ECE-521 Control Systems II
Laboratory 4

System Modelling, State Variable Controller Design, and the Real World

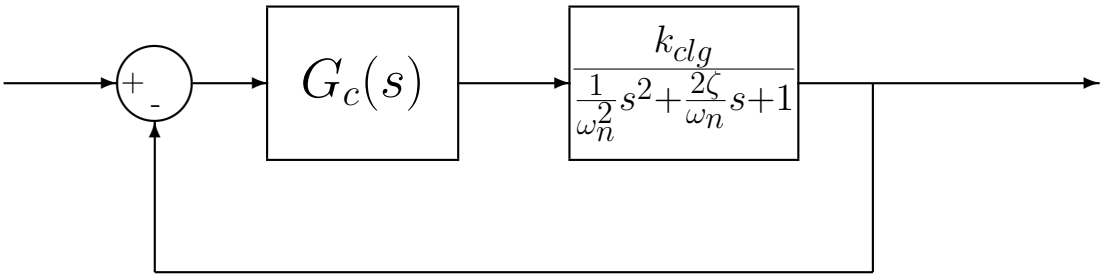
Preview In this Lab you will first obtain a second order model of your spring/mass/damper system, design a state variable feedback controller for it, implement the controller on the real system, and then compare the predicted response with the actual response. You will probably make at least two observations during this lab:

- models are not perfect, they are just guides
- real motors have real limits, which can make designing more difficult

Pre-Lab

1) Print out this lab and **read** it.

In what follows, assume we begin with the unity feedback system shown below. $G_c(s)$ is a classical controller (not state variable feedback), ω_n and ζ are estimates obtained using either time-domain or frequency domain methods, and k_{clg} is the system closed loop gain. Note that we are modelling the motor as contributing only a gain to the system, and we are lumping both the plant's gain and the motor gain together into one parameter.



For

$$G_c(s) \;=\; k_p$$

the steady state output y_{ss} due to a step input of amplitude Amp is given by

$$y_{ss} \;=\; \frac{Amp k_p k_{clg}}{1 + k_p k_{clg}}$$

2) Assume we have the plant transfer function

$$G(s) = \frac{k_{clg}}{\frac{1}{\omega_n^2}s^2 + \frac{2\zeta}{\omega_n}s + 1}$$

If the input is $u(t)$ and the output is $x(t)$ we can represent this system with the differential equation

$$\frac{1}{\omega_n^2}\ddot{x}(t) + \frac{2\zeta}{\omega_n}\dot{x}(t) + x(t) = k_{clg}u(t)$$

Assume $q_1(t) = x(t)$ and $q_2(t) = \dot{x}(t)$. Show that in terms of these variables we can write a state variable description of the system as

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} q_1(t) \\ q_2(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}\begin{bmatrix} q_1(t) \\ q_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ k_{clg}\omega_n^2 \end{bmatrix}u(t) \\ y(t) &= [1\ 0]\underline{q}(t) \end{aligned}$$

3) Assume we are using state variable feedback of the form $u(t) = k_{pf}r(t) - \underline{k}\underline{q}(t)$, so the new form of the system equations is

$$\begin{aligned} \dot{\underline{q}}(t) &= (A - B\underline{k})\underline{q} + Bk_{pf}r(t) \\ y(t) &= C\underline{q}(t) \end{aligned}$$

Assume the input $r(t)$ is a unit step with amplitude *Amp*. Show that in steady state for $y(t)$ to equal *Amp* we must have

$$k_{pf} = -1/C(A - B\underline{k})^{-1}B$$

4) Assume we want to use our state variable representation to implement a simple proportional controller. Hence we assume the feedback gain is $\underline{k} = [k_p\ 0]$ (we are only feeding back the position). Show that the closed loop transfer function is given by

$$H(s) = \frac{k_{clg}k_{pf}}{\frac{1}{\omega_n^2}s^2 + \frac{2\zeta}{\omega_n}s + k_{clg}k_p + 1}$$

and hence, for the state variable feedback system to be the same as the simple proportional controller system we must have $k_{pf} = k_p$. (*Hint: Compare steady state values for the same amplitude inputs.*)

We need to first **identify** the system:

1. Estimate an initial second order system model using time domain analysis (using either the **log_dec** or **fit** programs).
2. Measure the frequency response (make one measurement at 1Hz, 2Hz, ..., 7 Hz, and at least 4 points near the resonant peak)
3. Use the **fit_bode** command to estimate the gain of the system.
4. Use the **opt_fit_bode** command to fine tune the system model.
5. Determine the closed loop system gain k_{clg} (to be described below)

Estimating the Closed Loop Gain k_{clg}

0. Set the units

Click **Setup** → **User Units** and set the units to **cm**.

1. Setting up the controller

Click **Setup** → **Control Algorithm**. Be sure the system is set for *Continuous Time*. Select **PID** under **Control Algorithm**. Click on **Setup Algorithm**. Be sure **Feedback** is from **Encoder 1**. Set k_p to a small number (less than or equal to 0.05) and be sure $k_d = 0$ and $k_i = 0$. Then click **OK**. Next Click **Implement Algorithm**. The click **OK**.

2. Setting up the closed loop trajectory

Click **Command** → **Trajectory**. Select **Step** and click on **Setup**. Select **Closed Loop Step** and set **Step Size** to 0.5 to 1.5 cm. Be sure to record this step size (we'll refer to the amplitude as *Amp* below). Set the **Dwell Time** to something like 2000 ms, this is the time the system will be recording data. Finally click **OK**, then **OK** and you should be back to the main menu.

3. Executing the closed loop step

Click **Command Execute**. A menu box will come up with a number of options, and a big green **Run** button. Click on the **Run** button. When the system has finished collecting data, a box will appear indicating the how many sample points of data have been collected. (If you have hit a stop, the system stops recording data. This usually means you're input amplitude was too large or k_p was too large.) Click on **OK** to get back to the main menu.

4. Determining the steady state value

Click **Plotting** → **Setup Plot**, or just **Plotting Data** → **Plot Data**. Look at the steady state value (y_{ss}). You may need to change the dwell time if your system has not reached steady state.

5. Estimating the closed loop gain

Estimate the closed loop gain k_{clg} using the formula derived previously: $k_{clg} = \frac{y_{ss}}{k_p} \frac{1}{Amp - y_{ss}}$. You need to go through this procedure at least three times for each configuration. You must use at least two different values of k_p and two different values of input amplitude Amp. If none of the steady state values is larger than 0.4 cm, increase either k_p or Amp. Average the three results to get your k_{clg} (they should be similar). For the trials I've run, I've got k_{clg} between 10 and 20. Your's may be outside this range though.

6. Estimating the transfer function

The final plant transfer function we will use is then

$$G(s) = \frac{k_{clg}}{\frac{1}{\omega_n^2} s^2 + \frac{2\zeta}{\omega_n} s + 1}$$

where ω_n and ζ were found previously.

7. Comparison of State Feedback and Proportional Control

Choose a value of k_p and simulate a proportional controller. (It's probably easiest to just use the last of the value of k_p you used to determine k_{clg}). Save the output and edit the file for Matlab. Then implement a state variable controller that produces the same closed loop transfer function (see the prelab for how to do this.) Save the output and edit the file for Matlab. Use the program **compare_tf_sv** to plot the results on the same graph. There are three arguments to be passed to this file:

- The first argument is the file containing the step response with the proportional controller in single quotes.
- The second argument is the file containing the step response of the corresponding state variable model, again in single quotes.
- The final argument is the final time to display. Set the final time a short time after the system reaches steady state. It is not good to have a plot of 5 seconds, 4 second of which are the system at steady state.

If the two plots are not on top of each other you screwed up. Include this graph in your documentation for each system you model.

Designing and Implementing the State variable Controllers

Our general goals for the state variable controllers are as follows:

- produce a position error of less than 0.15
- reach steady state within 1 seconds (the faster the better)
- have as little overshoot as you can manage

Here are some general ideas:

- You need to try and use positive values for k_1 and k_2 ($\underline{k} = [k_1\ k_2]$). The system does not respond very well to negative values. In particular, the steady state values may be off.
- k_1 should be larger than k_2. k_2 is multiplying the derivative, and the estimate of the derivative tends to be noisy.
- Try to keep k_2 less than 0.05 and k_1 less than 1.0.
- Be sure that the gains for the second and third carriages are set to zero
- You may need to reset the controller often, such as every time you want to implement a new controller. Click **Utility** → **Reset Controller**. Only do this **before** you have implemented a controller.
- You may need to rephase the motor. Click **Utility** → **Rephase Motor**
- Be sure to **Implement** the controller you have designed.
- Try and track a step with an input amplitude of 0.5 to 1 cm.

Direct (Trial and Error) Method

1. Estimate the Gains

Use the program **state_variables_1cart** to guess values for k_1 and k_2. The program will print out the corresponding locations of the closed loop poles and the correct gain k_{pf}, as well as produce a plot of the system with state variable feedback.

The arguments to this program are:

- the amplitude of the input signal (in cm)
- the feedback gain matrix $\underline{k} = [k_1 \ k_2]$
- the closed loop system gain k_{clg}
- the estimated natural frequency of the system ω_n
- the estimated damping ratio of the system ζ
- the length of time to run the simulation for
- the file name with containing the response of the real system in single quotes. At this point, the filename is just ''

2. Implement the Gains on the ECP System

Once your simulated system has a reasonable response, and probably more importantly, reasonable gains, try running the ECP system with these gains. If the gains are not too large and the system works, save the results to a file. If the system buzzes and doesn't work, go back to step 1 and try again.

3. Comparing the Simulation and the ECP system

Edit the file you saved in part 2 so Matlab can read it. Run the program **state_variables_1cart** again, with the same gains as you used on the system. This time the last argument to the program is the name of the file you saved the response of the system into. You should get a plot containing both the real system and the simulated system. You may want to reduce the final time of the plot so there is not alot of time at steady state showing.

4. Practice Makes Perfect

Try at least three different combinations of gains before you move on the the next method. Be sure to produce a plot for each system, and record the gains and closed loop poles for each system.

Linear Quadratic Regulator Method

1. Estimating the Feedback Gains

Use the Matlab routine **lqr** to estimate the feedback gains k_1 and k_2. The arguments to this routine are

- the A matrix of the system (see the prelab)
- the B matrix of the system (see the prelab)
- a penalty matrix Q
- a penalty matrix R

(Note there is one more possible argument, but we won't use it. Type help lqr for more information).
The Linear Quadratic Regulator finds the gain $\underline{k}$ to minimize

$$J = \int_0^\infty \left[\underline{x}^T(t)Q\underline{x}(t) + u(t)Ru(t)\right] dt$$

where

$$\begin{aligned} \dot{\underline{x}}(t) &= A\underline{x}(t) + Bu(t) \\ u(t) &= -\underline{k}\underline{x}(t) \end{aligned}$$

In our case Q is a two by two positive definite matrix, and R is a scalar. Since Q is most likely a diagonal matrix, it's easiest to iterate using the following command in Matlab

```
> K = lqr(A,B,diag([q1 q2]),R)
```

where $q1$ and $q2$ are the desired diagonal elements of Q and R is a scalar. In general, as R gets larger (it may have to get very large), the size of the gains goes down. Also, if a gain is negative, try setting the weight on that gain very large. Note that if all of the gains are large there is no real effect! Iterate on values of Q and R until you think you have something that works.

2. Determining k_{pf}

Again used the program **state_variables_1cart**. The values for $\underline{k}$ have been determined by the lqr routine above. The program will print out the corresponding locations of the closed loop poles and the correct gain k_{pf}, as well as produce a plot of the system with state variable feedback.

The arguments to this program are:

- the amplitude of the input signal (in cm)
- the feedback gain matrix $\underline{k} = [k_1\ k_2]$
- the closed loop system gain k_{clg}

- the estimated natural frequency of the system ω_n
- the estimated damping ratio of the system ζ
- the length of time to run the simulation for
- the file name with containing the response of the real system in single quotes. At this point, the filename is just ''

3. Implement the Gains on the ECP System

Once your simulated system has a reasonable response, and probably more importantly, reasonable gains, try running the ECP system with these gains. If the gains are not too large and the system works, save the results to a file. If the system buzzes and doesn't work, go back to step 1 and try again.

4. Comparing the Simulation and the ECP system

Edit the file you saved in part 3 so Matlab can read it. Run the program **state_variables_1cart** again, with the same gains as you used on the system. This time the last argument to the program is the name of the file you saved the response of the system into. You should get a plot containing both the real system and the simulated system. You may want to reduce the final time of the plot so there is not alot of time at steady state showing.

4. Practice Makes Perfect

Try at least three different combinations of gains (corresponding to three different values of Q and R). Be sure to produce a plot for each system, and record the gains and closed loop poles for each system.

Memo

Your memo should compare (briefly) the response of the model and the response of the real system for the different gains you tried. You should have some description of the configuration of the system you were trying to control.

You should include the following items as attachments. Most of these are figures which should have reasonable captions.

- The step response of the time-domain model.
- The initial frequency response of the system.
- The optimized frequency response of the system.
- The data used to determine the closed loop gain.
- The final model of the system.
- The predicted and actual response of the system to each of the different controllers where you guessed the values of $\underline{k}$, and the corresponding closed loop pole locations.
- The predicted and actual response of the system to each of the different controllers where you used the lqr algorithm to determine the values of $\underline{k}$. Also record values of Q and R used and the corresponding closed loop pole locations.