Lab 5: ITAE, Deadbeat, and Quadratic Optimal Control

Overview

In this lab you will be controlling the systems you previously modeled using an ITAE, a deadbeat, and quadratic optimal controller (one of each type of controller.). For each system you modelled, you need to plot and examine the difference between how the model predicts the response and how the system actually responds.

If you modeled two rectilinear 1 dof systems, start with the rectilinear systems. If you modeled two torsional 1 dof systems, start with the torsional systems.

You will need to set up a folder for Lab 5 and copy all files from the folder ***basic_files*** *into this folder. You will need your Simulink model and the* ***closedloop_driver.m*** *file from your homework for this lab.*

Special note: When comparing the predicted response and the actual response, I do not want to see a graph that shows very little. For example, if both the predicted response and the real response have settled within 0.5 seconds, I do not want to see a graph that goes out to 4 seconds.

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 0.5 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 10%

Rectilineat Systems (Model 210)

- Settling time less than 0.5 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 10%

You should start with the lower step values, and work your way up to the higher step values. Not all systems or controllers will work for a 1 cm step or a 15 degree step. Your real systems may oscillate a bit. If this happens try to reduce the input level, since this limits the allowed control effort. *It may not be possible to eliminate all of the oscillations, since these types of controllers depend on canceling the plant dynamics,* and if you model is not accurate enough the controller will not cancel the real plant well enough. Do the best you can, *as though your grade depended on it.*

PART A

*Changes to **closedloop_driver.m***

Note: You may not need to make all of the changes below. It depends on how much you may have modified this file. It will speed up lab time if these changes are made before lab.

You should end up with a section of code which includes all the different types of controllers. You will comment out the types you are not using, and will be adding to these in the next few labs.

Step 1: Modify **closedloop_driver.m** to read in your state model file. For example, if the model file was named **state_model_1dof.mat** you need the following near the beginning of your Simulink file

```
load state_model_1dof
C = [1 0];
Gp = ss2tf(A,B,C,D);
```

Step 2: Modify **closedloop_driver.m** by adding the **abs** function to fix the following ``feature'' (not an error on my part, since I never make errors)

```
num_Gp = (abs(num_Gp) > tol*ones(1,length(num_Gp))).*num_Gp;
den_Gp = (abs(den_Gp) > tol*ones(1,length(den_Gp))).*den_Gp;
```

Step 3: Modify the **saturation_level** variable in **closedloop_driver.m** for the appropriate system.

```
saturation_level = 1000/2196;  % (rectilinear system, Model 210)
saturation_level = 1000/2546;  % (torsional system, Model 205)
```

Comment out these lines until you need them.

Step 4: Modify **closedloop_driver.m** for utilizing ITAE model matching control. Here, the desired closed loop transfer functions are:

$$G_o(s) = \frac{\omega_o^2}{s^2 + 1.4\omega_o s + \omega_o^2}$$

$$G_o(s) = \frac{\omega_o^3}{s^3 + 1.75\omega_o s^2 + 2.15\omega_o^2 s + \omega_o^3}$$

$$G_o(s) = \frac{\omega_o^4}{s^4 + 2.1\omega_o s^3 + 3.4\omega_o^2 s^2 + 2.7\omega_o^3 s + \omega_o^4}$$

You should enter these into **closedloop_driver.m**, but leave the frequency ω_o a variable.
Comment out these lines until you need them.

Step 5: Modify **closedloop_driver.m** for utilizing deadbeat model matching control. Here the desired closed loop transfer functions are:

$$G_o(s) = \frac{\omega_o^2}{s^2 + 1.82\omega_o s + \omega_o^2}$$

$$G_o(s) = \frac{\omega_o^3}{s^3 + 1.90\omega_o s^2 + 2.20\omega_o^2 s + \omega_o^3}$$

$$G_o(s) = \frac{\omega_o^4}{s^4 + 2.20\omega_o s^3 + 3.5\omega_o^2 s^2 + 2.8\omega_o^3 s + \omega_o^4}$$

You should enter these into **closedloop_driver.m**, but leave the frequency ω_o a variable. *Comment out these lines until you need them.*

Step 6: Modify closedloop_driver.m to utilize quadratic optimal control, similar to your homework. You should add the lines

```
q = 1;  % or whatever you want
Go = solve_quadratic(Gp,q);
```

Comment out these lines until you need them.

Step 7: Modify the Go obtained in Step 6 so that Go(0) = 1.
Comment out these lines until you need them.

Step 8: Modify **closedloop_driver.m** to determine the controller. For model matching control, this will be

```
Gc = minreal(Go/(Gp*(1-Go))
```

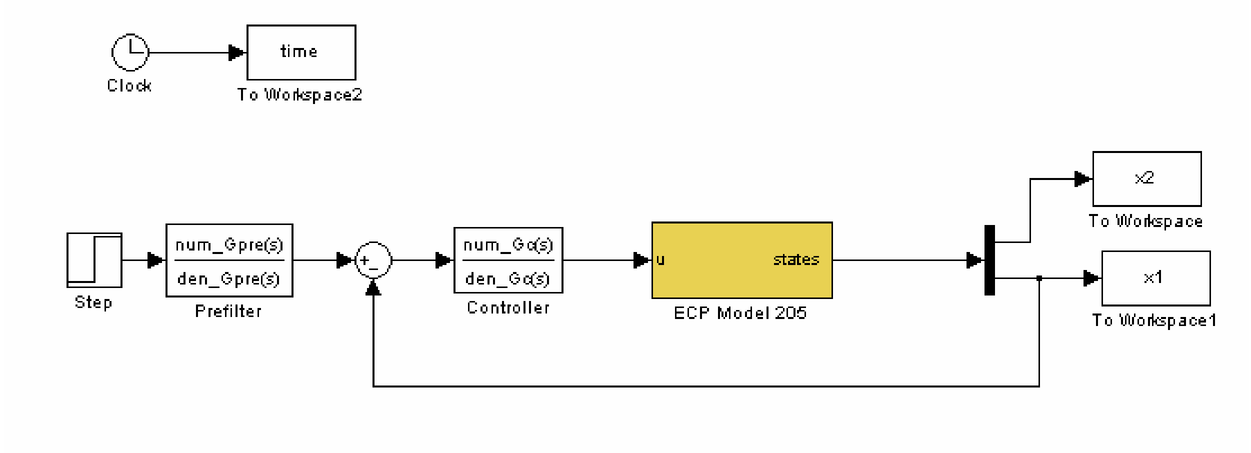
Be sure to use the **minreal** command.

Step 9: Be sure **closedloop_driver.m** computes the prefilter gain, Gpf. You did this in lab 1.

PART B

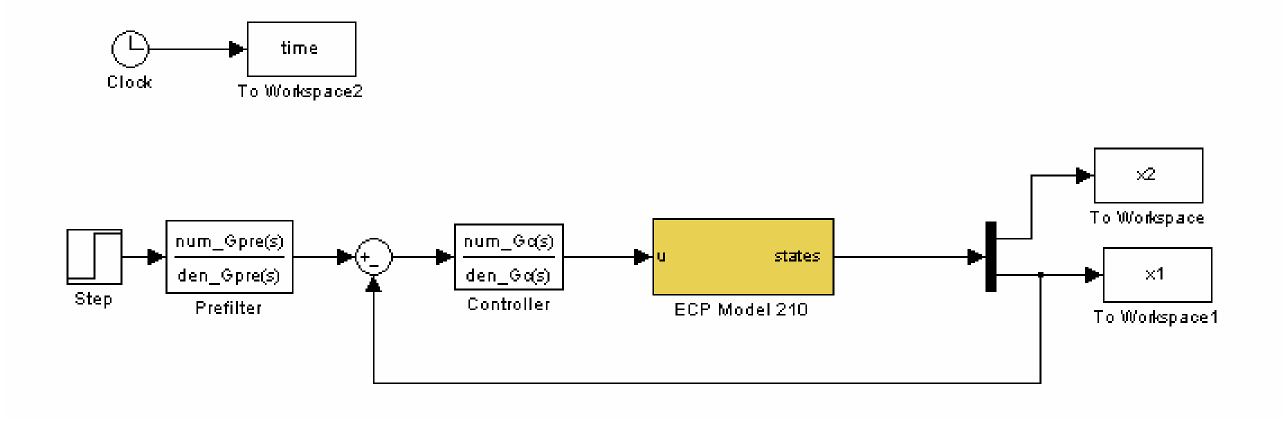
Creating Closed Loop Simulink Systems for the ECP Systems

Step 1: Open **Model205_Openloop.mdl** and save it as **Model205_Closedloop.mdl.** Then modify **Model205_Closedloop.mdl** so it looks like the following



This is a cross between **closedloop.mdl** and **Model205_Openloop.mdl.**, so you can pretty much just cut and paste. Note that if you want to scale to degrees, you must do this after the feedback branch.

Step 2: Open **Model210_Openloop.mdl** and save it as **Model210_Closedloop.mdl.** Then modify **Model210_Closedloop.mdl** so it looks like the following



This is a cross between **closedloop.mdl** and **Model210_Openloop.mdl.**, so you can pretty much just cut and paste.

PART C

Control of One Degree of Freedom Systems

For each of your three 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 it's 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: ITAE Control. *(Be sure to record the value of ω_o you use.)*

- Vary ω_o until you meet the design specs with the *simulation.* Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.
- 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.

Step 5: Deadbeat Control. *(Be sure to record the value of ω_o you use.)*

- Vary ω_o until you meet the design specs with the *simulation*. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.
- 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.

<u>Step 6:</u> Quadratic Optimal Control. *(Do not use the code that modifies Go(s) so that Go(0) =1, use a constant prefilter instead. Be sure to record the value of q you use.)*

- Vary *q* until you meet the design specs with the *simulation*. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.

- 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.

<u>Step 7:</u> Quadratic Optimal Control. *(Use the code that modifies Go(s) so that Go(0) =1, be sure to record the value of q you use.)*

- Vary *q* until you meet the design specs with the *simulation*. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.

- 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.

PART D

Control of a Two Degree of Freedom System

Note: It is much more difficult to control a two degree of freedom system, as you will probably see (especially those of you lucky enough to have the torsional system.) We will only be trying to control the position of the second cart/disk in this part.

Step 1: Set up the 2 dof system exactly the way it was when you determined it's 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: Modify **closedloop_driver.m** so the position of the second cart/disk is the output. This means you need to use

```
C = [0 0 1 0];
```

Step 5: Modify your ECP Simulink driver file so that the output is from the position of the second cart/disk. *This means the position of the second cart/disk is what is in the feedback loop.*

Step 6: ITAE Control *(be sure to record the value of ω_o you use.)*

- Vary ω_o until you meet the design specs with the *simulation*. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.

- 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. Remember you are trying to control the position of the second cart/disk, not the first!

<u>Step 7:</u> Deadbeat Control. *(Be sure to record the value of ω_o you use.)*

- Vary ω_o until you meet the design specs with the *simulation*. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.

- 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.

<u>Step 8:</u> Quadratic Optimal Control. *(Use the code that modifies Go(s) so that Go(0) =1, be sure to record the value of q you use.)*

- Vary *q* until you meet the design specs with the *simulation*. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.

- 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.

Your memo should include four graphs for each of the 1 dof systems you used, and three graphs for the 2 dof system. The values of q or ω_o used in the controller must be in the figure captions or as part of the graph (not handwritten). The figures should all be attachments, at least two per page. 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. You should discuss the difference between the performance of the quadratic optimal system when the system is a type 0 system and when it is a type 1 system. Attach your Matlab driver file ***closedloop_driver.m***