QUANSER
INNOVATE · EDUCATE

STUDENT WORKBOOK

QUBE-Servo Experiment for MATLAB®/Simulink® Users

Standardized for ABET* Evaluation Criteria

Developed by:
Jacob Apkarian, Ph.D., Quanser
Michel Lévis, M.A.SC., Quanser

QUBE educational solutions are powered by:

MATLAB SIMULINK

Course material complies with:

ABET

CAPTIVATE. MOTIVATE. GRADUATE.

*ABET Inc., is the recognized accreditor for college and university programs in applied science, computing, engineering, and technology, providing leadership and quality assurance in higher education for over 75 years.

QUBE-Servo Integration

Topics Covered

- Getting familiarized with the Quanser® QUBE-Servo Rotary Servo Experiment hardware.
- Using QUARC® to interact with QUBE-Servo system.
- Sensor calibration.

Prerequisites

- The QUBE-Servo has been setup and tested. See the QUBE-Servo Quick Start Guide for details.
- Inertia disk load is on the QUBE-Servo.
- You have the QUBE-Servo User Manual. It will be required for some of the exercises.
- You are familiar with the basics of Simulink® .

1 Background

1.1 QUARC Software

The QUARC® software is used with Simulink® to interact with the hardware of the QUBE-Servo system. QUARC® is used to drive the DC motor and read angular position of the disk.

The basic steps to create a Simulink® model with QUARC® in order to interact with the QUBE-Servo hardware are:

1. Make a Simulink® model that interacts with your installed data acquisition device using blocks from the *QUARC Targets* library.
2. Build the real-time code.
3. Execute the code.

Type `doc quarc` in Matlab® to access QUARC® documentation and demos.

1.2 DC Motor

Direct-current motors are used in a variety of applications. As discussed in the QUBE-Servo User Manual, the QUBE-Servo has a brushed DC motor that is connected to a PWM amplifier. See the QUBE-Servo User Manual for details.

1.3 Encoders

Similar to rotary potentiometers, encoders can also be used to measure angular position. There are many types of encoders but one of the most common is the rotary incremental optical encoder, shown in Figure 1.1. Unlike potentiometers, encoders are relative. The angle they measure depends on the last position and when it was last powered. It should be noted, however, that absolute encoders are available.

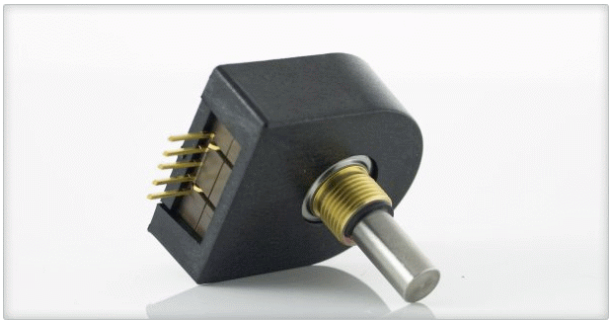


Figure 1.1: US Digital incremental rotary optical shaft encoder

The encoder has a coded disk that is marked with a radial pattern. This disk is connected to the shaft of the DC motor. As the shaft rotates, a light from a LED shines through the pattern and is picked up by a photo sensor. This effectively generates the A and B signals shown in Figure 1.2. An index pulse is triggered once for every full rotation of the disk, which can be used for calibration or homing a system.

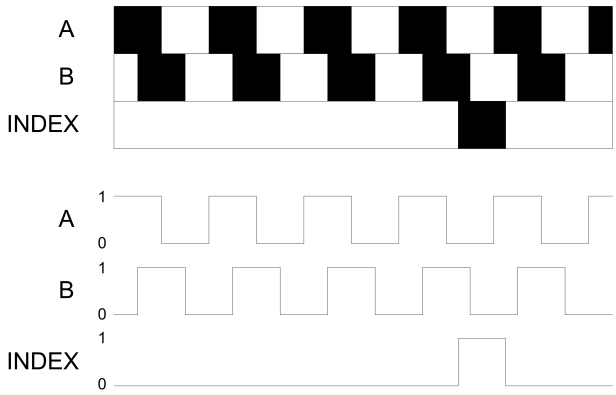


Figure 1.2: Optical incremental encoder signals

The A and B signals that are generated as the shaft rotates are used in a decoder algorithm to generate a count. The resolution of the encoder depends on the coding of the disk and the decoder. For example, a single encoder with 512 lines on the disk can generate a total of 512 counts for every rotation of the encoder shaft. However, in a quadrature decoder as depicted in Figure 1.2, the number of counts (and thus its resolution) quadruples for the same line patterns and generates 2048 counts per revolution. This can be explained by the offset between the A and B patterns: Instead of a single strip being either on or off, now there is two strips that can go through a variety of on/off states before the cycle repeats. This offset also allows the encoder to detect the directionality of the rotation, as the sequence of on/off states differs for a clockwise and counter-clockwise rotation.

2 In-Lab Exercises

In this lab, we will make a Simulink® model using QUARC® blocks to drive to the DC motor and the measure it corresponding angle - similarly as shown in Figure 2.1.

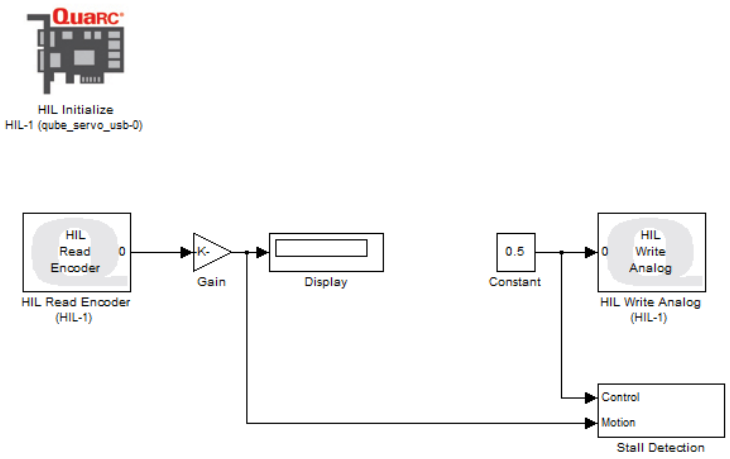


Figure 2.1: Simulink model used with QUARC® to drive motor and read angle on QUBE-Servo

2.1 Configuring a Simulink® Model for the QUBE-Servo

Follow these steps build a Simulink® model that will interface to the QUBE-Servo using QUARC®:

1. Load the Matlab® software.
2. Create a new Simulink® diagram by going to *File* | *New* | *Model* item in the menu bar.
3. Open the Simulink® Library Browser window by clicking on the *View* | *Library Browser* item in the Simulink® menu bar or clicking on the Simulink® icon.
4. Expand the *QUARC Targets* item and go to the *Data Acquisition* | *Generic* | *Configuration* folder, as shown in Figure 2.2.
5. Click-and-drag the HIL Initialize block from the library window into the blank Simulink® model. This block is used to configure your data acquisition device.
6. Double-click on the HIL Initialize block.
7. Make sure the QUBE-Servo is connected to your PC USB port and powered *ON* (the Power LED should be lit).

   External DAQ Users: If you are using the QUBE-Servo with an external data acquisition device, then make sure the QUBE-Servo has been properly connected to your DAQ as dictated in the QUBE-Servo User Manual.
8. In the *Board type* field, select *qube_servo_usb*.

   External DAQ Users: If you are using an external data acquisition device, then select the board that is installed in your PC. For example, if you are using a Quanser Q8-USB board to connect to the QUBE-Servo then select *q8_usb*.
9. Go to the *QUARC* | *Set default options* item to set the correct Real-Time Workshop parameters and setup the Simulink® model for external use (as opposed to the simulation mode).

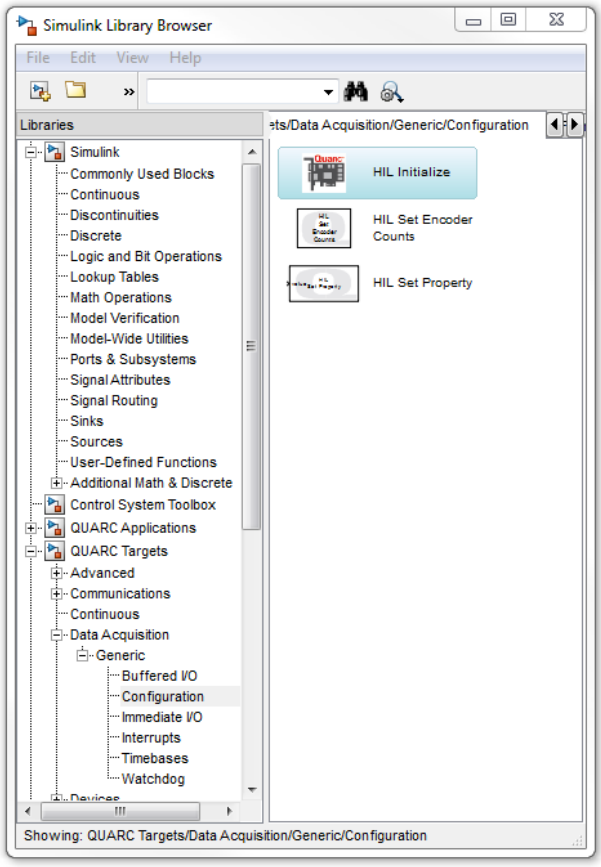


Figure 2.2: *QUARC Targets* in Simulink® Library Browser

10. Select the *QUARC* | *Build* item. Various lines in the Matlab® Command Window should be displayed as the model is being compiled. This creates a QUARC® executable (.exe) file which we will commonly refer to as a QUARC® controller.

11. Run the QUARC® controller. To do this, go to the Simulink® model tool bar, shown in Figure 2.3, and click on the *Connect to target* icon and then on the *Run* icon. You can also go *QUARC* | *Start* to run the code. The Power LED on the QUBE-Servo (or your DAQ card) should be blinking.

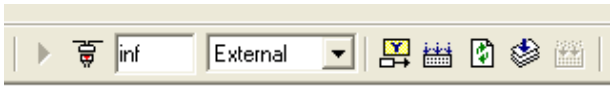


Figure 2.3: Simulink® model toolbar: connect to target and compilation

12. If you successfully ran the QUARC® controller without any errors, then you can stop the code by clicking on the *Stop* button in the tool bar (or go to *QUARC* | *Stop*).

2.2 Reading the Encoder

Follow these steps to read the encoder:

1. Using the Simulink® model you configured for the QUBE-Servo in the previous section, add the HIL Read Encoder block from the *QUARC Targets* | *Data Acquisition* | *Generic* | *Timebases* category in the Library Browser.

2. Connect the HIL Read Encoder to a Gain and Display block similar to Figure 2.1 (without the HIL Write Analog block). In the Library Browser, you can find the Display block from the *Simulink* | *Sinks* and the Gain block from *Simulink* | *Math Operations*.

3. Build the QUARC® controller. The code needs to be re-generated again because we have modified the Simulink diagram.

4. Run the QUARC® controller.

5. Rotate the disk back and forth. The Display block shows the number of counts measured by the encoder. The encoder counts are proportional to the angle of disk.

6. What happens to the encoder reading every time the QUARC® controller is started? Stop the controller, move around the disk, and re-start the controller. What do you notice about the encoder measurement when the controller is re-started?

7. Measure how many counts the encoder outputs for a full rotation. Briefly explain your procedure to determine this and validate that this matches the specifications given in the QUBE-Servo User Manual.

8. Ultimately we want to display the disk angle in degrees, not counts. Set the Gain block to a value that converts counts to degrees. This is called the *sensor gain*. Run the QUARC® controller and confirm that the Display block shows the angle of the disk correctly.

2.3 Driving the DC Motor

1. Add the HIL Write Analog block from the *Data Acquisition* | *Generic* | *Immediate I/O* category into your Simulink® diagram. This block is used to output a signal from analog output channel #0 on the data acquisition device. This is connected to the on-board PWM amplifier which drives the DC motor.

2. Add the Constant block found in the *Simulink* | *Sources* folder to your Simulink model. Connect the Constant and HIL Write Analog blocks together, as shown in Figure 2.1.

   Note: We recommend including a Stall Monitor block that is part of the Stall Detection block in Figure 2.1 and Figure 2.4. This block will monitor the applied voltage and speed of the DC motor to ensure that it does not stall. If the motor is motionless for more than 20 s with an applied voltage of over ± 5 V, the simulation is halted to prevent the QUBE-Servo from overheating and subsequent potential damage to the motor.

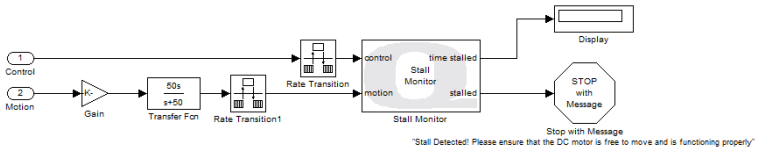


Figure 2.4: Stall Detection Subsystem

3. Build and run the QUARC® controller.

4. Set the Constant block to 0.5. This applies 0.5 V to the DC motor in the QUBE-Servo. Confirm that we are obtaining a *positive measurement when a positive signal is applied*. This convention is important, especially in control systems when the design assumes the measurement goes up positively when a positive input is applied. Finally, in what direction does the disk rotate (clockwise or counter-clockwise) when a positive input is applied?

5. Stop the QUARC® controller.

6. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Filtering

Topics Covered

- Using an encoder to measure speed.
- Low-pass filters.

Prerequisites

- QUBE-Servo Integration laboratory experiment.

1 Background

A low-pass filter can be used to block out the high-frequency components of a signal. A first-order low-pass filter transfer function has the form

$$G(s) = \frac{\omega_f}{s + \omega_f}, \tag{1.1}$$

where ω_f is the cut-off frequency of the filter in radians per seconds (rad/s). All higher frequency components of the signal will be attenuated by at least -3 dB $\approx 50\%$.

2 In-Lab Exercises

Based on the model developed in the Integration lab, the goal is to design a model that measures the servo velocity using the encoder as shown in Figure 2.1.

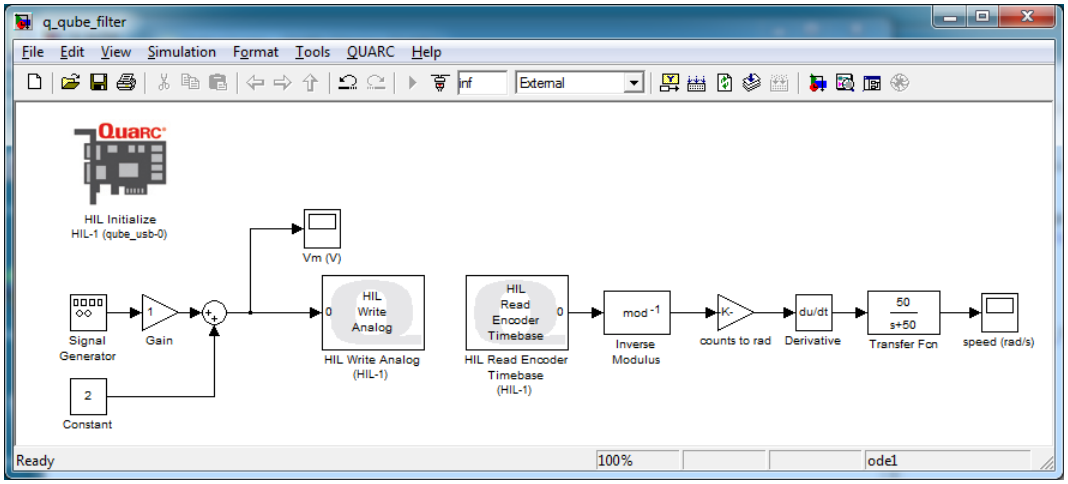
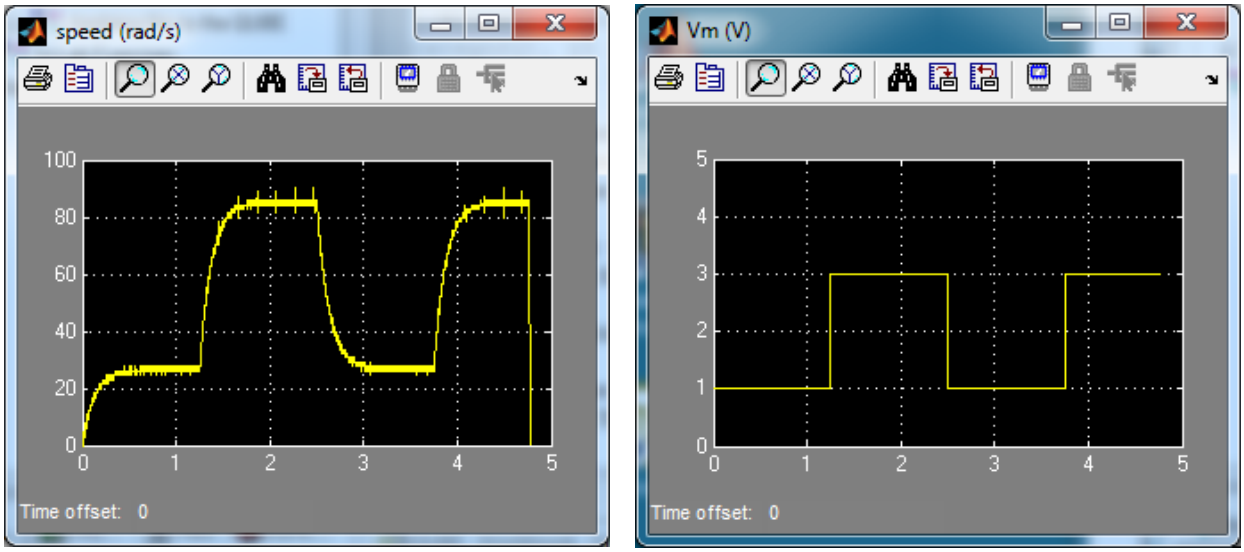


Figure 2.1: Measuring speed using the encoder

1. Take the model you developed in the QUBE-Servo Integration lab. Change the encoder calibration gain to measure the gear position in radians (instead of degrees). What is the value of your gain?
2. Build the Simulink® diagram shown in Figure 2.1 but, for now, do not include the Transfer Fcn block (will be added later).
   - **Inverse Modulus**: Since the QUBE-Servo DAQ has 16-bit counters, the valid count range is from $-2^{15} = -32768$ to $2^{15}-1 = 32767$. To eliminate the discontinuous jump that would occur when the encoder reaches the limits of this range, add an Inverse Modulus block from the *QUARC Targets* | *Discontinuous* category into the Simulink diagram, as shown in Figure 2.1. Set the modulus to the total buffer size, i.e. 2^{16}.
   - **Derivative**: Add a Derivative block to the encoder calibration gain output to measure the gear speed using the encoder (in rad/s).
   - **Scope**: Connect the output of the Derivative to a Scope.
3. Setup the source blocks to output a *step* voltage that goes from 1 V to 3 V at 0.4 Hz.
4. Build and run the QUARC controller. Examine the encoder speed response. Attach sample responses. They should look similar to Figure 2.2.



(a) Encoder Speed

(b) Motor Voltage

Figure 2.2: Measured servo speed using encoder

5. Explain why the encoder-based measurement is noisy.

   Hint: Measure the encoder position measurement using a new Scope. Zoom up on the position response and remember that this later enters derivative. Is the signal continuous?

6. One way to remove some of the high-frequency components is adding a low-pass filter (LPF) to the derivative output. From the *Simulink* | *Continuous Simulink* library, add a Transfer Fcn block after the Derivative output and connect LPF to the Scope. Set the Transfer Fcn block to $50/(s+50)$, as illustrated in Figure 2.1.

7. Build and run the QUARC controller. Show the filtered encoder-based speed response and the motor voltage. Has it improved?

8. What is the cutoff frequency of the low-pass filter $50/(s+50)$? Give you answer in both rad/s and Hz.

9. Vary the cutoff frequency, ω_f, between 10 to 200 rad/s (or 1.6 to 32 Hz). What effect does it have on the filtered response? Consider the benefit and the trade-off of lowering and increasing this parameter.

10. Stop the QUARC® controller.

11. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Stability Analysis

Topics Covered

- Stable, marginally stable, and unstable systems.
- Open-loop position and speed response of a servo.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- Filtering laboratory experiment.

1 Background

1.1 Servo Model

The QUBE-Servo voltage-to-speed transfer function is

$$P_{v-s}(s) = \frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s + 1}, \tag{1.1}$$

where $K = 23.0$ rad/(V-s) is the model steady-state gain, $\tau = 0.13$ s is the model time constant, $\Omega_m(s) = \mathcal{L}\left[\omega_m(t)\right]$ is the motor speed (i.e. speed of load disk), and $V_m(s) = \mathcal{L}\left[v_m(t)\right]$ is the applied motor voltage. If desired, you can conduct an experiment to find more precise model parameters, K and τ, for your particular servo (e.g. performing the Bump Test Modeling lab).

The voltage-to-position process transfer function the same as Equation 1.1 with an integrator in series

$$P_{v-p} = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)} \tag{1.2}$$

where $\Theta_m(s) = \mathcal{L}\left[\theta_m(t)\right]$ is the load gear position.

1.2 Stability

Definition for Bounded-Input Bounded-Output (BIBO) stability is:

1. A system is stable if every bounded input yields a bounded output.
2. A system is unstable if any bounded input yields a unbounded output.

The stability of a system can be determined from its poles:

- Stable systems have poles only in the left-hand plane.
- Unstable systems have at least one pole in the right-hand plane and/or poles of multiplicity greater than 1 on the imaginary axis.
- Marginally stable systems have one pole on the imaginary axis and the other poles in the left-hand plane.

2 In-Lab Exercises

Based on the models already designed in QUBE-Servo QUBE-Servo Integration and Filtering laboratory experiment, design a model that applies a step of 1 V to the motor and reads the servo velocity and the position as shown in Figure 2.1.

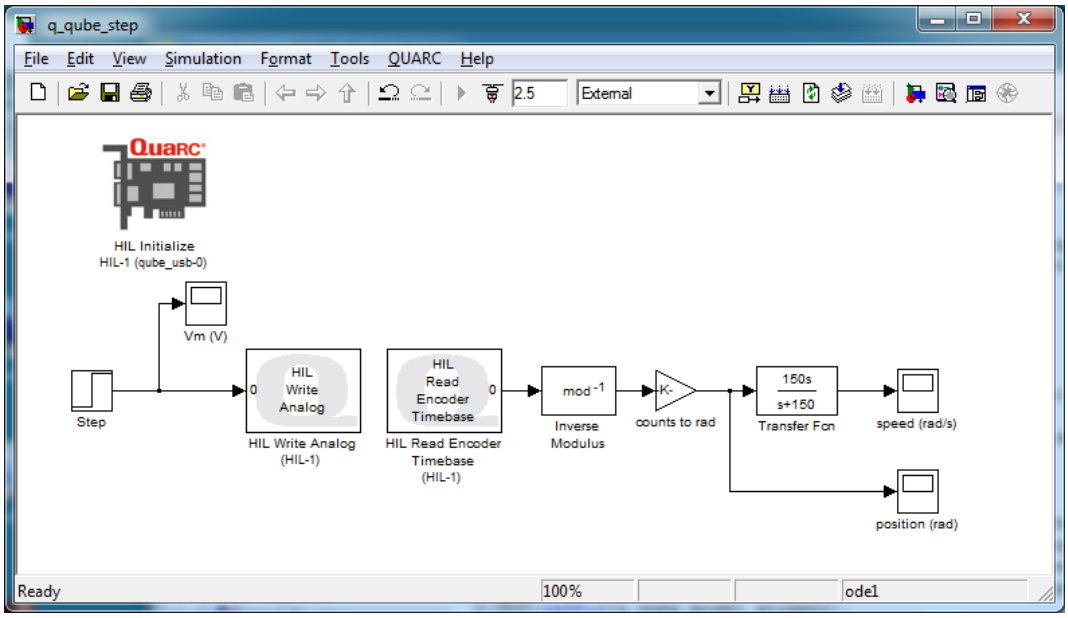
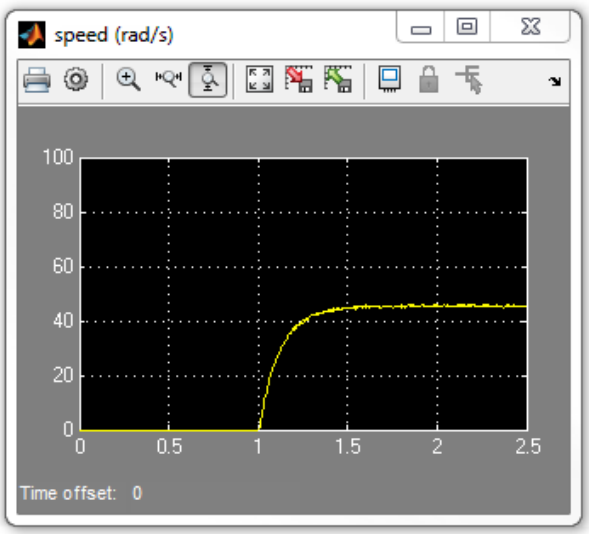
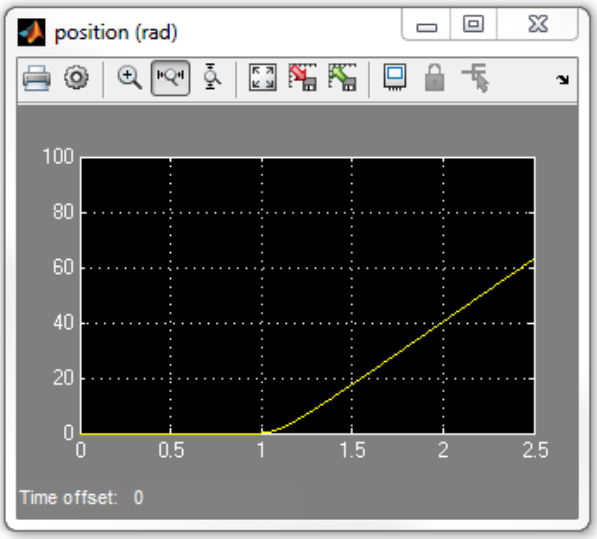


Figure 2.1: Measuring speed and position when applying a step

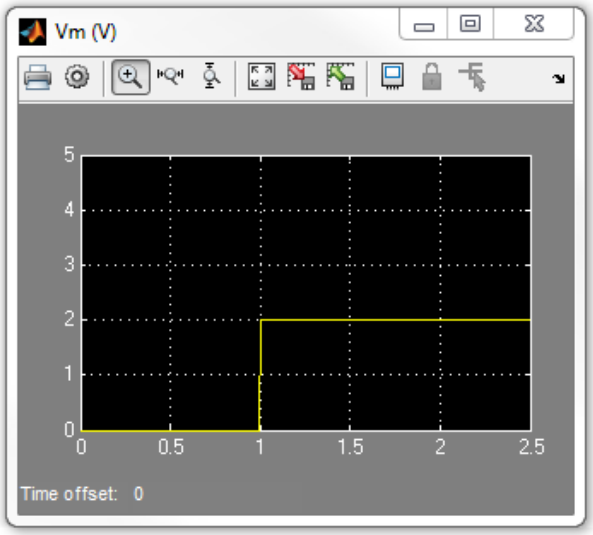
1. Determine the stability of the voltage-to-speed servo system from its poles.
2. Determine the stability of the voltage-to-position servo system from its poles.
3. Apply a unit step voltage to the servo by running the QUARC® model shown in Figure 2.1. The position and speed step response should be similar to Figure 2.2.



(a) Speed Response



(b) Position Response



(c) Voltage Input

Figure 2.2: Step Response

4. Based on the *speed* response and the BIBO principle, what is the stability of the system? How does this compare with your results from the pole analysis. Similarly, assess the stability of the system using the *position* response using BIBO and the pole analysis.

5. Based on the *position* response and the BIBO principle, what is the stability of the system? How does this compare with your results from the pole analysis.

6. Is there an input where the open-loop servo position response is stable? If so then modify the Simulink diagram to include your input, test it on the servo, and show the position response. Based on this result, how could you define marginal stability in terms of bounded inputs?

   Hint: Try an impulse (i.e. short step) or sinusoid input and compare the position response with the step response observed earlier.

7. Stop the QUARC® controller.

8. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Bump Test Modeling

Topics Covered

- First order transfer functions.
- Obtaining the QUBE-Servo model using the bump test method.
- Model validation.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- Filtering laboratory experiment.

1 Background

The bump test is a simple test based on the step response of a stable system. A step input is given to the system and its response is recorded. As an example, consider a system given by the following transfer function:

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1} \tag{1.1}$$

The step response shown in Figure 1.1 is generated using this transfer function with $K = 5$ rad/V.s and $\tau = 0.05$ s.

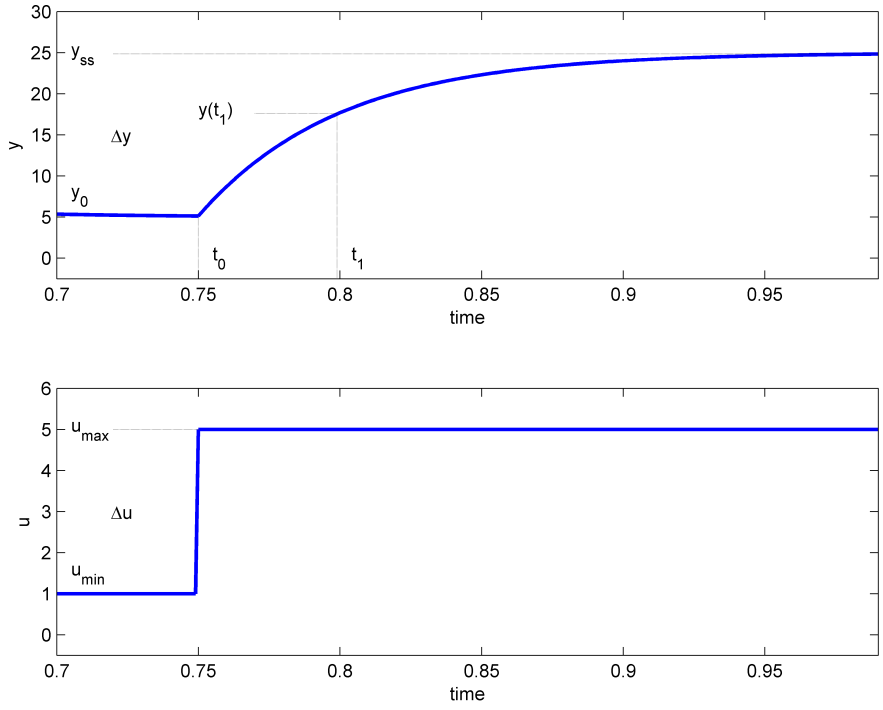


Figure 1.1: Input and output signal used in the bump test method

The step input begins at time t_0. The input signal has a minimum value of u_{min} and a maximum value of u_{max}. The resulting output signal is initially at y_0. Once the step is applied, the output tries to follow it and eventually settles at its steady-state value y_{ss}. From the output and input signals, the steady-state gain is

$$K = \frac{\Delta y}{\Delta u} \tag{1.2}$$

where $\Delta y = y_{ss} - y_0$ and $\Delta u = u_{max} - u_{min}$. The time constant of a system τ is defined as the time it takes the system to respond to the application of a step input to reach $1 - 1/e \approx 63.2\%$ of its steady-state value, i.e. for Figure 1.1

$$t_1 = t_0 + \tau,$$

where

$$y(t_1) = 0.632\Delta y + y_0. \tag{1.3}$$

Then, we can read the time t_1 that corresponds to $y(t_1)$ from the response data in Figure 1.1. From this, the model time constant can be found as:

$$\tau = t_1 - t_0. \tag{1.4}$$

1.1 Applying this to the QUBE-Servo

Going back to the QUBE-Servo system, the s-domain representation of a step input voltage with a time delay t_0 is given by

$$V_m(s) = \frac{A_v e^{(-st_0)}}{s}, \tag{1.5}$$

where A_v is the amplitude of the step and t_0 is the step time (i.e. the delay).

The voltage-to-speed transfer function is

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s + 1} \tag{1.6}$$

where K is the model steady-state gain, τ is the model time constant, $\Omega_m(s) = \mathcal{L}\left[\omega_m(t)\right]$ is the load gear rate, and $V_m(s) = \mathcal{L}\left[v_m(t)\right]$ is the applied motor voltage.

If we substitute input Equation 1.5 into the system transfer function Equation 1.6, we get:

$$\Omega_m(s) = \frac{K A_v e^{(-st_0)}}{(\tau s + 1)s}. \tag{1.7}$$

We can then find the QUBE-Servo motor speed step response in the time domain $\omega_m(t)$ by taking inverse Laplace of this equation

$$\omega_m(t) = K A_v \left(1 - e^{\left(-\frac{t-t_0}{\tau}\right)}\right) + \omega_m(t_0), \tag{1.8}$$

noting the initial conditions $\omega_m(0^-) = \omega_m(t_0)$.

2 In-Lab Exercises

Based on the models already designed in QUBE-Servo QUBE-Servo Integration and Filtering laboratory experiment, design a model that applies a step of 2 V to the motor and reads the servo velocity using the encoder as shown in Figure 2.1.

To apply your step for a certain duration (e.g. 2.5 s), set the *Simulation stop time* of the Simulink® model. Using the saved response, the model parameters can then be found as discussed in Background section of this lab. For information on saving data to Matlab for offline analysis, see the QUARC® help documentation (under *QUARC Targets* | *User's Guide* | *QUARC Basics* | *Data Collection*).

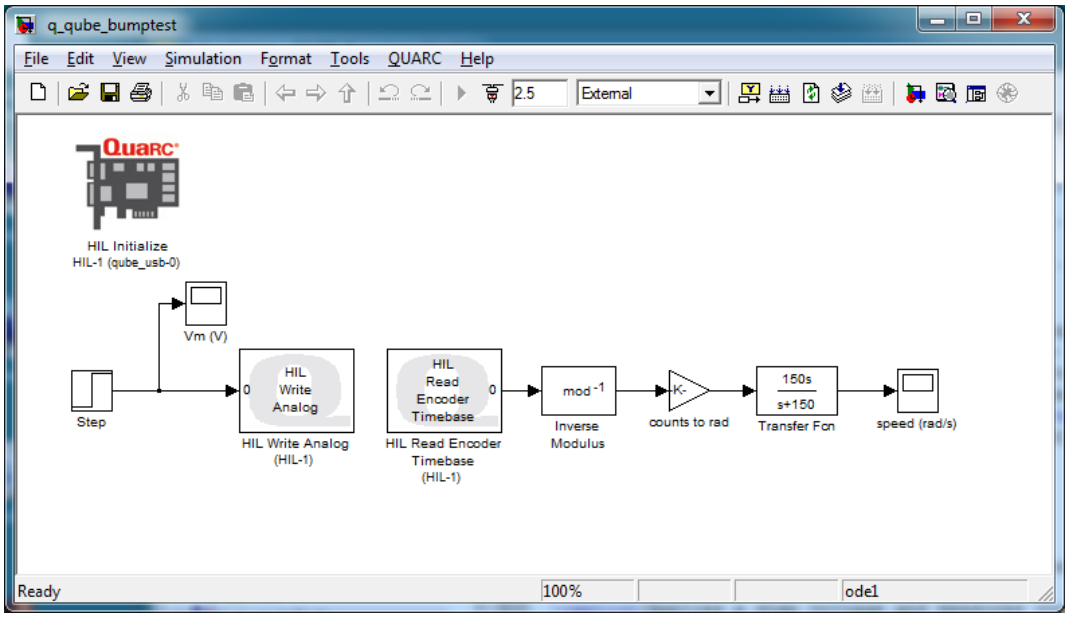
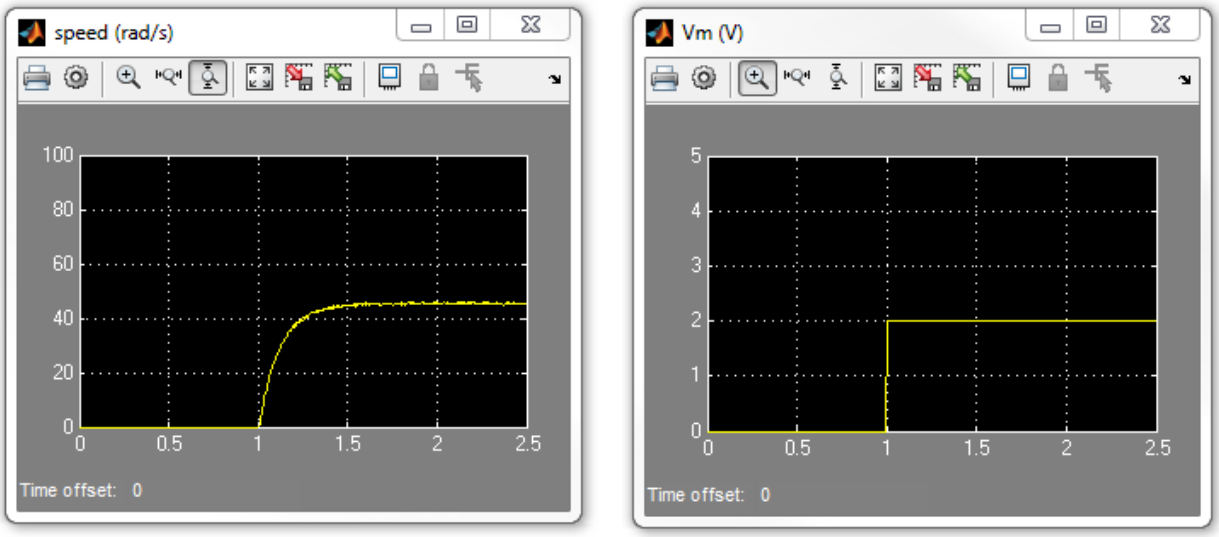


Figure 2.1: Applies a step voltage and measures corresponding servo speed

1. Run the QUARC controller to apply a 2 V step to the servo. The scope response should be similar to Figure 2.2.
2. Plot the response in Matlab® figure. For example, you can setup the scopes to save the measured load/disk speed and motor voltage to the Matlab® workspace in the variables `data_wm` and `data_vm`, where the `data_wm(:,1)` is the time vector and `data_wm(:,2)` is the measured speed.
3. Find the steady-state gain using the measured step response.

   Hint: Use the Matlab® *ginput* command to measure points off the plot.
4. Find the time constant from the obtained response.
5. To check if your derived model parameters K and τ are correct, modify the Simulink diagram to include a Transfer Fcn block with the first-order model in Equation 1.1, as shown in Figure 2.3. Connect both the measured and simulated QUBE-Servo responses to the scope using a Mux block (from the *Signal Routing* category). Build and run your QUARC® controller. Attach a Matlab® figure displaying both the measured and simulated response in one plot, as well as in the input voltage.
6. Did you derive the model parameters K and τ correctly? Explain.
7. Stop the QUARC® controller.
8. Power *OFF* the QUBE-Servo.



(a) Load Speed (b) Motor Voltage

Figure 2.2: QUBE-Servo Bump Test Response

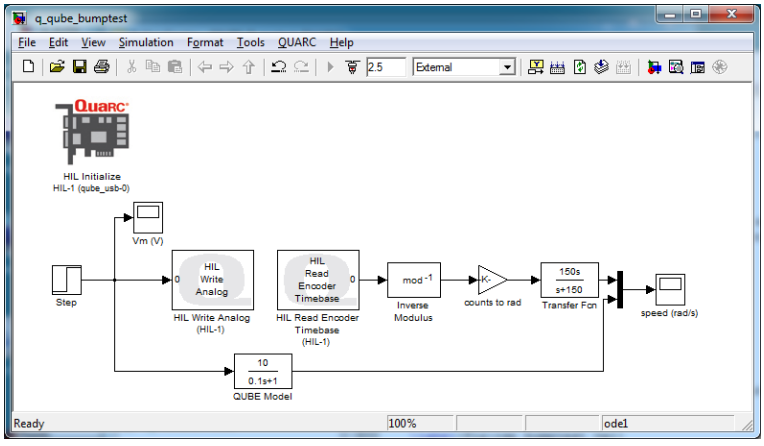


Figure 2.3: Validating bump test model

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

First Principles Modeling

Topics Covered

- Obtaining the equations of motion of a DC motor based rotary servo.
- Creating and validating a system model.
- Model validation.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- Filtering laboratory experiment.

1 Background

The Quanser QUBE-Servo is a direct-drive rotary servo system. Its motor armature circuit schematic is shown in Figure 1.1 and the electrical and mechanical parameters are given in Table 1.1. The DC motor shaft is connected to the *load hub*. The hub is a metal disk used to mount the disk or rotary pendulum and has a moment of inertia of J_h. A disk load is attached to the output shaft with a moment of inertia of J_d.

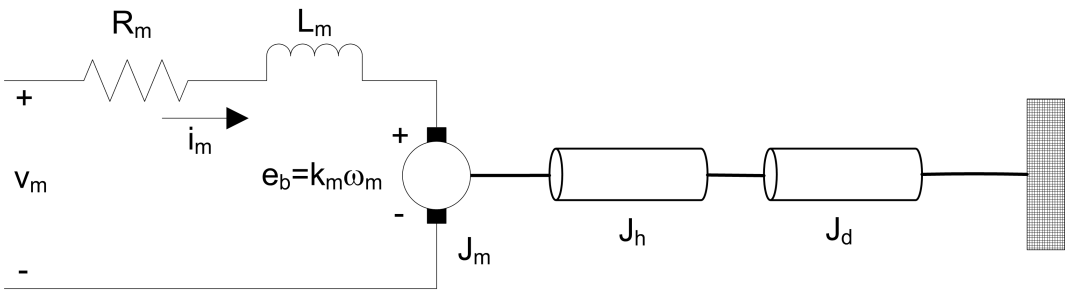


Figure 1.1: QUBE-Servo DC motor and load

The back-emf (electromotive) voltage $e_b(t)$ depends on the speed of the motor shaft, ω_m, and the back-emf constant of the motor, k_m. It opposes the current flow. The back emf voltage is given by:

$$e_b(t) = k_m \omega_m(t) \tag{1.1}$$

Symbol	Description	Value
DC Motor		
R_m	Terminal resistance	8.4Ω
k_t	Torque constant	0.042 N.m/A
k_m	Motor back-emf constant	0.042 V/(rad/s)
J_m	Rotor inertia	4.0×10^{-6} kg.m^2
L_m	Rotor inductance	1.16 mH
m_h	Load hub mass	0.0106 kg
r_h	Load hub mass	0.0111 m
J_h	Load hub inertia	0.6×10^{-6} kg.m^2
Load Disk		
m_d	Mass of disk load	0.053 kg
r_d	Radius of disk load	0.0248 m

Table 1.1: QUBE-Servo system parameters

Using Kirchoff's Voltage Law, we can write the following equation:

$$v_m(t) - R_m i_m(t) - L_m \frac{di_m(t)}{dt} - k_m \omega_m(t) = 0. \tag{1.2}$$

Since the motor inductance L_m is much less than its resistance, it can be ignored. Then, the equation becomes

$$v_m(t) - R_m i_m(t) - k_m \omega_m(t) = 0. \tag{1.3}$$

Solving for $i_m(t)$, the motor current can be found as:

$$i_m(t) = \frac{v_m(t) - k_m \omega_m(t)}{R_m}. \tag{1.4}$$

The motor shaft equation is expressed as

$$J_{eq}\dot{\omega}_m(t) = \tau_m(t), \tag{1.5}$$

where J_{eq} is total moment of inertia acting on the motor shaft and τ_m is the applied torque from the DC motor. Based on the current applied, the torque is

$$\tau_m = k_m i_m(t) \tag{1.6}$$

The moment of inertia of a disk about its pivot, with mass m and radius r, is

$$J = \frac{1}{2} m r^2. \tag{1.7}$$

2 In-Lab Exercises

Based on the models already designed in QUBE-Servo Integration and Filtering laboratory experiment, design a model that applies a $1-3$ V, 0.4 Hz square wave to the motor and reads the servo velocity using the encoder as shown in Figure 2.1.

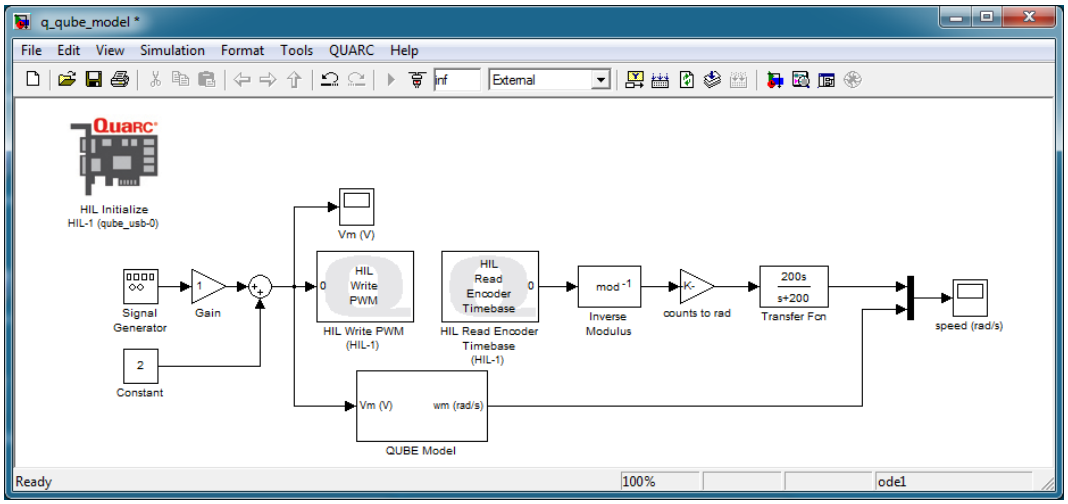


Figure 2.1: Applies a step voltage and displays measured and simulated QUBE-Servo speed.

Create subsystem called *QUBE-Servo Model*, as shown in Figure 2.1, that contains blocks to model the QUBE-Servo system. Thus using the equations given above, assemble a simple block digram in Simulink® to model the system. You'll need a few Gain blocks, a Subtract block, and an Integrator block (to go from acceleration to speed). Part of the solution is shown in Figure 2.2.

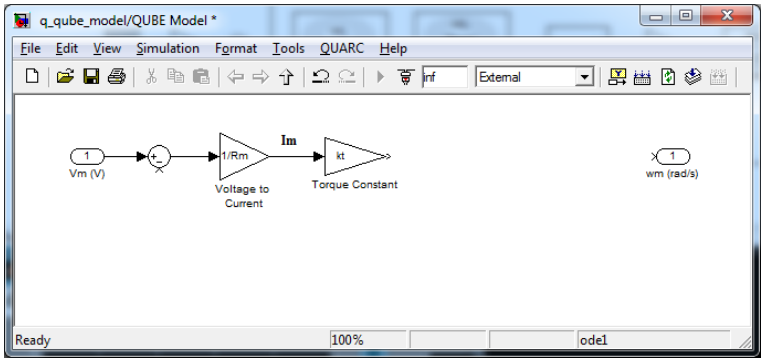
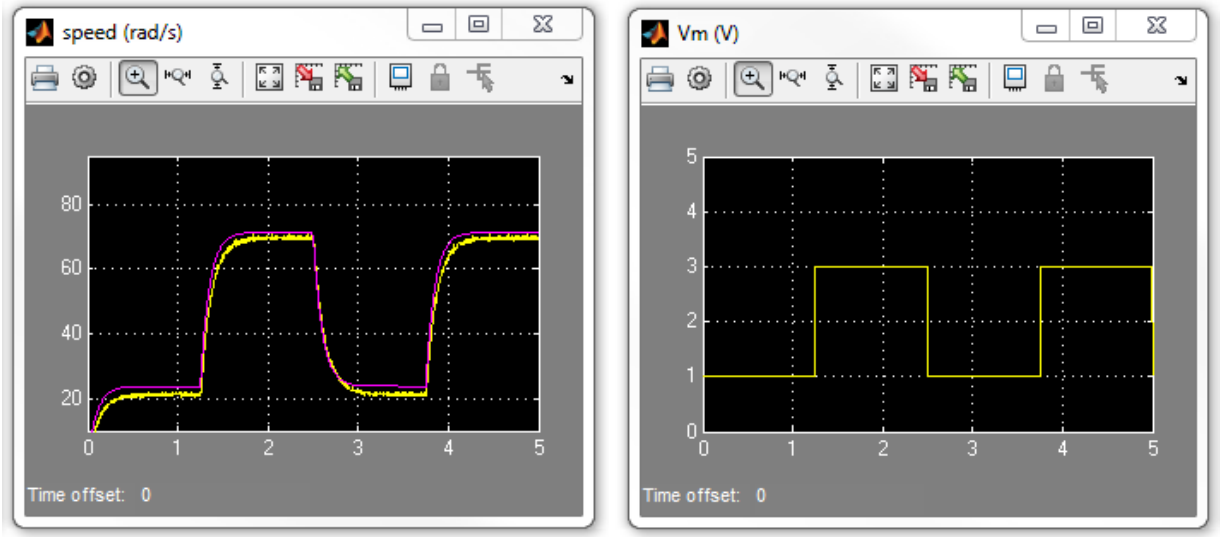


Figure 2.2: Incomplete *QUBE-Servo Model* subsystem.

It may also help to write a short Matlab® script that sets the various system parameters in Matlab®Matlab, so you can use the symbol instead of entering the value numerically in the Gain blocks. In the example shown in Figure 2.2, we are using *Rm* for motor resistance and *kt* for the current-torque constant. To define these, write a script like:

1. The motor shaft of the QUBE-Servo is attached to a *load hub* and a disk load. Based on the parameters given in Table 1.1, calculate the equivalent moment of inertia that is acting on the motor shaft.
2. Design the *QUBE-Servo Model* subsystem as described above. Attach a screen capture of your model and the Matlab® script (if you used one).
3. Build and run the QUARC controller with your QUBE-Servo model. The scope response should be similar to Figure 2.3. Attach a screen capture of your scopes. Does your model represent the QUBE-Servo well? Explain.



(a) Motor Speed

(b) Motor Voltage

Figure 2.3: QUBE Response

4. Stop the QUARC® controller.
5. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Second-Order Systems

Topics Covered

- Underdamped second-order systems.
- Damping ratio and natural frequency.
- Peak time and percent overshoot time-domain specifications.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- Filtering laboratory experiment.

1 Background

1.1 Second-Order Step Response

The *standard second-order* transfer function has the form

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{1.1}$$

where ω_n is the natural frequency and ζ is the damping ratio. The properties of its response depend on the values of the parameters ω_n and ζ.

Consider a second-order system as shown in Equation 1.1 subjected to a step input given by

$$R(s) = \frac{R_0}{s},$$

with a step amplitude of $R_0 = 1.5$. The system response to this input is shown in Figure 1.1, where the red trace is the output response $y(t)$ and the blue trace is the step input $r(t)$.

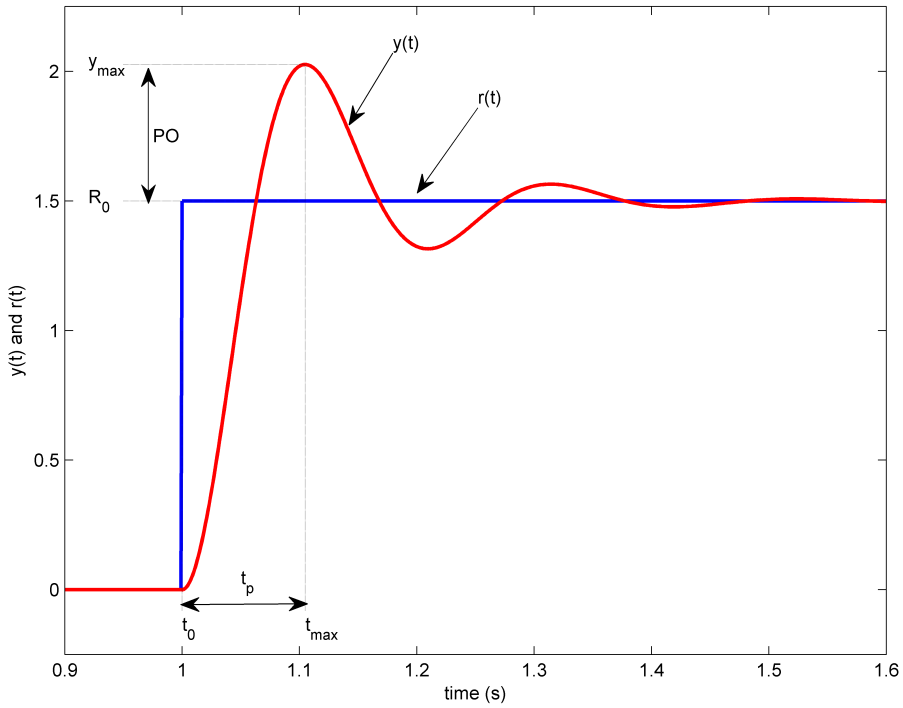


Figure 1.1: Standard second-order step response

1.2 Peak Time and Overshoot

The maximum value of the response is denoted by the variable y_{max} and it occurs at a time t_{max}. For a response similar to Figure 1.1, the percent overshoot is found using

$$PO = \frac{100\,(y_{max} - R_0)}{R_0}. \tag{1.2}$$

From the initial step time, t_0, the time it takes for the response to reach its maximum value is

$$t_p = t_{max} - t_0. \tag{1.3}$$

This is called the *peak time* of the system.

In a second-order system, the amount of overshoot depends solely on the damping ratio parameter and it can be calculated using the equation

$$PO = 100e^{\left(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}\right)}. \tag{1.4}$$

The peak time depends on both the damping ratio and natural frequency of the system and it can be derived as:

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}. \tag{1.5}$$

Generally speaking, the damping ratio affects the shape of the response while the natural frequency affects the speed of the response.

1.3 Unity Feedback

The unity-feedback control loop shown in Figure 1.2 will be used to control the position of the QUBE-Servo.

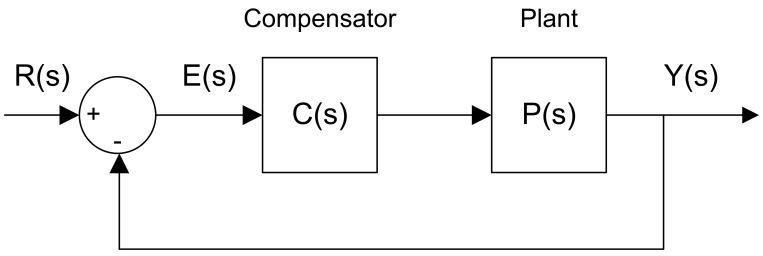


Figure 1.2: Unity feedback loop

The QUBE-Servo voltage-to-position transfer function is

$$P(s) = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s+1)}. \tag{1.6}$$

where $K = 23.0$ rad/(V − s) is the model steady-state gain, $\tau = 0.13$ s is the model time constant, $\Theta_m(s) = \mathcal{L}\left[\theta_m(t)\right]$ is the motor / disk position, and $V_m(s) = \mathcal{L}\left[v_m(t)\right]$ is the applied motor voltage. If desired, you can conduct an experiment to find more precise model parameters K and τ for your particular servo (e.g. performing the Bump Test Modeling laboratory experiment).

The controller is denoted by $C(s)$. In this lab, we are only going to use unity feedback therefore

$$C(s) = 1. \tag{1.7}$$

The closed-loop transfer function of the QUBE-Servo position control from the reference input $R(s) = \Theta_d(s)$ to the output $Y(s) = \Theta_m$ using unity feedback as shown in Figure 1.2 is

$$\frac{\Theta_d(s)}{V_m(s)} = \frac{\frac{K}{\tau}}{s^2 + \frac{1}{\tau}s + \frac{K}{\tau}}. \tag{1.8}$$

2 In-Lab Exercises

Design a Simulink® model similar to Figure 2.1. This implements the unity feedback control given in Figure 1.2 in Simulink®. A step reference (i.e. desired position or setpoint) of 1 rad is applied at 1 second and the controller runs for 2.5 seconds.

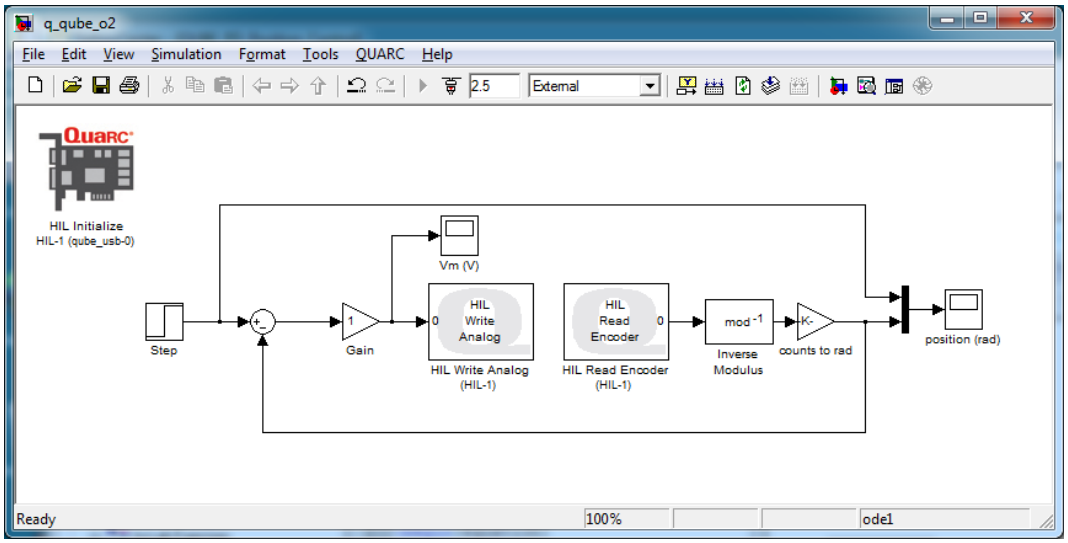
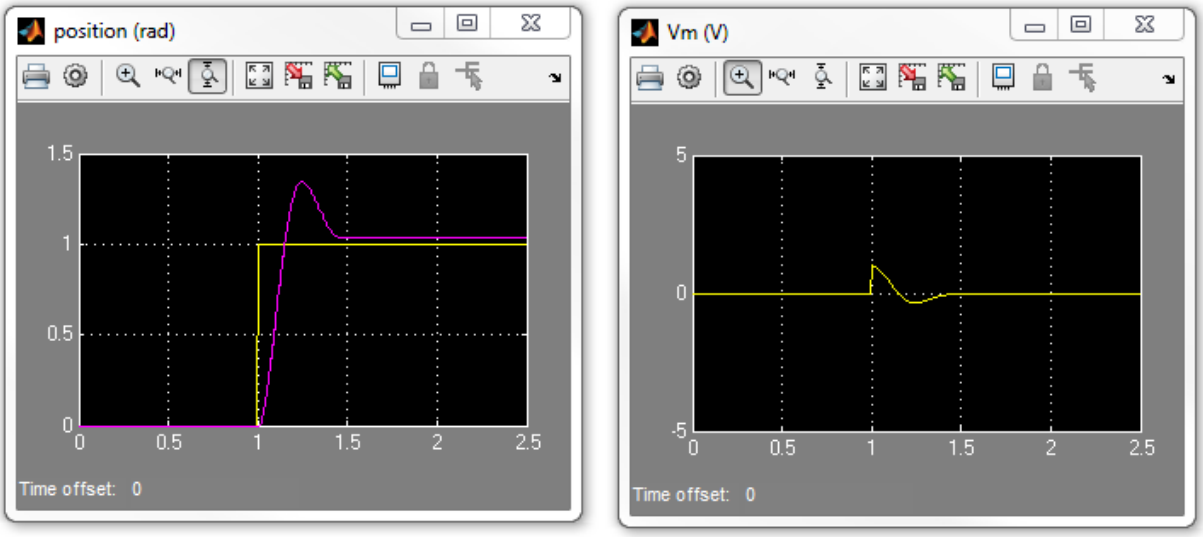


Figure 2.1: Unity feedback position control of QUBE-Servo

1. Given the QUBE-Servo closed-loop equation under unity feedback in Equation 1.8 and the model parameters above, find the natural frequency and damping ratio of the system.
2. Based on your obtained ω_n and ζ, what is the expected peak time and percent overshoot?
3. Build and run the QUARC® controller. The scopes should look similar to Figure 2.2.



(a) Position

(b) Voltage

Figure 2.2: Unity feedback QUBE-Servo step response

4. Attach the QUBE-Servo position response - showing both the setpoint and measured positions in one scopes - as well as the motor voltage.

Hint: For information on saving data to Matlab® for offline analysis, see the QUARC® help documentation (under *QUARC Targets* | *User's Guide* | *QUARC Basics* | *Data Collection*). You can then use the Matlab® `plot` command to generate the necessary Matlab figure.

5. Measure the peak time and percent overshoot from the response and compare that with your expect results.

   Hint: Use the Matlab® `ginput` command to measure points off the plot.

6. Stop the QUARC® controller.

7. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

PD Control

Topics Covered

- Servo position control.
- Proportional-derivative (PD) compensator.
- Designing control according to specifications.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- Filtering laboratory experiment.
- Second-Order Systems laboratory experiment.

1 Background

1.1 Servo Model

The QUBE-Servo voltage-to-position transfer function is

$$P(s) = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)}, \tag{1.1}$$

where $K = 23.2$ rad/(V-s) is the model steady-state gain, $\tau = 0.13$ s is the model time constant, $\Theta_m(s) = \mathcal{L}\left[\theta_m(t)\right]$ is the motor / disk position, and $V_m(s) = \mathcal{L}\left[v_m(t)\right]$ is the applied motor voltage. If desired, you can conduct an experiment to find more precise model parameters, K and τ, for your particular servo (e.g. performing the Bump Test Modeling laboratory experiment).

1.2 PID Control

The proportional, integral, and derivative control can be expressed mathematically as follows

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt}. \tag{1.2}$$

The corresponding block diagram is given in Figure 1.1. The control action is a sum of three terms referred to as proportional (P), integral (I) and derivative (D) control gain. The controller Equation 1.2 can also be described by the transfer function

$$C(s) = k_p + \frac{k_i}{s} + k_d s. \tag{1.3}$$

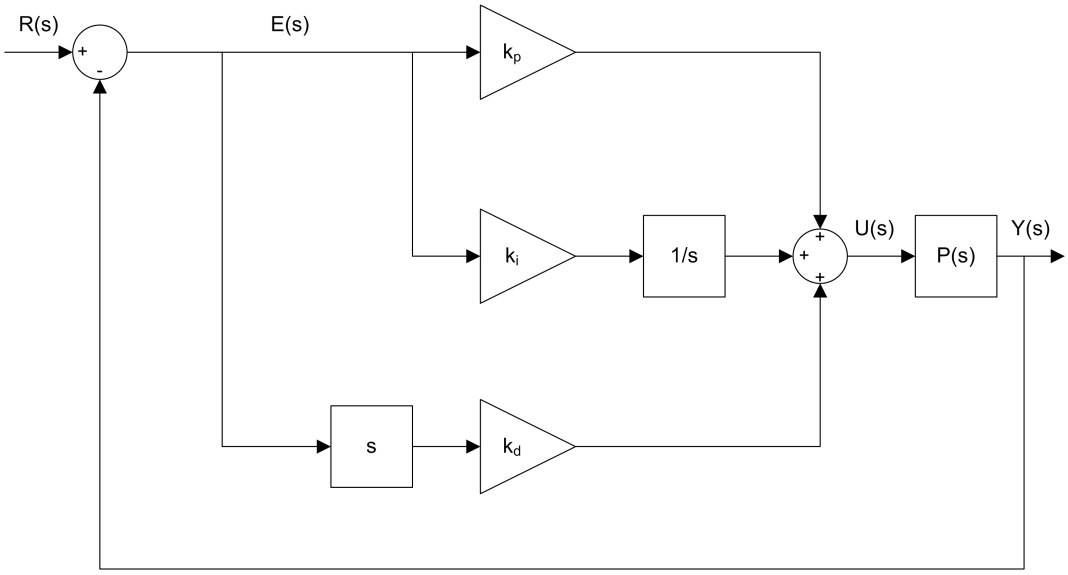


Figure 1.1: Block diagram of PID control

The functionality of the PID controller can be summarized as follows. The proportional term is based on the present error, the integral term depends on past errors, and the derivative term is a prediction of future errors.

The PID controller described by Equation 1.2 or Equation 1.3 is an ideal PID controller. However, attempts to implement such a controller may not lead to a good system response for real-world system. The main reason for this is that measured signals always include measurement noise. Therefore, differentiating a measured (noisy) signal will result in large fluctuations, thus will result in large fluctuations in the control signal.

1.3 PV Position Control

The integral term will not be used to control the servo position. A variation of the classic PD control will be used: the proportional-velocity control illustrated in Figure 1.2. Unlike the standard PD, only the negative velocity is fed back (i.e. not the velocity of the *error*) and a low-pass filter will be used in-line with the derivative term to suppress measurement noise. The combination of a first order low-pass filter and the derivative term results in a high-pass filter $H(s)$ which will be used instead of a direct derivative.

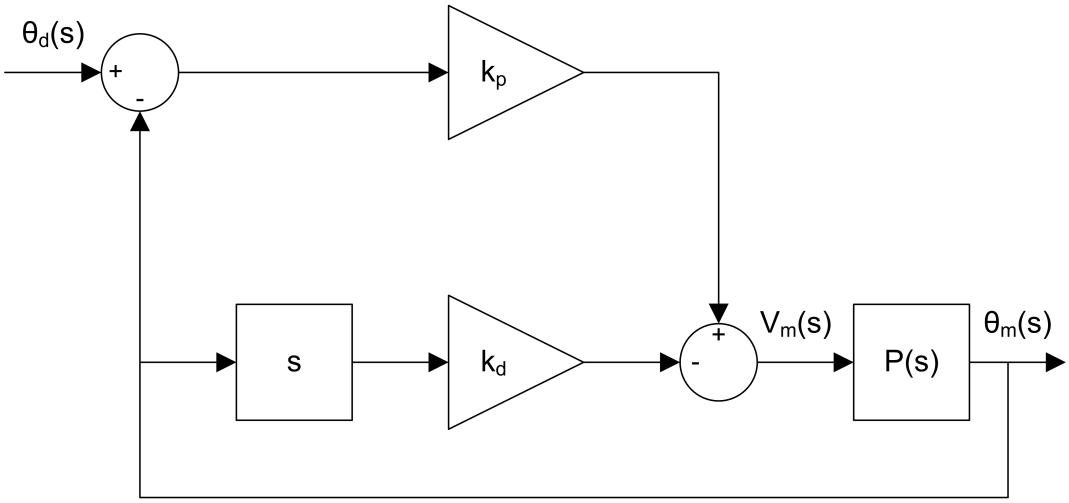


Figure 1.2: Block diagram of PV control

The proportional-velocity (PV) control has the following structure

$$u = k_p \left(r(t) - y(t) \right) - k_d \dot{y}(t), \tag{1.4}$$

where k_p is the proportional gain, k_d is the derivative (velocity) gain, $r = \theta_d(t)$ is the setpoint or reference motor / load angle, $y = \theta_m(t)$ is the measured load shaft angle, and $u = V_m(t)$ is the control input (applied motor voltage).

The closed-loop transfer function of the QUBE-Servo is denoted $Y(s)/R(s) = \Theta_m(s)/\Theta_d(s)$. Assume all initial conditions are zero, i.e. $\theta_m(0^-) = 0$ and $\dot{\theta_m}(0^-) = 0$, taking the Laplace transform of Equation 1.4 yields

$$U(s) = k_p(R(s) - Y(s)) - k_d s Y(s),$$

which can be substituted into Equation 1.1 to result in

$$Y(s) = \frac{K}{s\left(\tau s + 1\right)} \left(k_p \left(R(s) - Y(s) \right) - k_d s Y(s) \right).$$

Solving for $Y(s)/R(s)$, we obtain the closed-loop expression

$$\frac{Y(s)}{R(s)} = \frac{K k_p}{\tau s^2 + \left(1 + K k_d\right) s + K k_p}. \tag{1.5}$$

This is a second-order transfer function. Recall the standard second-order transfer function

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \tag{1.6}$$

2 In-Lab Exercises

Design the Simulink model shown in Figure 2.1. This implements the PV controller outlined in 1.3 with a high-pass filter of $100s/(s+100)$. Set the Signal Generator block such that the servo command (i.e. reference angle) is a square wave with an amplitude of 0.5 rad and at a frequency of 0.4 Hz.

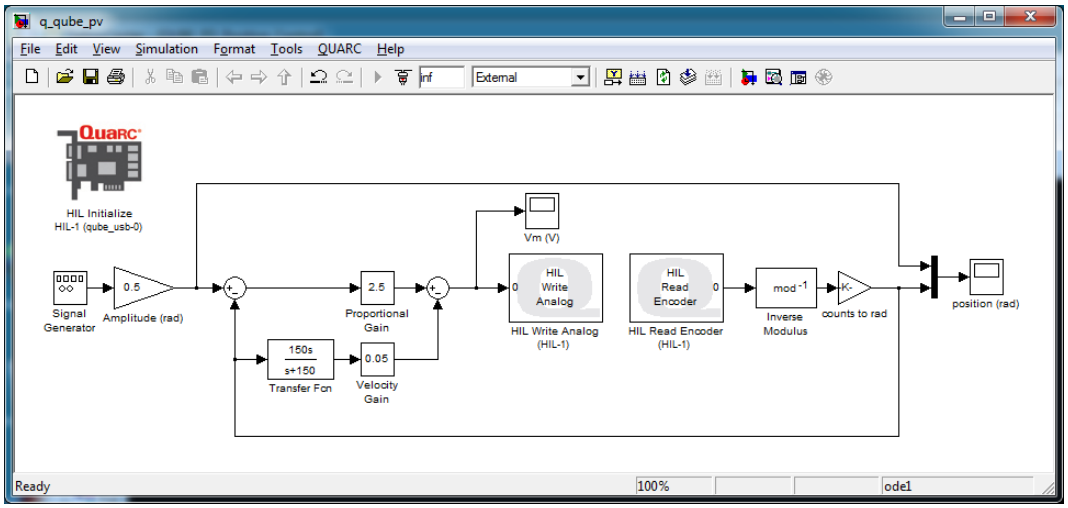
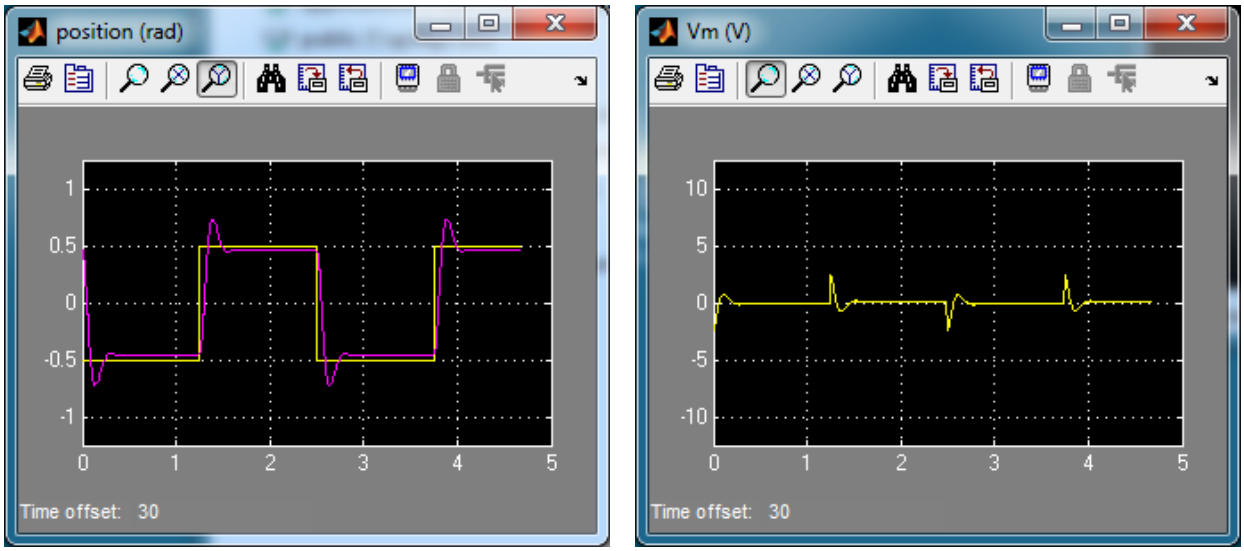


Figure 2.1: PV control on QUBE-Servo

1. Build and run the QUARC® controller. The response should look similarly as shown in Figure 2.2.



(a) Position

(b) Voltage

Figure 2.2: QUBE-Servo PV control with $k_p = 2.5$ V/rad and $k_d = 0.05$ V/(rad/s).

2. Set $k_p = 2.5$ V/rad and $k_d = 0$ V/(rad/s). Keep the derivative gain at 0 and vary k_p between 1 and 4. What does the proportional gain do when controlling servo position?
3. Set $k_p = 2.5$ V/rad and vary the derivative gain k_d between 0 and 0.15 V/(rad/s). What is its effect on the position response?
4. Stop the QUARC® controller.

5. Find the proportional and derivative gains required for the QUBE-Servoclosed-loop transfer function given in Equation 1.5 to match the standard second-order system in Equation 1.6. Your gain equations will be a function of ω_n and ζ.

6. Find the proportional and derivative gains required for the QUBE-Servo closed-loop transfer function given in Equation 1.5 to match the standard second-order system in Equation 1.6. Your gain equations will be a function of ω_n and ζ.

7. For the response to have a peak time of 0.15 s and a percentage overshoot of 2.5 %, the natural frequency and damping ratio needed are $\omega_n = 32.3$ rad/s and $\zeta = 0.76$. Using the QUBE-Servo model parameters K and τ given above in Background section of this lab (or those you found previously through a modeling lab), calculate the control gains needed to satisfy these requirements.

8. Run the PV controller with the newly designed gains on the QUBE-Servo. Attach the position response as well as the motor voltage used.

9. Measure the percent overshoot and peak time of the response. Do they match the desired percent overshoot and peak time specifications given in Step 7 without saturating the motor (going beyond ± 10 V)?

   Hint: Use the Matlab `ginput` command to measure points off the plot and the equations from the Second-Order Systems laboratory experiment

10. If your response did not match the above overshoot and peak time specification, try tuning your control gains until your response does satisfy them. Attach the resulting Matlab® figure, resulting measurements, and comment on how you modified your controller to arrive at those results.

11. Stop the QUARC® controller.

12. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Pendulum Moment of Inertia

Topics Covered

- Finding moment of inertia analytically and experimentally.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- Rotary pendulum module is attached to the QUBE-Servo.

1 Background

The free-body diagram of the QUBE-Servo pendulum system is shown in Figure 1.1.

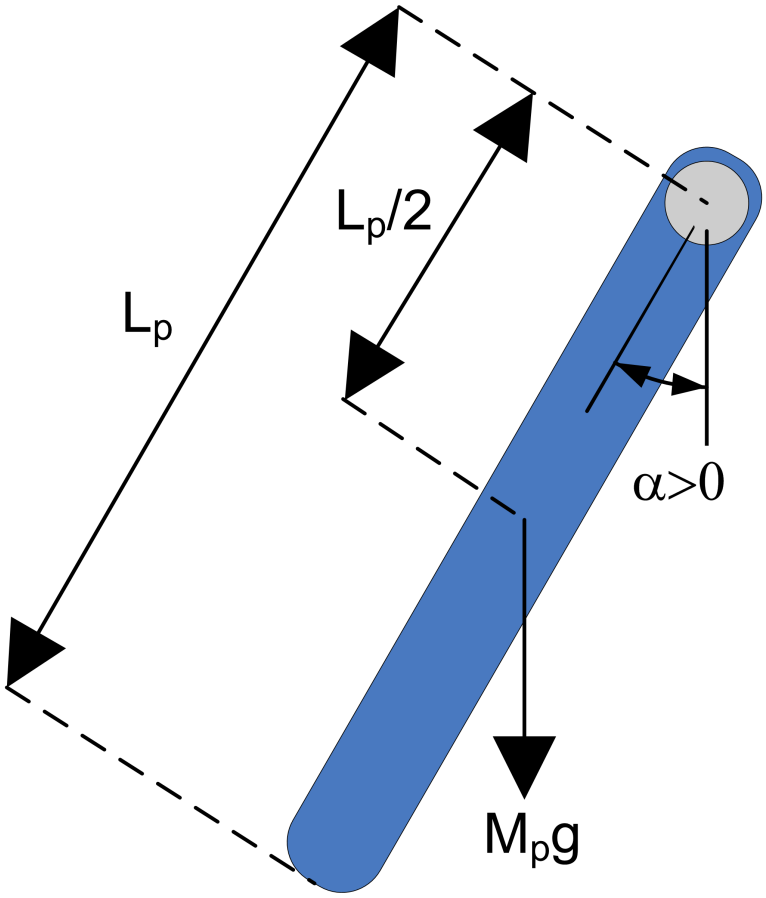


Figure 1.1: Free-body diagram of pendulum

From the free-body diagram in Figure 1.1, the resulting nonlinear equation of motion of the pendulum is

$$J_p \ddot{\alpha}(t) = M_p g \frac{L_p}{2} \sin(\alpha(t)), \tag{1.1}$$

where J_p is the moment of inertia of the pendulum at the pivot axis, M_p is the total mass of the pendulum assembly, and L_p is the length of the pendulum (from pivot to end). The center of mass position is at $L_p/2$, as depicted in Figure 1.1.

The moment of inertia of the pendulum can be found experimentally. Assuming the pendulum is not actuated, linearizing Equation 1.1 and solving for the differential equation yields

$$J_p = \frac{M_p g l_p}{(2\pi f)^2}, \tag{1.2}$$

where f is the measured frequency of the pendulum as the arm remains rigid. The frequency is calculated using

$$f = \frac{n_{cyc}}{\Delta t} \tag{1.3}$$

where n_{cyc} is the number of cycles and Δt is the duration of these cycles. Alternatively, J_p can be calculated using the moment of inertia expression

$$J = \int r^2 dm, \tag{1.4}$$

where r is the perpendicular distance between the element mass dm and the axis of rotation.

2 In-Lab Exercises

Based on the model already designed in QUBE-Servo Integration laboratory experiment, design a model that measures the pendulum angle using the encoder as shown in Figure 2.1.

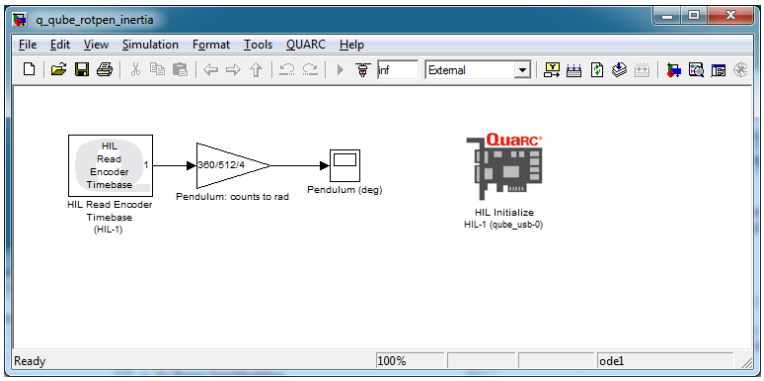


Figure 2.1: Displays measured pendulum angle

1. Find the moment of inertia acting about the pendulum pivot using the free-body diagram. Make sure you evaluate it numerically using the parameters defined in the QUBE-Servo User Manual.

   Hint: For solid objects with a uniform density, you can express the differential mass in terms of differential length.

2. Build the Simulink® diagram shown in Figure 2.1. Enter $360/(512 * 4)$ in the encoder sensor gain to measure the pendulum angle in degrees.

3. Build and run the QUARC® controller. With the controller running, manually perturb the pendulum while holding the rotary arm in place. The scope response should be similar to Figure 2.2.

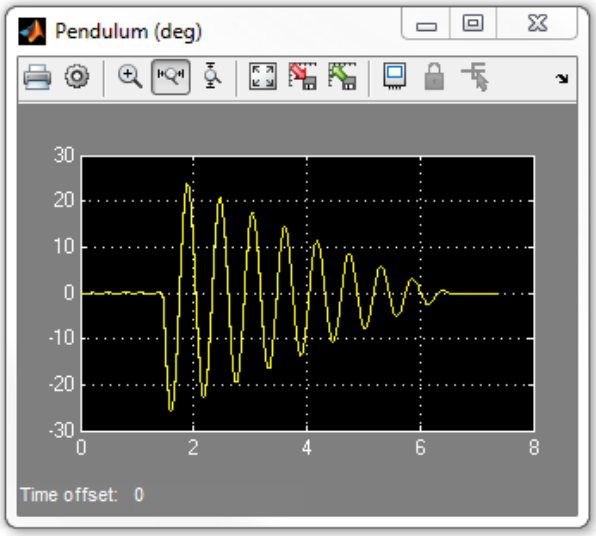


Figure 2.2: Free-oscillation response of pendulum

4. Find the frequency and moment of inertia of the pendulum using the observed results.

5. Compare the moment of inertia calculated analytically in Exercise 1 and the moment of inertia found experimentally. Is there a large discrepancy between them?

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Rotary Pendulum Modeling

Topics Covered

- Interact with the Quanser® QUBE-Servo Rotary Pendulum system.
- Configure sensor and actuator gains to match model conventions.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- Rotary pendulum module is attached to the QUBE-Servo.

1 Background

The rotary pendulum system, also known as the Furuta Pendulum, is a classic system often used to teach modeling and control in physics and engineering. The free-body diagram of a basic rotary pendulum is depicted in Figure 1.1.

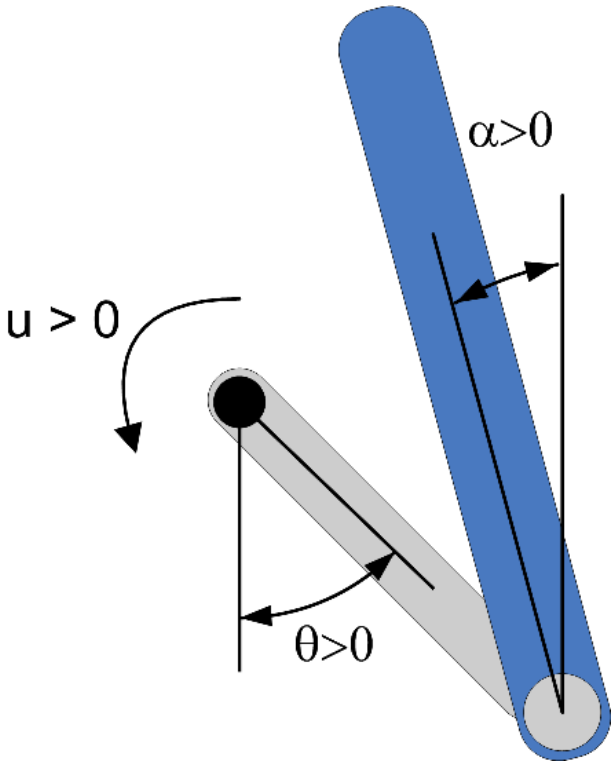


Figure 1.1: Free-body diagram of rotary pendulum

The rotary arm, which is attached to the motor pivot, is denoted by the variable θ and the pendulum angle, attached to the end of the rotary arm, is denoted by α. Note the following conventions:

- Angle α is defined as the *inverted pendulum angle*, i.e. the angle with respect to the upright vertical position where $\alpha = 0$ means the pendulum is perfectly upright. It is expressed mathematically using

$$\alpha = \alpha_{full} \text{ mod} 2\pi - \pi. \tag{1.1}$$

  where α_{full} is the pendulum angle measured by the encoder, i.e. the continuous angle measurement defined as zero when pendulum is in the downward configuration.

- Both angles are defined as positive when rotated in the counter-clockwise (CCW) direction.
- When a positive voltage is applied to the motor, the rotary arm moves in the positive CCW direction.

The goal is to design a system model that follows these conventions. The QUBE-Servo Integration laboratory experiment introduced the DC motor and encoders on the QUBE-Servo system. The pendulum angle is also measured using an encoder.

2 In-Lab Exercises

In this lab, we will make a Simulink® model using QUARC® blocks to drive to the dc motor and measure both the rotary arm and pendulum angles - similarly as shown in Figure 2.1.

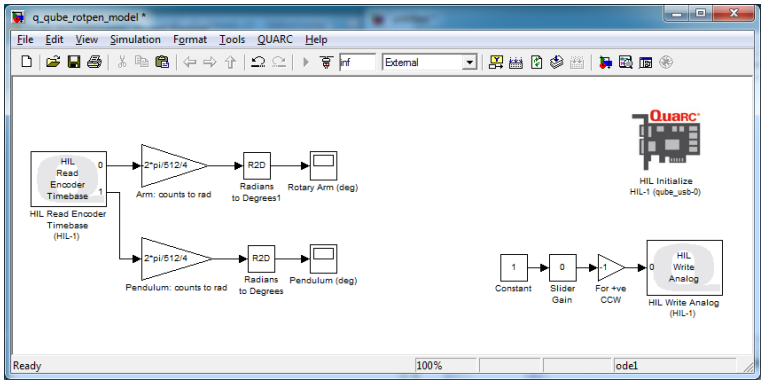
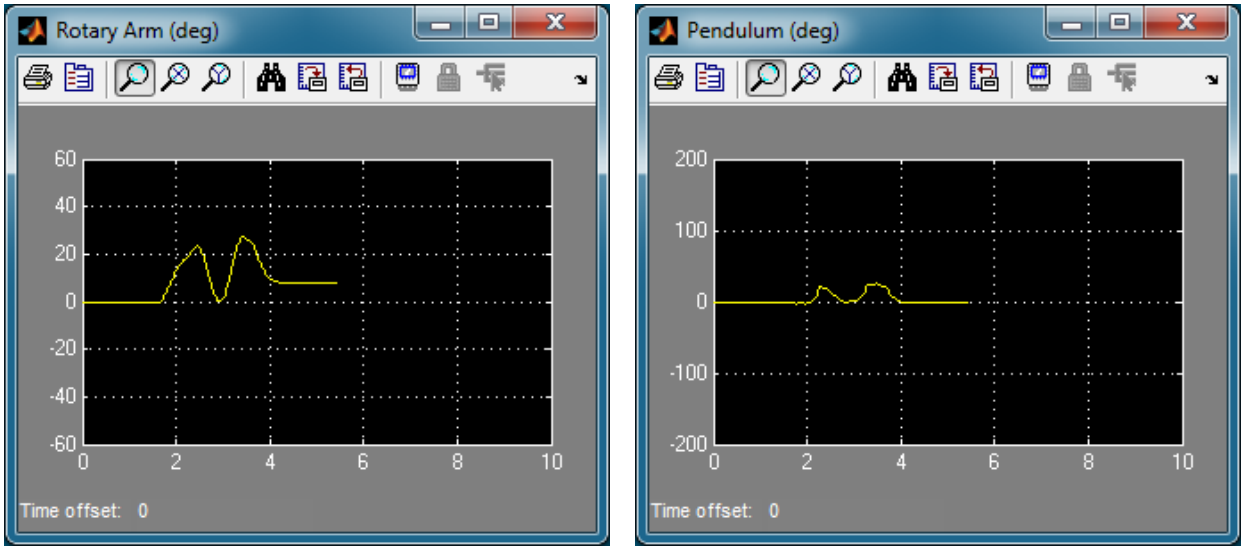


Figure 2.1: Simulink® model used with QUARC® to drive motor and read angles on QUBE-Servo Rotary Pendulum system

1. Using the Simulink® model you made in the QUBE-Servo Integration laboratory experiment, do the following:
   - Configure the HIL Read Encoder Timebase block to read from both channels #0 and #1. The pendulum is measured on channel #1.
   - Setup the encoder gains on each channel to read the angles in radians (instead of degrees as in the lab).
   - Connect the the measured angles to scopes, but display them in degrees. You can do this using the *Simulink Extras | Transformation* blocks or just by adding another Gain block that converts radians to degrees.
   - If desired, add a Slider Gain block between the Constant and Analog Output block. Make sure the Constant source block is set to 1 in this case.
2. Build and run the QUARC® controller.
3. Rotate the rotary arm and pendulum counter-clockwise and examine the response on the scopes. Example responses are shown in Figure 2.2. Do the measured angles follow the modeling conventions given in Section 1



(a) Rotary Arm (b) Pendulum

Figure 2.2: Measured rotary arm and pendulum angles

4. Apply a small voltage (e.g. 0.5 V) to the motor by changing the Constant block connected to Analog Output. Does this adhere to the modeling conventions?
5. Modify the Simulink model such that the measured angles and applied voltage follow by the modeling conventions. Briefly list any changes made.
6. Add modulus and bias blocks, as shown in Figure 2.3, to measure *inverted pendulum angle*, defined as Equation 1.1.

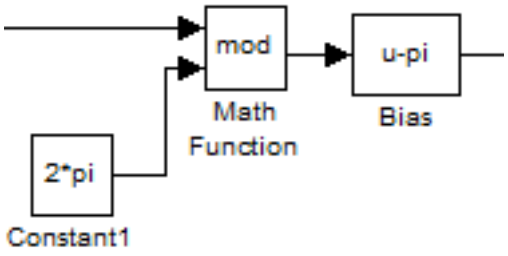


Figure 2.3: Simulink® modulus and bias blocks

7. Make sure the pendulum is hanging in the downward position and lies motionless before starting the controller.
8. Build and run the QUARC® controller.
9. Rotate the pendulum to the upright vertical position and ensure the angle is measured correctly and it follows the free-body diagram in Figure 1.1. Capture the response of the pendulum being raise to the inverted position. Explain what the bias and modulus functions do?
10. Stop the QUARC® controller.
11. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Balance Control

Topics Covered

- Control enabling logic.
- PID-based balance control.

Prerequisites

- Filtering laboratory experiment.
- PD Control laboratory experiment.
- Rotary Pendulum Modeling laboratory experiment.
- Rotary pendulum module is attached to the QUBE-Servo.

1 Background

Balancing is a common control task. In this experiment we will find control strategies that balance the pendulum in the upright position while maintaining a desired position of the arm. When balancing the system, the pendulum angle α is small and balancing can be accomplished with a simple PD controller, as shown in Figure 1.1. If we are further interested in keeping the arm in a desired position, a feedback loop from the arm position will also be introduced. The control law can then be expressed as

$$u = k_{p,\theta}(\theta_r - \theta) - k_{p,\alpha}\alpha - k_{d,\theta}\dot{\theta} - k_{d,\alpha}\dot{\alpha} \tag{1.1}$$

where $k_{p,\theta}$ is the arm angle proportional gain, $k_{p,\alpha}$ is the pendulum angle proportional gain, $k_{d,\theta}$ is the arm angle derivative gain, and $k_{d,\alpha}$ is the pendulum angle derivative gain. The desired angle of the arm is denoted by θ_r and the reference for the pendulum angle is zero (i.e. upright position).

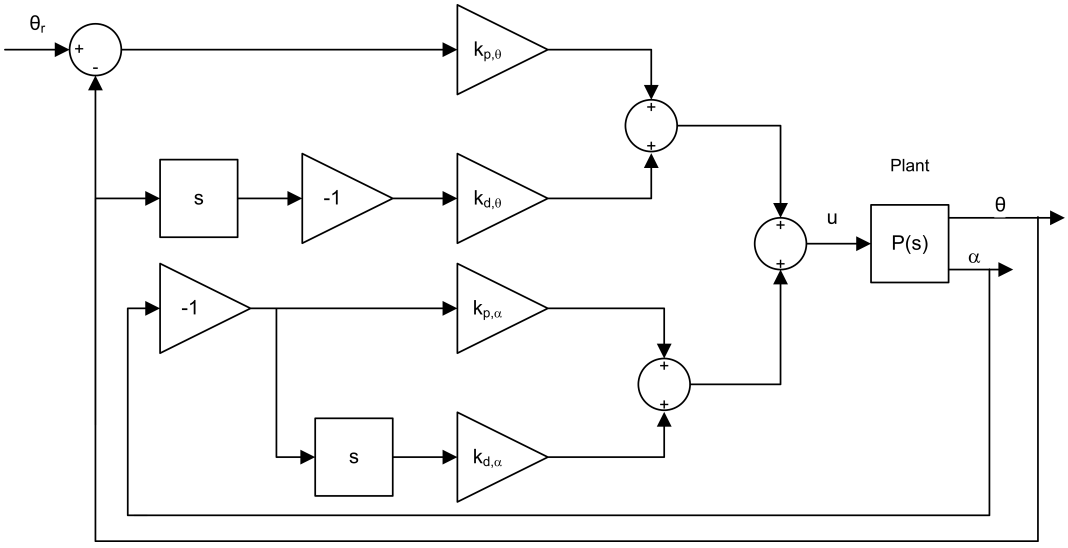


Figure 1.1: Block diagram of balance PD control for rotary pendulum

There are many different ways to find the controller parameters. One method is discussed in the Optimal LQR Control lab. Initially, however, the behavior of the system will be explored using default parameters.

Recall that the pendulum angle α is defined as zero when the pendulum is about its upright vertical position and expressed mathematically using $\alpha = \alpha_{full} \bmod 2\pi - \pi$, as defined in the Rotary Pendulum Modeling Lab as

$$\alpha = \alpha_{full} \bmod 2\pi - \pi.$$

The balance control is to be enabled when the pendulum is within the following range:

$$|\alpha| \leq 10^{\circ}. \tag{1.2}$$

Given that the pendulum starts in the downward vertical position, it needs to be manually brought up to its upright vertical position. Once the pendulum is within $\pm 10^{\circ}$, the balance controller is engaged. It remains in balance mode until the pendulum goes beyond $\pm 10^{\circ}$.

If desired, you can integrate this with an algorithm that swings-up the pendulum automatically. See the Swing-Up Control laboratory experiment for details.

2 In-Lab Exercises

Construct a QUARC® controller similarly as shown in Figure 2.1 that balances the pendulum on the QUBE-Servo rotary pendulum using the PD control detailed in Section 1.

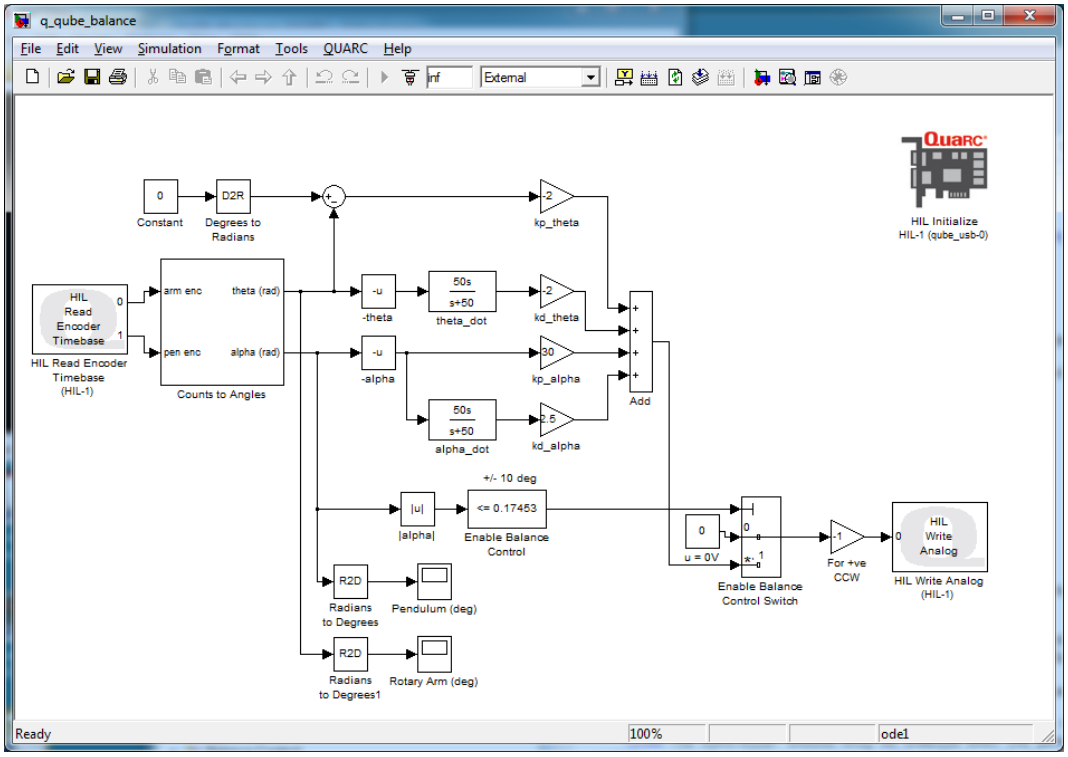
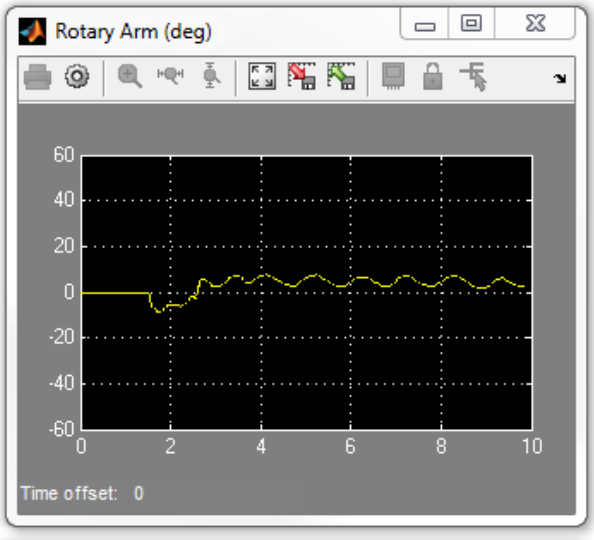
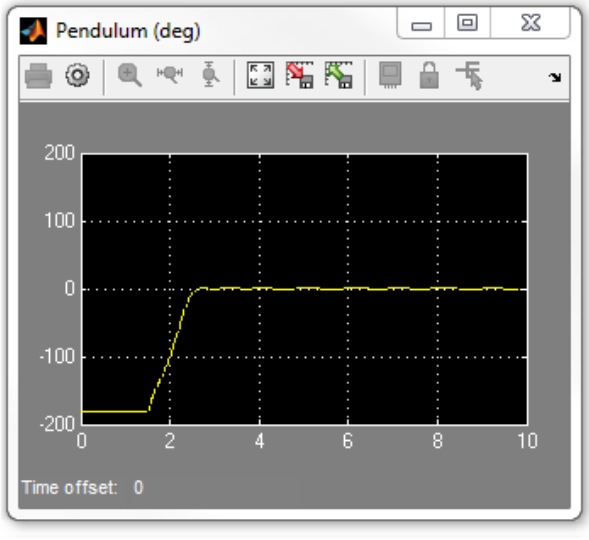


Figure 2.1: Simulink® model used with QUARC® run balance controller

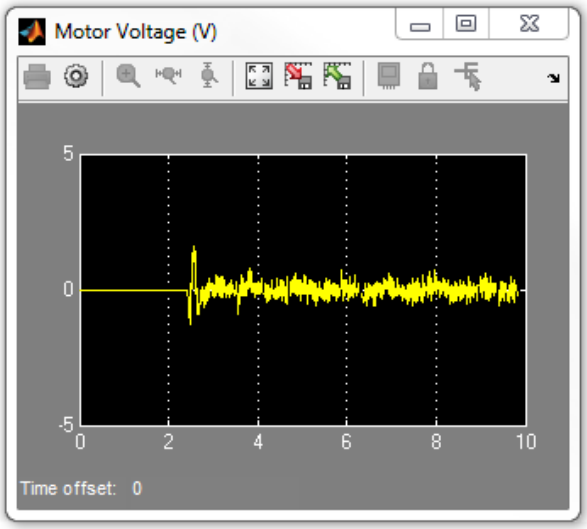
1. Using the Simulink® model you made in the *Rotary Pendulum Model* lab, construct the controller shown in Figure 2.1:
   - The Counts to Angles subsystem contains the same blocks used in the Rotary Pendulum Modeling laboratory experiment to convert encoder counts to radians. Make sure you use the inverted pendulum angle.
   - To find the velocity of the rotary arm and pendulum, add the high-pass filters $50s/(s+50)$ similarly as in the Filtering laboratory experiment.
   - Add the necesary Sum and Gain blocks to implement the PD control given in Equation 1.1.
   - The controller should only be enabled when the pendulum is $\pm 10^\circ$ about the upright vertical position (or ± 0.175 rad). Add the absolute value, constant comparison, and selector blocks to implement this.
2. Set the PD gains as follows: $k_{p,\theta} = -2$, $k_{p,\alpha} = 30$, $k_{d,\theta} = -2$, and $k_{d,\alpha} = 2.5$.
3. Build and run the QUARC® controller.
4. Manually rotate the pendulum in the upright position until the controller engages. The scopes should read something similar as shown in Figure 2.2. Attach response of the rotary arm, pendulum, and controller voltage.



(a) Rotary Arm



(b) Pendulum



(c) Motor Voltage

Figure 2.2: QUBE-Servo rotary pendulum response

5. As the pendulum is being balanced, describe the responses in the *Rotary Arm (deg)* and the *Pendulum Angle (deg)* scopes.
6. Vary the Constant block that is connected to the positive input of the summer block in the PD control. Observe the response in the *Arm Angle (deg)* scope. **Do not set the value too high, keep it within $\pm 45^\circ$ to start.** What variable does this represent in the balance control?
7. Stop the QUARC® controller.
8. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

State Space Modeling

Topics Covered

- Inverted pendulum modeling.
- Introduction to state-space models.
- Model validation.

Prerequisites

- QUBE-Servo Integration laboratory experiment.
- First Principles Modeling laboratory experiment.
- Rotary Pendulum Modeling laboratory experiment.
- Rotary pendulum module is attached to the QUBE-Servo.

1 Background

1.1 Pendulum Model

The rotary pendulum model is shown in Figure 1.1. The rotary arm pivot is attached to the QUBE-Servo system and is actuated. The arm has a length of L_r, a moment of inertia of J_r, and its angle θ increases positively when it rotates counter-clockwise (CCW). The servo (and thus the arm) should turn in the CCW direction when the control voltage is positive ($V_m > 0$).

The pendulum link is connected to the end of the rotary arm. It has a total length of L_p and it center of mass is at $L_p/2$. The moment of inertia about its center of mass is J_p. The inverted pendulum angle α is zero when it is hanging downward and increases positively when rotated CCW.

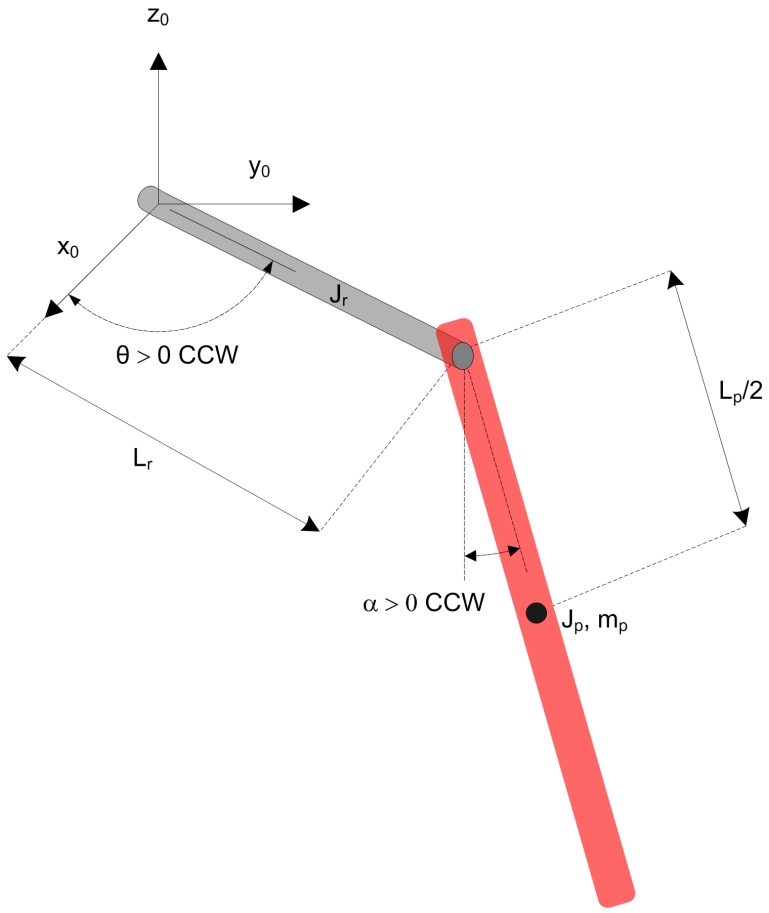


Figure 1.1: Rotary inverted pendulum model

The equations of motion (EOM) for the pendulum system were developed using the Euler-Lagrange method. This systematic method is often used to model complicated systems such as robot manipulators with multiple joints. The total kinetic and potential energy of the system is obtained, then the Lagrangian can be found. A number of derivatives are then computed to yield the EOMs. The complete derivation of the EOM for the pendulum system are presented in the *Rotary Pendulum Modeling Summary* and Maple workbook.

The resultant nonlinear EOM are:

$$\left(m_pL_r^2+\frac{1}{4}m_pL_p^2-\frac{1}{4}m_pL_p^2\cos(\alpha)^2+J_r\right)\ddot{\theta}-\left(\frac{1}{2}m_pL_pL_r\cos(\alpha)\right)\ddot{\alpha}$$
$$+\left(\frac{1}{2}m_pL_p^2\sin(\alpha)\cos(\alpha)\right)\dot{\theta}\dot{\alpha}+\left(\frac{1}{2}m_pL_pL_r\sin(\alpha)\right)\dot{\alpha}^2=\tau-D_r\dot{\theta} \tag{1.1}$$

$$\frac{1}{2}m_pL_pL_r\cos(\alpha)\ddot{\theta}+\left(J_p+\frac{1}{4}m_pL_p^2\right)\ddot{\alpha}-\frac{1}{4}m_pL_p^2\cos(\alpha)\sin(\alpha)\dot{\theta}^2+\frac{1}{2}m_pL_pg\sin(\alpha)=-D_p\dot{\alpha}. \tag{1.2}$$

with an applied torque at the base of the rotary arm generated by the servo motor as described by the equation:

$$\tau=\frac{k_m\left(V_m-k_m\dot{\theta}\right)}{R_m} \tag{1.3}$$

When the nonlinear EOM are linearized about the operating point, the resultant linear EOM for the inverted pendulum are defined as:

$$\left(m_pL_r^2+J_r\right)\ddot{\theta}-\frac{1}{2}m_pL_pL_r\ddot{\alpha}=\tau-D_r\dot{\theta}. \tag{1.4}$$

and

$$\frac{1}{2}m_pL_pL_r\ddot{\theta}+\left(J_p+\frac{1}{4}m_pL_p^2\right)\ddot{\alpha}+\frac{1}{2}m_pL_pg\alpha=-D_p\dot{\alpha}. \tag{1.5}$$

Solving for the acceleration terms yields:

$$\ddot{\theta}=\frac{1}{J_T}\left(-\left(J_p+\frac{1}{4}m_pL_p^2\right)D_r\dot{\theta}+\frac{1}{2}m_pL_pL_rD_p\dot{\alpha}+\frac{1}{4}m_p^2L_p^2L_rg\alpha+\left(J_p+\frac{1}{4}m_pL_p^2\right)\tau\right). \tag{1.6}$$

and

$$\ddot{\alpha}=\frac{1}{J_T}\left(\frac{1}{2}m_pL_pL_rD_r\dot{\theta}-\left(J_r+m_pL_r^2\right)D_p\dot{\alpha}-\frac{1}{2}m_pL_pg\left(J_r+m_pL_r^2\right)\alpha-\frac{1}{2}m_pL_pL_r\tau\right). \tag{1.7}$$

where

$$J_T=J_pm_pL_r^2+J_rJ_p+\frac{1}{4}J_rm_pL_p^2. \tag{1.8}$$

1.1.1 Linear State-Space Model

The linear state-space equations are

$$\dot{x}=Ax+Bu \tag{1.9}$$

and

$$y=Cx+Du \tag{1.10}$$

where x is the state, u is the control input, A, B, C and D are state-space matrices. For the rotary pendulum system, the state and output are defined

$$x=\begin{bmatrix}\theta & \alpha & \dot{\theta} & \dot{\alpha}\end{bmatrix}^T \tag{1.11}$$

and

$$y=\begin{bmatrix}x_1 & x_2\end{bmatrix}^T. \tag{1.12}$$

2 In-Lab Exercises

2.1 Pendulum State-Space Model

1. Based on the sensors available on the pendulum system, find the C and D matricies in Equation 1.10.
2. Using Equation 1.6 and Equation 1.7 and the defined state in Equation 1.11, derive the linear state-space model of the pendulum system.
3. Based on the state space model derived in Step 2 and the Matlab® script `rotpen_ABCD_eqns.m` provided, create the appropriate matricies that correspond to the linear state space model of the pendulum.
4. Based on the Simulink® model already designed in the Rotary Pendulum Modeling laboratory experiment, design a similar model to the one shown in Figure 2.1 that applies a $0-1$ V, 1 Hz square wave to the pendulum system and state-space model.

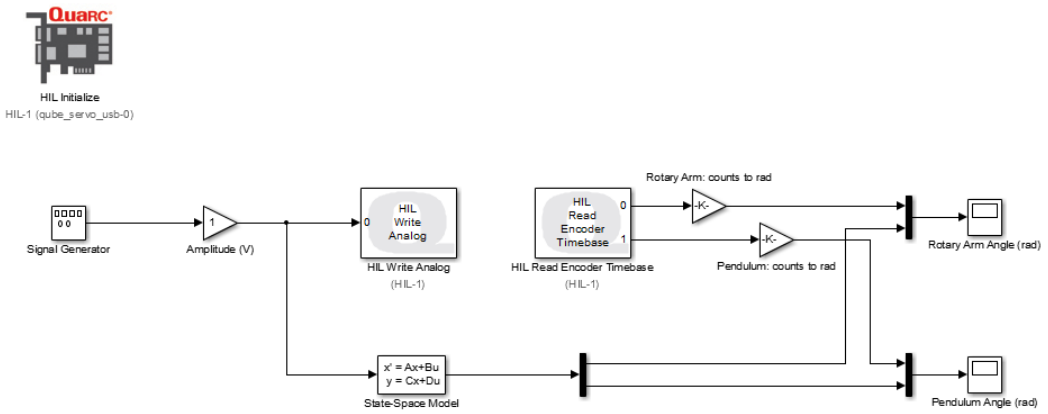
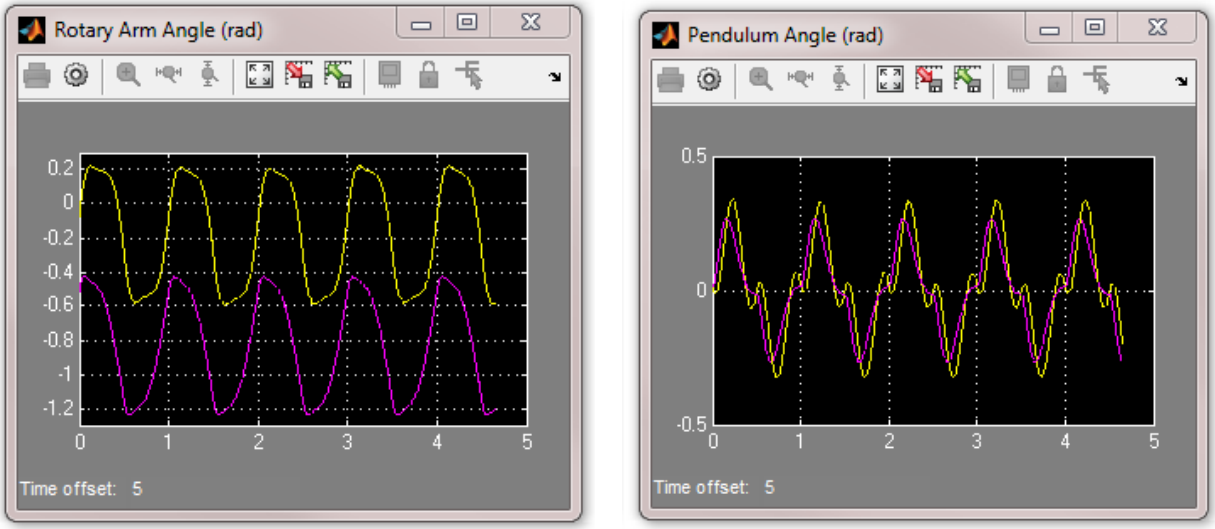


Figure 2.1: Applies a step voltage and displays measured and simulated pendulum response.

5. Run `setup_ss_model.m` to create the state space model parameters in the Matlab® workspace. Ensure that the generated matricies match your solution in Step 2.
6. In `setup_ss_model.m`, set the rotary arm viscous damping coefficient D_r to 0.0015 N.m.s/rad, and the pendulum damping coefficient D_p to 0.0005 N.m.s/rad. These parameters were found experimentally to reasonably accuractly reflect the viscous damping of the system due to effects such as friction, when subject to a step response.
7. Build and run the model. The scope response should be similar to Figure 2.2. Attach a screen capture of your scopes. Does your model represent the actual pendulum well? If not, explain why there might be discrepancies.



(a) Rotary Arm Angle (rad)

(b) Pendulum Angle (rad)

Figure 2.2: Step response of the pendulum system

8. The viscous damping of each inverted pendulum can vary slightly from system to system. If your model does not accuratly represent your specific pendulum system, try modifying the damping coefficients D_r and D_p to obtain a more accurate model.
9. Stop the QUARC® controller.
10. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Swing-Up Control

Topics Covered

- Energy control.
- Nonlinear control.
- Control switching logic.

Prerequisites

- Filtering laboratory experiment.
- Balance Control laboratory experiment.
- Rotary pendulum module is attached to the QUBE-Servo.

1 Background

1.1 Energy Control

In theory, if the arm angle is kept constant and the pendulum is given an initial perturbation, the pendulum will keep on swinging with constant amplitude. The idea of energy control is based on the preservation of energy in ideal systems: The sum of kinetic and potential energy is constant. However, friction will be damping the oscillation in practice and the overall system energy will not be constant. It is possible to capture the loss of energy with respect to the pivot acceleration, which in turn can be used to find a controller to swing up the pendulum.

The dynamics of the pendulum can be redefined in terms of the pivot acceleration u as

$$J_p\ddot{\alpha} + \frac{1}{2}M_p g L_p \sin\alpha = \frac{1}{2}M_p g u \cos\alpha. \tag{1.1}$$

Here, u is the linear acceleration of the pendulum.

The potential energy of the pendulum is

$$E_p = \frac{1}{2}M_p g L_p \left(1 - \cos\alpha\right),$$

and the kinetic energy is

$$E_k = \frac{1}{2}J_p\dot{\alpha}^2.$$

The pendulum angle α and the lengths of the pendulum are illustrated in the free body diagram in Figure 1.1.

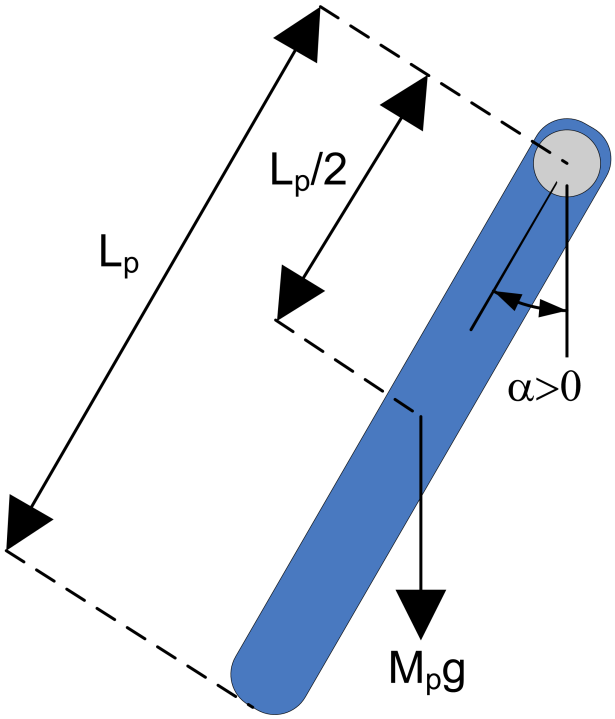


Figure 1.1: Free-body diagram of pendulum

The potential energy is zero when the pendulum is at rest at $\alpha = 0$ and equals $M_p g L_p$ when the pendulum is upright at $\alpha = \pm\pi$. The sum of the potential and kinetic energy of the pendulum is

$$E = \frac{1}{2}J_p\dot{\alpha}^2 + \frac{1}{2}M_p g L_p \left(1 - \cos\alpha\right). \tag{1.2}$$

Differentiating Equation Equation 1.2 yields

$$\dot{E} = \dot{\alpha}\left(J_p\ddot{\alpha} + \frac{1}{2}M_p g L_p \sin\alpha\right). \tag{1.3}$$

Recalling Equation (Equation 1.1) and rearranging terms as

$$J_p\ddot{\alpha} = -M_p g l_p \sin\alpha + M_p u l_p \cos\alpha$$

eventually yields

$$\dot{E} = M_p u l_p \dot{\alpha}\cos\alpha.$$

Since the acceleration of the pivot is proportional to current driving the arm motor and thus also proportional to the drive voltage, it is possible to control the energy of the pendulum with the proportional control law

$$u = (E_r - E)\dot{\alpha}\cos\alpha. \tag{1.4}$$

By setting the reference energy to the pendulum potential energy ($E_r = E_p$), the control law will swing the link to its upright position. Notice that the control law is nonlinear because the proportional gain depends on the cosine of the pendulum angle α. Further, the control changes sign when $\dot{\alpha}$ changes sign and when the angle is ± 90 degrees.

For the system energy to change quickly, the magnitude of the control signal must be large. As a result the following swing-up controller is implemented in the controller as

$$u = \text{sat}_{u_{max}}\left(\mu(E_r - E)\text{sign}(\dot{\alpha}\cos\alpha)\right) \tag{1.5}$$

where μ is a tunable control gain and the $\text{sat}_{u_{max}}$ function saturates the control signal at the maximum acceleration of the pendulum pivot, u_{max}. The expression $\text{sign}(\dot{\alpha}\cos\alpha)$ is used to enable faster control switching.

1.2 Hybrid Swing-Up Control

The energy swing-up control in Equation 1.4 (or Equation 1.5) can be combined with the balancing control law from the Balance Control Lab to obtain a control law that swings up the pendulum and then balances it.

Similarly as described in the Balance Control laboratory experiment, the balance control is to be enabled when the pendulum is within ± 20 degrees. When it is not enabled, the swing-up control is engaged. Thus the switching can be described mathematically by:

$$u = \begin{cases} u_{bal} & \text{if } |\alpha| - \pi \leq 20^\circ \\ u_{swing\_up} & \text{otherwise} \end{cases} \tag{1.6}$$

2 In-Lab Exercises

The controller in QUARC® is shown in Figure 2.1 that swings-up and balances the pendulum on the QUBE-Servo rotary pendulum system. The *Swing-Up Control* subsystem implements the energy control described in Section 1.

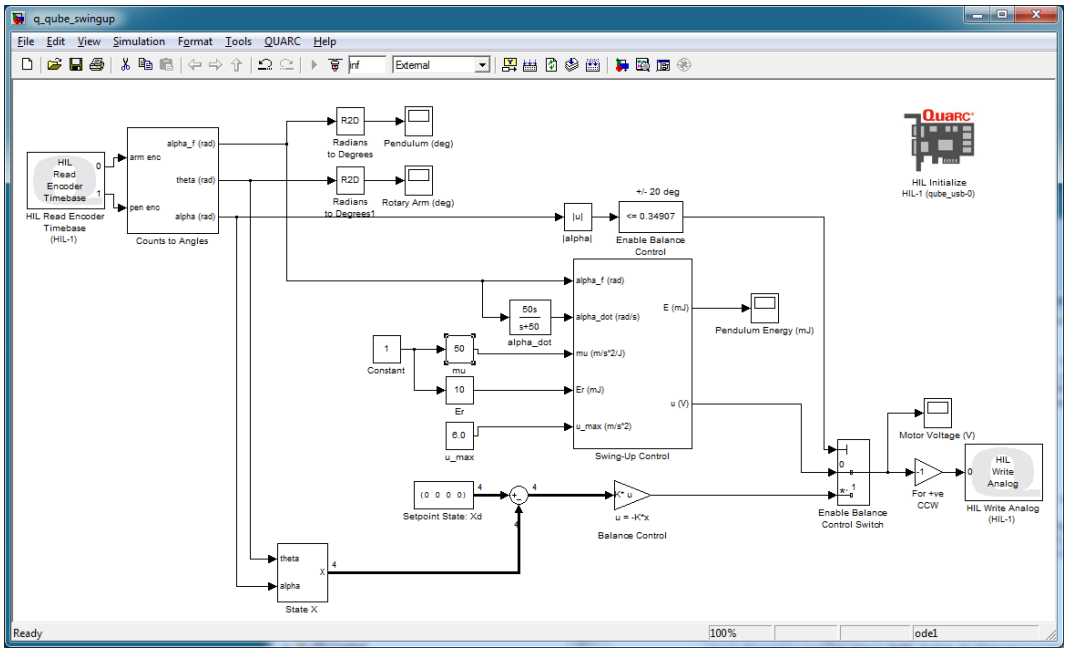


Figure 2.1: Simulink® model used with QUARC® to run swing-up controller

2.1 Energy Control

1. Open the `q_qube_swingup.mdl` Simulink® model.
2. Run the `setup_qube_rotpen.m` Matlab® script. This loads the pendulum parameters that is used by the Simulink® model.
3. To turn the swing-up control off, set the Slider Gain block called *mu* to 0.
4. Build and run the QUARC® controller.
5. Manually rotate the pendulum at different levels and examine the pendulum angle and energy in the *Pendulum (* deg*)* and *Pendulum Energy (* mJ*)* scopes.
6. What do you notice about the energy when the pendulum is moved at different positions? Record the energy when the pendulum is being balanced (i.e. fully inverted in the upright vertical position). Does this reading make sense in terms of the equations developed in Section 1?
7. Click on the Stop button to bring the pendulum down to the initial, downward position.
8. Set the swing-up control parameters (i.e. the Constant and Gain blocks connected to the inputs of the Swing-Up Control subsystem) to the following:
   - mu = 50 m/s^2/J
   - Er = 10.0 mJ
   - u_max = 6 m/s^2
9. If the pendulum is not moving, gently perturb the pendulum with your hand to get it going.

10. Vary the reference energy, *Er*, between 10.0 mJ and 20.0 mJ. As it is changed, examine the pendulum angle and energy response in *Pendulum (deg)* and the *Pendulum Energy (mJ)* scopes and the control signal in the *Motor Voltage (V)* scope. Attach the responses showing how changing the reference energy affects the system.
11. Fix *Er* to 20.0 mJ and vary the swing-up control gain *mu* between 20 and 60 $\mathrm{m/s^2/J}$. Describe how this changes the performance of the energy control.
12. Stop the QUARC® controller.

2.2 Hybrid Swing-Up Control

1. Open the `q_qube_swing_up.mdl` Simulink® model.
2. Run the `setup_qube_rotpen.m` Matlab® script. This loads the pendulum parameters that is used by the Simulink® model.
3. Set the swing-up control parameters to the following:
   - mu = 20 $\mathrm{m/s^2/J}$
   - u_max = 6 $\mathrm{m/s^2}$
4. Based on your observations in the previous lab, 2.1, what should the reference energy be set to?
5. Make sure the pendulum is hanging down motionless and the encoder cable is not interfering with the pendulum.
6. Build and run the QUARC® controller.
7. The pendulum should begin going back and forth. If not, manually perturb the pendulum with your hand. **Click on the Stop button in the Simulink® tool bar if the pendulum goes unstable.**
8. Gradually increase the swing-up gain, μ, denoted as the *mu* Slider Gain block, until the pendulum swings up to the vertical position. Capture a response of the swing-up and record the swing-up gain that was required. Show the pendulum angle, pendulum energy, and motor voltage.
9. Stop the QUARC® controller.
10. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

Optimal LQR Control

Topics Covered

- Introduction to state-space models.
- State-feedback control.
- Linear Quadratic Regulator (LQR) optimization.

Prerequisites

- Filtering laboratory experiment.
- Balance Control laboratory experiment.
- Rotary pendulum module is attached to the QUBE-Servo.

1 Background

Linear Quadratic Regulator (LQR) theory is a technique that is ideally suited for finding the parameters of the pendulum balance controller in the Balance Control laboratory experiment. Given that the equations of motion of the system can be described in the form

$$\dot{x} = Ax + Bu,$$

where A and B are the state and input system matrices, respectively, the LQR algorithm computes a control law u such that the performance criterion or cost function

$$J = \int_0^\infty (x_{ref} - x(t))^T Q (x_{ref} - x(t)) + u(t)^T R u(t) dt \tag{1.1}$$

is minimized. The design matrices Q and R hold the penalties on the deviations of state variables from their setpoint and the control actions, respectively. When an element of Q is increased, therefore, the cost function increases the penalty associated with any deviations from the desired setpoint of that state variable, and thus the specific control gain will be larger. When the values of the R matrix are increased, a larger penalty is applied to the aggressiveness of the control action, and the control gains are uniformly decreased.

In our case the state vector x is defined

$$x = \begin{bmatrix} \theta & \alpha & \dot{\theta} & \dot{\alpha} \end{bmatrix}^T. \tag{1.2}$$

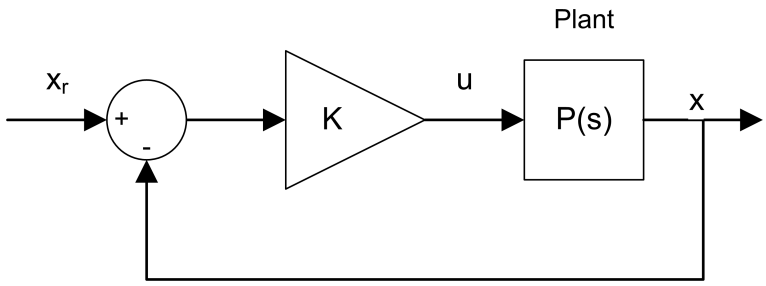


Figure 1.1: Block diagram of balance state-feedback control for rotary pendulum

Since there is only one control variable, R is a scalar. The reference signal x_{ref} is set to $\begin{bmatrix} \theta_r & 0 & 0 & 0 \end{bmatrix}^T$, and the control strategy used to minimize cost function J is thus given by

$$u = K(x_{ref} - x) = k_{p,\theta}(\theta_r - \theta) - k_{p,\alpha}\alpha - k_{d,\theta}\dot{\theta} - k_{d,\alpha}\dot{\alpha}. \tag{1.3}$$

This control law is a state-feedback control and is illustrated in Figure 1.1. It is equivalent to the PD control explained in the Balance Control laboratory experiment.

2 In-Lab Exercises

Construct a QUARC® controller similarly to Figure 2.1 that balances the pendulum on the QUBE-Servo rotary pendulum system using a generated control gain K.

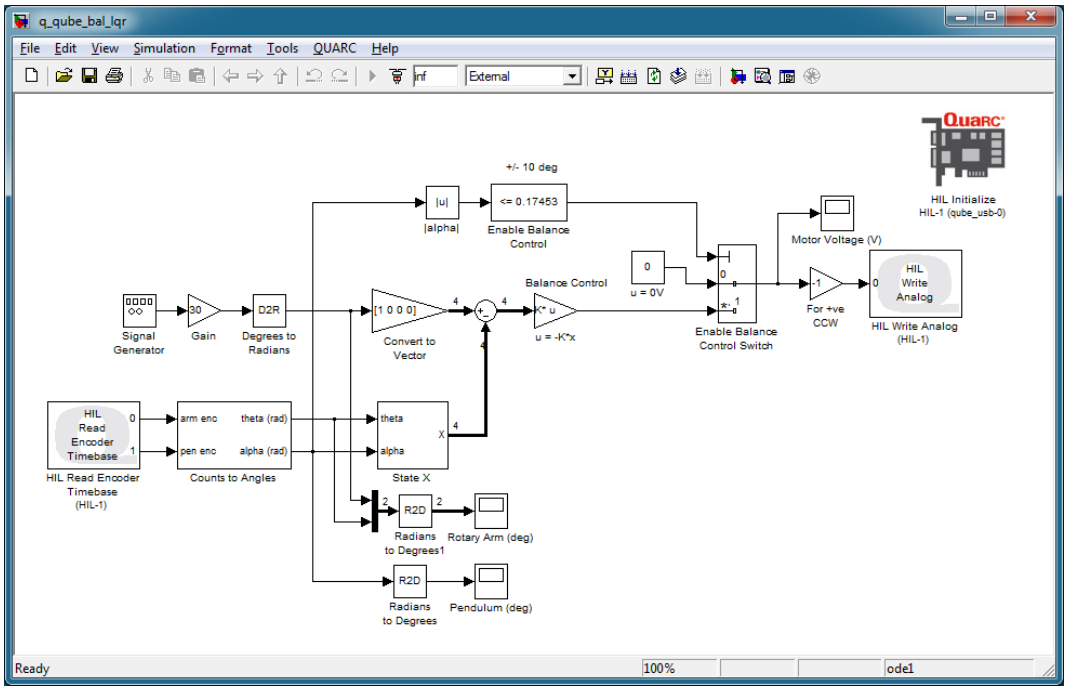


Figure 2.1: Simulink® model used with QUARC® run optimized balance controller

The LQR theory has been packaged in the Matlab® *Control Design Module*. Given the model of the system,in the form of the state-space matrices A and B, and the weighting matrices Q and R, the LQR function computes the feedback control gain automatically.

In this experiment, the state-space model is already available. In the laboratory, the effect of changing the Q weighting matrix while R is fixed to 1 on the cost function J will be explored.

2.1 LQR Control Design

1. In Matlab®, run the `setup_qube_rotpen.m` script. This loads the QUBE-Servo rotary pendulum state-space model matrices A, B, C, and D. The A and B matrices should be displayed in the Command Window:

```
A =

         0         0    1.0000         0
         0         0         0    1.0000
         0  149.2751   -0.0104         0
         0  261.6091   -0.0103         0


B =

         0
         0
   49.7275
   49.1493
```

2. Use the `eig` command to find the open-loop poles of the system. What do you notice about the location of the open-loop poles? How does that affect the system?

3. Using the `lqr` function with the loaded model and the weighting matrices

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad R = 1,$$

generate gain K. Give the value of the control gain generated.

4. Change the LQR weighting matrix to the following and generate a new gain control gain:

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad R = 1.$$

Record the gain generated. How does changing q_{11} affect the generated control gain? Based on the description of LQR in 2.1, is this what you expected?

2.2 LQR-Based Balance Control

1. Run the `setup_qube_rotpen.m` script in Matlab.
2. Using the Simulink® model you made in the Balance Control laboratory experiment, construct the controller shown in Figure 2.1:
   - Using the angles from the Counts to Angles subsystem you designed in the Balance Control laboratory experiment (which converts encoder counts to radians), build state x given in Equation 1.2. In Figure 2.1, it is bundled in the subsystem called *State X*. Use high-pass filters $50s/(s+50)$ to compute the velocities $\dot{\theta}$ and $\dot{\alpha}$.
   - Add the necessary Sum and Gain blocks to implement the state-feedback control given in Equation 1.3. Since the control gain is a vector, make sure the gain block is configured to do matrix type multiplication.
   - Add the Signal Generator block in order to generate a varying, desired arm angle. To generate a reference state, make sure you include a Gain block of $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$.
3. Load the gain designed in Step 3. Make sure it is set as variable `K` in the Matlab® workspace.
4. Set the Signal Generator block to the following:
   - Type = Square
   - Amplitude = 1
   - Frequency = 0.125 Hz
5. Set the Gain block that is connected to the Signal Generator to 0.
6. Build and run the QUARC® controller.
7. Manually rotate the pendulum in the upright position until the controller engages.

8. Once the pendulum is balanced, set the Gain to 30 to make the arm angle go between $\pm 30^\circ$. The scopes should read something similar as shown in **??**. Attach your response of the rotary arm, pendulum, and controller voltage.

9. In Matlab®, generate the gain using $Q = \texttt{diag}(\begin{bmatrix} 5 & 1 & 1 & 1 \end{bmatrix})$ performed in Step 4 in the first part of this laboratory experiment. The `diag` command specifies the diagonal elements in a square matrix.

10. To apply the newly designed gain to the running QUARC controller, go to the Simulink model and select *Edit* | *Update Diagram* (or press CTRL-D).

11. Examine and describe the change in the *Rotary Arm (deg)* and *Pendulum (deg)* scopes.

12. Adjust the diagonal elements of Q matrix to reduce how much the pendulum angle deflects (overshoots) when the arm angle changes. Describe your experimental procedure to find the necessary control gain.

13. List the resulting LQR Q matrix and control gain K used to yield the desired results. Attach the responses using this new control gain and briefly outline how the response changed.

14. Stop the QUARC® controller.

15. Power *OFF* the QUBE-Servo.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

QUANSER
INNOVATE · EDUCATE

YOU CAN RELY ON QUANSER TO ADVANCE CONTROL EDUCATION

For over two decades Quanser has focused solely on the development of solutions for advanced control education and research. Today, over 2,500 universities, colleges and research institutions around the world rely on a growing portfolio of Quanser control systems.

Our Rotary Control solutions offer quality, convenience, ease of use, ongoing technical support and affordability. They are part of a wider range of Quanser control lab solutions designed to enhance students' academic experience. They come as complete workstations and can captivate undergraduate and graduate students, motivate them to study further and encourage them to innovate.

Engineering educators worldwide agree that Quanser workstations are reliable and robust. Choose from a variety of mechatronics experiments and control design tools appropriate for teaching at all levels as well as advanced research. Take advantage of engineering expertise that includes mechatronics, electronics, software development and control system design. Leverage the accompanying **ABET-aligned courseware** which have been developed to the highest academic standards. Last but not least, rely on Quanser's engineers for ongoing technical support as your teaching or research requirements evolve over time.

Learn more at www.quanser.com or contact us at info@quanser.com

Follow us on:

ABET, Inc., is the recognized accreditor for college and university programs in applied science, computing, engineering, and technology. Among the most respected accreditation organizations in the U.S., ABET has provided leadership and quality assurance in higher education for over 75 years.

 v2.3