SFWR ENG 3DX4: Lab 2

Empirical Estimation of Transfer Functions for First Order Systems

First week of lab: Week of Feb 5, 2024
Prelab Due Start of Lab Period, week of: Week of Feb 12, 2024
Demo Due End of Lab Period, week of: Week of Feb 12, 2024

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 checkpoints indicated in your laboratory instructions by the phrase "show your work/output to your TA." 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. Prelab questions are to be completed *individually*, Lab exercises are to be performed in groups of two. This lab consists of 6 checkpoints.

Goals:

- Introduction to Simulink and QUARC Library.
- Create Simulink Quarc models and run the model in real time on NI MyRIO to control SRV02.
- Read the quadrature encoder from Quanser's SRV02, and derive shaft angular position change and gears' angular velocity from the readings.
- Perform basic parameter estimation from data captured.
- Experimentally derive a transfer function for a DC motor using the "bump test."
- Test the accuracy of the derived transfer function by comparing it to real-time experimental data.

1 Prelab Exercises

Prelabs due the week of: Feb. 12, 2024

In Lab 1, the model we used for our DC motor was derived theoretically from first principles. Such a derivation is not possible in all cases, so in Lab 2 we will be finding the transfer function modelling our physical plant using experimental methodology, and comparing our approximation to the system's actual behaviour.

Once again, our first order approximation of the transfer function of a DC electric motor with respect to angular velocity is:

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

where A and τ are positive, real-valued constants. $\Omega(s)$ and $V(s)$ are angular velocity and voltage as functions of s in the Laplace domain. (Note: Ω is capital ω so we are using $\Omega(s) := \mathcal{L}\{\omega(t)\}$.)

1. We will now develop a formula for the motor DC gain constant A in terms of a step input change ΔV and step output change $\Delta\omega$.

   a) Using the final value theorem, find an expression for the steady state value of $\omega(t)$ when a step input of amplitude V_x is applied. NOTE: $\tau > 0$ so the pole of $G_\omega(s)$ at $s = -\frac{1}{\tau}$ is in the open Left Half Plane (LHP) so the system is stable! Therefore we can use the final value theorem.

   b) Give the expression for $\omega(t)$ in response to a step input V_x at time $t = 0$. Assume a non-zero initial condition for $\omega(t)$, i.e., $\omega(0) = \omega_0$. Note that the response due to a non-zero initial condition ω_0 can be modeled as the response due to input $v(t) = \omega_0\delta(t)$ where $\delta(t)$ in the impulse function. Since $G_\omega(s)$ is a linear system, the response to step input V_x with non-zero initial condition ω_0 is just the sum of the responses due to the step and the initial condition.

   c) For $\omega(t)$ you computed in 1b), what is the $\lim_{t\to\infty}\omega(t)$? How does it compare to your answer from 1a)?

   d) Now assume that you run the motor with an initial step input of V_{min} until time t_0 At time t_0, assume that the system has reached steady state and the step input is changed to V_{max} at time t_0. In other words, your system input will take the form

   $$v(t) = \begin{cases} V_{max}, & t \geq t_0 \\ V_{min}, & 0 \leq t < t_0 \end{cases} \tag{1}$$

   where $t_0 >> \tau$, and V_{min} and V_{max} may be non-zero.
   Use the results above to show that:

   $$A = \frac{\Delta\omega}{\Delta V}$$

   where

   - $\Delta V = V_{max} - V_{min}$
   - $\Delta\omega = \omega_{ss} - \omega_0$
     where ω_0 is the steady state response to a constant input V_{min}, and ω_{ss} is the steady state response to the input V_{max}.

2. Using the formula derived in question 1, use the following graphs to calculate A for this system.

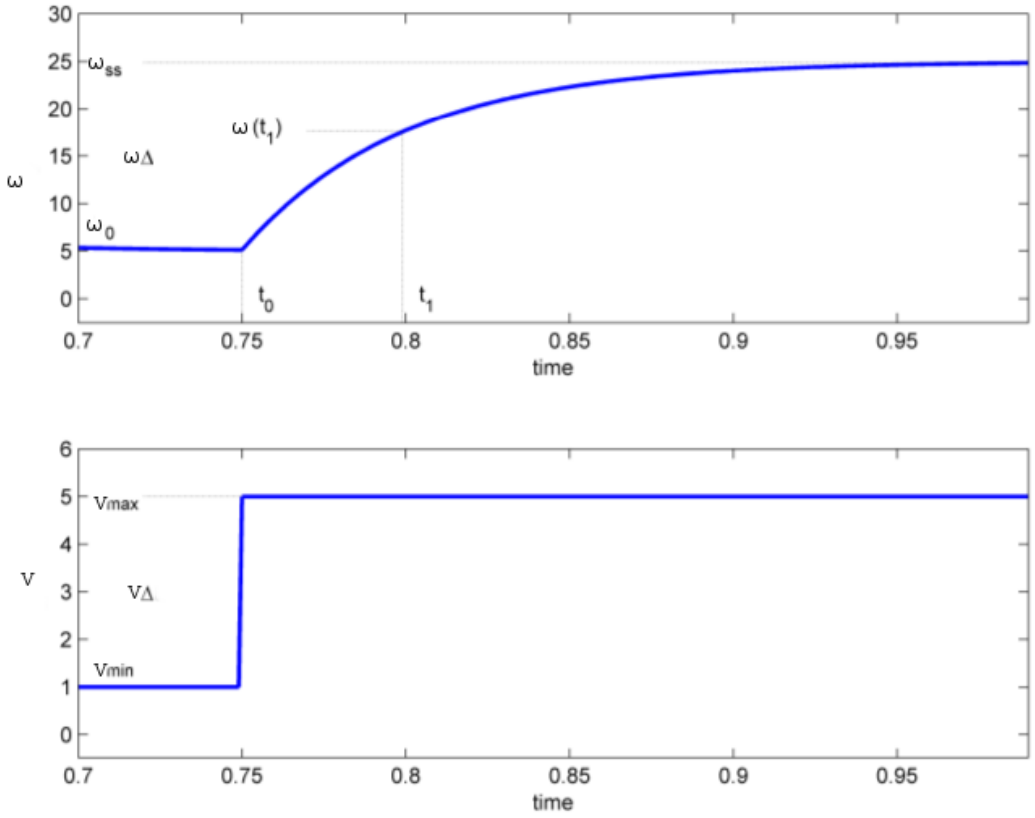

Figure 1: A First Order Response to Step Input

3. For a first order system, the time it takes a step response to reach 63.2% of its steady state value ($t_1 - t_0$ in Fig. 1) is the response's time constant τ. i.e., at time t_1, $\omega(t_1) = 0.632\Delta\omega + \omega_0$. Find the time constant τ for the above system in Fig. 1.

4. Using A and τ, give the transfer function $G_\omega(s)$ in terms of s.

5. Complete problem 21(a) in Chapter 4 of Control Systems Engineering 8th ed. (problem 32(a) in Chapter 4 of Control Systems Engineering 7th ed).

6. In one or two paragraphs, describe a situation in which it is preferable to derive a transfer function experimentally, and then design/calibrate a controller using simulation software, rather than experimenting with the plant to design/calibrate the controller directly. Describe a situation in which this is not preferable. (Answer in complete sentences!)

2 Laboratory Procedure

Although we will not use National Instrument's LabView to measure the quadrature encoder in this lab, the following document gives a good introduction on quadrature encoder theory, and the basics of measuring them: `http://ni.com/tutorial/7109/`

2.1 Acquire and Display Temperature data

In this part of the lab, we will connect a temperature sensor's output to the pin AI0 in MyRIO's MXP B connector. We will build a Simulink model using Quarc blocks, and make it run on the MyRIO. The model will measure the temperature in real time by reading the analog signal and converting it to temperature readings.

Please read the datasheet for the LM35 temperature sensor, which is available in this lab's folder on Avenue. Pay special attention to the diagram for identifying the three legs of the sensor. If the legs for GND and for 5V are not connected correctly, the sensor may be fried and stop working.

Turn off power to your MyRIO. Make connections for your hardware, double check your connections or ask a TA to approve your connections before powering on the MyRIO.

Follow the steps below to build and run your Simulink model on your MyRIO:

1. Launch Matlab and start Simulink.
2. In the **New** tab of the opened widow, select and click the **Blank Model** template to start with a blank Simulink model. (*To build a new model for running on MyRIO, you can also start with the* ***QUARC>>Hardware I/O Model*** *template.*)
3. Save your model as "Temperature.slx", or something descriptive, in your working directory. In our lab, you are recommended to save your lab projects in a folder on the Z: drive.
4. In the **Simulink Library Browser**, navigate to QUARC Targets>> Data Acquisition>>Generic>>Configuration, find the **HIL Initialize** block and place one on the canvas of your Simulink model.
5. The **HIL Initialize** block is required by all the models that require hardware access. This block will not receive any connections, but is used to initialize an HIL board and associate a name with the board. Double click the **HIL Initialize** block to open its configuration window. In the **Main** tab of the window:

(a) In the "Main" tab, set the "Board name" as you wish (or leave it as its default "HIL-1"). `If you change the ''Board name'' here, you must make sure to configure the other Quarc I/O blocks to use the board with this name!`

(b) Set the "Board type" as "ni_myrio_1900", which is the external target we will use in the lab.

(c) Once the board type is set to MyRIO, the **HIL Initialize** block will make all the analog input and output channels available, but not the encoder input channels. To use an encoder channel later, we need to make them available:

- Select the "Encoder Inputs" tab.
- Click the button with the ellipsis beside the input box of "Encoder input channels" and select the available encoder channels so that they appear in the "Channels Selected" box (*in the lab, we will use the encoder channels on the MSPC port; most of the time we will use MSPC-ENC0*).

(d) Click "Apply" and "OK" to close the configuration window.

Because the **HIL Initialize** block will not be connected to any blocks in the model, after it is configured, you can put it off to the side, away from your working area in the model.

6. In the **Simulink Library Browser**, navigate to **QUARC Targets>>Data Acquisition>>Generic>>Immediate I/O**, find block **HIL Read Analog** and place one in the model.

7. Double click the **HIL Read Analog** block on the model canvas to open its block parameter configuration window. Set its "Channel" to 4, as we will read the analog input signal from the AI0 pin in MyRIO's MXP port B.

   The information for the channel or port numbers for configuring Quarc blocks for using MyRIO is at: `http://quanser-update.azurewebsites.net/quarc/documentation/ni_myrio_1900.html`.

8. Place a Simulink **Gain** block in the model and change the gain to 100. This coefficient will convert the electrical analog signal reading to a temperature reading in $°C$.

9. Place a Simulink **Scope** block in the model. Connect the output port of the **HIL Read Analog** block to the Gain block, and connect the output of the Gain block to the scope.

   You have finished constructing your model to measure the temperature from a temperature sensor that will be connected to the AI0 pin on MyRIO's MXP port B. Next we need to configure the model to let it run externally on NI-MyRIO.

10. Select the menu "QUARC" and then select "Model Settings" to open the "Configuration Parameters" window for the model. (This window can also be opened by right clicking on the blank area of the model canvas and selecting "Model Configuration Parameters" from the context menu.)

11. On the left side of the same window, select "Code Generation".

12. On the right side of the window, set the "System target file" as "quarc_linux_rt_armv7.tlc".

13. Click "Apply" to save the change, and click "Apply" for the "Configuration Parameters" window.

    Notice that after this step, a new tab **HARDWARE** appears on the Simulink toolstrip.

14. Now let's tailor the numerical method that will be used to solve our model. On the left side of the "Configuration Parameters" window, select "Solver". Then on the right side of the window, leave the "Simulation time" and the "Stop time" at default. Under the "Solver selection" section, set the solver's "Type" to "Fixed step", leave the 'Solver" as default for now, but change the system sampling time for purpose of optimizing the real time control model performance. Let's set the step size (fundamental sampling time) to 0.01. (Click "Solver details" and then set the "Fixed-step size" to 0.01).

15. Click "Apply" and then "OK" to close the "Configuration Parameters" window.

16. Go to Quarc>>Preferences to open the window "Quarc Preferences".

17. On the **Model** tab, set "Target Type" as "Linux_rt_armv7", set the "Default model URI" as "tcpip://XXX.XXX.XXX.XXX:17001?nagle=no"

    Here, xxx.xxx.xxx.xxx is the IP address that is assigned to your MyRIO. In our lab, the IP address assigned to a MyRIO is printed on the tag on the MyRIO. If the MyRIO is connected to the host computer by a USB cable, the default IP address will be 172.22.11.2. You need to change the default IP address section from 172.22.11.2, which the MyRIO would use if USB connection is used, to the IP address that is assigned to your MyRIO by the wireless network.

18. Click "Apply" and then "OK" to close the preferences window.

19. Save your model (under the **SIMULATION** tab), and then go to the **HARDWARE** tab to click the "Monitor & Tune" button to run the model on NI MyRIO. This button allows users to build, deploy and run the real-time model on remote targets.

20. Check the scope display. Set the scope's Y limits to 25 to 35°C to better display readings around room temperature.

21. Touch the thermistor with your finger or other object to observe the change of measured temperature.

Show your output and your model to your TA before proceeding further (checkpoint 1).

2.2 Measure SRV02 gear's angular position change by reading the quadrature encoder

1. If you have not already done so, please read "Quadrature Encoder Measurements: How-to Guide" at: `https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000x1riCAA&l=en-CA`

2. Download the model "QuadratureEncoder.slx" from the Lab 2 folder on Avenue and save it in a folder on your Z: drive.

3. Open and study the model.

   (a) The quadrature encoder discs in the SRV02 unit have 1024 slots per disc, but the encoder uses X4 encoding. Therefore each revolution will correspond to a pulse count of 4096. (See the above reference.)

   (b) If the SRV02 keeps running, eventually the DAQ's counter which is used to hold the encoder's pulse count will overflow and wrap around at 0. Therefore an **Inverse Modulus** block (QUARC Targets >>Discontinuities) is used in the model to deal with this issue. Because MyRIO-1900 has 32-bit counters, we set the modulus for the **Inverse Modulus** block to 2^{32}.

   (c) The **Unary Minus** or **Negate** block is to make the sign or direction of measured gear position change (and the velocity derived from these changes) have the same sign as the driving voltage to the motor, and have the same sign as the velocity measured by SRV02's tachometer.

4. According to the encoder used in the model, connect SRV02's encoder port to the appropriate port on your MyRIO. Don't forget that the port numbers used by Simulink QUARC blocks for the MyRIO ports or pins are in the document "Quarc MyRIO support", which is at: `http://quanser-update.azurewebsites.net/quarc/documentation/ni_myrio_1900.html`

5. Go to **QUARC>>Preferences** to set a correct URI for your MyRIO.

6. Power on your MyRIO, and turn on the power supply module for your SRV02.

7. Run the model on MyRIO by clicking **HARDWARE >> Monitor & Tune**.

8. (CAUTION: Turning SRV02 gears may hurt your fingers! If you feel unsafe doing this step by yourself, please see a TA for help). Use your finger to rotate SRV02's gear and check the model's output, to make sure the system works properly.

Show your work to your TA (checkpoint 2).

2.3 Measure SRV02 gear's angular speed from quadrature encoder readings

When "QuadratureEncoder.slx" runs on MyRIO, it samples the gear's angular position change in real time; as a result the gear's angular speed can be readily derived. In this section, you are required to create a new model based on "QuadratureEncoder.slx" to measure SRV02's LOAD gear speed.

PLEASE NOTE: In SRV02, the encoder is attached to the load shaft (the middle shaft of SRV02). Therefore the encoder measures the position change of the LOAD gears (the gears on the middle shaft). Thus the angular velocity derived from these position changes is the angular velocity of the LOAD gears. Please also note that because the number of teeth on the gears for different shafts may not be the same, the angular velocities of different shafts may be different.

1. Notice that in "QuadratureEncoder.slx", the angular position change is measured in Degree. In this section we would like to measure the gear's angular change and angular velocity in Rad. Change your model or the parameters in the model accordingly to make the angular position change to be measured in Rad.

2. Find the **Derivative** block in **Simulink>>Continuous**, and use it to calculate your angular speed. Connect the angular speed to a scope display.

3. Place the **Quarc>>Data Acquisition>>Generic>>Immediate I/O>>HIL Write Analog** block in your model. Set its channel number to channel 4, which is for MSP C port AO0 channel to drive SRV02. Send it a constant value of 2. This will drive the motor to run at 2 Volts.

4. Save your model as a new file with an appropriate name.

5. Run your model to check the motor angular speed.

6. Run the model with different analog output voltages to check the load gear's velocities. `To protect SRV02, please uses voltages that are less than 9V`.

Show your output and your model to your TA (checkpoint 3).

2.4 Create Simulink model for Bump Test

The bump test is a simple way to find a system's characteristics based on a step input signal. For a system that can be well modelled by a first order system, the system's time constant and DC gain can be easily obtained from the bump test results.

Follow the steps below to build a new model to perform the bump test on your SVR02:

1. Start with the model you created in Section 2.3 that measures SRV02 load gear's angular speed. You may remove unnecessary blocks or portions of the model to make your model simpler.

2. Place a **Simulink>>Source>>Step** block into the model. Connect the **Step**'s output to an input port of a **Product** block, and connect the other input port of the **Product** block to a **Constant** block. Name the **Constant** block "Amplitude".

3. Connect the output of the **Product** block to an input port of an **Add** block. Connect the other input port of the **Add** block to a constant value of 1.5.

   The bump test assumes that the effect of static friction is negligible. In order for this to be physically true, the motor needs to be in a state that static friction has been overcome, i.e., the motor must be running at the time the bump test is performed. This is what the constant driving voltage of 1.5 is for.

4. Remove the constant that is connected to the **HIL Write Analog** block. Connect the result of the **Add** block to the **HIL Write Analog** block. This is the step voltage that will be applied to your SRV02 motor.

5. Feed the voltage signal applied to the SRV02 to another input port of the scope.

6. Name the signals fed into the scope.

   Your model should be similar to Figure 2.

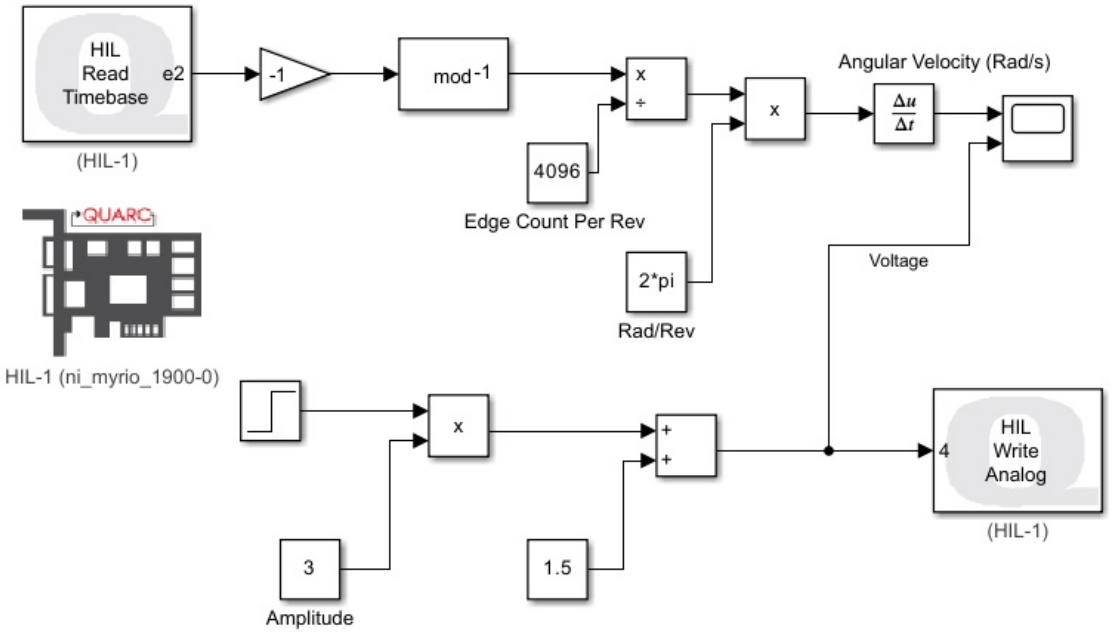

Figure 2: A sample Simulink model to do a bump test with a DC motor

7. Set the model's Stop Time to 10 Seconds.

8. Make sure the connections of your physical lab set up are correct.

Show your model and system set up to your TA before proceeding (checkpoint 4).

2.5 Finding the Servo's Transfer Function

1. You may now run the Simulink model for the bump test.

2. Use the scope's zoom or pan tools to accurately read the input voltage and the shaft velocity.

3. Create a table in a notebook or spreadsheet like the following:

Amplitude ΔV	$\Delta\omega$	DC Gain A	Time Constant τ
1.5 V			
2.0 V			
2.5 V			
3.0 V			
3.5 V			
4.0 V			
4.5 V			
5.0 V			
average			

4. For a given Amplitude ΔV, run the bump test, read the $\Delta\omega$ and the time constant τ from the scope display, and calculate the DC gain A. Record the results in the table.

5. Repeat the above procedure for the other amplitudes listed in the table, and calculate the **average** A **and** τ of all trials.

6. Using the average values for A and τ, estimate a transfer function for this first order system.

Show your chart and final transfer function to your TA for (checkpoint 5)

2.6 Testing the Accuracy of your Model

1. In this section, we will test the accuracy of our experimentally derived transfer function by comparing its output to the velocity readings from your motor.

2. Add a **Transfer Fcn** block (**Simulink**..**Continuous**) into your bump test model. Double click it and change the Numerator and Denominator coefficient vectors to reproduce the transfer function you found in Section 2.5.

- Alternatively, you may define environment variables for A and τ, and use the variables as your transfer function coefficients.

3. Connect the step input to the input port of the transfer function block. Connect the output of the transfer function block to the scope. The model should now look something like Figure 3.

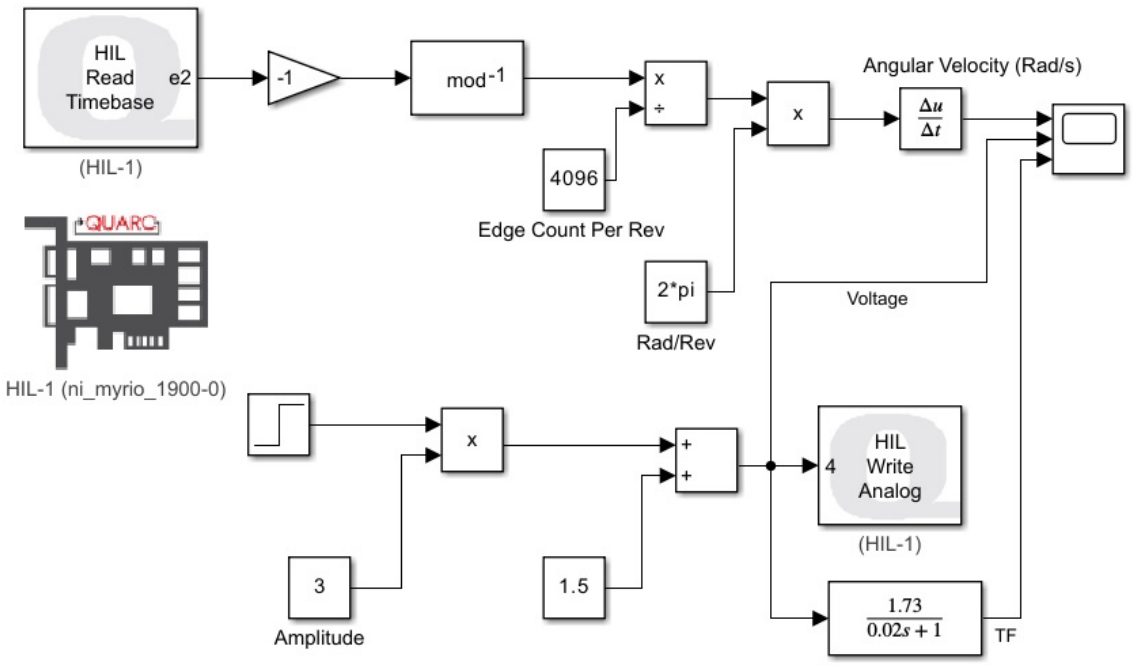

Figure 3: Testing Accuracy of Estimated Transfer Function

4. Run the model and qualitatively evaluate its error. Try this for several different voltage levels (do not exceed ± 9V).

5. For *checkpoint 6:*

   - Show your TA your graph.
   - Describe how good your approximation is.
   - If your approximation could be improved, describe to your TA what could be changed, by how much, and in what direction.

3 Closing Notes

Don't be disappointed if your two curves in Section 2.6 didn't match. It is often the case that a model will be off when compared with measured results.

While it's a fairly accurate model of this type of system, a first-order approximation is just that: an approximation. The real system is much more complex than a first order transfer function, even though it roughly behaves like one with respect to angular velocity. There are a number of nonlinearities that exist in our physical system, such as gear backlash, static friction, and inductive forces, which can't be accurately modelled using a first order approximation.

It is often the case that a model will get us in the ballpark, and then subsequent tweaking produces a progressively more accurate approximation of the desired behaviour. As an optional exercise, try playing with your transfer function to get a better fit.