QUANSER

INNOVATE. EDUCATE.

STUDENT WORKBOOK

QUBE-Servo Experiment for LabVIEW™ 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: LabVIEW™

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.

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

ACKNOWLEDGEMENTS

Quanser, Inc. would like to thank the following contributors:

Dr. Hakan Gurocak, Washington State University Vancouver, USA, for his help to include embedded outcomes assessment, and

Dr. K. J. Åström, Lund University, Lund, Sweden for his immense contributions to the curriculum content.

CONTENTS

1 QUBE-SERVO INTEGRATION

Topics Covered

- Getting familiarized with the Quanser QUBE-Servo hardware.
- Using LabVIEW™ to interact with Quanser QUBE-Servo system.
- Sensor calibration.

Prerequisites

Before starting this lab make sure:

- The QUBE-Servo has been setup and tested. See the QUBE-Servo Quick Start Guide for details.
- Inertial disc 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 LabVIEW™ .

1.1 Background

1.1.1 Using LabVIEW with Quanser Rapid Control Prototyping Toolkit

The Quanser Rapid Control Prototyping Toolkit® software is used with LabVIEW™ to interact with the hardware of the QUBE-Servo system. LabVIEW is used to drive the dc motor and read angular position of the disc.

The basic steps to create a LabVIEW™ Virtual Instrument (VI) using Quanser Rapid Control Prototyping Toolkit® (RCP) add-on in order to interact with the QUBE-Servo hardware are:

1. Make a LabVIEW Virtual Instrument (VI) that interacts with your installed data acquisition device using blocks from the *Quanser Rapid Control Prototyping* palette.
2. Run the VI.

For examples VIs using RCP, in LabVIEW go to Help | Find Examples and look under *Toolkits and Modules* | *Quanser Rapid Control Prototyping*.

1.1.2 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.

1.1.3 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.

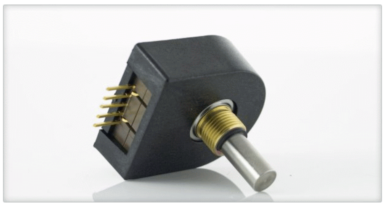


Figure 1.1: US Digital incremental rotary optical shaft encoder

The encoder has a coded disk that is marked with a radial pattern. As the disk rotates (with the shaft), the light from an 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.

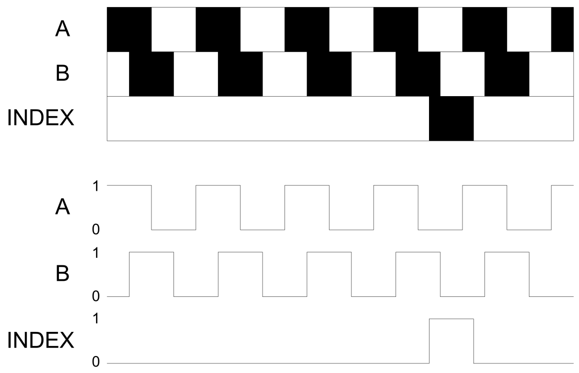
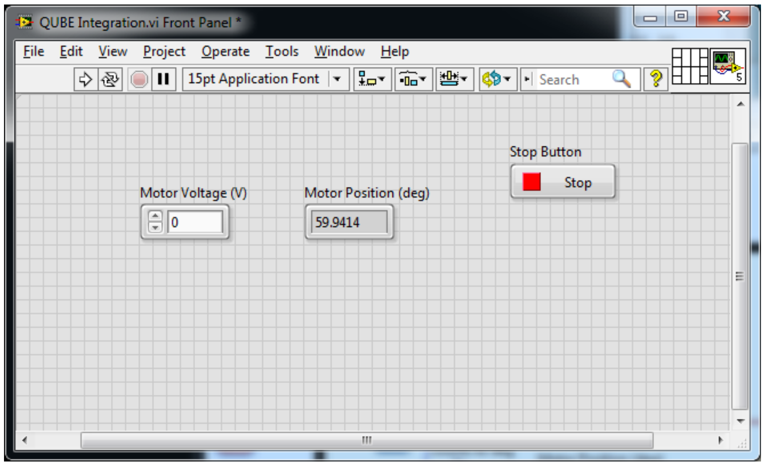


Figure 1.2: Optical incremental encoder signals

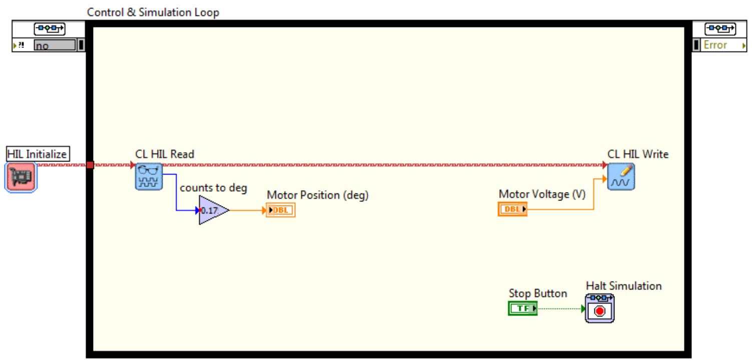
The A and B signals that are generated as the shaft rotates are used in a decoder algorithm to generate a count. The resolution of the encoder depends on the coding of the disk and the decoder. For example, an encoder with 512 lines on the disk can generate a total of 512 counts for every rotation of the encoder shaft. However, in a quadrature decoder the number of counts quadruples, therefore the encoder would generate 2048 counts per revolution.

1.2 In-Lab Exercises

In this lab, we will make a LabVIEW™ Virtual Instrument (VI) using Quanser Rapid Control Prototyping Toolkit® blocks to drive to the dc motor and the measure it corresponding angle - similarly as shown in Figure 1.3.



(a) Front Panel



(b) Block diagram

Figure 1.3: VI used with RCP to drive motor and read angle on QUBE-Servo

1.2.1 Configuring a LabVIEW VI for the QUBE-Servo

Follow these steps build a LabVIEW VI that will interface to the QUBE-Servo using Quanser Rapid Control Prototyping Toolkit®:

1. Load the LabVIEW™ software.
2. Create a new, blank Virtual Instrument (VI).
3. Go to the block diagram and create a Simulation loop, as depicted in Figure 1.4. It is found in the *Control Design & Simulation* | *Simulation* palette.
4. Double-click on the Simulation loop input node (or right-click on the border and select *Configure Simulation Parameters*) to access the *Simulation Parameters* box shown in Figure 1.5.
5. As shown in Figure 1.5, in the *Simulation Parameters* tab set the following:
    - Final time (s): Inf

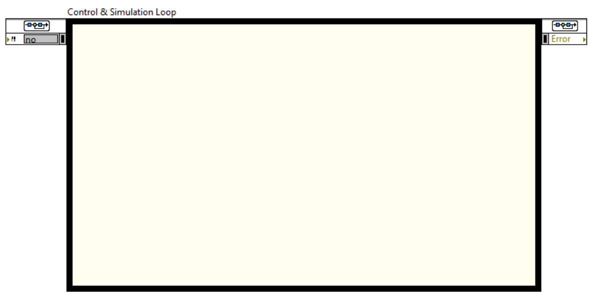


Figure 1.4: Create a Simulation Loop in the block diagram

- ODE Solver: Runge-Kutta 1 (Euler)
- Step Size (s): 0.002

This configures the simulation to run until it is stopped by the user at a sampling rate of 500 Hz. When performing control, any of the fixed solvers can be used but Runge-Kutta 1 is typically the default.

6. As shown in Figure 1.5, in the *Timing Parameters* tab set the following:
   - Select *Synchronize Loop to Timing Source*
   - Timing Source: 1 kHz Clock
   - Select *Auto Period*

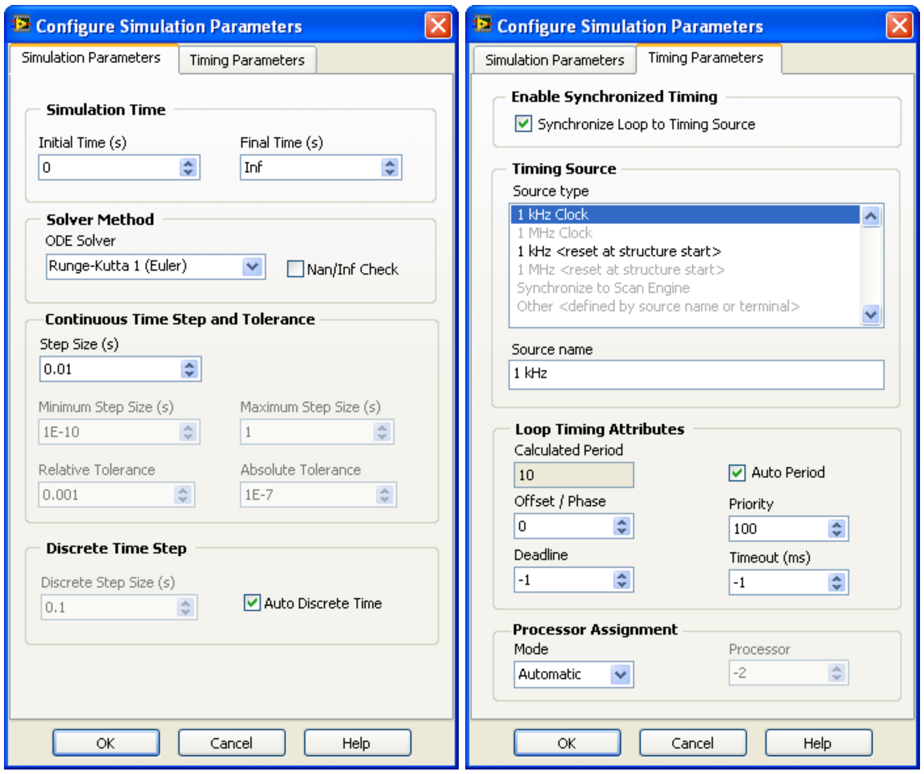
   This synchronizes the simulation to the PC clock. Otherwise, the simulation would run as fast as possible (which is fine when doing pure simulation, but not when interfacing to hardware).

7. Click on the OK button to apply the changes.
8. Add the *HIL Initialize* VI from the *Quanser Rapid Control Prototyping Toolkit* palette.
9. Double-click on the HIL Initialize block.
10. In the *Board type* field, select *qube_servo_usb* under *Quanser Devices*.
    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*.
11. Run the VI by clicking on the white arrow in the top-left corner. The *Power* LED on the QUBE-Servo (or your DAQ card) should be blinking.
12. If you successfully ran the VI without any errors, then you can stop the code by clicking on the *Abort* button in the tool bar.

1.2.2 Reading the Encoder

Follow these steps to read the encoder:

1. Add the *HIL Read* VI from the *Quanser Rapid Control Prototyping* palette.
2. Connect the *board out* terminal from HIL Initialize to the *board in* terminal on HIL Read, similarly as shown in Figure 1.3.
3. Double-click on HIL Read to configure the encoder channel.
4. As shown in Figure 1.6, set the Polymorphic Instance to *Encoder (Scalar)* in order to read from Encoder channel #0.


Configure Simulation Parameters

Simulation Parameters | Timing Parameters

Simulation Time
Initial Time (s): 0
Final Time (s): Inf

Solver Method
ODE Solver: Runge-Kutta 1 (Euler)
☐ Nan/Inf Check

Continuous Time Step and Tolerance
Step Size (s): 0.01
Minimum Step Size (s): 1E-10
Maximum Step Size (s): 1
Relative Tolerance: 0.001
Absolute Tolerance: 1E-7

Discrete Time Step
Discrete Step Size (s): 0.1
☑ Auto Discrete Time

OK | Cancel | Help

(a) Simulation Parameters

Configure Simulation Parameters

Simulation Parameters | Timing Parameters

Enable Synchronized Timing
☑ Synchronize Loop to Timing Source

Timing Source
Source type:
1 kHz Clock
1 MHz Clock
1 kHz <reset at structure start>
1 MHz <reset at structure start>
Synchronize to Scan Engine
Other <defined by source name or terminal>

Source name: 1 kHz

Loop Timing Attributes
Calculated Period: 10
☑ Auto Period
Offset / Phase: 0
Priority: 100
Deadline: -1
Timeout (ms): -1

Processor Assignment
Mode: Automatic
Processor: -2

OK | Cancel | Help

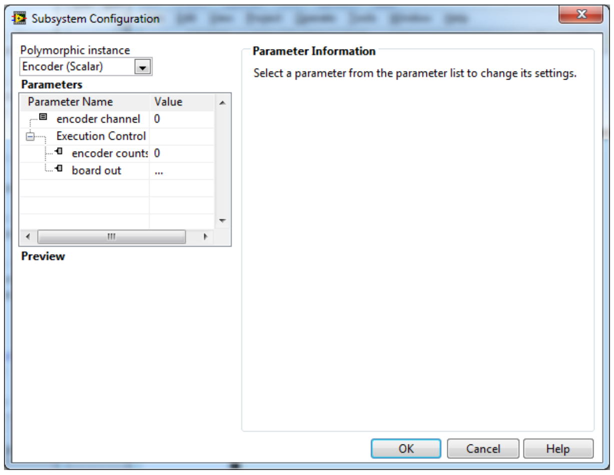
(b) Timing Parameters

Figure 1.5: Simulation Loop Parameters Dialog

5. Connect the HIL Read to a Gain and Numeric indicator block similar as shown in Figure 1.3 (without the HIL Write Analog block).
   - You can find the Gain block in the Control Design & Simulation | Simulation | Signal Arithmetic palette.
   - The indicator can be added by right-clicking on the Gain output and going to Create | Indicator.
6. Go to the front panel. You should see the numeric indicator.
7. Run the VI.
8. Rotate the disc back and forth. The numeric indicator shows the number of counts measured by the encoder. The encoder counts are proportional to the angle of disc.
9. What happens to the encoder reading every time the VI is started? Stop the VI, move around the disc, and re-start VI. What do you notice about the encoder measurement when the VI is re-started?
10. 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.
11. Ultimately we want to display the disc 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 VI and confirm that the numeric indicator shows the angle of the disc correctly.

1.2.3 Driving the DC Motor

1. Add the *HIL Write* block from the *Quanser Rapid Control Prototyping Toolkit* palette. 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.


Subsystem Configuration

Polymorphic instance: Encoder (Scalar)

Parameters

Parameter Name	Value
encoder channel	0
Execution Control	
encoder counts	0
board out	...

Preview

Parameter Information

Select a parameter from the parameter list to change its settings.

Figure 1.6: HIL Read Configuration

2. Wire the *board out* terminal from HIL Read to the *board in* terminal on HIL Write.
3. Add a Numeric Control by right-clicking on *analog voltage* terminal on HIL Write and going go Create | Control. Connect the Constant and HIL Write Analog blocks together, as shown in Figure 1.3.
4. Instead of using the *Abort* button on the VI tool bar, its better practice to add your own stop function. To add your own stop as shown in Figure 1.3 do the following:
   (a) Add the *Halt Simulation* block from the *Control Design & Simulation* | *Simulation* | *Utilities* palette.
   (b) To add a *Stop* button, go the front panel of the VI and look through the *Silver* | *Boolean* palette (or *Modern* | *Boolean*).
   (c) In the block diagram, connect the *Stop* button and *Halt Simulation* VI as shown in Figure 1.3.
5. Run the VI.
6. 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 disc rotate (i.e., clockwise or counter-clockwise) when a positive input is applied?
7. Click on the Stop button to stop the VI.
8. Power OFF the QUBE-Servo.

2 FILTERING

Topics Covered

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

Prerequisites

- QUBE-Servo Integration Lab

2.1 Background

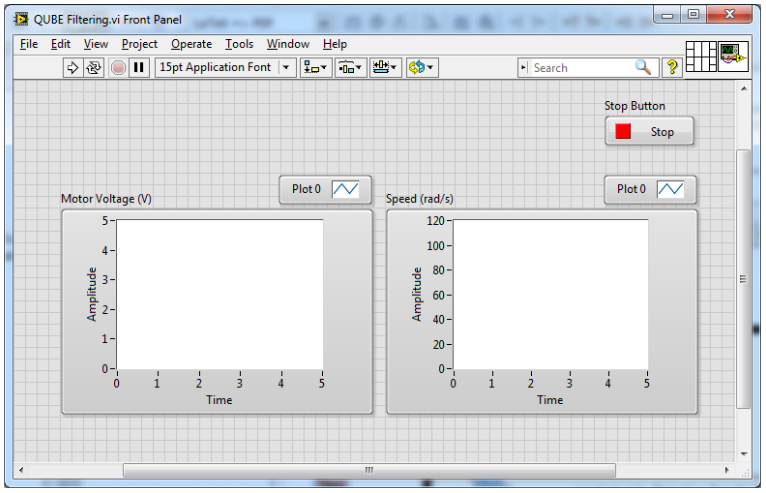
The low-pass filter is used to block out the high-frequency components of a signal. First-order filter transfer function has the form

$$G(s) = \frac{\omega_f}{s + \omega_f}, \tag{2.1}$$

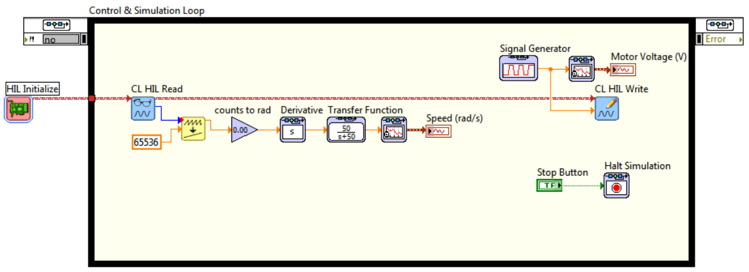
where ω_f is the cut-off frequency of the filter in radians per seconds (rad/s).

2.2 In-Lab Exercises

Based on the model developed in Section 1, the goal is to design a VI that measures the servo velocity using the encoder as shown in Figure 2.1.



(a) Front Panel



(b) Block Diagram

Figure 2.1: Measuring speed using the encoder

1. Take the model you developed in Section 1. Change the encoder calibration gain to measure the gear position in radians, i.e., instead of degrees as in the lab.
2. Build the VI 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 $2^{16} = 65536$. To eliminate the discontinous jump that would occur when the encoder reaches the limits of this range, add the Inverse Modulus block from the Rapid Control Prototyping | Utilities category into the VI, as shown in Figure 2.1. Connect a constant of 65536 to the *Modulus* terminal.
   - **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 SimTime Waveform block. It can be found in the Control Design & Simulation | Simulation | Graph Utilities palette.
3. Add a Signal Generator from the Control Design & Simulation | Simulation | Signal Generation palette. Setup the Signal Generator to output a *square* voltage that goes from 1 V to 3 V at 0.4 Hz.
4. Run the VI. Examine the encoder speed response. Attach sample responses. They should look similar to Figure 2.2.
5. Explain why the encoder-based measurement is noisy.

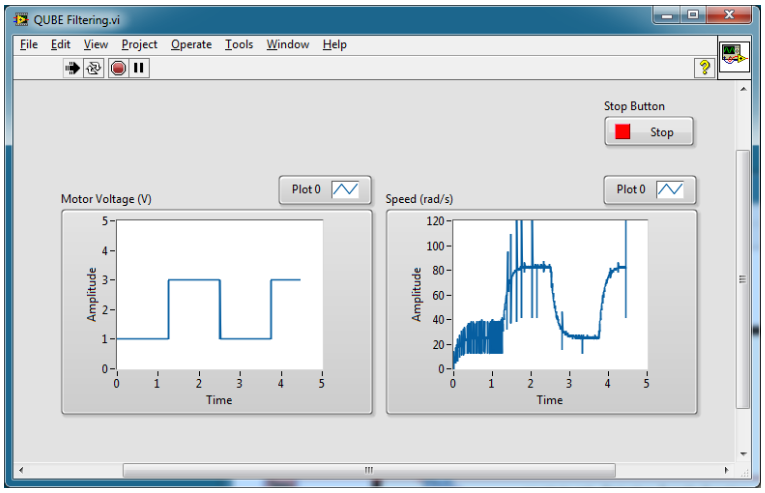


Figure 2.2: Measured servo speed using encoder

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 *Control Design & Simulation* | *Simulation* | *Continuous* palette, 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. Run the VI. 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.

3 STABILITY ANALYSIS

Topics Covered

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

Prerequisites

- QUBE-Servo Integration Lab
- Filtering Lab

3.1 Background

3.1.1 Servo Model

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

$$P(s) = \frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s + 1}, \tag{3.1}$$

where $K = 23.0$ rad/(V-s) is the model steady-state gain, $\tau = 0.13$ s is the model time constant, $\Omega_m(s) = \mathcal{L}[\omega_m(t)]$ is the motor speed (i.e., speed of load disc), and $V_m(s) = \mathcal{L}[v_m(t)]$ 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 3.1 with an integrator in series

$$P(s) = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)} \tag{3.2}$$

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

3.1.2 Stability

Definition for Bounded-Input Bounded-Output (BIBO) stability is ([2]):

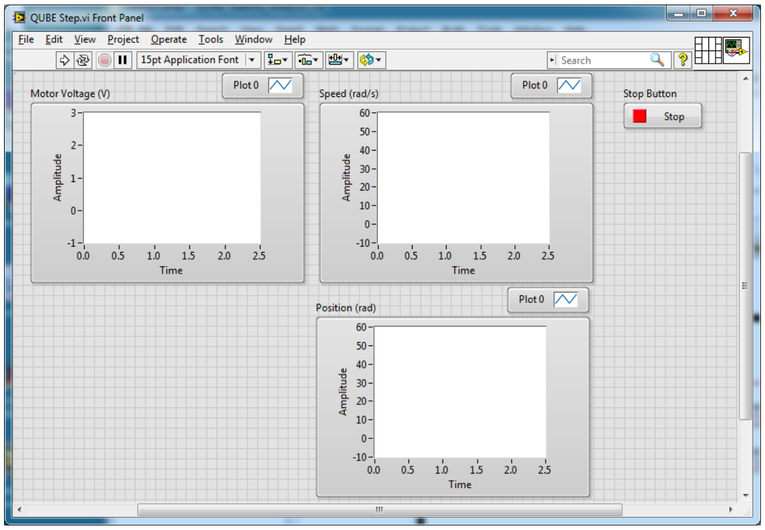
1. A system is stable if every bounded input yields a bounded output.
2. A system is unstable if any bounded input yields a unbounded output.

The stability of a system can be determined from its poles ([2]):

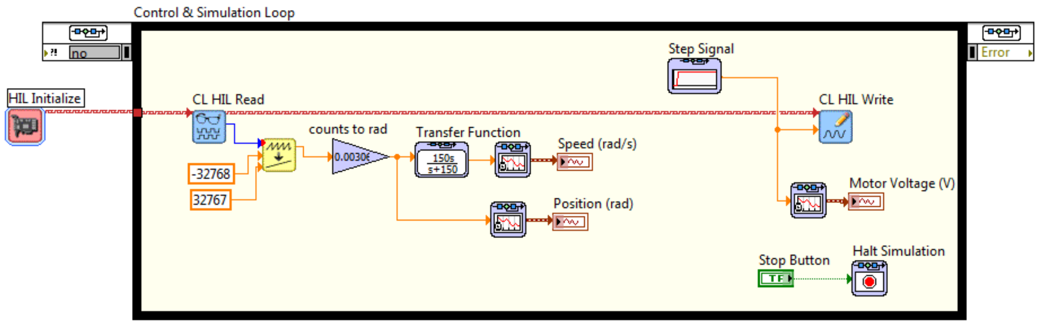
- Stable systems have poles only in the left-hand plane.
- Unstable systems have at least one pole in the right-hand plane and/or poles of multiplicity greater than 1 on the imaginary axis.
- Marginally stable systems have one pole on the imaginary axis and the other poles in the left-hand plane.

3.2 In-Lab Exercises

Based on the VIs already designed in QUBE-Servo Integration and Filtering labs, design a VI that applies a step of 1 V to the motor and reads the servo velocity and the position as shown in Figure 3.1.



(a) Front Panel



(b) Block Diagram

Figure 3.1: 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 VI shown in Figure 3.1. Configure the Simulation Loop to run for 2.5 seconds. The position and speed step response should be similar to Figure 3.2.
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 VI 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?

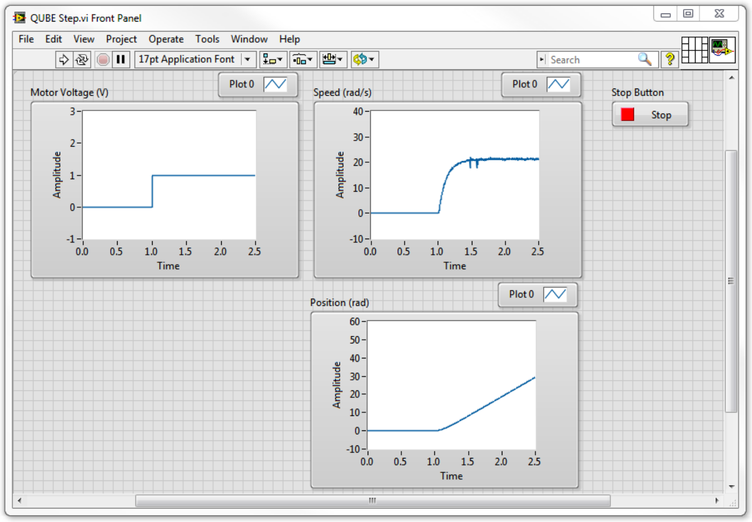


Figure 3.2: Step Response

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

4 BUMP TEST MODELING

Topics Covered

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

Prerequisites

- Lab #1: QUBE-Servo Integration.
- Lab #2: Filtering.

4.1 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:

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1} \tag{4.1}$$

The step response shown in Figure 4.1 is generated using this transfer function with $K = 5$ rad/V-s and $\tau = 0.05$ s.

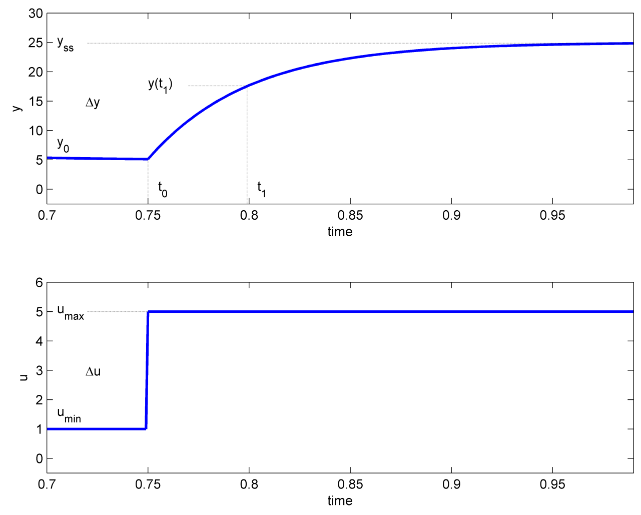


Figure 4.1: Input and output signal used in the bump test method

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

$$K = \frac{\Delta y}{\Delta u} \tag{4.2}$$

where $\Delta y = y_{ss} - y_0$ and $\Delta u = u_{max} - u_{min}$. In order to find the model time constant, τ, we can first calculate where the output is supposed to be at the time constant from:

$$y(t_1) = 0.632\Delta y + y_0. \tag{4.3}$$

Then, we can read the time t_1 that corresponds to $y(t_1)$ from the response data in Figure 4.1. From the figure we can see that the time t_1 is equal to:

$$t_1 = t_0 + \tau$$

From this, the model time constant can be found as:

$$\tau = t_1 - t_0 \tag{4.4}$$

4.1.1 Applying this to the QUBE-Servo

Going back to the QUBE-Servo system, a step input voltage with a time delay t_0 can be expressed as follows in the Laplace domain:

$$V_m(s) = \frac{A_v\, e^{(-s\,t_0)}}{s} \tag{4.5}$$

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

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s + 1} \tag{4.6}$$

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

If we substitute input 4.5 into the system transfer function 4.6, we get:

$$\Omega_m(s) = \frac{K A_v\, e^{(-s\,t_0)}}{(\tau\, s + 1)\, s}$$

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

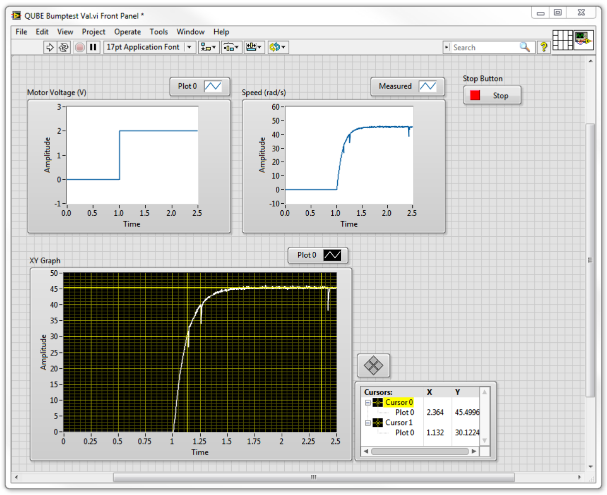
$$\omega_m(t) = K\, A_v \left(1 - e^{\left(-\frac{t-t_0}{\tau}\right)}\right) + \omega_m\,(t_0)$$

Here we need to be careful with the time delay t_0 and note that the initial condition is $\omega_m(0^-) = \omega_m(t_0)$.

4.2 In-Lab Exercises

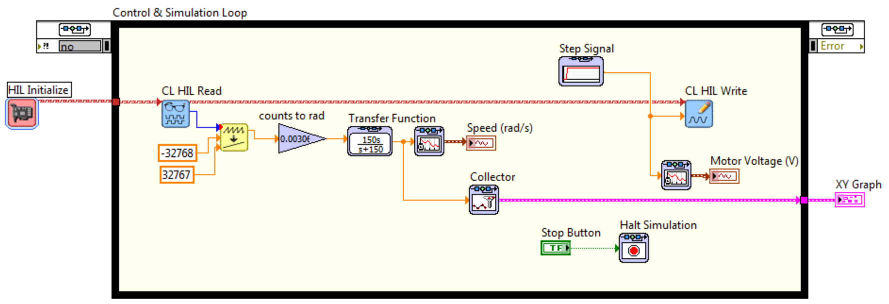
Based on the VIs already designed in QUBE-Servo Integration and Filtering labs, design a VI that applies a 2V step to the motor and reads the servo velocity using the encoder as shown in Figure 4.2.

To apply your step for a 2.5 seconds, set the *Final Time* of the Simulation Loop to 2.5 (instead of *Inf*). Using the saved response, the model parameters can then be found as discussed in Section 4.1. As shown in Figure 4.2, the bumptest response is "saved" using the Collector block from the Control Design & Simulation | Simulation | Utilities palette and displayed in an XY Graph. LabVIEW graphs (as opposed to charts) have cursors that can be used to take measurements.



Cursors:	X	Y
Cursor 0		
Plot 0	2.364	45.4996
Cursor 1		
Plot 0	1.132	30.1224

(a) Front Panel



(b) Block Diagram

Figure 4.2: Applies a step voltage and measures corresponding servo speed

1. Run the VI to apply a 2 V step to the servo. The scope response should be similar to Figure 4.3.
2. Plot the motor speed response and the input voltage. For example, you can use the Export | Export Simplified Image to save the measured load/disc speed and motor voltage to a picture file and attach that to your report.
3. Find the steady-state gain using the measured step response. **Hint:** Use the cursor palette in the XY Graph 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 VI to include a Transfer Function block with the first-order model in Equation 4.1, as shown in Figure 4.4. Display both the measured and simulated QUBE-Servo responses in one scope (using Build Array from the Array category). Run the VI. Attach a 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.

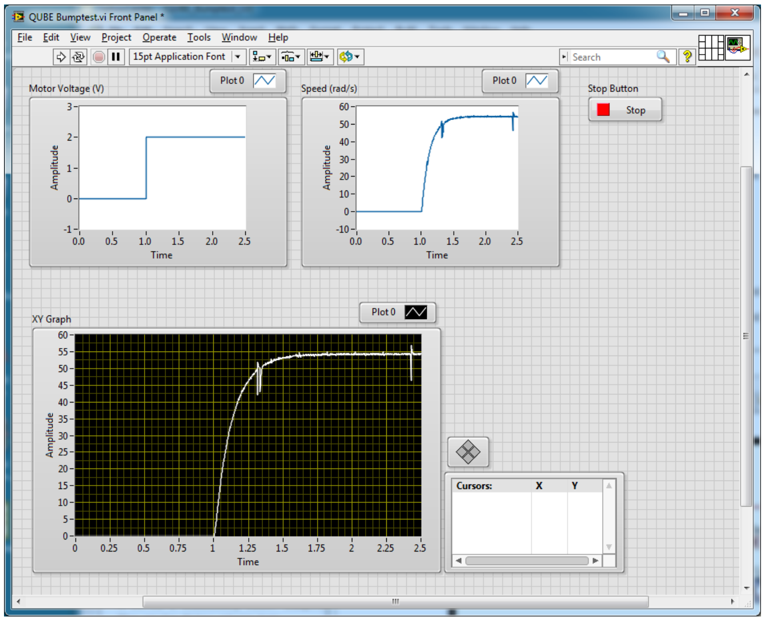


Figure 4.3: QUBE-Servo Bump Test Response

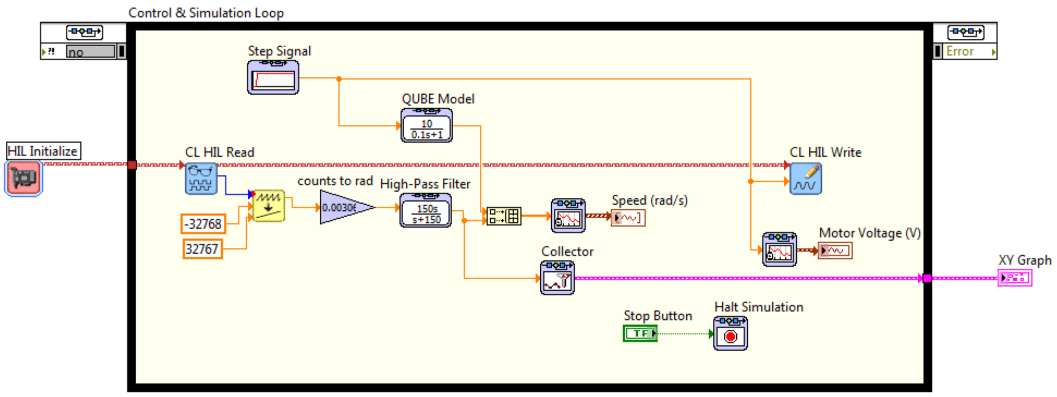


Figure 4.4: Validating bump test model

5 FIRST PRINCIPLES MODELING

Topics Covered

- Obtaining the equations of motion of a dc motor based rotary servo.
- Modeling rotary servo in LabVIEW.
- Model validation.

Prerequisites

- QUBE-Servo Integration Lab
- Filtering Lab

5.1 Background

The Quanser QUBE-Servo is a direct-drive rotary servo system. Its motor armature circuit schematic is shown in Figure 5.1 and the electrical and mechanical parameters are given in Table 5.1. The dc motor shaft is connected to the *load hub*. The hub is a metal disc used to mount the disc or rotary pendulum and has a moment of inertia of J_h. A disc load is attached to the output shaft with a moment of inertia of J_d.

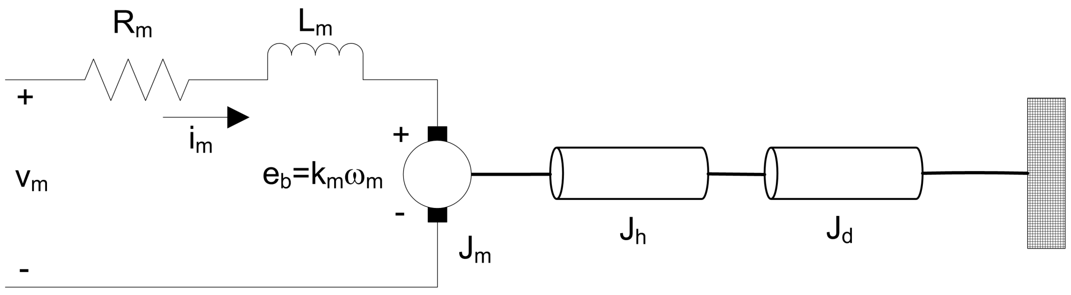


Figure 5.1: QUBE-Servo dc motor and load

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:

$$e_b(t) = k_m \omega_m(t)$$

Symbol	Description	Value
DC Motor		
R_m	Terminal resistance	6.3 Ω
k_t	Torque constant	0.036 N-m/A
k_m	Motor back-emf constant	0.036 V/(rad/s)
J_m	Rotor inertia	4.0×10^{-6} kg-m^2
L_m	Rotor inductance	0.85 mH
m_h	Load hub mass	0.0087 kg
r_h	Load hub mass	0.0111 m
J_h	Load hub inertia	1.07×10^{-6} kg-m^2
Load Disc		
m_d	Mass of disc load	0.054 kg
r_d	Radius of disc load	0.0248 m

Table 5.1: QUBE-Servo System Parameters

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

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

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

$$v_m(t) - R_m i_m(t) - k_m \omega_m(t) = 0.$$

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

$$i_m(t) = \frac{v_m(t) - k_m \omega_m(t)}{R_m}. \tag{5.1}$$

The motor shaft equation is expressed as:

$$J_{eq} \dot{\omega}_m(t) = \tau_m(t) \tag{5.2}$$

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

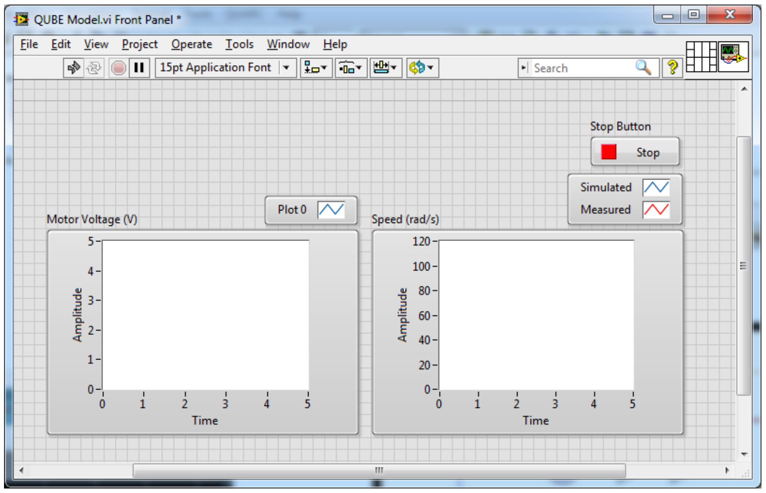
$$\tau_m = k_m i_m(t)$$

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

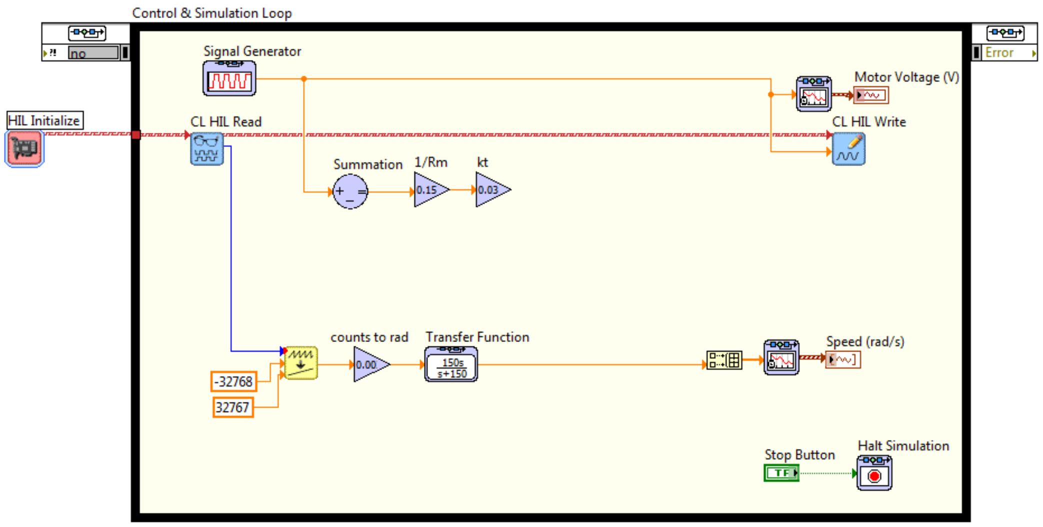
$$J = \frac{1}{2} m r^2. \tag{5.3}$$

5.2 In-Lab Exercises

Based on the VIs already designed in QUBE-Servo Integration and Filtering labs, design a VI 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 5.2.



(a) Front Panel



(b) Block Diagram

Figure 5.2: Applies a step voltage and displays measured and simulated QUBE-Servo speed (incomplete block diagram).

Thus using the equations given above, assemble a simple block digram in the block diagram of the VI 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 the block diagram in Figure 5.2.

1. The motor shaft of the QUBE-Servo is attached to a *load hub* and a disc load. Based on the parameters given in Table 5.1, calculate the equivalent moment of inertia that is acting on the motor shaft.
2. Design the QUBE-Servo model using LabVIEW Control Design & Simulation blocks as described above. Attach a screen capture of your model.
3. Run the VI with your QUBE-Servo model. The scope response should be similar to Figure 5.3. Attach a screen capture of your scopes. Does your model represent the QUBE-Servo well? Explain.

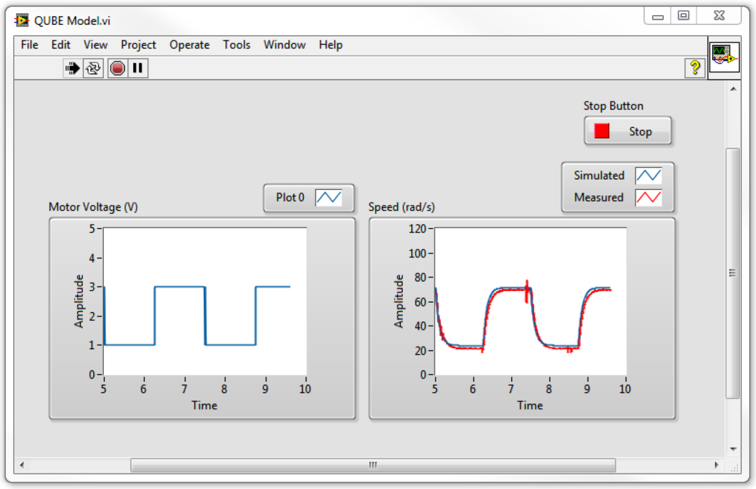


Figure 5.3: QUBE-Servo Measured and Simulated Responses

6 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 Lab
- Filtering Lab

6.1 Background

6.1.1 Second-Order Step Response

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

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\,\omega_n\,s + \omega_n^2} \tag{6.1}$$

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

Consider a second-order system as shown in Equation 6.1 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 6.1, where the red trace is the response (output), $y(t)$, and the blue trace is the step input $r(t)$.

6.1.2 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 6.1, the percent overshoot is found using

$$PO = \frac{100\,(y_{max} - R_0)}{R_0} \tag{6.2}$$

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

$$t_p = t_{max} - t_0 \tag{6.3}$$

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

$$PO = 100\,e^{\left(-\frac{\pi\,\zeta}{\sqrt{1-\zeta^2}}\right)} \tag{6.4}$$

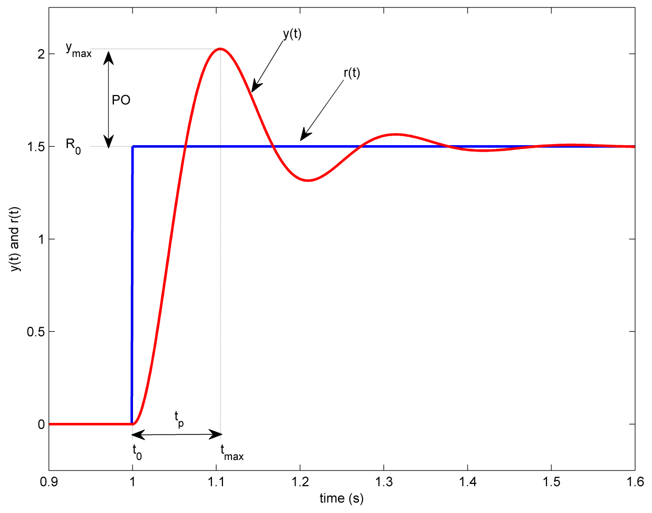


Figure 6.1: Standard second-order step response.

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

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \tag{6.5}$$

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

6.1.3 Unity Feedback

The unity-feedback control loop shown in Figure 6.2 will be used to control the position of the QUBE-Servo.

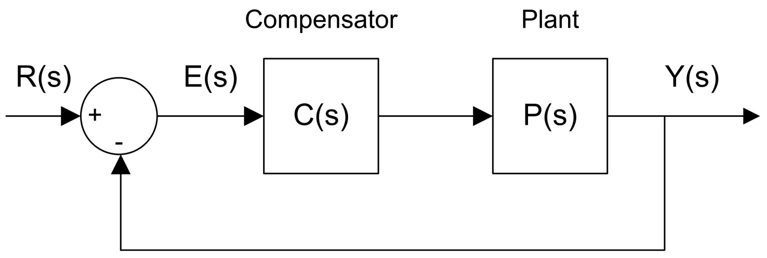


Figure 6.2: Unity feedback loop

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

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

where $K = 23.0$ rad/(V-s) is the model steady-state gain, $\tau = 0.13$ s is the model time constant, $\Theta_m(s) = \mathcal{L}[\theta_m(t)]$ is the motor / disc position, and $V_m(s) = \mathcal{L}[v_m(t)]$ 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 controller is denoted by $C(s)$. In this lab, we are only going to do unity feedback therefore

$$C(s) = 1.$$

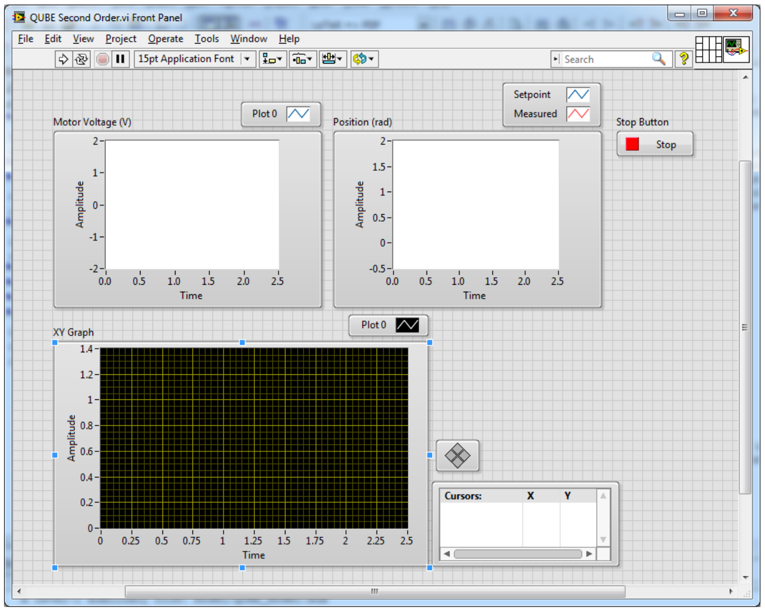
The closed-loop transfer function of the QUBE-Servo position control using unity feedback as shown in Figure 6.2 is

$$\frac{\Theta_m(s)}{V_m(s)} = \frac{\frac{K}{\tau}}{s^2 + \frac{1}{\tau}s + \frac{K}{\tau}} \tag{6.6}$$

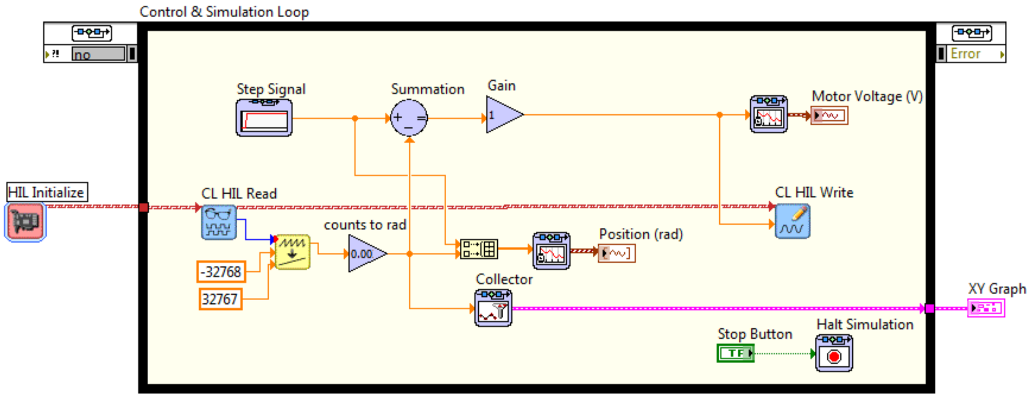
6.2 In-Lab Exercises

Design the VI shown in Figure 6.3. This implements the unity feedback control given in Figure 6.2 in LabVIEW. 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.

To apply your step for a 2.5 seconds, set the *Final Time* of the Simulation Loop to 2.5 (instead of *Inf*). Using the saved response, the peak time and overshoot can be found as discussed in Section 6.1. As shown in Figure 6.3, the unity feedback step response is "saved" using the Collector block from the Control Design & Simulation | Simulation | Utilities palette and displayed in an XY Graph. LabVIEW graphs (as opposed to charts) have cursors that can be used to take measurements.



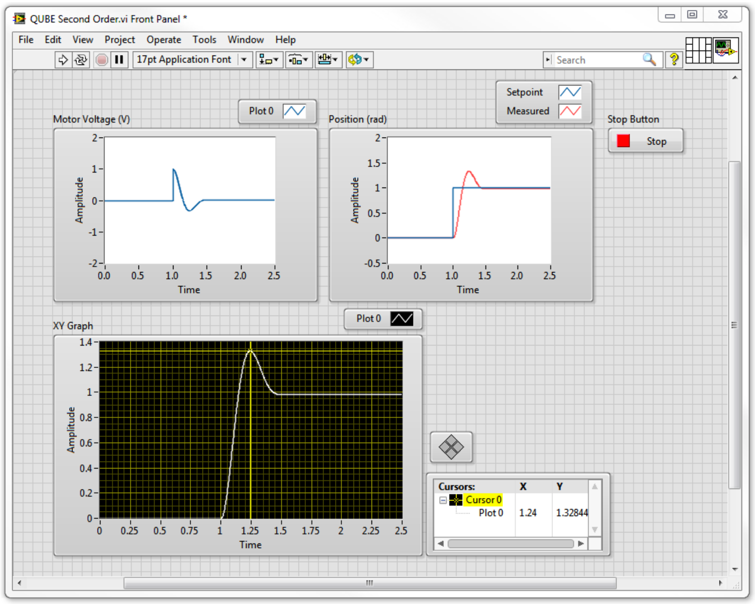
(a) Front Panel



(b) Block Diagram

Figure 6.3: Unity feedback position control of QUBE-Servo

1. Given the QUBE-Servo closed-loop equation under unity feedback in Equation 6.6 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. Run the VI. The scopes should look similar to Figure 6.4.



Cursors:	X	Y
Cursor 0		
Plot 0	1.24	1.32844

Figure 6.4: 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. For example, you can use the Export | Export Simplified Image to save the measured load/disc speed and motor voltage to a picture file and attach that to your report.

5. Measure the peak time and percent overshoot from the response and compare that with your expect results. **Hint:** Use the cursor palette in the XY Graph to measure points off the plot.

7 PD CONTROL

Topics Covered

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

Prerequisites

- QUBE-Servo Integration Lab
- Filtering Lab
- Second-Order Systems Lab

7.1 Background

7.1.1 Servo Model

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

$$P(s) = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)}. \tag{7.1}$$

where $K = 23.2$ rad/(V-s) is the model steady-state gain, $\tau = 0.13$ s is the model time constant, $\Theta_m(s) = \mathcal{L}[\theta_m(t)]$ is the motor / disc position, and $V_m(s) = \mathcal{L}[v_m(t)]$ 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).

7.1.2 PID Control

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

$$u(t) = ke(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt} \tag{7.2}$$

The corresponding block diagram is given in Figure 7.1. The control action is thus a sum of three terms referred to as proportional (P), integral (I) and derivative (D). The controller Equation 7.2 can also be described by the transfer function

$$C(s) = ks + \frac{k_i}{s} + k_d s \tag{7.3}$$

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. Advanced model-based controllers differ from the PID controller by using a model of the process for prediction.

The PID controller described by Equation 7.2 or Equation 7.3 is the ideal PID controller. Attempts to implement these formulas may not lead to good controllers. For example, most measurement signals have noise and taking the differentiation of a noisy signal gives very large fluctuations.

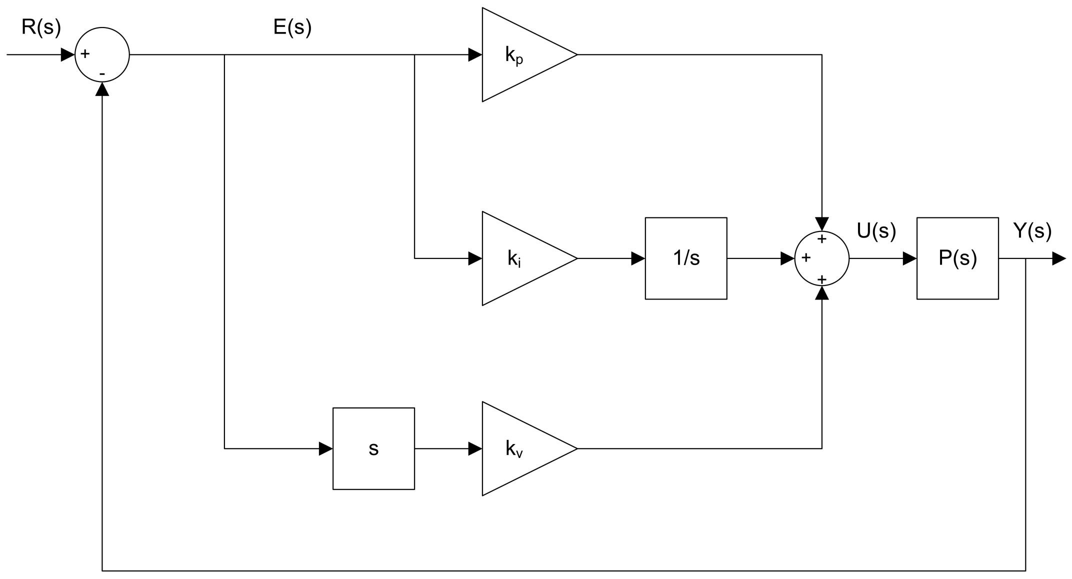


Figure 7.1: Block diagram of PID control.

7.1.3 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 7.2. Unlike the standard PD, only the negative velocity is fed back (i.e., not the velocity of the *error*) and a high-pass filter, $H(s)$ is used instead of a direct derivative. Filtering can remove a lot of the noise that occurs when taking the derivative.

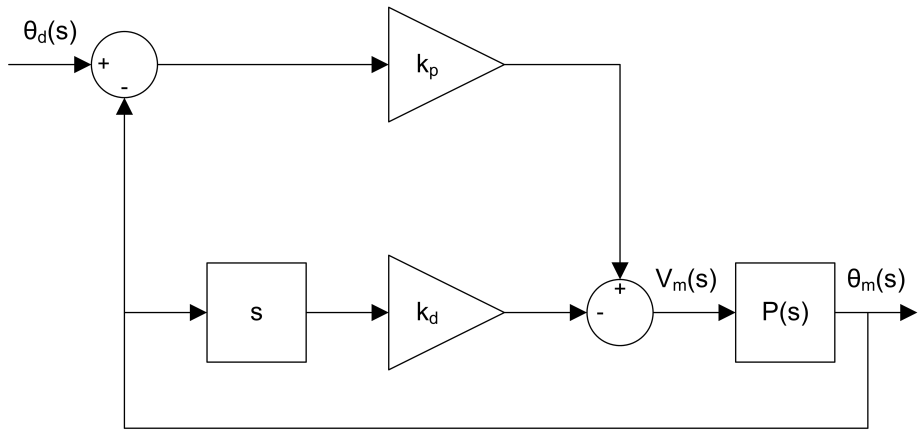


Figure 7.2: Block diagram of PV control.

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

$$u = k_p\ (r(t) - y(t)) - k_d\, \dot{y}(t) \tag{7.4}$$

where k_p is the proportional gain, k_d is the derivative 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 (i.e., in this case, the 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 of Equation 7.4 gives

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

and substituting that into Equation 7.1 we get

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

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

$$\frac{Y(s)}{R(s)} = \frac{K\,k_p}{\tau s^2 + (1 + K\,k_d)\,s + K\,k_p}. \tag{7.5}$$

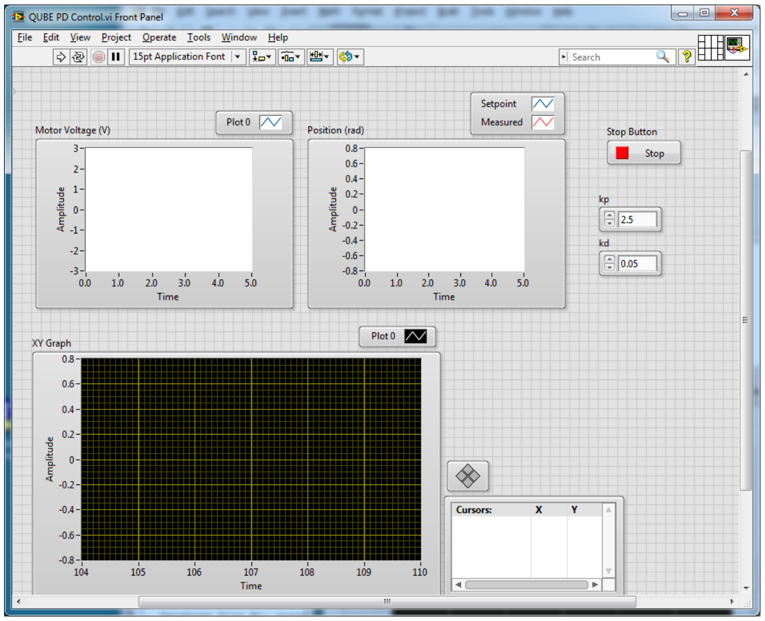
This is a second-order transfer function. Recall the standard second-order transfer function

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\,\omega_n\,s + \omega_n^2}. \tag{7.6}$$

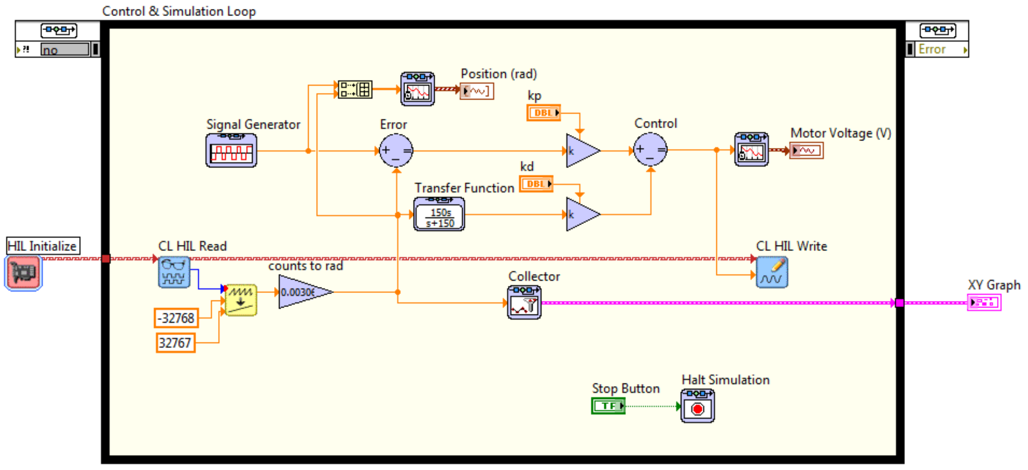
7.2 In-Lab Exercises

Design the VI shown in Figure 7.3. This implements the PV controller outlined in Section 7.1.3 with a high-pass filter of $150s/(s + 150)$. 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.

Using the saved response, the peak time and overshoot can be found as discussed in Section 6.1. As shown in Figure 7.3, the PD response is "saved" using the Collector block from the Control Design & Simulation | Simulation | Utilities palette and displayed in an XY Graph. LabVIEW graphs (as opposed to charts) have cursors that can be used to take measurements.



(a) Front Panel



(b) Block Diagram

Figure 7.3: PV control on QUBE-Servo

1. Design and run the VI. The response should look similarly as shown in Figure 7.4.
2. Set $k_p = 2.5$ V/rad and $k_d = 0$. Keep the derivative gain at 0 and vary k_p between 1 and 4. What does the proportional gain do when controlling servo position?

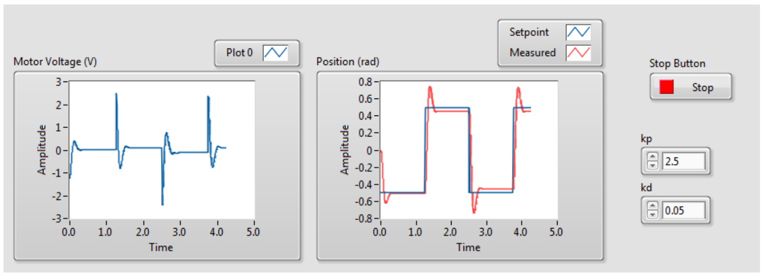


Figure 7.4: QUBE-Servo PV control with $k_p = 2.5$ and $k_d = 0.05$.

Hint: It will make it easier to adjust the gain if you set the *Increment* property of the numeric control to 0.1, or some other value.

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?
   Hint: It will make it easier to adjust the gain if you set the *Increment* property of the numeric control to 0.01, or some other value.

4. Stop the VI.

5. Find the proportional and derivative gains required for the QUBE-Servo closed-loop transfer function given in Equation 7.5 to match the standard second-order system in Equation 7.6. Your gain equations will be a function of ω_n and ζ.

6. 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 Section 7.1.1 (or those you found previously through a modeling lab), calculate the control gains needed to satisfy these requirements.

7. Run the PV controller with the newly designed gains on the QUBE-Servo. Attach the position response as well as the motor voltage used. For example, you can use the Export | Export Simplified Image to save the measured load/disc speed and motor voltage to a picture file and attach that to your report.

8. Measure the percent overshoot and peak time of the response. Do they match the desired percent overshoot and peak time specifications given in Step 6 without saturating the motor, i.e., going beyond $\pm$ 10 V?
   Hint: Use the cursor palette in the XY Graph to measure points off the plot and the equations from Lab #6.

9. 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 response, measurements, and comment on how you modified your controller to arrive at those results.

8 PENDULUM MOMENT OF INERTIA

Topics Covered

- Finding moment of inertia analytically and experimentally.

Prerequisites

Before starting this lab make sure:

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

8.1 Background

The free-body diagram of the QUBE-Servo pendulum system is shown in Figure 8.1.

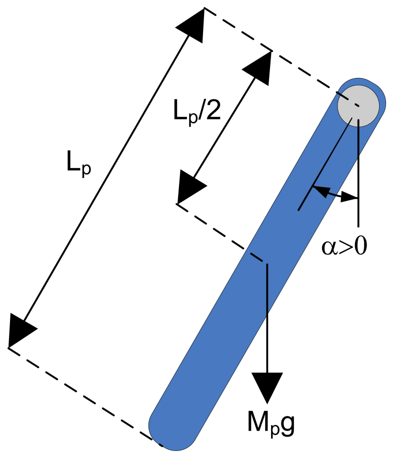


Figure 8.1: Free-body diagram of pendulum

From the free-body diagram in Figure 8.1, the resulting nonlinear equation of motion of the pendulum is

$$J_p\ddot{\alpha}(t) = M_p\, g\, l_p \sin\alpha(t) \tag{8.1}$$

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 8.1.

The moment of inertia of the pendulum can be found experimentally. Assuming the pendulum is unactuated, linearizing Equation 8.1 and solving for the differential equation gives the expression

$$J_p = \frac{M_p\, g\, l_p}{(2\pi f)^2} \tag{8.2}$$

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

$$f = \frac{n_{cyc}}{\Delta t} \tag{8.3}$$

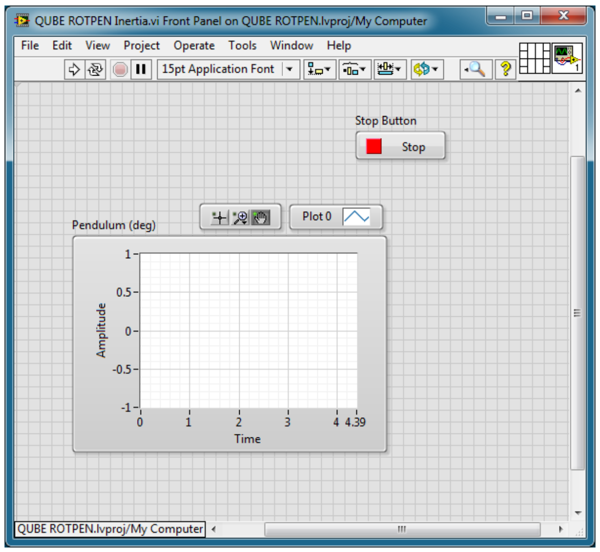
where n_{cyc} is the number of cycles and Δt is the duration of these cycles. Alternatively, J_p can be calculated using the moment of inertia expression

$$J = \int r^2 dm \tag{8.4}$$

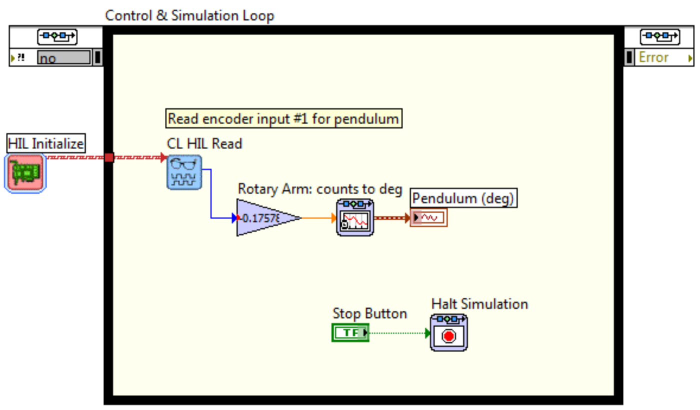
where r is the perpendicular distance between the element mass, dm, and the axis of rotation.

8.2 In-Lab Exercises

Based on the model already designed in QUBE-Servo Integration lab, design a model that measures the pendulum angle using the encoder as shown in Figure 8.2.


QUBE ROTPEN Inertia.vi Front Panel on QUBE ROTPEN.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Stop Button
Stop
Pendulum (deg)
Plot 0
Amplitude
Time
QUBE ROTPEN.lvproj/My Computer

(a) Front Panel


Control & Simulation Loop
no
Error
Read encoder input #1 for pendulum
HIL Initialize
CL HIL Read
Rotary Arm: counts to deg
0.17578
Pendulum (deg)
Stop Button
Halt Simulation

(b) Block Diagram

Figure 8.2: 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 VI shown in Figure 8.2. Enter $360/512/4$ in the encoder sensor gain to measure the pendulum angle in degrees.
3. Run the VI. With the controller running, manually perturb the pendulum while holding the rotary arm in place. The scope response should be similar to Figure 8.3.

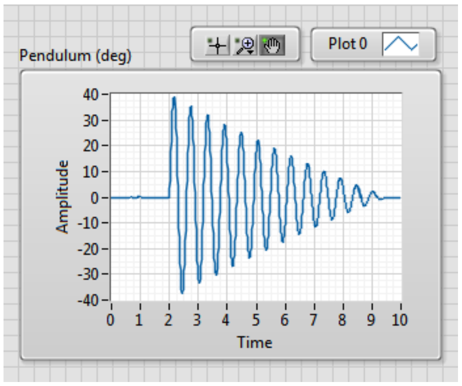


Figure 8.3: Free-oscillation response of pendulum

4. Find the frequency and moment of inertia of the pendulum using the observed results. If needed, use the Graph Palette to zoom on the response.
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?

9 ROTARY PENDULUM MODELING

Topics Covered

- Using LabVIEW™ to interact with Quanser QUBE-Servo Rotary Pendulum system.
- Configure sensor and actuator gains to match model conventions.

Prerequisites

Before starting this lab make sure:

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

9.1 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 engineeering. The free-body diagram of a basic rotary pendulum is depicted in Figure 9.1.

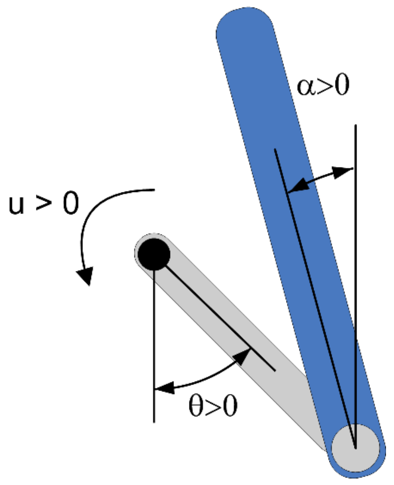


Figure 9.1: Free-body diagram of rotary pendulum

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

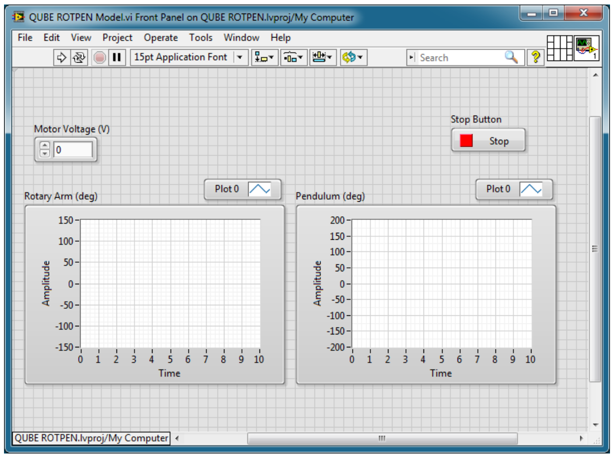
  $$\alpha = \alpha_{full} \text{ mod } 2\pi - \pi. \tag{9.1}$$

  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 LabVIEW VI that follows these modeling conventions. The QUBE-Servo Integration laboratory introduced the DC motor and encoders on the QUBE-Servo system. The pendulum angle is also measured using an encoder.

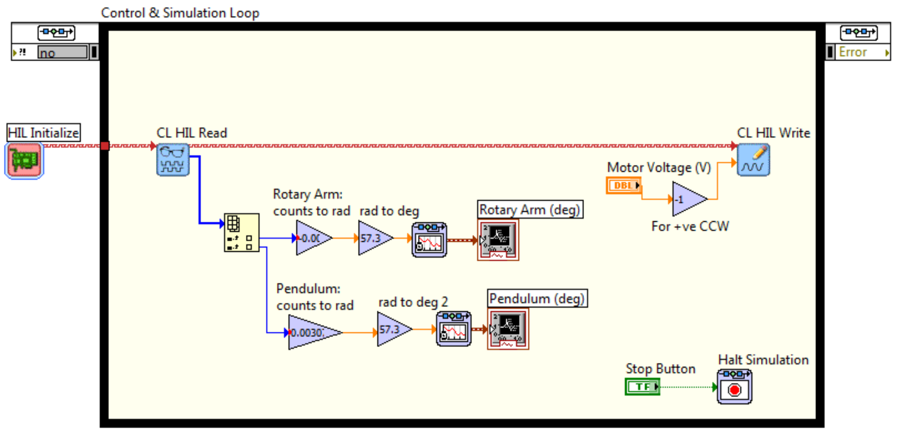
9.2 In-Lab Exercises

In this lab, we will make a LabVIEW™ Virtual Instrument (VI) to drive to the dc motor and measure both the rotary arm and pendulum angles - similarly as shown in Figure 9.2.



QUBE ROTPEN Model.vi Front Panel on QUBE ROTPEN.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Motor Voltage (V)
0
Stop Button
Stop
Rotary Arm (deg)
Plot 0
Pendulum (deg)
Plot 0
Amplitude
Time
QUBE ROTPEN.lvproj/My Computer

(a) Front Panel



(b) Block Diagram

Figure 9.2: VI used to drive motor and read angles on QUBE-Servo ROTPEN system

1. Using the VI you made in the *QUBE-Servo Integration* lab, do the following:
   - Configure the HIL Read Encoder 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, i.e., instead of degrees as in the lab.
   - Connect the the measured angles to scopes, but display them in degrees (usually more intuitive). You can do this by adding Gain blocks that convert radians to degrees.
   - Connect a Numeric Control to the Analog Output channel to change the motor voltage.
2. Run the VI.
3. Rotate the rotary arm and pendulum counter-clockwise and examine the response on the scopes. Example responses are shown in Figure 9.3. Do the measured angles follow the modeling conventions given in Section 9.1.
4. Apply a small voltage, e.g., 0.5 V, to the motor. Does this adhere to the modeling conventions?

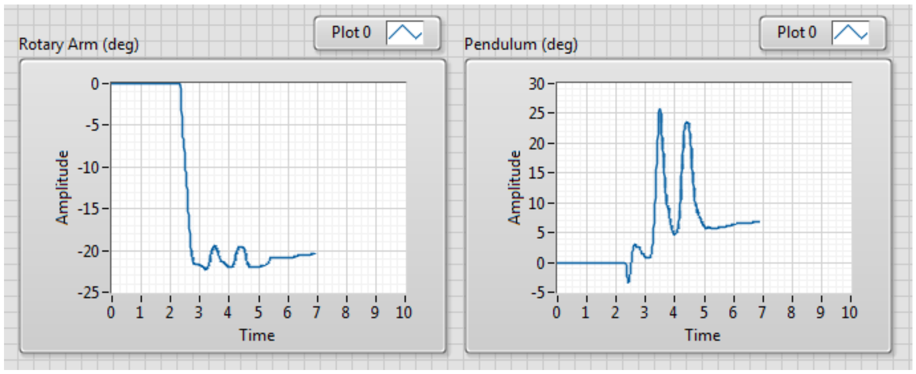


Figure 9.3: Measured rotary arm and pendulum angles

5. Modify the VI 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 9.4, to measure *inverted pendulum angle*, defined as Equation 9.1.

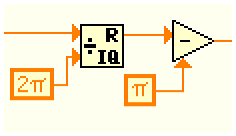


Figure 9.4: LabVIEW modulus and bias blocks

7. Make sure the pendulum is hanging in the downward position and lies motionless before starting the controller.
8. Run the VI.
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 9.1. Capture the response of the pendulum being raise to the inverted position. Explain what the bias and modulus functions do?
10. Stop the VI.
11. Power OFF the QUBE-Servo if no more experiments will be conducted.

10 BALANCE CONTROL

Topics Covered

- Control enabling logic
- PID-based balance control

Prerequisites

Before starting this lab make sure:

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

10.1 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 simply with a PD controller, as shown in Figure 10.1. If we are also interested in keeping the arm in a fixed position a feedback from the arm position will also be introduced. The control law can then be expressed as

$$u = k_{p,\theta}(\theta_r - \theta) - k_{p,\alpha}\alpha - k_{d,\theta}\dot{\theta} - k_{d,\alpha}\dot{\alpha} \tag{10.1}$$

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 there is no reference for the pendulum angle because the desired position is zero.

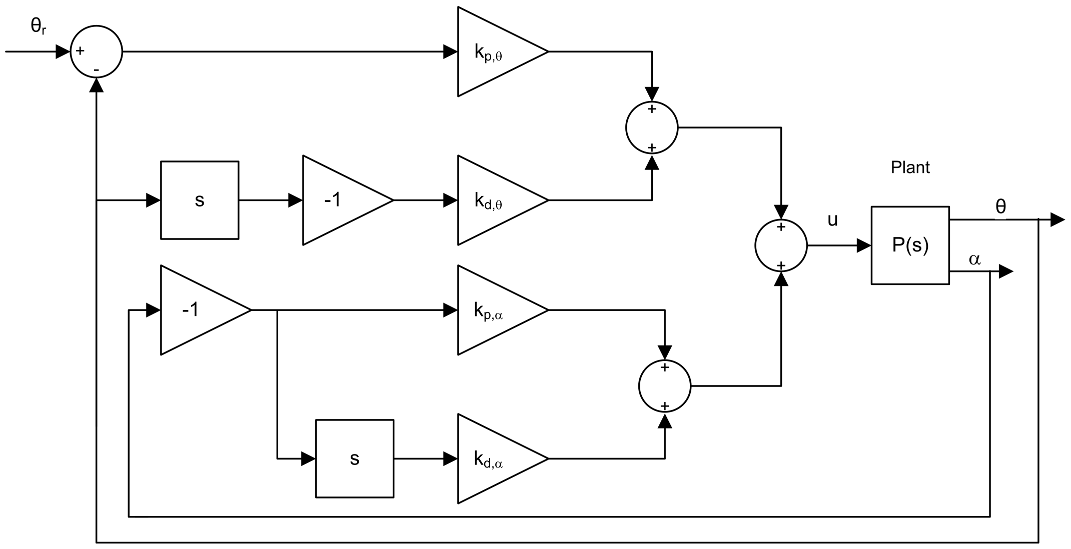


Figure 10.1: Block diagram of balance PD control for rotary pendulum

There are many different ways to find the controller parameters. As discussed in Section 12, one method is based on LQR-optimal control. Initially, however, the behaviour 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 $\alpha = \alpha_{full}$ mod 2π, as defined in Equation 9.1.

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

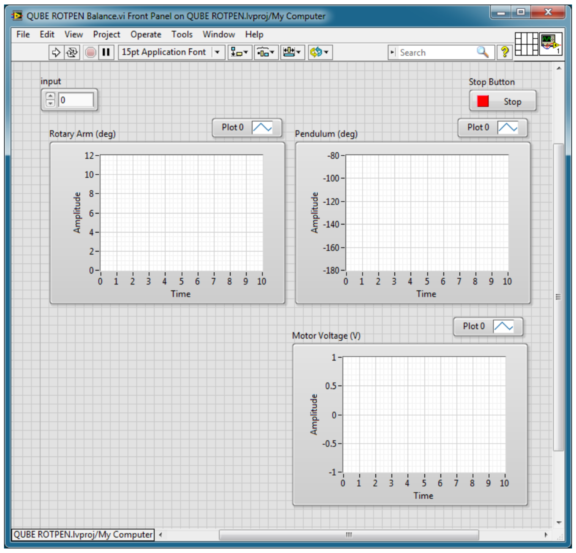
$$|\alpha| \leq 10 \text{ deg.}$$

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, the balance controller is engaged. It remains in balance mode until the pendulum goes beyond $\pm$ 10 deg.

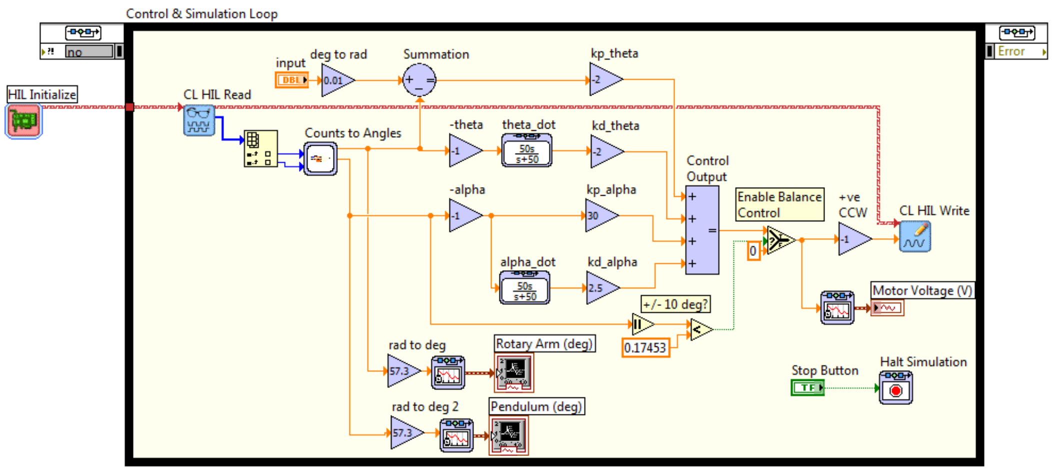
If desired, you can integrate this with an algorithm that swings-up the pendulum automatically. See Lab #11: Swing-Up Control for details.

10.2 In-Lab Exercises

Construct a LabVIEW™ Virtual Instrument (VI) similarly as shown in Figure 10.2 that balances the pendulum on the QUBE-Servo rotary pendulum using the PD control detailed in Section 10.1.


QUBE ROTPEN Balance.vi Front Panel on QUBE ROTPEN.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
input
Stop Button
Stop
Rotary Arm (deg)
Plot 0
Pendulum (deg)
Amplitude
Time
Motor Voltage (V)
QUBE ROTPEN.lvproj/My Computer

(a) Front Panel



(b) Block Diagram

Figure 10.2: VI to run PD balance controller

1. Using the VI you made in the *Rotary Pendulum Model* lab, construct the controller shown in Figure 10.2:
   - The *Counts to Angles* subsystem contains the same blocks used in the *Rotary Pendulum Model* lab 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 done in the *Filtering* lab.
   - Add the necesary Sum and Gain blocks to implement the PD control given in Equation 10.1.
   - The controller should only be enabled when the pendulum is $\pm$ 10 deg about the upright vertical position, or $\pm$ 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. Make sure the pendulum is hanging in the downward position and lies motionless before starting the controller.

4. Run the VI.

5. Manually rotate the pendulum in the upright position until the controller engages. The scopes should read something similar as shown in Figure 10.3. Attach response of the rotary arm, pendulum, and controller voltage.

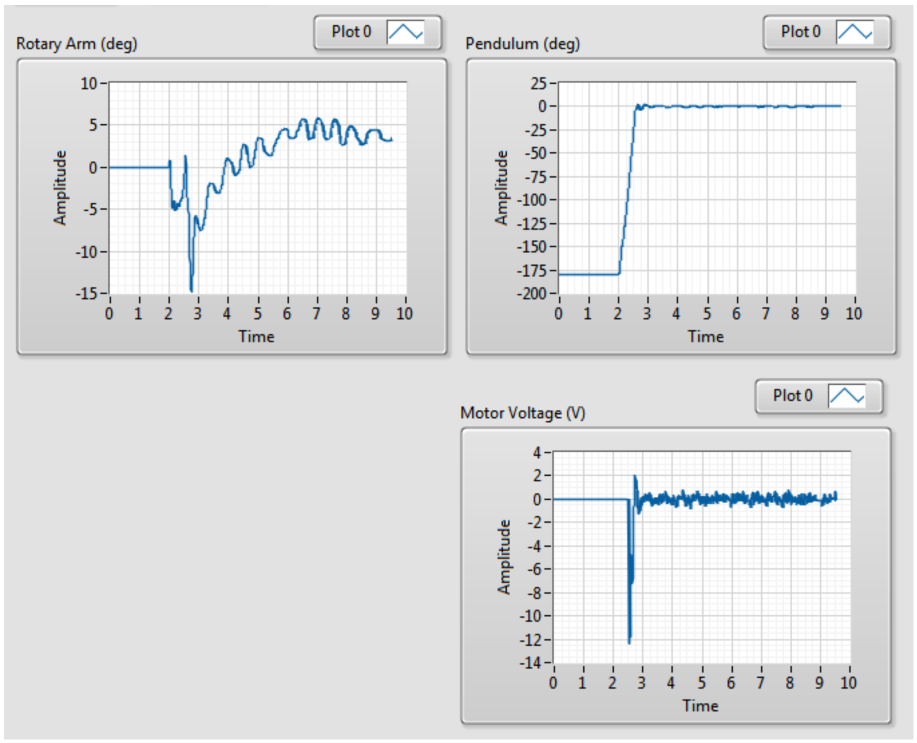


Figure 10.3: QUBE-Servo rotary pendulum response

6. As the pendulum is being balanced, describe the responses in the *Rotary Arm (deg)* and the *Pendulum Angle (deg)* scopes.

7. Vary 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 degrees to start.** What variable does this represent in the balance control?

8. Click on the *Stop* button to stop running the controller.

11 SWING-UP CONTROL

Topics Covered

- Energy control
- Nonlinear control
- Control switching logic

Prerequisites

Before starting this lab make sure:

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

11.1 Background

11.1.1 Energy Control

If the arm angle is kept constant and the pendulum is given an initial position it would swing with constant amplitude. Because of friction there will be damping in the oscillation. The purpose of energy control is to control the pendulum in such a way that the friction is constant. The potential energy of the pendulum is

$$E_p = M_p\, g\, l_p\, (1 - \cos\alpha)$$

and the kinetic energy is

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

The pendulum angle, α, and the lengths are illustrated in Figure 11.1. The moment of inertia in this case is $l_p = L_p/2$.

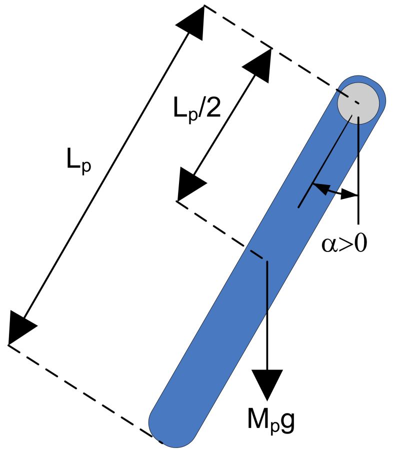


Figure 11.1: Free-body diagram of pendulum

The potential energy is zero when the pendulum is at rest at $\alpha = 0$ and equals $2M_p\, g\, l_p$ when the pendulum is upright at $\alpha = \pm\pi$. The sum of the potential and kinetic energy of the pendulum is

$$E = \frac{1}{2}J_p\dot{\alpha}^2 + M_p\, g\, l_p\,(1 - \cos\alpha)\,. \tag{11.1}$$

Differentiating 11.1 results in the differential equation

$$\dot{E} = \dot{\alpha}\left(J_p\ddot{\alpha}^2 + M_p\, g\, l_p \sin\alpha\right)\,. \tag{11.2}$$

Substituting the pendulum equation of motion

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

for pendulum acceleration into Equation 11.2 above gives

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

Since the acceleration of the pivot is proportional to current driving the arm motor and thus also proportional to the drive voltage we find that it is easy to control the energy of the pendulum. The proportional control law

$$u = (E_r - E)\,\dot{\alpha}\cos\alpha \tag{11.3}$$

drives the energy towards the reference energy E_r. Notice that the control law is nonlinear because the proportional gain depends on the pendulum angle, α. Also, notice that the control changes sign when $\dot{\alpha}$ changes sign and when the angle is $\pm$ 90 deg.

However, for 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

$$u = \text{sat}_{u_{max}}\left(\mu(E_r - E)\text{sign}(\dot{\alpha}\cos\alpha)\right) \tag{11.4}$$

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}.

11.1.2 Hybrid Swing-Up Control

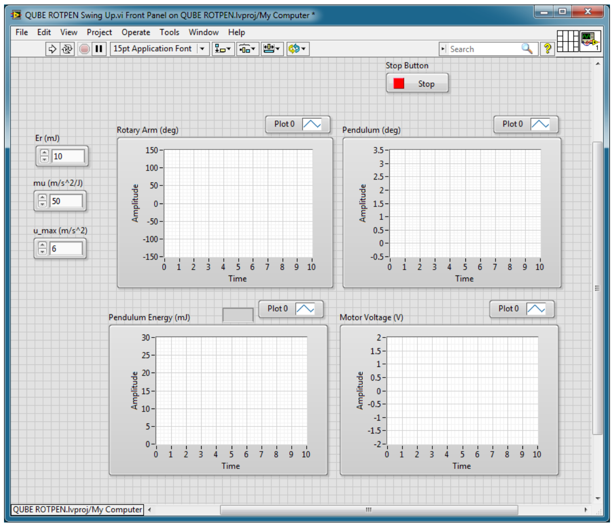
The energy swing-up control in 11.3 (or 11.4 can be combined with the balancing control law in 10.1 to obtain a control law which performs the dual tasks of swinging up the pendulum and balancing it.

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

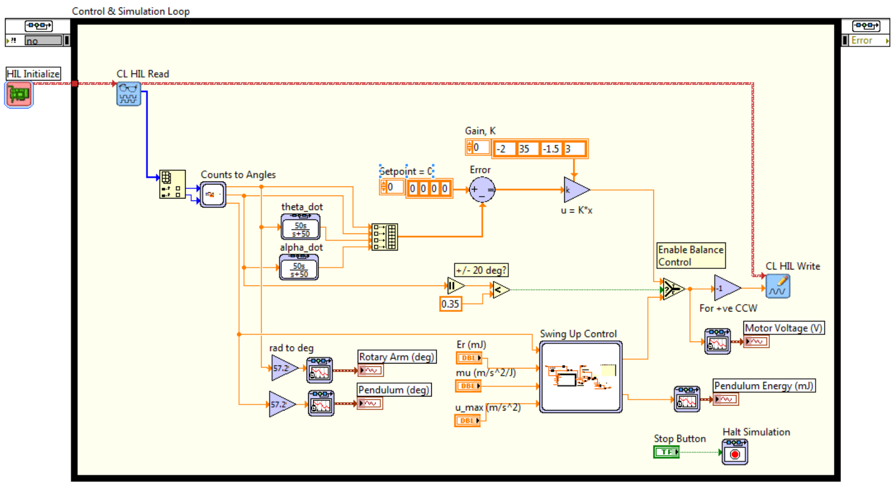
$$u = \begin{cases} u_{bal} & \text{if } |\alpha| - \pi \leq 20 \text{ deg} \\ u_{swing} & \text{otherwise} \end{cases}$$

11.2 In-Lab Exercises

The LabVIEW™ Virtual Instrument (VI) shown in Figure 11.2 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 11.1.



(a) Front Panel



(b) Block Diagram

Figure 11.2: VI that implements the swing-up controller

11.2.1 Energy Control

1. Open the QUBE-Servo ROTPEN Swing Up.vi in LabVIEW™ .
2. To turn the swing-up control off, set *mu* to 0.
3. Run the VI.
4. Manually rotate the pendulum at different levels and examine the pendulum angle and energy in the *Pendulum (deg)* and *Pendulum Energy (mJ)* scopes.
5. 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 11.1?

6. Click on the *Stop* button to bring the pendulum down to the initial, downward position.
7. Set the swing-up control parameters (i.e., the Constant and Gain blocks connected to the inputs of the *Swing-Up Control* subystem) to the following:
   - mu = 50 m/s^2/J
   - Er = 10.0 mJ
   - u_max = 6 m/s^2
8. If the pendulum is not moving, gently perturb the pendulum with your hand to get it going.
9. 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.
10. Fix *Er* to 20.0 mJ and vary the swing-up control gain *mu* between 20 and 60 m/s^2/J. Describe how this changes the performance of the energy control.
11. Click on the *Stop* button to stop running the controller.

11.2.2 Hybrid Swing-Up Control

1. Open the QUBE-Servo ROTPEN Swing Up.vi in LabVIEW™ .
2. To turn the swing-up control off, set *mu* to 0.
3. Run the VI.
4. Set the swing-up control parameters to the following:
   - mu = 20 m/s^2/J
   - u_max = 6 m/s^2
5. Based on your observations in the previous lab, Section 11.2.1, what should the reference energy be set to?
6. Make sure the pendulum is hanging down motionless and the encoder cable is not interfering with the pendulum.
7. Run the VI.
8. The pendulum should begin going back and forth. If not, manually perturb the pendulum with your hand. **Click on the Stop button if the pendulum goes unstable.**
9. 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.
10. Click on the *Stop* button to stop running the VI.
11. Power OFF the QUBE-Servo if no more experiments will be conducted.

12 OPTIMAL LQR CONTROL

Topics Covered

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

Prerequisites

Before starting this lab make sure:

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

12.1 Background

A rich collection of methods for finding parameters of control strategies have been developed. Several of them have also been packaged in tools that are relatively easy to use. Linear Quadratic Regulator (LQR) theory is a technique that is suitable for finding the parameters of the balancing controller in Equation 10.1 in Section 10. Given that the equations of motion of the system can be described in the form

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

the LQR algorithm computes a control task, u, to minimize the criterion

$$J = \int_0^\infty x(t)^T Q x(t) + u(t)^T R u(t) dt \tag{12.1}$$

The matrix Q defines the penalty on the state variable and the matrix R defines the penalty on the control actions. Thus when Q is made larger, the controller must work harder to minimize the cost function and the resulting control gain will be larger. In our case the state vector x is defined

$$x = \begin{bmatrix} \theta & \alpha & \dot{\theta} & \dot{\alpha} \end{bmatrix}^T \tag{12.2}$$

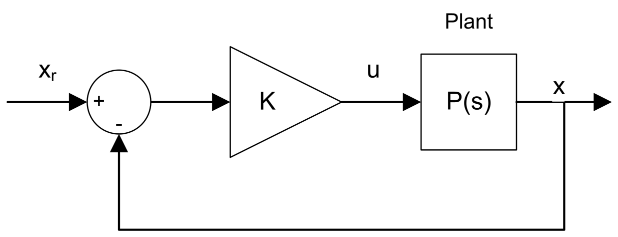


Figure 12.1: Block diagram of balance state-feedback control for rotary pendulum

Since there is only one control variable, R is a scalar and the control strategy used to minimize cost function J is given by

$$u = K(x_r - x) = k_{p,\theta}(\theta_r - \theta) - k_{p,\alpha}\alpha - k_{d,\theta}\dot{\theta} - k_{d,\alpha}\dot{\alpha}. \tag{12.3}$$

This strategy is know as state-feedback control and is illustrated in Figure 12.1. It is equivalent to the PD control explained in the *Balance Control* lab.

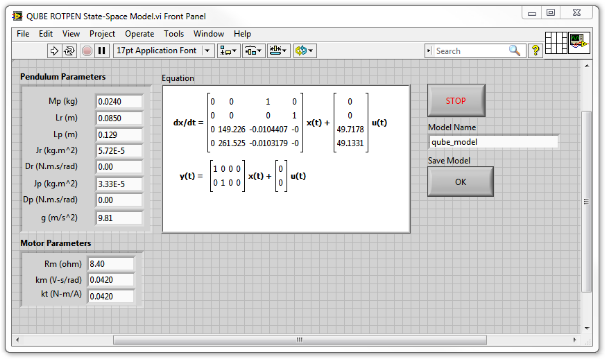
12.2 In-Lab Exercises

12.2.1 LQR Control Design

The LQR theory has been packaged in the LabVIEW™ *Control Design and Simulation Module*. Thus given a 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 in the *Control Design* Toolkit 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.

1. Run the *QUBE-Servo ROTPEN State-Space Model.vi* shown in Figure 12.2. This loads the QUBE-Servo rotary pendulum state-space model matrices A, B, C, and D and saves it to a file, e.g., *qube_rotpen*.


QUBE ROTPEN State-Space Model.vi Front Panel
File Edit View Project Operate Tools Window Help
17pt Application Font
Search
STOP
OK

Pendulum Parameters

Mp (kg)	0.0240
Lr (m)	0.0850
Lp (m)	0.129
Jr (kg.m^2)	5.72E-5
Dr (N.m.s/rad)	0.00
Jp (kg.m^2)	3.33E-5
Dp (N.m.s/rad)	0.00
g (m/s^2)	9.81

Motor Parameters

Rm (ohm)	8.40
km (V-s/rad)	0.0420
kt (N-m/A)	0.0420

Equation

$$dx/dt = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 149.226 & -0.0104407 & -0 \\ 0 & 261.525 & -0.0103179 & -0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 49.7178 \\ 49.1331 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(t)$$

Model Name: qube_model

Save Model

Figure 12.2: VI to generate QUBE-Servo ROTPEN state-space model

2. Design a VI similarly as shown in Figure 12.3 to find the open-loop poles of the system. Use the *Read Model from File* to load the model file you just saved (i.e., typically saved in the *Rotpen Model* folder). What do you notice about the location of the open-loop poles? How does that affect the system?

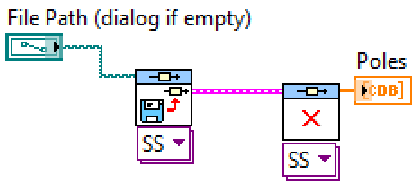
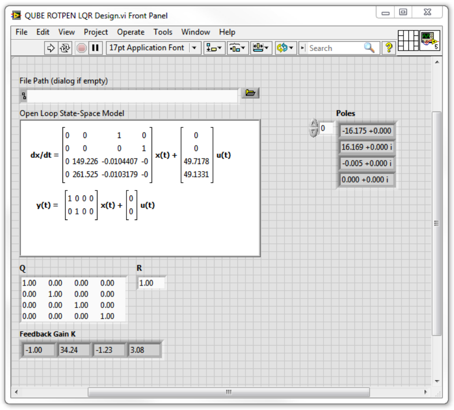


Figure 12.3: Find poles of loaded state-space model

3. Use the *Read Model from File* and *Linear Quadratic Regular* VIs as shown in Figure 12.4 to generate a control gain K. Generate K using the following weighting matrices:

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


QUBE ROTPEN LQR Design.vi Front Panel
File Edit View Project Operate Tools Window Help
17pt Application Font
Search

File Path (dialog if empty)

Open Loop State-Space Model

$$dx/dt = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 149.226 & -0.0104407 & -0 \\ 0 & 261.525 & -0.0103179 & -0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 49.7178 \\ 49.1331 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(t)$$

Poles

0

-16.175 +0.000
16.169 +0.000 i
-0.005 +0.000 i
0.000 +0.000 i

Q

1.00	0.00	0.00	0.00
0.00	1.00	0.00	0.00
0.00	0.00	1.00	0.00
0.00	0.00	0.00	1.00

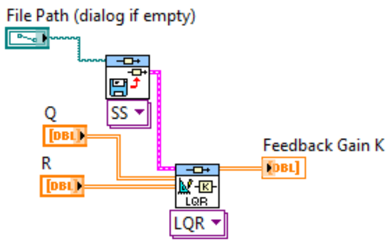
R

1.00

Feedback Gain K

-1.00	34.24	-1.23	3.08

(a) Front Panel



(b) Block Diagram

Figure 12.4: Generate LQR control gain using loaded state-space model

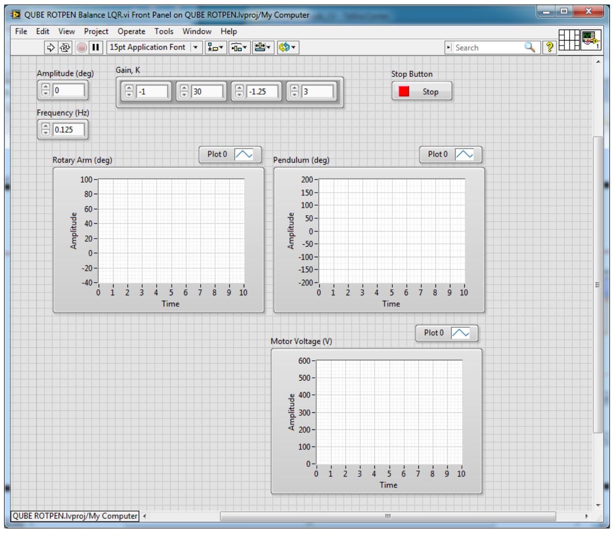
4. Change the LQR weighting matrix to the following and generate a new gain control gain:

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

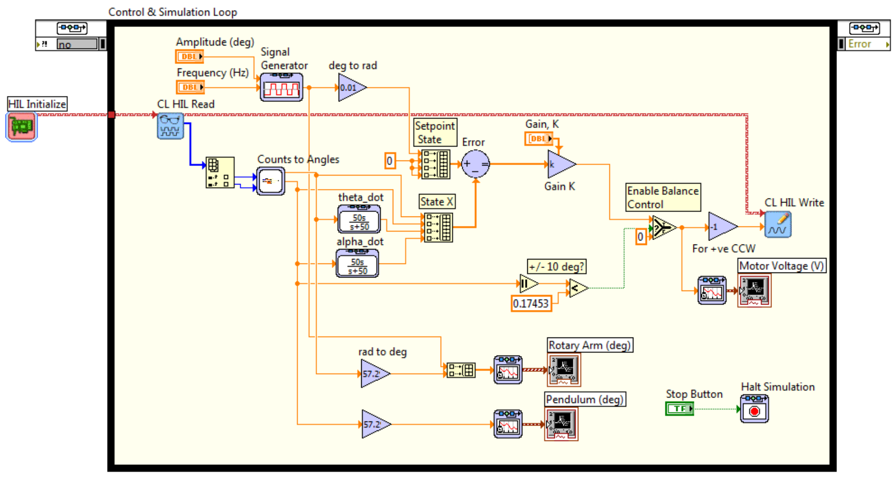
   Record the gain generated. How does changing q_{11} affect the generated control gain? Based on the description of LQR in Section 12.2.1, is this what you expected?

12.2.2 LQR-Based Balance Control

Construct a VI similarly as shown in Figure 12.5 that balances the pendulum on the QUBE-Servo Rotary Pendulum system using an adjustable feedback control gain K.



(a) Front Panel



(b) Block Diagram

Figure 12.5: VI to run optimized balance controller

1. Either open the *QUBE-Servo ROTPEN LQR Design.vi* or use the VI generated in Section 12.2.1 to generate the control gain K based on LQR and the QUBE-Servo model.
2. Using the VI you made in the *Balance Control* lab, construct the controller shown in Figure 12.5:
   - Using the angles from the *Counts to Angles* subsystem you designed in the *Balance Control* lab (which converts encoder counts to radians), build state x given in Equation 12.2. As shown in Figure 12.5 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 12.3. 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 θ_r. To generate a reference state x_r, make sure you include a *Build Array* VI to get $[\theta_r \quad 0 \quad 0 \quad 0]$.
3. Set K in the *QUBE-Servo ROTPEN Balance LQR.vi* to the gain that was generated in Step 3.
4. Set the Signal Generator block to the following:
   - Type = Square
   - Amplitude = 1

- Frequency = 0.125 Hz

5. Set the *Amplitude (deg)* control on the front panel to 0.
6. Run the VI.
7. Manually rotate the pendulum in the upright position until the controller engages.
8. Once the pendulum is balanced, set the *Amplitude (deg)* control to 30 to make the arm angle go between ± 30 deg. The scopes should read something similar as shown in Figure 10.3. Attach your response of the rotary arm, pendulum, and controller voltage.

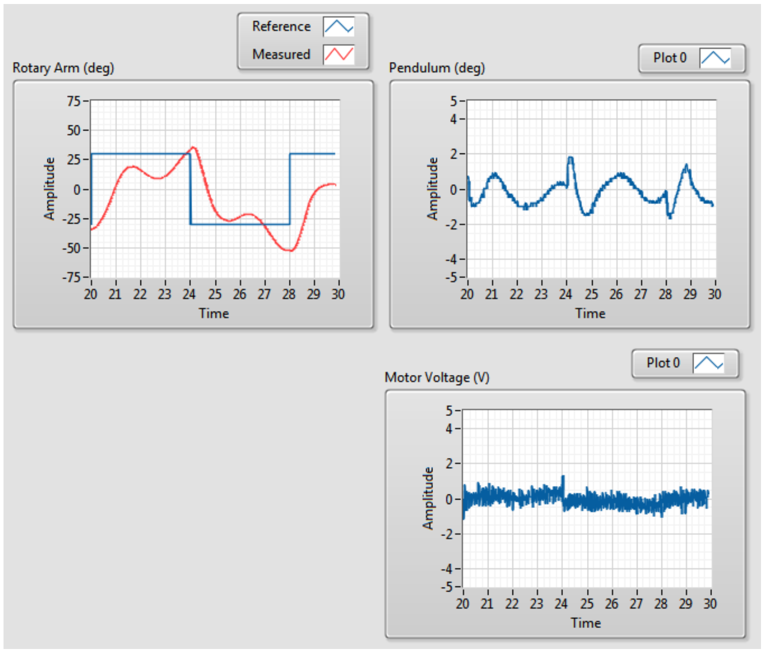


Figure 12.6: QUBE-Servo rotary pendulum response

9. Using the LQR design VI, generate the gain using $Q = \text{diag}([5 \quad 1 \quad 1 \quad 1])$ performed in Step 4 in Section 12.2.1. The *diag* term specifies the diagonal elements in a square matrix.
10. Enter the newly designed gain in the *QUBE-Servo ROTPEN Balance LQR.vi*.
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 (i.e., 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. Click on the *Stop* button to stop running the controller.
15. Power off the QUBE-Servo if no more experiments will be performed in this session.

13 SYSTEM REQUIREMENTS

Required Software

Make sure LabVIEW™ is installed with the following required add-ons:

1. LabVIEW™
2. NI-DAQmx
3. NI LabVIEW™ Control Design and Simulation Module
4. NI LabVIEW™ MathScript RT Module
5. Quanser Rapid Control Prototyping Toolkit®

Note: Make sure the Quanser Rapid Control Prototyping (RCP) Toolkit is installed after LabVIEW. See the RCP Toolkit Quick Start Guide for more information.

Required Hardware

- QUBE-Servo Rotary Servo Experiment
- Inertial wheel attachment
- Rotary pendulum (ROTPEN) attachment
- **QUBE-Servo Direct I/O users**: Data acquisition (DAQ) device that is compatible with Quanser Rapid Control Prototyping Toolkit®. This includes Quanser devices (e.g., Q2-USB) and some National Instruments DAQ devices (e.g., NI USB-6251, NI PCIe-6259). For a full listing of compliant DAQ cards, see the *Data Acquisition Card Support* in the Rapid Control Prototyping (RCP) Toolkit help page.

Before Starting Lab

Before you begin this laboratory make sure:

- LabVIEW™ is installed on your PC.
- *QUBE-Servo Direct I/O users*: DAQ device has been successfully tested (e.g., using the test procedure outlined in the Quick Start Guide or the *Analog Loopback Demo*).
- QUBE-Servo Rotary Servo Experiment is connected as described in the QUBE-Servo User Manual ([1]).

13.1 Overview of Files

File Name	Description
QUBE-Servo User Manual.pdf	This manual describes the hardware of the Quanser QUBE-Servo Rotary Servo Experiment system and explains how to setup and wire the system for the experiments.
QUBE-Servo Workbook (Student).pdf	This manual contains pre-lab questions and lab experiments for the Quanser QUBE-Servo Rotary Servo Experiment plant using LabVIEW™ .
QUBE-Servo ROTPEN Swing Up.vi	Run this LabVIEW™ Virtual Instrument (VI) to run the swing-up controller on the QUBE-Servo Rotary Pendulum system.
QUBE-Servo ROTPEN State-Space Model.vi	Run this to load the state-space model of the QUBE-Servo Rotary Pendulum system and save it to a file (so it can be loaded by another VI).

Table 13.1: Files supplied with the Quanser QUBE-Servo Rotary Servo Experiment.

13.2 Setup for Pendulum Swing-Up

Before performing the Swing-Up Control in-lab exercises in Section 11.2, follow these steps to get the system ready for this lab:

1. Setup the QUBE-Servo module as detailed in the QUBE-Servo User Manual ([1]).
2. Make sure the QUBE-Servo DVD files have been copied to your local hard drive.
3. Load LabVIEW™ .
4. Browse through the *Current Directory* window in LabVIEW and find the folder that contains the *QUBE-Servo ROTPEN Swing-Up.vi*.
5. **QUBE-Servo Direct I/O users:** If you are using an external DAQ device to interface to the QUBE-Servo, then make sure the *Board type* in the HIL Initialize VI is configured for the DAQ device that is installed in your system.
6. To run the VI, click on the white arrow in the top-left corner of the front panel.

REFERENCES

[1] Quanser Inc. *QUBE-Servo User Manual*, 2012.

[2] Norman S. Nise. *Control Systems Engineering*. John Wiley & Sons, Inc., 2008.

QUANSER
INNOVATE. EDUCATE.

niversity of Toronto • Monash University • Kyoto U
niversity of Manchester • California Institute of Tec
olytechnic School of Lausanne • Hong Kong Univers
niversity of Waterloo • Carnegie Mellon University
niversity of Melbourne • ETH Zurich • Yale Univers
niversity of Houston • KAIST • Karlsruhe University
f Alberta • Gifu University • Loughborough Universi
niversity of California, Berkeley • KTH • McMaster
niversity Munich • Rice University • Kyoto Institute
niversity of Auckland • MIT • Imperial College Lond
he Chinese University of Hong Kong • Virginia Tech
f Cincinnati • McGill University • Australian National
niversity of Bristol • Purdue University • Osaka Uni
ing Soud University • I.I.T Kharagpur •Memorial Uni
niversity of British Columbia • Delft University of Te
niversity of Texas at Austin • Beijing Institute of Te
f Tokyo • Princeton University • Hebei University of
niversity of Wisconsin-Madison • Holon Institute of
of Klagenfurt • Harvard University • Tokyo Institute
niversity of Reading • Tsinghua University • Cornell
niversity of Michigan • Korea University • Queen's
niversity of Stuttgart • Georgia Tech • Ben-Gurion
niversity Eindhoven • Ajou University • Kobe Uni
niversity of Maryland College Park • Nanyang Tech
niversity of New South Wales • Washington Univers
ational University of Singapore • Harbin Institute of
niversity of Victoria • Boston University • Donghua
orthwestern University • Tongji University • Royal
niversity of Quebec • Clemson University • Fukuoka
delaide University • University of Barcelona • SUNY
ueen's University Belfast •Istanbul Technical Unive
e Los Andes • Louisiana Tech • Norwegian Universi
nited States Military Academy • CINVESTAV • Drex

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 course materials** 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: