Lab 3, SFWR ENG 3DX4

Introduction to Computer-Based Control

First week of lab: February 26, 2024
Prelab Due start of Lab Period, week of: March 4, 2024
Demo Due End of Lab Period, week of: March 4, 2023

1 Announcements

In this lab, you will be evaluated on prelab questions and demonstration of working laboratory exercises. Evaluation of lab exercises will be performed on a checkpoint system, with the checkpoints indicated in your laboratory instructions. In order to receive full marks for a lab, you must show a TA your work/output at each checkpoint, and your work must be correct and complete. In addition, lab work must be completed in the allotted laboratory time. Prelab questions are to be completed *individually*, This lab consists of 5 checkpoints.

1.1 Goals

- Introduction to computer-based control of a simple voltage-controllable electromechanical system - the DC motor
- Estimate the steady-state Gain A of a motor
- Experiment with different set-points and constants of a PID controller while controlling the DC motor
- Estimate the time constant τ

Note: This lab consists of 5 activities. There is a corresponding checkpoint for each activity.

Before you go for your lab session, please read the following NI documents at your convenience, in addition to the class notes:

- Fundamentals of Motion Control PDF (see Lab 3 files)

2 Equipment Arrangement

Quanser's SRV02 rotary servo plant consists of a DC motor, a gear box, a potentiometer, and a quadrature encoder. Some models also have a tachometer.

The potentiometer output is used to determine the gear angular position, which can be used to implement the proportional (P) part of the PID controller. This is an analog signal and must be converted into a digital signal before it is passed on to the computer. The potentiometer is connected through a 6-pin DIN connector to the S_1 terminal on the Universal Power Module (UPM). The UPM supplies voltage to the potentiometer and picks up the analog signal from its wiper, proportional to the position of the motor shaft. The S_1 output from the UPM is connected to one of the analog input channels on the myRIO.

The tachometer output is also an analog voltage signal which is proportional to the angular velocity of the shaft, $\frac{d\theta}{dt}$, which can be used to implement the derivative (D) part of the PID controller. The 6-pin DIN connector of the tachometer is connected to the S_3 terminal of the UPM.

The encoder is connected directly to the myRIO $ENC0$ port using a 5-pin DIN connection. The output of the quadrature encoder can be used to determine the gear's angular position change, which may also be used to determine angular velocity.

One analog output channel of the myRIO (AO0) provides a voltage signal, but the myRio provides nowhere near enough current to drive the motor. Instead, we use the UPM to amplify the signal, and provide current to the motor.

If the voltage applied to the Quanser rotary servo is close to or exceeds 10 volts, sufficient internal forces can be generated that may damage the servo's gear box. Fast voltage changes or rapid oscillations in voltage may also generate sufficient force to damage the gear box. Therefore, when we use a PID controller to control the SRV02, to protect the servo device, we normally saturate the output of the PID controller to within $\pm 9V$ (meaning, we do not allow the motor voltage to exceed this value).

3 Laboratory Procedure

3.1 Improving our Estimation Mechanism

In this exercise we will estimate the angular velocity of the motor shaft and use it to calculate the steady-state gain:

$$A = \frac{\dot{\theta}}{input\ voltage} = \frac{\omega}{input\ voltage}$$

3.1.1 Background

The open-loop transfer function of a motor can be approximated as:

$$M_\theta(s) = \frac{\theta(s)}{V(s)} = \frac{A}{s(\tau s + 1)}$$

where A is the steady-state gain and τ is the time constant (both are positive real valued constants).

The transfer function in terms of angular velocity $\omega(t) = \frac{d\theta}{dt}(t)$ can be approximated as:

$$M_\omega(s) = \frac{\Omega(s)}{V(s)} = sM_\theta(s) = \frac{A}{\tau s + 1}$$

$$\Omega(s) = \frac{A}{\tau s + 1} V(s)$$

Note that the transfer function has one pole, at $-\frac{1}{\tau}$. Assuming $\tau > 0$, all poles are in the left hand plane, making this system stable. We may therefore use the Final Value Theorem (FVT):

$$\lim_{t\to\infty} \omega(t) = \lim_{s\to 0} \frac{sA}{\tau s + 1} \times \frac{V_0}{s} = V_0 A,$$

where V_0 is the amplitude of the applied voltage.

In the lab, if we estimate the steady-state angular velocity for a given input voltage V_0, we can determine A. We'll make some modifications to the model we used in the previous lab, in order to measure the steady-state angular velocity more accurately. We will use a sliding array of 100 angular velocity readings, and average the values, as well as calculating and displaying the DC gain, based on this measured average angular velocity.

1. Download "openloop.slx" from the Lab 3 folder on Avenue and save it in a folder on your Z drive.

2. Open and study "openloop.slx". This model is similar to the bump test model we have used in Lab 2, but has the following differences:

   - In "openloop.slx", the step input signal is replaced by a constant input.
   - The **Saturation** block is to limit the voltage applied to the SRV02 to $\pm 9V$ to protect the servo plant.
   - The velocity unit used in this model is Degrees/s instead of Rad/s, therefore the gain to convert the encoder readings is 360/4096.

3. As in Lab 2, you can use this model to estimate the steady-state angular velocity values by looking at the output graphs. This time, we're going to implement a *running average* to get more accurate velocity readings. The running average

continuously calculates the average over a certain number of observations. It is a simple way to reduce system noise and to get a more stable reading. Use Figure 1 as a reference to modify your "openloop.slx" model, to calculate the running average of SRV02 load gear's angular velocity over 100 observations:

Simulink/Discrete/Tapped Delay
Simulink/Math Operation/Vector Concatenate
Simulink/Math Operation/Sum of Elements
Simulink/Signal Attributes/Width

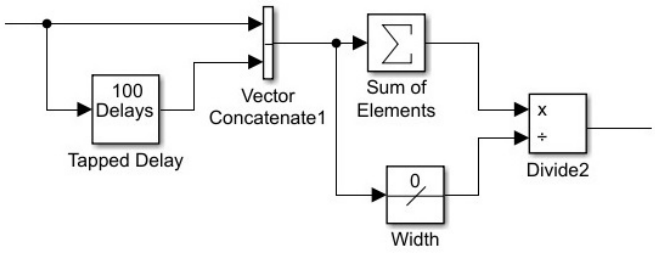

Figure 1: A sample section to calculate rolling average

4. Send the running average velocity, as well as the `actual` input voltage applied to the servo motor, to the same scope displaying the velocity directly derived from the encoder readings. Also send the running average to a digital **Display** block.

5. Next, add whatever blocks are necessary in order to calculate the DC gain from the running average. This value should also be displayed in a digital **Display** block.

6. Name the signals that are connected to the scope, and set the scope to display legend.

7. Show your model to your TA to complete *CHECKPOINT 1*.

3.2 Estimating the steady-state gain A

- Power on your lab system. Make sure the connections are correct.

- Run your model with an input voltage of 1V. Examine the scope display. If the running average of the velocity is not stable, consider increasing the number of observations to average until you are satisfied with the result. Then run the model for a sufficiently long period of time to get the final velocity and DC gain.

- Using the chart below, run some trials, and record the values measured by your model.

Input Voltage (V)	Average Gear Speed ($\frac{Deg}{s}$)	DC Gain ($\frac{Deg}{V \cdot s}$)
-8		
-7		
-6		
-5		
-4		
-3		
-2		
-1		
1		
2		
3		
4		
5		
6		
7		
8		
average		

- Calculate the mean value of the steady-state gain. Show it to your TA to complete *CHECKPOINT 2*, along with answers to the following questions.
  - Do you notice a pattern in the DC gain values you calculated? If so, what is that pattern?
  - Why do you think this is happening?
  - What does this say about our model of the DC motor?

3.3 A PID Controller for Angular Position

The purpose of this part of the lab is to observe how an integral compensator (K_I) can improve steady-state error. To do this we first need to construct a model with a PID controller to control the SRV02 gear's angular position.

1. Start with a blank model. Place the following blocks on the model canvas: **HIL Initialize, HIL Write Analog, HIL Read Analog Timebase, Scope, Saturation**. We will use the same AO0 channel to drive the servo motor. We will also read the potentiometer's analog signal, which we normally connect to myRIO's MSP C port AI0, which is channel #8 for configuring Quarc blocks. Configure the model for it to run on myRIO, and configure the Quarc write and read blocks.

   An alternative is to use your "openloop.slx" model as a starting point and make the appropriate modifications.

2. The potentiometer will give a reading between -5V and +5V, corresponding to positions between -180 and +180 degrees (or π radians). In this lab, we wish

to control the motor in terms of degrees, not radians. Calculate the conversion factor to convert the potentiometer readings in volts to degrees. (This is the same as saying, how many degrees do we get per volt?)

3. Apply the factor you calculated in the previous step to convert the signals read from the potentiometer to degrees, and connect the converted result to the scope. The converted result is the position (in degrees) of the servo plant gear, which is our process variable and the feedback signal.

4. Our system is going to need some input! Rather than using a constant block, this time let's be a bit fancier and use a slider!

   - Add a constant block with a value of 1 and a **Slider Gain** block (Simulink>>Math Operations).
   - You can access the slider by bringing up the properties window of the slider gain block. Set the minimum value to -170, and the maximum to 170.

     According to the SRV02 user manual, if the position reading exceeds ±180 degrees, a wrap-around error will be introduced, and unstable system behaviour may occur. In practice, some potentiometers cannot read values when the gear's angle is outside ±170 degrees. This is why we set the limits for the slider block to ±170, because the output of the slider block will be our *set-point signal* (degrees). Note also that the gain is a multiplicative factor, so we need to feed a constant value of 1 into the input in order for it to output the slider value directly.

5. Use an **Add** or **Sum** block to calculate the difference between the set-point and the feedback signal. This is the error between the requested set-point and the actual position of the servo gear. The PID controller will operate on this error.

6. This time we're not going to build the PID controller from scratch. Go to the Simulink library: **Simulink>>Discrete**, find the **Discrete PID Controller** and place one into your model. Set K_P to 1.2, and K_D and K_I to zero.

7. The input to the PID block has units of degrees. Unless an extra coefficient is implied, the output of the PID block should also be in degrees. Before using the PID controller's output to drive the servo motor to adjust the process variable, we need to convert this result to volts. Use the factor you derived from Step 2 and a **Divide** or **Product** block to implement the adjustment.

8. As a safety feature to protect the servo plant, use a **Saturation** block to limit the converted PID output before feeding it to the **HIL Write Analog** block. Set the upper and lower bounds of this saturation to ±9. This is to limit the PID controller's output to ±9 V.

9. Connect the output of the **Saturation** block to the **HIL Write Analog** block. This is the actual applied voltage to the servo motor.

10. Connect the set-point to the same scope displaying the measured gear position. Also, connect the actual applied voltage to the motor to the scope. Name the signals fed to the scope and set the scope to display legend.

11. Save your model as "PIDdemo.slx", or other appropriate name. Run your model. Play with the **Slider Gain** to watch how the motor reacts.

12. Show your model to your TA to complete *CHECKPOINT 3*

3.4 The Importance of K_I

1. In this part, we will run a number of trials comparing the steady-state error when K_I is zero or when K_I is positive.

2. You have two options for measuring this steady-state error. Take a look at the situation and use the option you think will be faster!

   - The procedure used in Lab 2, where values are read from the graph, with the assistance of the Cursor Measurements tool.
   - An adaptation of the procedure used earlier in this lab, where a running average of the output is calculated, and the difference between this average and the set-point signal is automatically calculated.

3. Run your trials for $K_I = 0$, as given in the chart below. In this set of trials, the servo needs to be "zeroed" between each trial. This means you need to set the set-point to 0 and run the model so as to set the gear position close to 0 degrees. You also need to make sure that you are running the model for a sufficiently long period of time so that the system has adequate time to reach steady state.

4. Now, set K_I to some value greater than zero and re-run the trials. This value is left to your discretion. You may wish to run a few trials at different K_I values until you get a response you like. It is possible through your choice of K_I to push the response into the wrap-around region, for larger set-points. If this occurs, try lowering your K_I value.

5. Once your trials have been run, calculate the average steady-state errors of the two sets of trials.

Set-point	e_{ss} with $K_p = 1.2 \wedge K_I = 0$	e_{ss} with $K_p = 1.2 \wedge K_I =$ ____
-150		
-120		
-90		
-60		
-30		
30		
60		
90		
120		
150		
average		

6. Show your work to your TA to complete *CHECKPOINT 4*. Be ready to answer the following questions:

   (a) Did adding K_I improve the steady-state error? By how much?

   (b) What negative effects did adding K_I introduce?

   (c) If you wanted to reduce the overshoot, what parameter would you use, and in which direction would you tweak it?

3.5 Estimating the time constant τ

This exercise examines two methods for estimating the time constant τ.

3.5.1 Background

The closed-loop transfer function for our motor is as follows:

$$G_{\theta d} = \frac{\theta(s)}{R(s)} = \frac{K_p A}{\tau s^2 + s + K_p A}$$

In the previous section and in this section, we wish to control the motor's angular position (in degrees). The units for the motor's steady-state gain are $\frac{Degrees}{V}$. If we consider the conversion factors needed to correctly interpret the feedback signal, our transfer function becomes:

$$G_{\theta d} = \frac{\theta(s)}{R(s)} = \frac{K_p A}{36\tau s^2 + 36s + K_p A} \tag{1}$$

Critical Damping is that amount of damping just sufficient to prevent overshoot or oscillations. When a system is critically damped, it has two poles located at $s = -\sigma$. We can use this observation to calculate τ based upon the solutions to the quadratic equation being equal and real.

Consider a general quadratic equation:

$$as^2 + bs + c = 0$$

This has solutions:

$$s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

In the case when both roots are the same, it must be of the form

$$s = \frac{-b \pm 0}{2a}$$

Therefore,

$$b^2 - 4ac = 0$$

- **Suppose you know that when the system is critically damped, K_p=K_* and the system's steady-state gain is A. Use the above result, and the quadratic expression in (1), to find an equation for τ. You'll need this during the lab procedure.**
- **Your value for A was derived experimentally earlier in this lab.**
- **Your value of K_* is the value of K_P that results in a critically damped system.**

Another approach to calculate τ is to consider that the Root Mean Square (RMS) value of a signal $f(t)$ that is periodic with period T is given by the equation:

$$\sqrt{\frac{1}{T}\int_0^T (f(t))^2 dt}$$

RMS values are often easier to measure and more accurate than trying to determine peak amplitudes. It can be shown that the RMS value of $u(t) = B\sin\omega t$ is $\frac{B}{\sqrt{2}}$. Suppose $u(t) = B\sin\omega t$ is the input to a strictly stable transfer function $G(s) = \frac{Y(s)}{U(s)}$. In steady-state, the RMS value of $y(t)$ is $|G(j\omega)|\frac{B}{\sqrt{2}}$.

Thus

$$\frac{RMS\ value\ of\ y(t)}{RMS\ value\ of\ u(t)} = |G(j\omega)|$$

This method of estimating τ is to experimentally determine the bandwidth of the closed-loop system $G_{\theta cl}(s)$ corresponding to the -3dB frequency ω_{bw} for a critically damped system. Then we have:

$$|G(j\omega_{bw})| = \frac{1}{\sqrt{2}}$$

For a critically damped system $K_p = K_*$, at ω_{bw}:

$$\left|\frac{K_* A}{36(j\omega_{bw})^2\tau + 36j\omega_{bw} + K_* A}\right| = \frac{1}{\sqrt{2}} \tag{2}$$

If we know the values of K_* and A, the above equation can be used to calculate τ.

Given a complex number $a + bj$, its magnitude is $\sqrt{a^2 + b^2}$. To convert $\frac{1}{c+dj}$ into the form $a + bj$, multiply top and bottom by the complex conjugate of $c + dj$.

3.5.2 Procedure

1. Find the equation for τ. You may need to review the background.

2. Take your model from the previous part and remove the steady-state error measuring components (if you decided to add them). Set K_I back to zero on your PID controller. While you're there, create a variable in the model workspace for K_P. This will help us tune our gain while the model is running.

3. Replace the set-point slider apparatus with a **Pulse Generator** block (Simulink >>Sources). Set its "Period" to something large, like 10s, its "Amplitude" to 60, and set its "Pulse Width" to 50.

4. Set the proportional gain to 15.

5. Run the model. The square wave generator allows you to send a repeating sequence of step inputs, so that we can see the system's step response many times in succession.

6. As the model is running, adjust the value of the proportional gain until the overshoot has *just* disappeared. This is your critical damping gain K_*. Record this value.

7. Use the equation you found in Section 3.5.1 to calculate the system's time constant τ.

8. Now we're going to measure the same thing using the RMS method described above. The short version is that we need to send a sinusoidal signal instead of a square wave, measure the reduction in amplitude at various frequencies, and thereby discover the -3dB frequency.

9. Replace the square wave generator with a sine wave generator. Set the amplitude to 60, and create a variable in the model workspace for the frequency. The initial value of this variable should be 1 Rad/s. `(Attention: the Frequency parameter for the Sine Wave block is in Rad/s, NOT in Hz)`

10. Run the model. You should already see how the amplitude of the motor output doesn't quite reach the set-point waveform, and that it lags slightly.

11. Run a number of trials of your choosing at various frequencies, enough so that you can make a Bode plot that demonstrates the cut-off frequency in a sufficiently convincing manner. Keep in mind that the cut-off frequency will occur when the amplitude has reached $\frac{1}{\sqrt{2}}$.

    - Once again, you can perform this operation by either reading values off the graph directly, or by setting up some components within the model to automatically read out the amplitude of the motor's position signal.

12. Remember! Bode plots take gain in decibels, not the motor's raw position readings! You will need to convert your values accordingly.

13. Your cut-off frequency will be the desired ω_{bw}. Use the equation you derived to calculate τ.

14. Show the TA your values for K_*, ω_{bw}, and τ to complete the *FIFTH AND FINAL CHECKPOINT!*

4 Closing Notes

Remember the scientific method! In order to demonstrate a phenomenon, it needs to be repeatable and observable, and the more independent methods of verification we can produce, the more confident we can be in our observed phenomenon.

However, some systems derived from pure scientific theory may not be practical in reality for various reasons. For example, if a system is marginally stable, a minor experimental error can ruin the expected results. This may be the case with the last part of this lab when trying to find the time constant τ using the cut-off frequency method with a closed-loop system.

In the last couple of labs, we have come at the problem of empirical model estimation from a number of angles. How do your results from this lab compare with your results from Lab 2? Are your estimates close, or way off?

If you have time this week, compare your lab results to those of your classmates (after you've run your experiments of course), to see how they match up. Although each set of lab apparatus may behave slightly differently, your lab results should be within the range of others. But remember! Experimentation should be entered into with the spirit of maximum openmindedness. You should never adjust your results so that you get what is expected. You should verify that your experiment is being run according to procedure, and that the theory is tested correctly by the experiment. Modifying your results because you didn't get what someone else got represents a corruption of the scientific enterprise!