ECE-520 Lab 3
Discrete-Time PID and PI Controllers and sisotool

Overview

In this lab you will be controlling both of the one degree of freedom systems you previously modeled using discrete-time PID and PI controllers. Both one degree of freedom systems must be controlled, and if there are two people in your lab group each lab partner should do a different system.

You will need your Matlab model files for you systems, the files ***DT_PID_driver.m*** *and* ***DT_PID.mdl*** *from your homework, and* ***Model210_DT_PID.mdl*** *and* ***Model205_DT_PID.mdl*** *from the class website. You will also need the files from the basic files group (****ECPDSPReset.c, ECPDSPReset.dll, ECPDSPRest.c, ECPDSPDriver.c, ECPDSPDriver.dll, ECPDSPDriver.h****)*

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 1.5 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 1.5 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%

Your memo should include ***four*** *graphs* ***for each*** *of the 1 dof systems you used (a PID controller and a PI controller for each system, two different sampling intevals), and* ***one*** *final graphs when you include a lowpass filter in one of the PID controlled systems.. 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.*

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 **DT_PID_driver.m** to read in the correct model file. You may have to copy this model file to the current folder.

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

Step 4: Set the sampling interval to 0.1 seconds (the first time) and then 0.05 seconds (the second time).

Step 5: PID and PI Control (see the *Hints* at the end of the lab)

- Design a PID and then a PI controller 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 **DT_PID_driver.m**
- Simulate the system for 3.0 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). Try and stay away from the maximum allowed control values as much as possible, they are not as good a predictor with discrete-time systems as with continuous time systems.
- Reset the system using **ECPDSPReset.mdl**
- Compile the correct closed loop ECP Simulink driver (**Model210_DT_PID.mdl** or **Model205_DT_PID.mdl**), connect to the system, and run the simulation.
- Use the **Compare_DT1.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.

Step 6: It is generally a very bad practice to take the derivatives of signals without first smoothing (or lowpass filtering) the signals. For one of your systems insert a discrete-time filtering block before the derivative is estimated. Use a simple four point running averaging filter: $H(z) = 0.25 + 0.25z^{-1} + 0.25z^{-2} + 0.25z^{-3}$. With this filter, rerun one of your systems using the PID controllers, and turn in your results and a copy of your Simulink model file (.mdl). (*Note: you may need to set the sampling time in the filter to Ts*).

PID Controller Hints

For stability, ***all poles of the system must be within the unit circle.*** However, zeros can be outside of the unit circle.

The closer to the origin your dominant poles are, the faster your system will respond.

<u>Controllers:</u>

Proportional (P) : $C(z) = K_p E(z)$

Integral (I) : $C(z) = \frac{K_i}{1 - z^{-1}} = \frac{K_i z}{z - 1}$

Derivative (D) : $C(z) = K_d (1 - z^{-1}) = \frac{K_d (z-1)}{z}$

<u>Proportional+Integral+Derivative (PID)controller:</u>

$$C(z) = K_p + \frac{K_i z}{z-1} + \frac{K_d (z-1)}{z} = \frac{K_p z(z-1) + K_i z^2 - K_d (z-1)^2}{z(z-1)} = \frac{(K_p + K_i + K_d) z^2 + (-K_p - 2K_d) z + K_d}{z(z-1)}$$

sisotool form: $C(z) = \frac{K(z^2 + az + b)}{z(z-1)}$

parameter extraction: $K_d = Kb, \quad K_p = -Ka - 2K_d, \quad K_i = K - K_p - K_d$

<u>Proportional+Integral (PI) controller:</u>

$$C(z) = K_p + \frac{K_i z}{z-1} = \frac{(K_p + K_i) z - K_p}{z-1}$$

sisotool form: $C(z) = \frac{K(z^2 + az)}{z(z-1)} = \frac{K(z+a)}{(z-1)}$

parameter extraction: $K_p = -Ka, \quad K_i = K - K_p$