



Lab 1

Classical PID control experiences with the QUBE-Servo 2



(a) QUBE-Servo 2 with Inertia Disc Module



(b) QUBE-Servo 2 with Pendulum Module

Hugues Garnier

September 2023

Contents

| | |
|--|-----------|
| Steps in the design of a control system | 2 |
| 1 QUBE-Servo 2 software and hardware interfacing | 4 |
| 1.1 The QUBE-Servo 2 from Quanser | 4 |
| 1.2 Hardware and software interfacing | 5 |
| 1.2.1 Background | 5 |
| 1.2.2 First experiments with the QUBE-Servo 2 | 7 |
| 1.3 Estimating angular velocity by high-pass filtering | 10 |
| 1.3.1 Low-pass and high-pass filtering | 10 |
| 1.3.2 Second experiment with the QUBE-Servo 2 | 11 |
| 2 PID-based control of the DC motor angular position | 14 |
| 2.1 Download of the files required for the lab | 14 |
| 2.2 Pre-lab questions | 15 |
| 2.3 Control performance requirements | 15 |
| 2.4 Transfer function model identification from step response experiment | 15 |
| 2.4.1 Recording of the step response experiment | 16 |
| 2.4.2 Transfer function model identification and validation | 17 |
| 2.5 Servo-motor control using PD feedback in simulation | 18 |
| 2.6 PD control implementation to the QUBE-Servo 2 | 18 |
| 2.7 Robustness test of the servo-motor PD control | 19 |

Acknowledgements

The contents of this lab has been largely inspired and adapted from initial versions of the different chapters provided by Quanser Inc¹. This is fully acknowledged.

¹www.quanser.com

Steps in the design of a control system

The various stages that lead to the design of a feedback control are detailed in Figure 1.

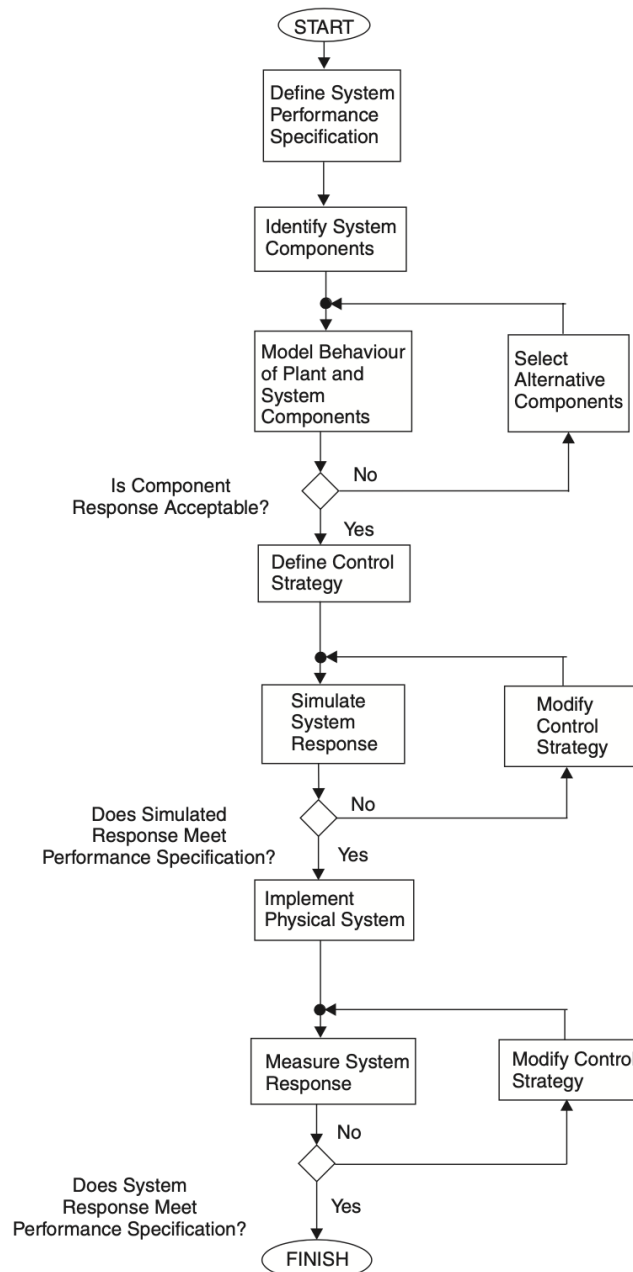


Figure 1: Steps in the design of a control system (From *S. Burns, Advanced Control Engineering, Butterworth-Heinemann, 2001*)

The methodology will be used throughout the different labs and you should always come back to it if your design is not successful. The determination of a model and the simulation of the behaviour of the closed-loop system are crucial steps for the design of a feedback control system. You will use the Matlab/Simulink environment to design, develop, and validate a variety of feedback control strategies before their implementation on the physical systems.

Part 1

QUBE-Servo 2 software and hardware interfacing

1.1 The QUBE-Servo 2 from Quanser

The Quanser QUBE-Servo 2, pictured in Figure 1.1, is a compact rotary servo system that can be used to perform a variety of classic servo control and inverted pendulum based experiments. The QUBE can be controlled by a computer via USB connection.

The system is driven using a direct-drive 18V brushed DC motor. The motor is powered by a built-in Pulse Width Modulation (PWM) amplifier with integrated current sense. Two add-on modules are supplied with the system:

- an inertia disc;
- a rotary pendulum.

The two modules can be easily attached or interchanged using magnets mounted on the QUBE-Servo 2 module connector.

Single-ended rotary encoders are used to measure the angular position of the DC motor and pendulum, while the angular velocity of the motor can be either estimated from the angular position encoder-based measurement or directly measured using an integrated software-based tachometer.



(a) QUBE-Servo 2 with Inertia Disc Module

(b) QUBE-Servo 2 with Pendulum Module

Figure 1.1: QUBE-Servo 2 with the two different modules

Download of the QUBE-Servo 2 user manual and quick start guides

Download the zipped file `QUBE-Servo2_pdf_guides.zip` from the course website and unzip it in the local disk folder `C:/temp/`.

1.2 Hardware and software interfacing

Topics covered

- Getting familiarized with the Quanser QUBE-Servo 2 hardware (sensor and actuator).
- Using Quanser QUARC software to interact with the QUBE-Servo 2 system.
- Sensor calibration.

Prerequisites

- Inertia disc load is on the QUBE-Servo 2.
- The QUBE-Servo 2 has been setup. See the QUBE-Servo 2 Quick Start Guide for details.
- You have the QUBE-Servo 2 User Manual. It can be useful for some of the experiments.
- You are familiar with the basics of Matlab and Simulink.

1.2.1 Background

1.2.1.1 QUARC software

The QUARC software is used to interact with the hardware of the QUBE-Servo 2 system. QUARC will be used to drive the DC motor and read the angular position of the inertia disc. The basic steps to create a Simulink model with QUARC in order to interact with the QUBE-Servo 2 hardware are:

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

If you want to know more about QUARC:

- type `doc quarc` in Matlab to access QUARC documentation and demos.
- watch a 18 mn video *Getting Started with QUARC* from Quanser (follow the link given below) that demonstrates how to interface with hardware, build a simple Simulink model and implement it on an actual system:
www.quanser.com/tutorial/quarc-essentials-hardware-interfacing/

1.2.1.2 The actuator: the DC motor

Direct-current motors are used in a variety of applications. As discussed in the QUBE-Servo 2 User Manual, the QUBE-Servo 2 has a brushed DC motor that is connected to a Pulse-Width Modulation (PWM) amplifier. See the QUBE-Servo 2 User Manual for details.

1.2.1.3 The sensors: the 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 1.2. Unlike potentiometers, encoders are relative. The encoder count is reset to 0 every time it is powered. 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 1.2: Digital incremental rotary optical shaft encoder

The encoder has a coded disc that is marked with a radial pattern. This disc 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 1.3. An index pulse is triggered once for every full rotation of the disc, which can be used for calibration or homing a system.

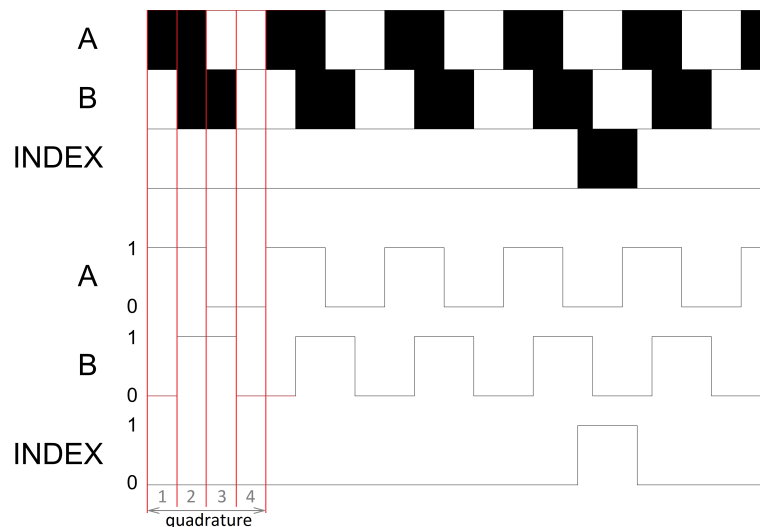


Figure 1.3: Optical incremental encoder signals

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 disc and the decoder. For example, a single encoder with 512 lines on the disc can generate a total of 512 counts for every rotation of the encoder shaft. However, in a quadrature decoder as depicted in Figure 1.3, 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 are 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.

1.2.2 First experiments with the QUBE-Servo 2

The objective here is to build a Simulink model using QUARC blocks to drive the DC motor and then measure its corresponding angle, as shown in Figure 1.4.

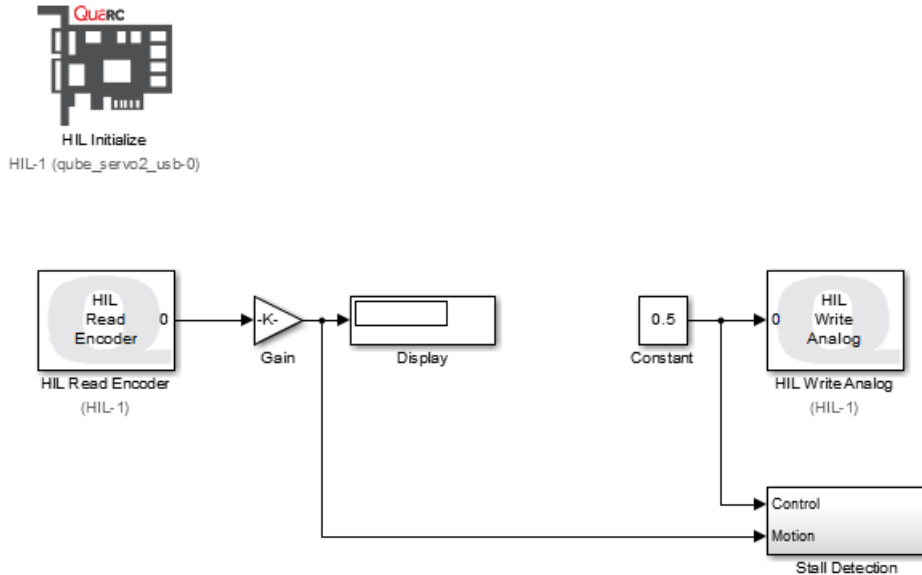


Figure 1.4: Simulink model used with QUARC to drive the DC motor and read angle on QUBE-Servo 2

1.2.2.1 Configuring a Simulink model for the QUBE-Servo 2

Follow the steps below to build a Simulink model that will interface to the QUBE-Servo 2 hardware using QUARC:

1. Open the the Quick Start Guide and follows the instruction to set up and connect the QUBE-Servo 2 to your PC USB port.
2. Open Matlab and then start Simulink by typing `Simulink` in the Command window or by clicking on its icon in the menu bar.
3. Make sure the QUBE-Servo 2 is connected to your PC USB port and the Power LED at the back of the QUBE-Servo 2 is lit green.
4. Create a new blank Simulink model (do not create a new blank QUARC model!) model by clicking on the icon in the Simulink Start page window or by going to `File | New | Simulink Model` item in the menu bar.
5. Go to `SIMULATION` item in the menu bar | Open the Library Browser window by clicking on its icon.
6. Expand the `QUARC Targets` item and go to the `Data Acquisition | Generic | Configuration` folder, as shown in Figure 1.5.

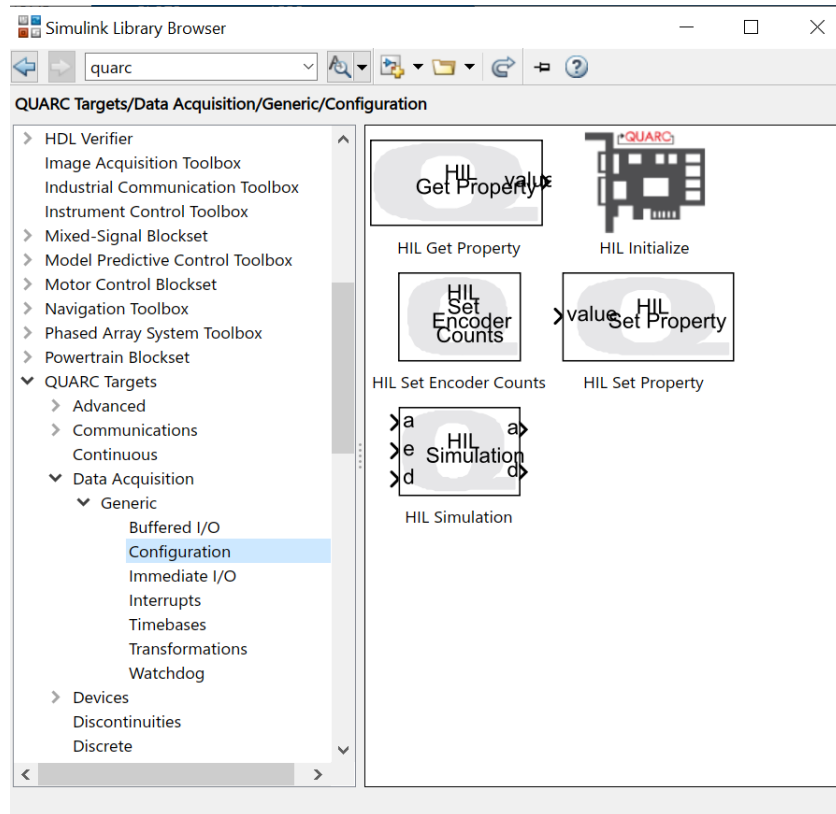



Figure 1.5: QUARC Targets in the Library Browser

7. Click-and-drag the HIL Initialize block from the library window into the blank model. This block is used to configure the data acquisition board.
8. Double-click on the HIL Initialize block.
9. In the Board type field, select qube_servo2_usb. Apply and close the window.
10. Go to QUARC item in the menu bar | Set default options item to set the correct Real-Time Workshop parameters and setup the model for external use (as opposed to the simulation mode).
11. Go to QUARC item in the menu bar | QUARC targets item to set the correct quarc_win64.tlc for targets running on 64-bits Windows.
12. Save the file as QUBE_iotest.slx in the following folder C:/temp/Lab_QUBE_Test.
13. Important ! Go to the Matlab window. By clicking on the browse for folder icon , **change the current folder of Matlab so that it becomes your C:/temp/Lab_QUBE_Test folder** that contains the file you have just saved.
14. Go back to Simulink. Select the HARDWARE | Monitor & Build item to build the code. Various lines in the Diagnosis Viewer Window (open the window) should be displayed as the model is being compiled. This creates a QUARC executable (.exe) file which we will commonly refer to as the **QUARC controller**.
If the experimental procedure was followed correctly, no errors should be obtained in when running the QUARC controller. Time should flow in Simulink and the LED strip at the top of the QUBE-Servo 2 should turn from red to green.

Great, you have successfully set up the connection to the QUBE-Servo 2. Congratulations !

15. 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**).

1.2.2.2 Reading the Encoder

Follow the steps below to read the encoder:

1. Using the Simulink model you configured in the previous section, add the **HIL Read Encoder** block by double clicking in the Simulink model and search for **HIL Read Encoder**. This block can also be found in the Library Browser from the **QUARC Targets | Data Acquisition | Generic | Immediate I/O** category.
2. Connect the **HIL Read Encoder** to a **Gain and Display** block similar to Figure 1.4 (without the **HIL Write Analog** block. It will added later). Double click in the Simulink model and search for the **Gain and Display** blocks. In the Library Browser, you can find the **Display** block from the **Simulink | Sinks** and the **Gain** block from **Simulink | Math Operations**.
3. Select the **HARDWARE | Monitor & Build** item to build the code. The code needs to be re-generated again because you have modified the Simulink model. No errors should be obtained and the QUARC controller should be running.
4. Rotate the disc by hand, back and forth. The **Display** block should show the number of counts measured by the encoder. Remind that the encoder counts are proportional to the angle (angular position) of disc.
5. Stop the controller, rotate the disc by 90 degrees clockwise, and re-start the controller. What do you notice about the encoder measurement when the controller is re-started? As the encoder is relative, the encoder count is reset to 0 every time the controller is ran.
6. Measure how many counts the encoder outputs for a full rotation. To do so, stop the controller and move the disc to the 0 degree position marked on the QUBE-Servo 2. Start the controller and rotate the disc one full rotation. The encoder count should read approximately 2048, which is in-line with the specifications given in the background section.
7. Now we want to display the disc angle in degrees, not in counts. Given that there is 2048 counts per revolution, to get a measurement in degrees we need a gain of $360/2048 = 0.1748$ deg/count. Set the **Gain** block to the value that converts counts to degrees. This is called the **sensor gain**. To confirm that the sensor gain is correct, start the controller with the disc at the 0 degree position marked on the QUBE-Servo 2. Rotate it one full rotation, and verify that the **Display** block reads 360.
8. Ultimately we want to display the disc angle in radians, not in counts. Given that there is 2048 counts per revolution, to get a measurement in radians we need a gain of $2\pi/2048 = 0.0031$ rad/count. Set the **Gain** block to the value that converts counts to radians. To confirm that the new sensor gain is correct, start the controller with the disc at the 0 position marked on the QUBE-Servo 2. Rotate it one full rotation, and verify that the **Display** block reads $2\pi \approx 6.28$.

1.2.2.3 Driving the DC motor

1. Add the **HIL Write Analog** block. 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 a **Constant** block. Connect the **Constant** and **HIL Write Analog** blocks together, as shown in Figure 1.4.
3. Set the **Constant** block to 0.5. This applies 0.5 V to the DC motor of the QUBE-Servo 2.
4. Click on **Monitor & tune** to build and run the QUARC controller.
5. Confirm that you 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 (clockwise or counter-clockwise) when a positive input is applied?
6. Keep the Controller running and modify the **Constant** block to -0.5 . Verify that the disc rotates in the counter-clockwise direction.
7. Stop the QUARC controller.

1.3 Estimating angular velocity by high-pass filtering

Topics covered

- Using an encoder to measure angular velocity.
- Low-pass and high-pass filters.

1.3.1 Low-pass and high-pass filtering

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

$$H_{LP}(s) = \frac{\omega_f}{s + \omega_f} \quad (1.1)$$

where s is the Laplace transform and ω_f is the cut-off frequency of the filter in (rad/s).

All higher frequency components of the signal will be attenuated by at least -3 dB ($\approx 70\%$ by amplitude).

A high-pass filter can be used to approximate the time-derivative of a signal. The Laplace transfer function of a first-order high-pass analog filter has the form

$$H_{HP}(s) = \frac{\hat{\Omega}(s)}{\Theta(s)} = \frac{\omega_f s}{s + \omega_f} \quad (1.2)$$

Note that the high-pass filter can be seen as the cascade of a low-pass filter and the pure derivate as shown in Figure 1.6.

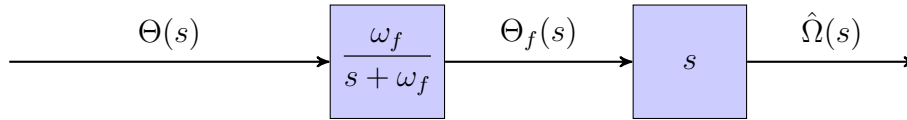


Figure 1.6: High-pass filter seen as a cascaded low-pass and derivative blocks

This high-pass filter can be very useful to provide an estimate of the angular velocity $\hat{\theta}(t) = \hat{\omega}(t)$ from a measured angular position $\theta(t)$.

1.3.2 Second experiment with the QUBE-Servo 2

Based on the Simulink model developed in the previous section, the goal is now to build a Simulink model that estimates the angular velocity of the motor shaft using the angular position measure provided by the encoder as shown in Figure 1.7.

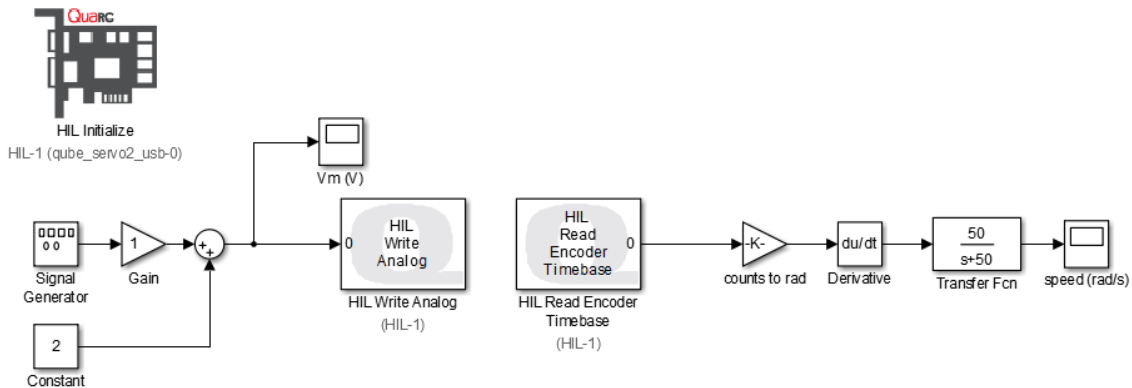
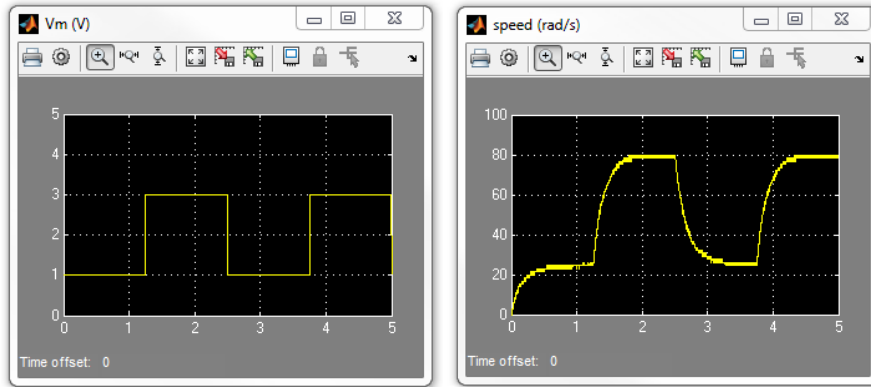


Figure 1.7: Estimating angular speed from the angular position measure provided by the encoder

1. Open the model you developed in the previous section. Change the encoder calibration gain to measure the angular position in radians (instead of degrees). Change the HIL Read Encoder by an HIL Read Encoder Timebase.
2. Build the Simulink model as shown in Figure 1.7. Add a Derivative block to the encoder calibration gain output to estimate the angular speed using the encoder (in rad/s). Connect the output of the Derivative to a Scope. For now, do not include the low-pass Transfer Fcn filter block $\frac{50}{s+50}$, it will be added later.
3. Setup the source blocks (Signal generator + Constant) to output a square wave voltage that goes from 1 V to 3 V at 0.4 Hz.
4. Run the QUARC controller. Examine the angular speed response. It should look similar to Figure 1.8.



(a) Motor Voltage (b) Encoder Speed
Figure 1.8: Measured servo speed using encoder

5. Explain why the encoder-based measurement is noisy. Plot the encoder angular position measurement using a new Scope. Zoom up on the position response. Is the signal continuous or piecewise constant?

The angular position measurement is therefore not continuous. The signal is piecewise constant that is, segmented into small steps. Remember that this later signal enters derivative. Differentiating these small steps result in large values in the response, as shown in Figure 1.9.

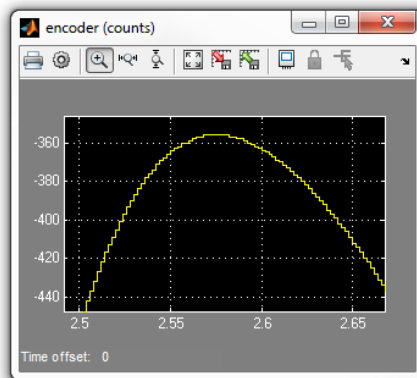


Figure 1.9: Encoder measurement

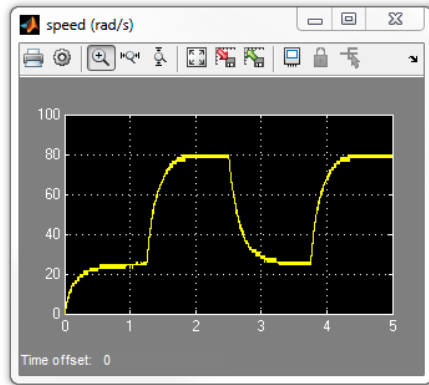
6. One way to remove some of the high-frequency components is adding a low-pass filter (LPF) to the derivative output. Add a Transfer Fcn block after the Derivative output and connect LPF to the Scope. Set the Transfer Fcn block to $50/(s + 50)$, as illustrated in Figure 1.7.

Note that the cascade of the derivative component $\frac{du}{dt}$ and the low-pass filter $\frac{50}{s + 50}$ can be implemented in one single high-pass transfer function block $\frac{50s}{s + 50}$.

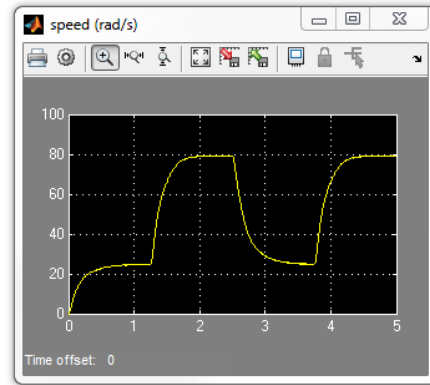
7. Run the QUARC controller. Look at the filtered encoder-based speed response and the motor voltage.

If the filter was applied properly, the filtered encoder-based speed measurement should be as shown in Figure 1.10. The filtered response in Figure 1.10b is a lot less noisy than

in Figure 1.10a. Zoom up if you want to see better the noise level in Figure 1.10a.



(a) Encoder speed without low-pass filter



(b) Encoder speed with low-pass filter

Figure 1.10: Estimated servo speed from encoder with and without low-pass filter

8. What is the cutoff frequency of the low-pass filter $50/(s + 50)$? Give your 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.

Lowering the cutoff removes more noise from the signal but causes it to slow down. Having a higher cutoff allows for more high-frequency components (noise), but the signal has less delay.

10. Stop the QUARC controller.

Part 2

PID-based control of the DC motor angular position

One of the most common tasks that control engineers are called upon to perform when creating industrial systems is to control the angular position of a DC motor. From automation in manufacturing to autonomous robots and even in aerospace, DC motors are used to actuate systems, and their (angular) position needs to be controlled to perform within specific design criteria.

During this part, we will mainly use the QUBE-Servo 2 with the inertia disk. The rotary pendulum will serve as a possible load disturbance to test the robustness of the implemented control only.

The objective is to illustrate the various stages of design that lead to the implementation of a PD control of the DC motor angular position. The stages are mainly:

1. to identify a linear low-order model from real data coming from a step test;
2. to design and tune a PD controller based on the identified linear model and evaluate its control performance in simulation by using Simulink;
3. to implement and refine the designed PD control on the physical servo-motor.
4. to test the robustness/sensitivity of the control to external load disturbance.

2.1 Download of the files required for the lab

1. Download the zipped file `Lab_QUBE_DCmotor.zip` from the course website. Save and unzip it in the local disk folder `C:/temp/`.
2. Start Matlab.
3. In the Current Folder window of Matlab, click right on the folder `C:/temp/Lab_QUBE_DCmotor` and select `add to path the selected folder`.
4. Double-click on the folder `Lab_QUBE_DCmotor` so that it becomes your current folder. You should see the different `.slx` and `m` files needed for this Lab.

2.2 Pre-lab questions

The pre-lab questions of this part are related to the problem solving session. The solutions to all questions will possibly require to be slightly adapted to the experimental conditions (mainly due to a different identified model and/or the presence of non-linear dry friction effects in the motor shaft).

It is assumed you have a full understanding of these solutions, and have a clear idea of the tasks that will have to perform during the lab.

2.3 Control performance requirements

The performance requirements for the angular position control are described in Table 2.1.

| Requirement | Assessment criteria | Level |
|----------------------|----------------------------|------------------------------|
| Control the position | Position setpoint tracking | No steady-state error |
| | Motor input voltage | limited to [-10V ; +10 V] |
| | Peak Overshoot | $D_1 = 4.3 \%$ |
| | Settling time at 5 % | $T_s^{5\%} = 0.05 \text{ s}$ |
| | Disturbance rejection | Rejection of load effects |

Table 2.1: Performance requirements for angular position control

2.4 Transfer function model identification from step response experiment

The determination of a model is the first crucial step for the design of a feedback control system.

The input and output of the DC motor of the QUBE with the inertia disk are:

- input: voltage of the motor $u(t)$ in V;
- output: inertia disk angular position $\theta(t)$ in rad.

The motor voltage-to-angular position transfer function takes the form of a first-order plus pure integrator model

$$\frac{\Theta(s)}{U(s)} = \frac{K}{s(1 + Ts)} \quad (2.1)$$

where $\Theta(s) = \mathcal{L}[\theta(t)]$ and $U(s) = \mathcal{L}[u(t)]$. K in rad/(V-s) is the model steady-state gain, T in s is the model time-constant.

As the angular velocity is the time-derivative of the angular position, both variables are linked in the Laplace domain by an integrator (or derivator) so that (2.1) can be expressed as:

$$\frac{\Theta(s)}{U(s)} = \frac{\Theta(s)}{\Omega(s)} \times \frac{\Omega(s)}{U(s)} = \frac{1}{s} \times \frac{K}{1 + Ts} \quad (2.2)$$

where $\Omega(s) = \mathcal{L}[\omega(t)]$ is the angular velocity (or speed) of the inertia disc.

Identifying a system having a pure integrator is tricky and it is better when the measure is available to identify the response between the motor angular velocity and the input voltage since the model takes the form of a simple first-order system.

The QUBE-Servo 2 motor voltage-to-angular velocity transfer function has therefore the well-known first-order form which parameters can be easily estimated from a step response experiment:

$$\frac{\Omega(s)}{U(s)} = \frac{K}{1 + T_s s}, \quad (2.3)$$

2.4.1 Recording of the step response experiment

1. Open the file *Step_resp_Qube.mdl* in Simulink.
2. Click on **Monitor & Tune** in the Hardware panel to run the step test that lasts 5 seconds only. The red color of the Qube should turn green if the test succeeds.
3. Observe the angular velocity response for a positive step from 0 to 2 V sent after 1s followed by a negative step from 2 V to 0 on the motor voltage after 3 seconds. The amplitude of the steps can be modified according to your own reasoning and choice. To do so, click on the two step blocks and modify the value.

The angular position (in rad) and velocity (in rad/s) responses to the positive and negative step input sent to the motor voltage have been recorded and saved in the `data_step_Qube.mat` file. They are plotted in Figure 2.1, which can be reproduced with Matlab by executing the `step_response_plot.m` file.

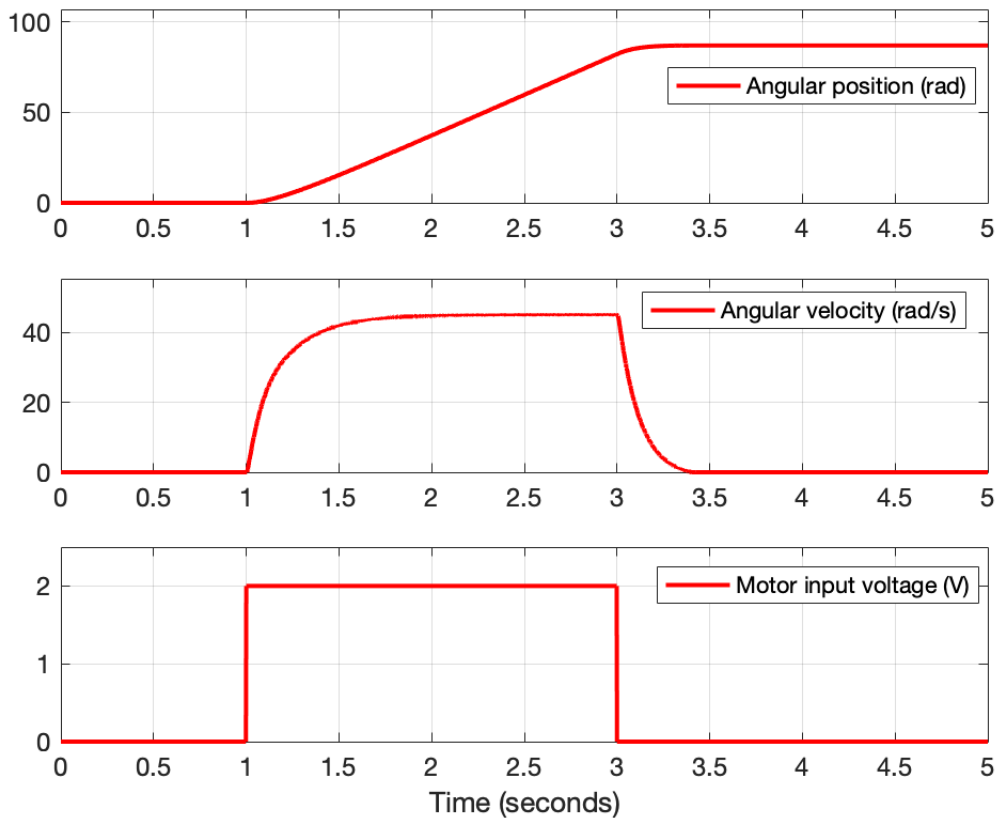


Figure 2.1: Angular position and velocity responses to a positive and negative step input sent to the motor voltage

2.4.2 Transfer function model identification and validation

It is possible to use very basic methods to determine the parameters of a first-order Laplace transfer function model from the positive step response for example. As we saw last year, there now exist more advanced model learning/identification methods available in Matlab toolboxes like the CONTSID toolbox developed by the research team of CRAN hosted at Polytech Nancy which determine the parameters of a (continuous-time) Laplace transfer function model directly from measured input/output data. We will make use of the CONTSID algorithms here.

To use these powerful data-driven high-fidelity model learning algorithms, follows the steps :

- Download the CONTSID toolbox from its website¹ to the local folder of your PC.
- Add the CONTSID folder to the Matlab path. If this is not possible, add all the files of CONTSID folder in your working directory.
- Run the file `ident_via_contsid.m` in Matlab (if you use your own laptop, the Matlab System Identification and Control toolboxes must be installed for the CONTSID to work).
- Compare the measured and model angular velocity responses.
- Note the numerical values of the estimated model steady-state gain K and time-constant T .

¹www.cran.univ-lorraine.fr/contsid

2.5 Servo-motor control using PD feedback in simulation

A variation of the classic PD control will be used as shown in Figure 2.2. Unlike the standard PD where the derivative action is applied to the error, it is applied to the output and a low-pass filter will be used in cascade with the derivative term to suppress measurement noise (the low-pass filter is not represented below but is included in the Simulink file).

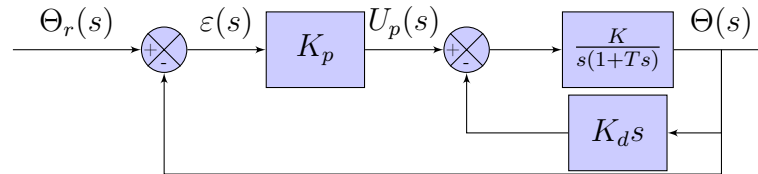



Figure 2.2: Block-diagram of the PD feedback configuration of the positional servo system with derivative effect on the output.

1. Determine the natural frequency ω_n and damping ratio z requirements that translate from the percent overshoot and settling time specifications of 4.3 % and 0.05s respectively.
2. Determine the values for K_p and K_d that make the closed-loop transfer function step response to have the required percent overshoot and settling time. Adapt the values obtained in the problem solving session to your identified model.
3. Open the file `simul_PD_control.mdl` in Simulink.
4. Enter the parameter values of your first-order model by clicking on the blue *Qube model* block.
5. Click on the proportional gain block and set your value for K_p .
6. Click on the derivative gain block and set your value for K_d .
7. Run the simulation and observe the angular position response in servo control as well as the DC-motor input voltage. Determine the maximum and minimum values of the input voltage.
8. Are the requirement specifications met?
9. If necessary, refine the value of K_p and K_d to get the best answer to the specification requirements (overshoot, steady-state error, settling time à 5% and amplitude of the motor voltage) with this PD feedback control.
10. By clicking on the  icon, measure the peak-time from the response.
11. Close the file.

2.6 PD control implementation to the QUBE-Servo 2

We will now proceed with the implementation and test of the PD control on the QUBE-Servo 2.

1. Open the file `PD_control_Qube.mdl` in Simulink.

2. Enter the numerical value of the two gains K_p and K_d of your best PD controller obtained in simulation.
3. Click on **Monitor & Tune** in the Hardware panel to run the controller.
4. Observe the angular position response to a square reference along with the control signal.
5. Are the experimental response close to the simulation results?
6. Are the requirement specifications fulfilled ? If not, modify the PD controller gains K_p and K_d to get the best answer to the specification requirements (overshoot, steady-state error and amplitude of the motor voltage).
7. Determine the peak-time.
8. Set the setpoint to a sine wave instead of the square wave and repeat the previous experiment. Can the PD controller track the sine wave reference?
9. Stop the QUARC controller.

2.7 Robustness test of the servo-motor PD control

1. Set back the setpoint to a square wave. Place an object (your smartphone for example) on the inertia disk and repeat the experiment. Are the requirement specifications fulfilled ? If not, modify the PD controller gains K_p and K_d to get the best answer to the specification requirements.
2. Replace the inertia disc by the rotary pendulum and repeat the previous experiment for a square wave setpoint.
3. Observe the angular position response to the square reference along with the control signal.
4. Stop the QUARC controller.