Lab 6: PI-D and I-PD Control with Dynamic Prefilters

Overview

In this lab you will be controlling the one degree of freedom systems you previously modeled using PI-D and I-PD controllers with and without dynamic prefilters.

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%

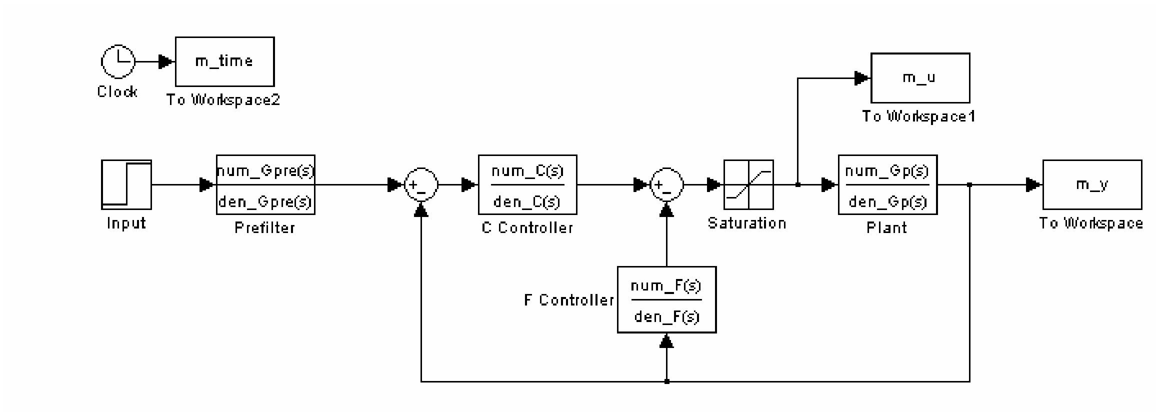
As a start, you should initially limit your gains as follows:

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

Your memo should include four graphs for each of the 1 dof systems you used (one PI-D and one I-PD controller with and without dynamic prefilters.) Be sure to include the values of k_p, k_i, and k_d 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***

Background: While PID controllers are very versatile, they have a number of drawbacks. One of the major drawbacks is that for a unit step input, the control effort $u(t)$ can be infinite at the initial time. This is referred to as a *set-point kick*. There are two commonly used configurations of PID controls schemes that utilize a different structure, the PI-D and the I-PD controllers. These are a bit more difficult to model using Matlab's *sisotool*, but it can be done and we get to explore more of *sisotool*.

The PI-D controller avoids the set-point kick by putting the derivative in the feedback path, while the I-PD controller avoids the set-point kick by placing both the derivative and proportional terms in the feedback path. Both types of controllers can be implemented using the following Simulink model, which you should construct and name appropriately:



For the PI-D controller, we have

$$C(s) = k_p + \frac{k_i}{s}, \quad F(s) = \frac{k_d s}{\frac{1}{50}s + 1}$$

while for the I-PD controller we have

$$C(s) = \frac{k_i}{s}, \; F(s) = k_p + \frac{k_d s}{\frac{1}{50}s + 1}$$

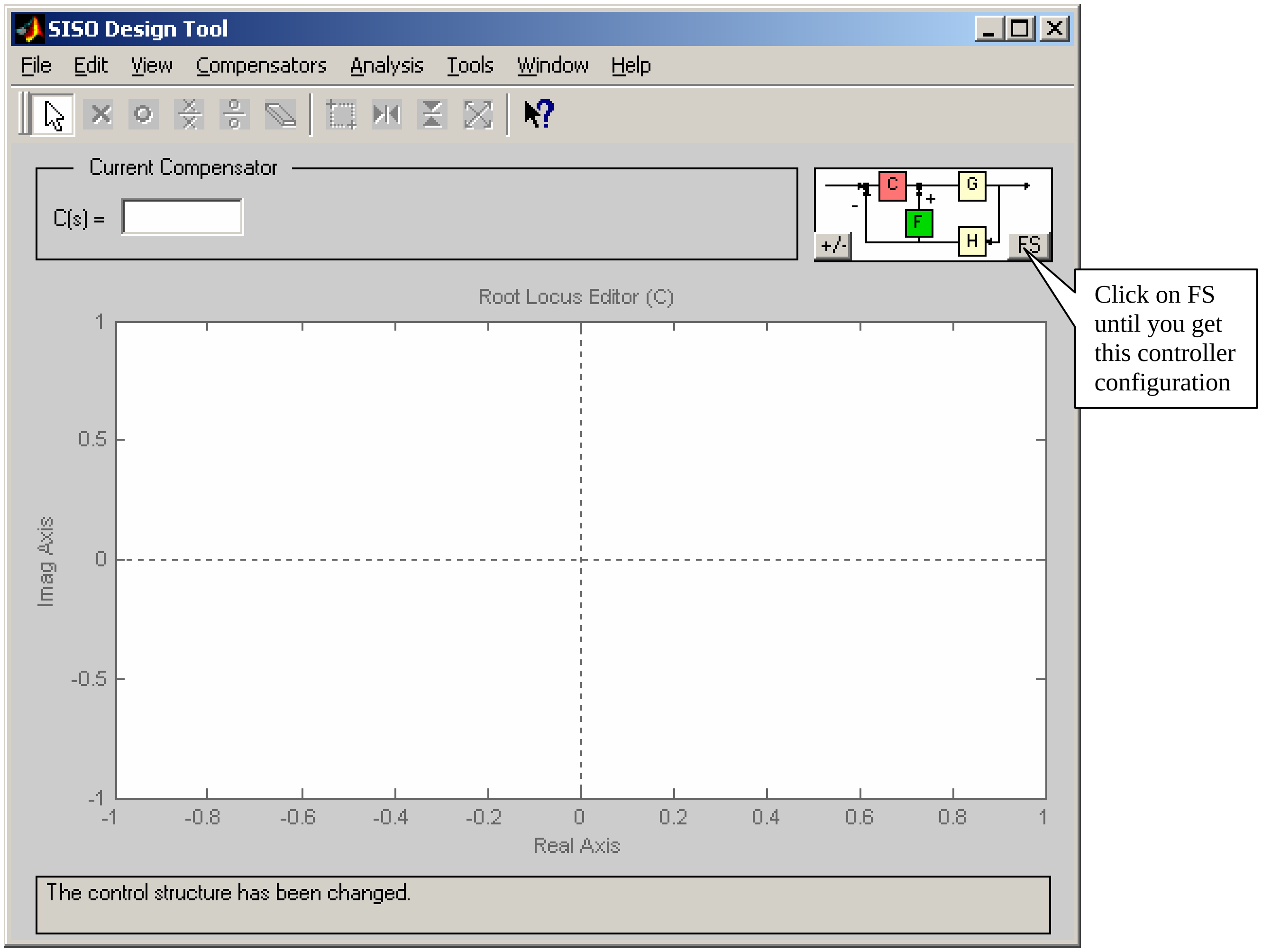
Note that for both controllers we continue to use a lowpass filter (with a cutoff of 50 rad/sec) in series with a differentiator.

For both of these controllers, if we ignore the prefilter (assume it is unity), the transfer function from input to output is

$$\frac{Y(s)}{R(s)} = \frac{C(s)G_p(s)}{1 + F(s)G_p(s) + C(s)G_p(s)}$$

Next we need to use *sisotool* to help determine reasonable values for k_p, k_i, and k_d.

When you start sisotool, you need to click on FS to get the proper configuration. Note that sisotool uses a + sign after the $F(s)$, where we are using a – sign.

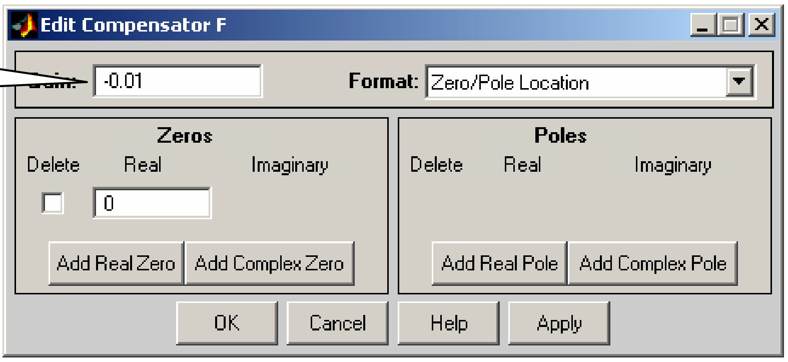

SISO Design Tool
File Edit View Compensators Analysis Tools Window Help
Current Compensator
C(s) =
C
G
F
H
+/-
FS
Click on FS until you get this controller configuration
Root Locus Editor (C)
Imag Axis
Real Axis

The control structure has been changed.

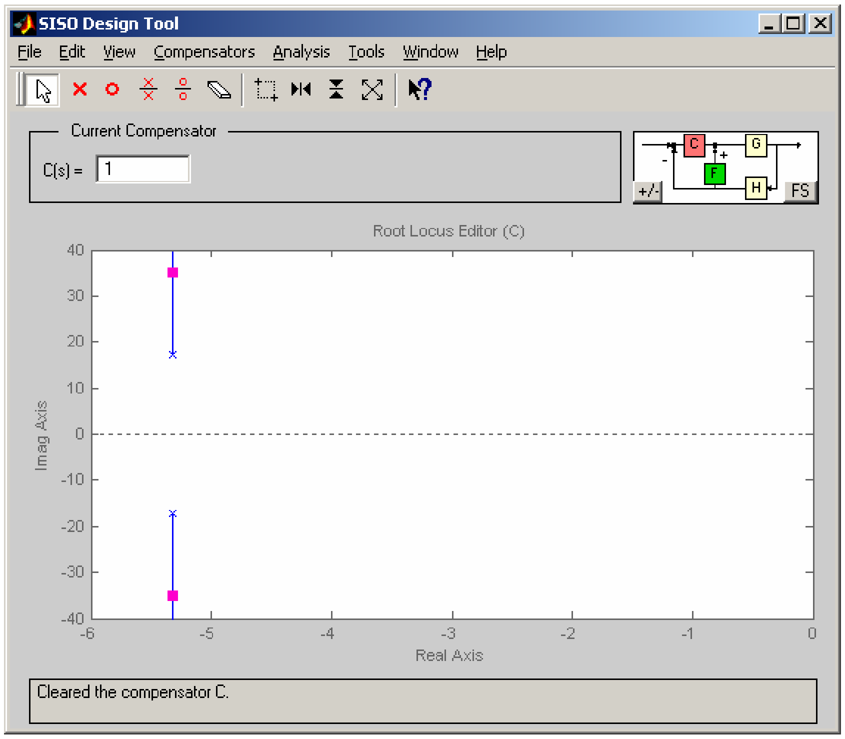
Sisotool will allow us to modify both $C(s)$ and $F(s)$, but it is much easier to modify $C(s)$. We will have to iterate between both of these.

For practice, let's assume our plant is $G_p(s) = \dfrac{938.4}{s^2 + 1.25s + 329.8}$ and we want to use a PI-D controller. In this case, for simulation purposes, we have $F(s) = k_d s$ (we won't worry about the lowpass filter here) . We will start off assuming $k_d = 0.01$, but this is just a guess! However, since we have the wrong sign on the summer after $F(s)$, we enter the gain as a negative number in sistool. Hence for this guess we enter a gain of -0.01 and a zero at 0 (which gives us a differentiator). This is shown below :

Note the gain is negative here!



After entering this into sisotool (and leaving $C(s) = 1$, the default), we get the following root locus plot:



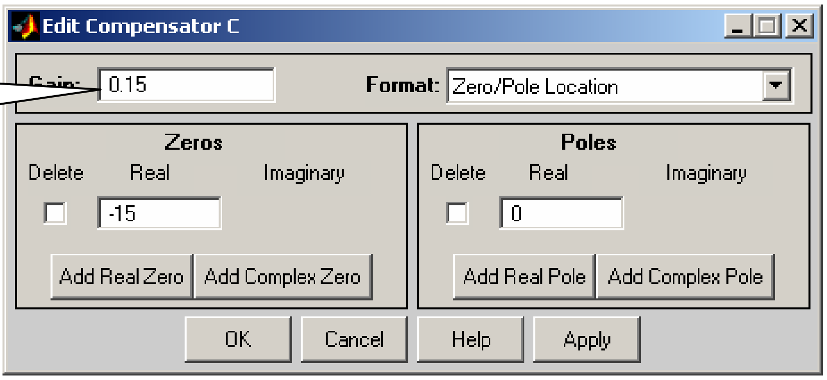
Cleared the compensator C.

What's important to notice here is that *we do not see any of the effects of our differentiator (we expect to see a zero at the origin)!*

Next we'll enter the PI part of the controller into $C(s)$. As a starting point, let's assume

$$C(s) = \frac{0.15(s+15)}{s}$$

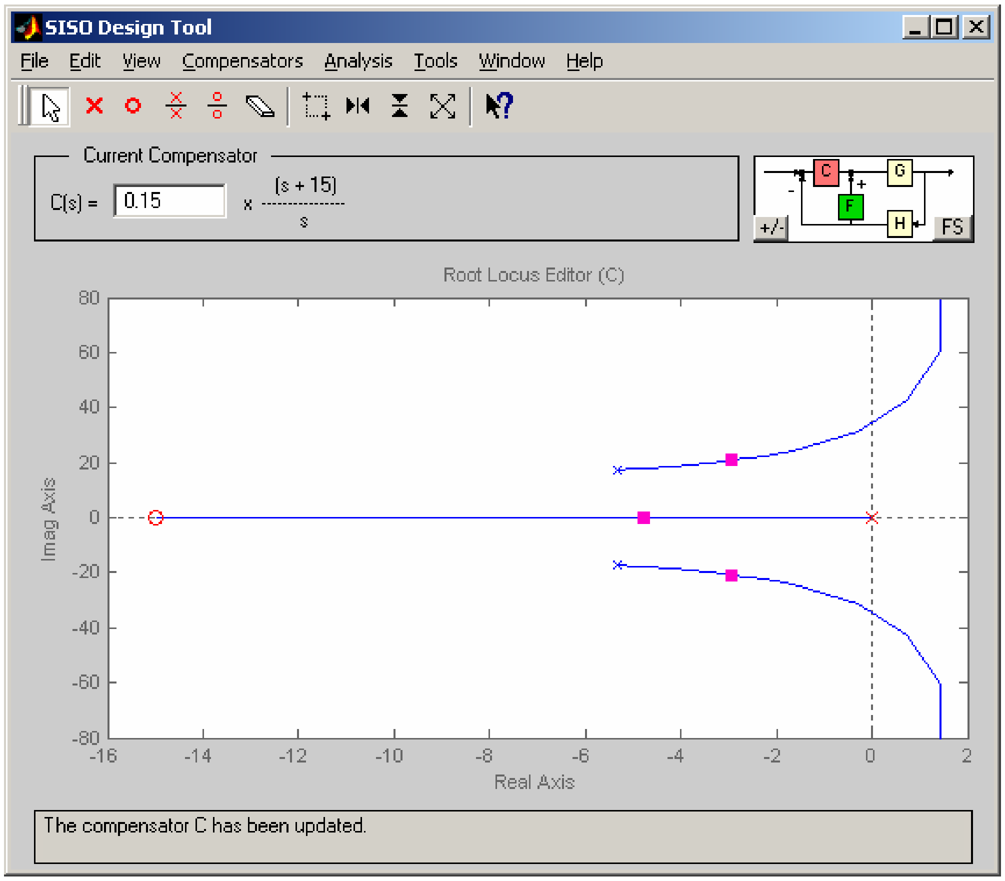
The gain is not negative here



Edit Compensator C

Gain: 0.15 Format: Zero/Pole Location

We will get the following root locus plot



SISO Design Tool

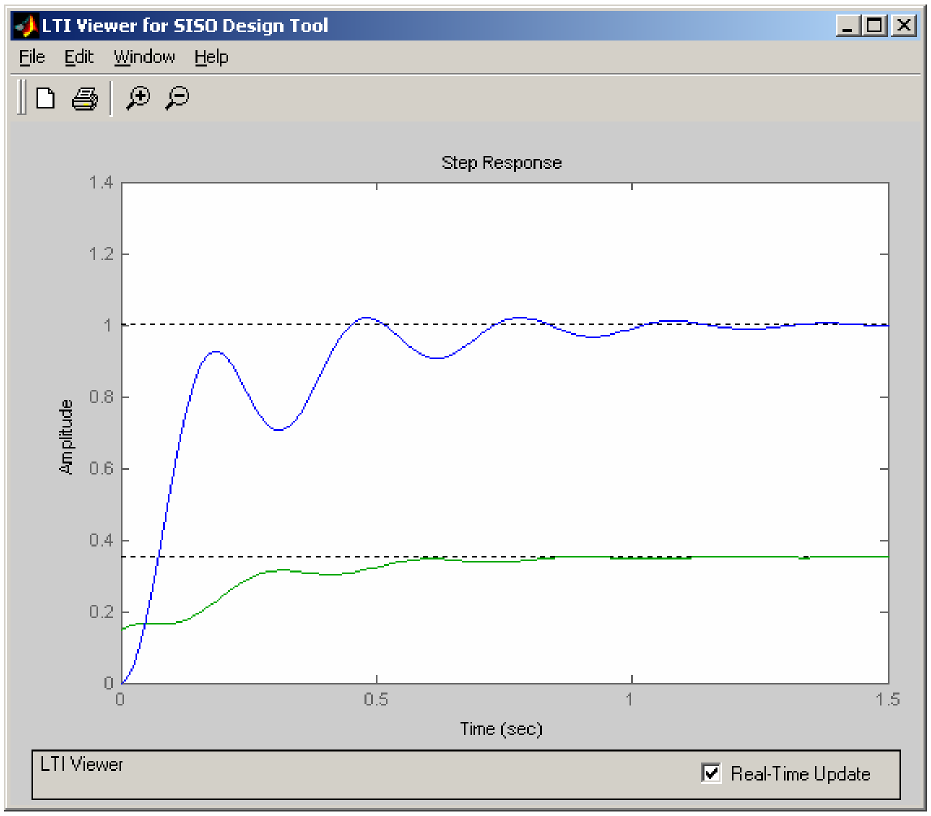
Current Compensator

C(s) = 0.15 x (s + 15) / s

The compensator C has been updated.

Here we can see the pole at the origin and the zero at -15 due to $C(s)$. This looks like the root locus plot for a PI controller. At this point we can drag the zero around and also drag the red squares around (which change the gain value). However, we can do nothing to change $F(s)$ except to edit it.

The step response for this controller configuration is:



Note that, compared to a normal PID controller, the control effort is not infinite at 0, and actually builds as time goes on (like an integral controller). At this point we might want to go back modify $C(s)$ to see if we get acceptable performance. If we cannot get acceptable performance we may then have to try another value for $F(s)$.

If we assume these controllers are OK, then for our systems we will enter

$$C(s) = \frac{0.15(s+15)}{s}, \quad F(s) = \frac{0.01s}{\frac{1}{50}s+1}$$

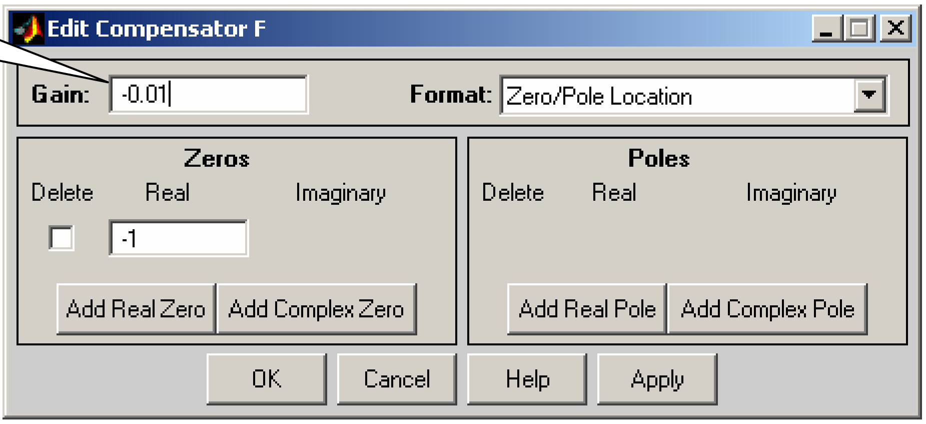
Note that we now use the positive coefficient for $F(s)$.

Now we'll assume we want to control the same plant, but this time use an I-PD controller. We first have to guess a PD controller. For the systems I tired, I found that the zero should be fairly small, between -1 and -5. As a start, we'll try the following PD controller

$$F(s) = 0.01(s+1)$$

When we implement this controller in sisotool, we need to be sure the gain is negative, as shown below:

Note the gain is negative

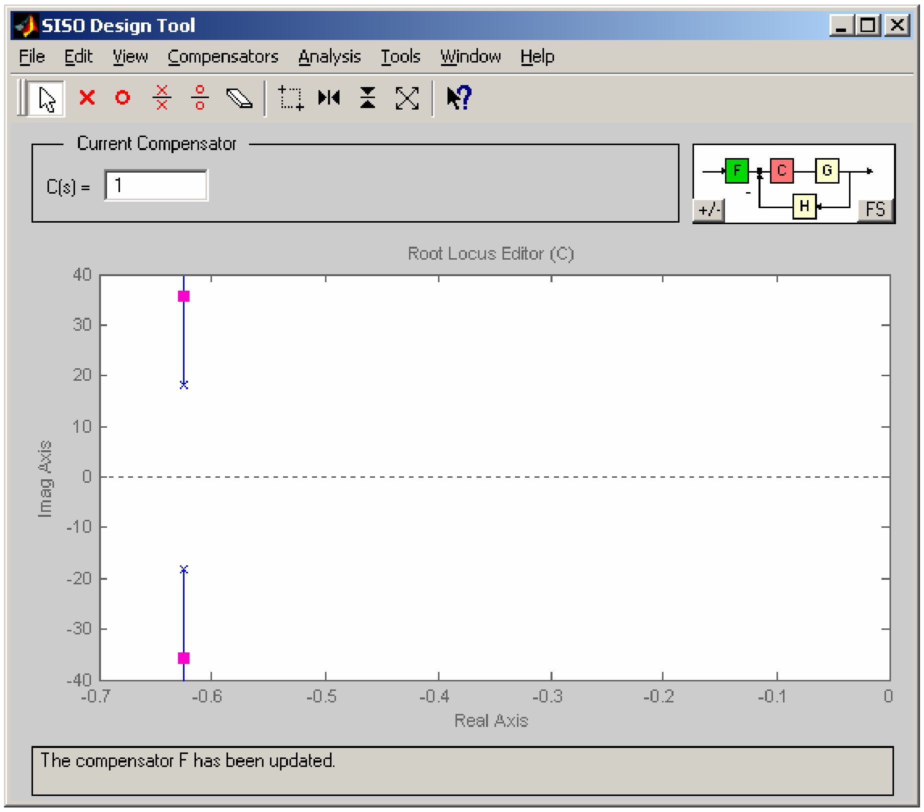


Edit Compensator F

Gain: -0.01

Format: Zero/Pole Location

The root locus plot at this point ($C(s) = 1$) is shown below



SISO Design Tool

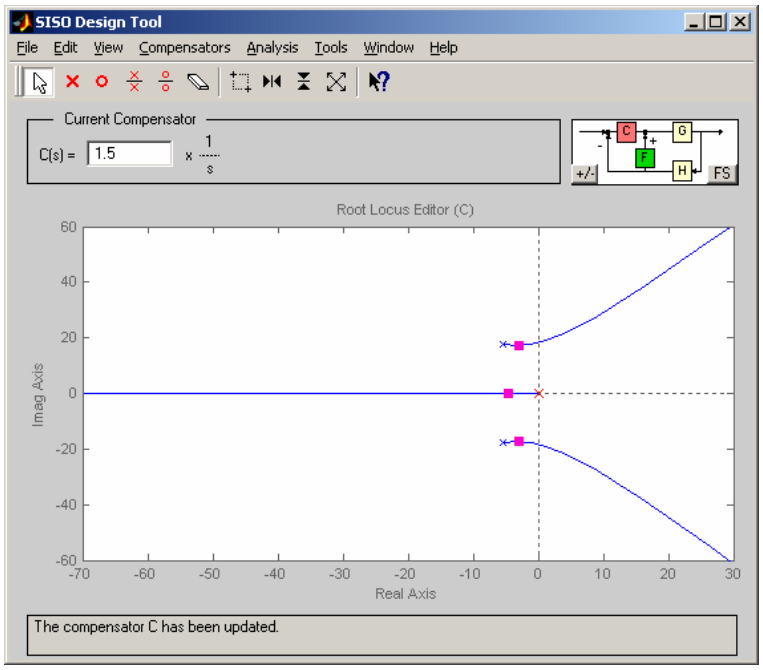
Current Compensator

C(s) = 1

The compensator F has been updated.

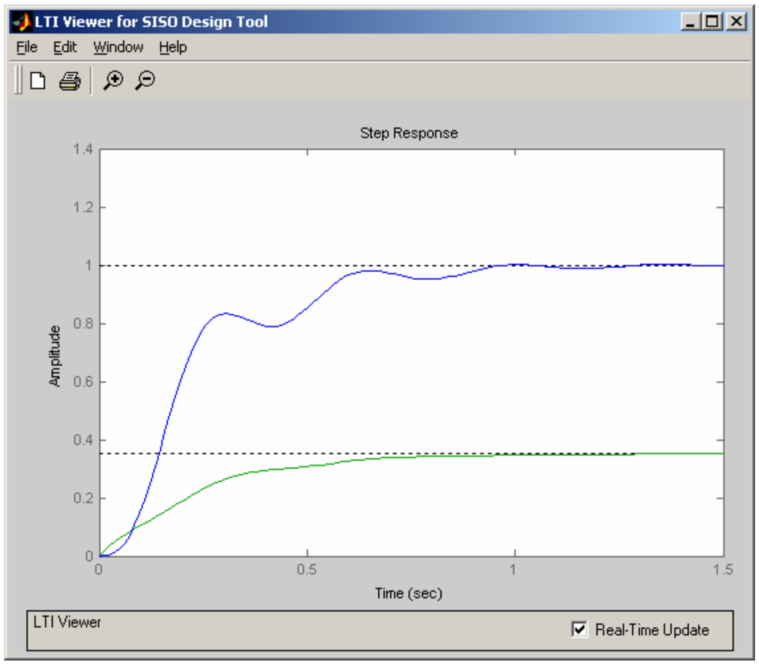
Note that, again, we cannot see any of the poles or zeros due to $F(s)$.

Next we have to try an I controller for $C(s)$. Let's assume $C(s) = \frac{1.5}{s}$. After entering this into sisotool, we have the following root locus plot.



The compensator C has been updated.

Note that this looks like the root locus plot for an I controller! The step response for this configuration is shown below. Note that the control effort is very small at the initial time. If the step response is not acceptable, first try modifying $C(s)$ then $F(s)$.



To implement this controller in Simulink, we would have

$$C(s) = \frac{1.5}{s},\ F(s) = 0.01 + \frac{0.01s}{\frac{1}{50}s + 1}$$

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

<u>Step 2:</u> Modify **closedloop_driver.m** to read in the correct model file and implement the new structure. In particular, it must now determine C(s) and F(s), as well as the prefilter.

<u>Step 3:</u> Modify the ECP driver file and rename appropriately. You cannot use **Model205_Closedloop.mdl** or **Model210_Closedloop.mdl** as before.

<u>Step 4:</u> PI-D Control

- Design a PI-D controller 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). Be sure to observe the limits on the other gains.

- 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 position 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: I-PD Control

- Design an I-PD controller 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). Keep the zero of the PD controller small, between -1 and -5, and be sure to observe the limits on the other gains.

- 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 position 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.*