*

- Change the prefilter to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation, recompile the ECP system, run the ECP system, and compare the predicted with the measured response. You also need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*

Step 7: I Control

- Design an I controller with real zeros using sisotool to meet the design specs (It may be difficult to meet the settling time constraint, do the best you can.) Use a **constant prefilter** (i.e., a number)

- Implement the correct gains into **closedloop_model.m**

- Simulate the system for 1.5 seconds or until your system comes to steady state. *Be sure to use radians for the Model 205 system!* If the design constrains are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)

- Compile the correct closed loop ECP Simulink driver, connect to the system, and run the simulation.

- Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on one nice, neatly labeled graph. The results for the torsional systems must be displayed in degrees. You need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*

- Change the prefilter to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation, recompile the ECP system, run the ECP system, and compare the predicted with the measured response. You also need to include this graph in your memo. *Be sure to include the values of* k_p, k_i, *and* k_d *in your memo.*