ECE-520 Lab 6

State Variable Feedback and Integral Control For One and Two Degree of Freedom Systems

Overview

In this lab you will be utilizing state variable feedback control to place the poles in a closed loop system to improve the performance of your open loop one and two degree of freedom systems. In addition, you will be incorporating integral control to try and produce zero steady state error for a step input.

For each of the systems you use in this lab (and for the remainder of the labs in this course) you will go through the following basic procedure:

1) *Modify the Simulink driver you are using to load the mathematical model file (.mat file) that corresponds to the way you have the system configured.*
2) *Simulate the system to determine if your model meets the desired specifications. If it does not, modify the pole locations until it does meet the specifications. In addition, you need to be sure the control effort does not reach the saturation level. The simulated control effort for the discrete-time system does not model the real control effort as well as it does for the continuous time system. It would work better if we were to sample at a higher rate. Hence, if your control effort is near the saturation level it is not likely to work well.*
3) *Once you have simulated the system all of the variables you used are in Matlab's workspace. Now compile the correct ECP driver file that replaces the model of the ECP system with the ECP system driver (and the real ECP system). Reset the ECP system, and run the system.*
4) *Finally, compare the predicted response (from your mathematical model) with the real response (from the ECP system). A graph showing the predicted and real response is to be included in your memo (as an attachment) for each system you simulate.*

Notes and Guidlines:

- Although it should not matter, only use *positive* pole locations. Apparently the ECP system are not particularly happy with negative pole locations.

- Start with poles at around 0.5 or 0.6, and then move them in closer once you see how the system is responding. We are not trying to make the systems go particularly fast here, but just see how discrete-time control systems work.

- The ECP systems really do not like poles at the origin, so don't put any poles there (no deadbeat control.)

- Run the systems for at least one second, but don't run the system so long most of your graph shows the system at steady state.

- Reset the system each time before you run it.

- As soon as you start you controller (click on play) be prepared to stop the system. In particular, listen for vibrations that are growing louder and stop the system as soon as possible after this.

Design Specifications

For each of your systems try and have the simulated systems meet the following design specifications

a) Settling time less than or equal to one second
b) Steady state error is zero for a 1 cm step input (or a 15 degree step input)
c) Percent overshoot less than 20%

You should try for the 1cm (or 15 degree) inputs, but if your system is unwilling to cooperate try a 0.5 cm input (or a 10 degree input). This is particularly true when trying to control the position of the second cart/disk.

You system should (eventually) reach the correct steady state level. It won't be exact due to measuring errors, but it should be close. Remember we are only guaranteed a zero steady state error at some point in time, it may take a while for the system to actually get to steady state.

Part A: One Degree of Freedom Rectiliner Systems

a) Load the files from the **Lab 5** folder into a folder for **Lab 6**.
b) Determine which of the models you created for the one degree of freedom rectilinear system seemed to work best last week. *Use only that model. (You only need to use one model per system.)*
c) Configure your one degree of freedom system the way you did in **Lab 1**.
d) Modify the Matlab driver file **DT_sv1_servo_driver.m** to read in the appropriate mathematical model.
e) Use state variable feedback coupled with an integral controller to place the poles in such a way that you think you will meet the system requirements given above. Remember there is usually a tradeoff between speed of response and the required control effort, which is limited by our motors.
f) Run the simulation for at least one second (*Tf should be at least one second*). This is because the ECP systems tend to hang up if they run for less than a second.
g) Simulate the systems, check to see that they meet the design requirements and the control effort does not reach the saturatation level. If there is a problem, go to step (e) and try a different set of pole locations.
h) Create the file **Model210_DT_sv1_servo.mdl** to implement the controller. The easiest way to do this is as follows:
    - open up your file **DT_sv1_servo.mdl** and then save it as **Model210_DT_sv1_servo.mdl**.
    - replace (copy and paste) the mathematical model of the system used in **Model210_DT_sv1_servo.mdl** with the ECP drivers from **Model210_DT_sv1.mdl**
    - edit the outputs of **Model210_DT_sv1_servo.mdl** so the states and time do not have the `m' prefix.
i) Reset the ECP system.
j) Connect **Model210_DT_sv1_servo.mdl** to the ECP system and run it.
k) Use the program **Compare_DT1.m** to produce a plot comparing the predicted response using the mathematical model of the system with the real response of the system. Note that the third state is not plotted since it is not really all that important.
l) Copy and paste this graph into your **Word** document that will eventually become your memo for this lab. It is a real good idea to write a short caption at this time so you don't forget what you just did.

Part B: Two Degree of Freedom Rectiliner Systems

For this system you need to basically go through the same steps you used in **PART A**, except you will need to use the programs **DT_sv2_servo.mdl** and **DT_sv2_servo_driver.m** you wrote for your homework.

You will need to copy the program **Model210_DT_sv1_servo.mdl** to **Model210_DT_sv2_servo.mdl** and then modify the program **Model210_DT_sv2_servo.mdl** to work with the two degree of freedom system (you need to change the outputs of the demux)**.**

Determine which of the mathematical models you created for the two degree of freedom rectilinear system seemed to work best last week and use that model.

Configure your two degree of freedom system the way you did in **Lab 2**.

Assume we are trying to control the position of the first cart, and then we are trying to control the position of the second cart. You should compare the predicted response with the response of the real system. Hence you will have two plots (each showing all four states) as you analyze your system.

Part C: One Degree of Freedom Torsional Systems

For this system you need to basically go through the same steps you used in **PART A**, except you will need to modify the programs **DT_sv1.mdl** and **DT_sv1_driver.m** to work with a the model you created for the one degree of freedom torsional system that seemed to work best last week.

Configure your one degree of freedom system the way you did in **Lab 1**.

Even though I indicated the desired input in degrees, your mathematical model (and the ECP system) works in radians, so ***your input must be in radians.***

You should plot your outputs in degrees or degrees/second

You will need to create program the **Model205_DT_sv1_servo.mdl** to interface with the torsional system**. The easiest way to do this is as follows:**

- open **Model210_DT_sv1_servo.mdl** and save it as **Model205_DT_sv1_servo.mdl**
- **open Model205_DT_sv1.mdl,** copy the **Model205 ECP driver,** and paste it into **Model205_DT_sv1_servo.mdl**

You will also have to modify **Compare_DT1.m** to plot the predicted states compared to the real states in ***degrees or degrees/second*** (we don't care about the delayed input state).

You should compare the predicted response using your mathematical model with the response of the real system.

Part D: Two Degree of Freedom Torsional Systems

For this system you need to basically go through the same steps you used in **PART B**, except you will need to use the programs **DT_sv2_servo.mdl** and **DT_sv2_servo_driver.m** you wrote for your homework.

You will need to copy the program **Model205_DT_sv1_servo.mdl** to **Model205_DT_sv2_servo.mdl** and then modify the program **Model205_DT_sv2_servo.mdl** to work with the two degree of freedom system (you need to change the outputs of the demux)**.**

Determine which of the models you created for the two degree of freedom torsional system seemed to work best last week and use that model.

Configure your two degree of freedom system the way you did in **Lab 2**.

Even though I indicated the desired input in degrees, your mathematical model (and the ECP system) works in radians, so ***your input must be in radians.***

You should plot your outputs in degrees or degrees/second

Assume we are trying to control the position of the first disk, and then we are trying to control the position of the second disk. You should compare the predicted response with the response of the real system. Hence you will have two plots (each showing all four states) as you analyze your system.

Your memo should summarize how well the mathematical model predicted the response of the systems, and whether including the integrator improved on the steady state error (compared to not having an integrator in the system).