Lab 3: Quanser Hardware and Proportional Control

"The worst wheel of the cart makes the most noise." – Benjamin Franklin

1 Objectives

The goal of this lab is to:

1. familiarize you with Quanser's QuaRC tools and the Q4/Q2-usb Data Acquisition (DAQ) board.
2. derive and understand a model for the dynamics of the cart (without the pendulum).
3. use proportional control to generate a step response on the actual hardware.

2 Equipment

Cart system (no attachments), Quanser Q4/Q2-usb DAQ board w/ terminal board, amplifier and cables.

The terminal board in Figure 1 provides connectors for the inputs and outputs of the Q4/Q2-usb DAQ boards in the computers. Specifically, you will use a single analog output (this will provide the motor voltage) and two encoder inputs (for cart position and pendulum angle). Since the analog out ports of the Quanser Q4/Q2-usb board are not powerful enough to drive the DC motor on the cart, the signal needs to be amplified (in fact, the gain will be unity, but the amplifier can provide much higher currents than the DAQ boards). Figure 2 shows the amplifier that drives the DC motor on the cart. In all of the coming labs, you will only use the ports "From analog output (D/A)" and "To Motor". The cables we use are special cables that have resistors between the connector ports built in so that the connection results in the correct op-amp circuit. The GSI will give you a demo on how to properly connect terminal board, amplifier and cart.

3 Theory

3.1 Simulink Coder, QuaRC, and the Q4 DAQ board

MATLAB's Simulink Coder (formerly Real-Time Workshop) generates and executes C and C++ code from Simulink diagrams, Stateflow charts, and MATLAB functions. The generated source code can be used for real-time and nonreal-time applications, including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing.

QuaRC is Quanser's rapid prototyping and production system for real-time control. QuaRC integrates seamlessly with Simulink to allow Simulink models to be run in real-time on Windows. It uses a host and target relationship that allows code generation and execution to occur on separate machines. However, we will be using QuaRC in "Single User Mode" or "Local Configuration", where we will be generating and executing code on the same computer, as shown in Figure 3.

The QuaRC Simulink Development Environment (SDE) is used to generate/build code to be later run on a real-time target from MATLAB/Simulink models. The QuaRC Windows Target feature is required to

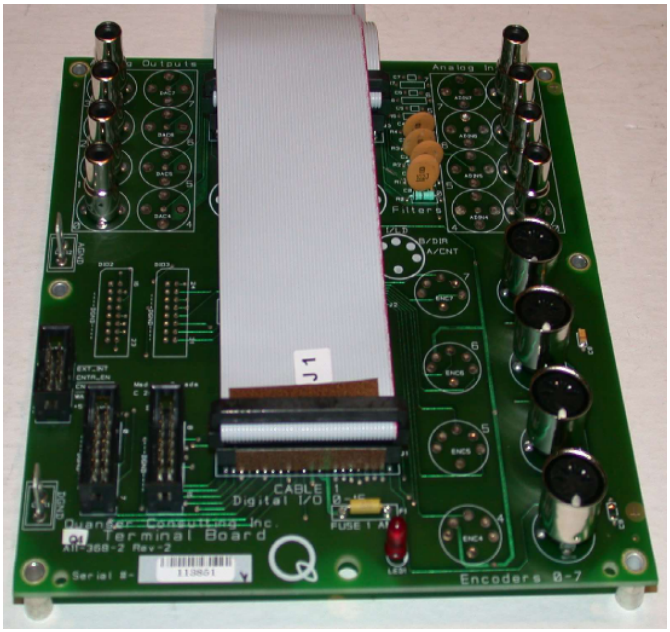

Figure 1: Terminal board of the Quanser Q4 DAQ card

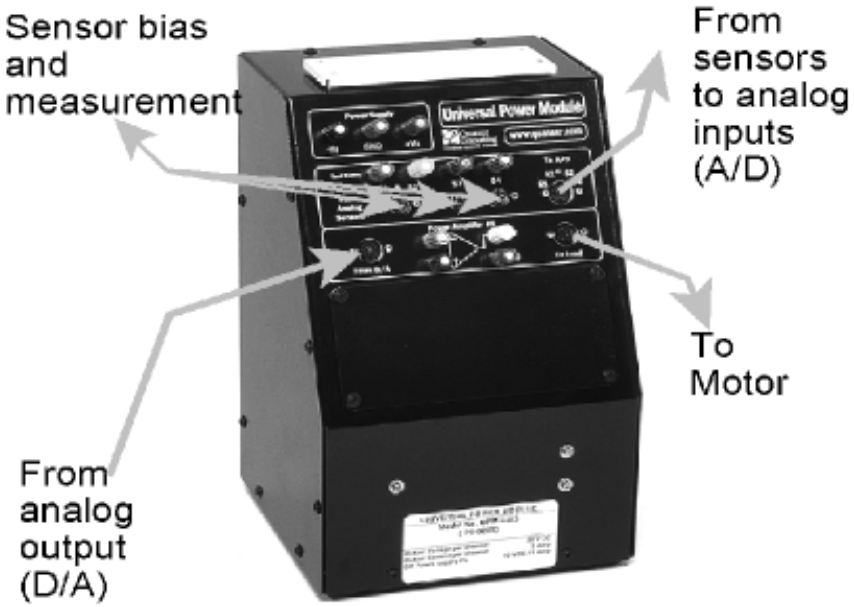

Figure 2: Amplifier for the cart's DC motor

run the generated code from MATLAB/Simulink models on a real-time Windows target (local or remote). QuaRC Windows Target needs to be open to run any QuaRC-generated code.

While interesting, understanding the details of the implementation of the QuaRC software is not the focus of this lab. Your task is to design the controller based on either classic or state-space techniques. Then you will implement the controller in Simulink. This is then downloaded to the QuaRC target, which interfaces the plant through the Q4 DAQ board. This board supports 4 A/D converters, 4 D/A converters, 16 Digital I/Os, 2 Realtime clocks, and up to 4 Quadrature input decoders/counters. The Q4

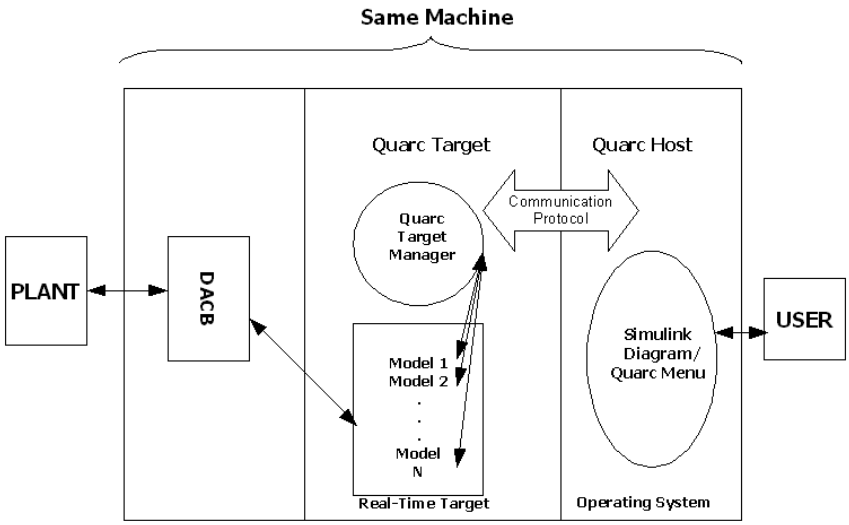

Figure 3: "Local Configuration" – QuaRC Host and Target on the same PC

board's functionality has also been abstracted from the user. The board has been set up to work with the cart and pendulum for all stations.

Important: Do not change any of the hardware of the Quanser stations without first consulting the GSI! Also, make sure you get permission from one of the GSIs before working on the hardware outside regular section times.

3.2 Dynamics of the Cart

Figure 4 shows a picture of the cart used in the lab setup for the pendulum experiments. The main components are the Cart motor pinion (5) with attached 6V DC motor and gearbox (not visible in the figure), the Cart position pinion (4) with attached encoder (8) and the pendulum axis (7) with attached encoder (9). **Warning: The motor pinion is the weakest part of the setup and can easily be damaged by high torque values**, so be careful when running a controller on the hardware. Always start with **less aggressively tuned controllers** first if possible.

Figure 5 shows the cart's free body diagram. For simplicity, we will ignore the effects of friction. In the diagram, F_a is the input force exerted on the cart by the voltage applied to the motor, m_c is the mass of the cart. The encoder is used to keep track of the position of the cart on the track.

Using Figure 5 and basic Newtonian dynamics you can derive the equations governing the system.

3.3 Motor Dynamics

The input to your system is actually a voltage to the cart's motor. Thus, you need to derive the dynamics of the system that converts the input voltage to the force exerted on the cart. These are the dynamics of the motor.

Figure 6 shows a diagram of the electrical components of the motor.

For this derivation of the motor dynamics we assume the following:

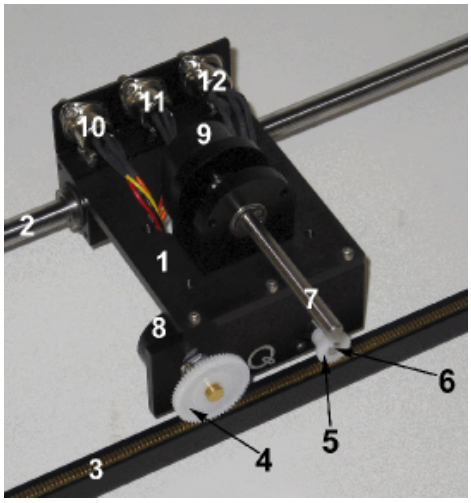

Figure 4: Quanser IP02 cart

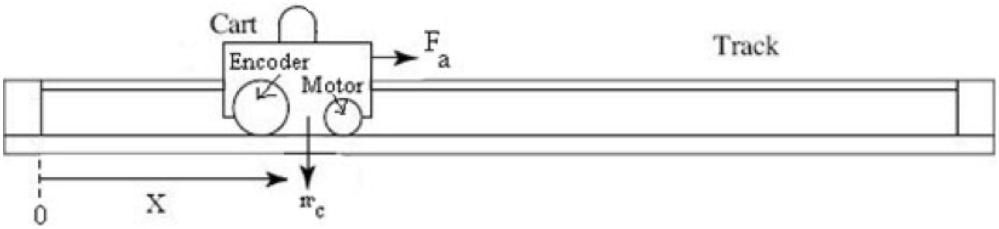

Figure 5: Free body diagram of the cart (ignoring friction)

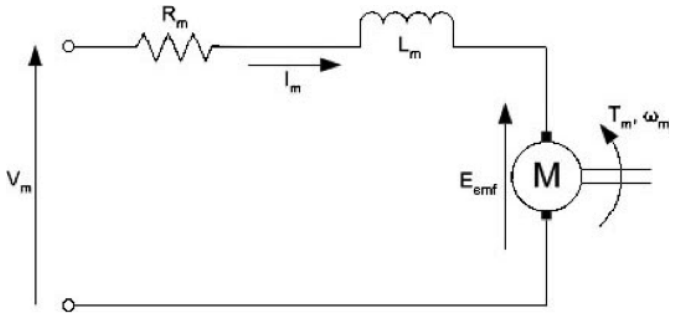

Figure 6: Classic armature circuit of a standard DC motor

- We disregard the motor inductance: $L_m \ll R_m$, so we can use the approximation $L_m \approx 0$.
- Perfect efficiency of the motor and gearbox: $\eta_m = \eta_g = 1$.

The torque generated by the motor is proportional to the current flowing through the motor windings, but is lessened due to the moment of inertia:

$$T_m = K_t I_m - J_m \ddot{\theta} \tag{1}$$

Here K_t is the motor torque constant, I_m is the current flowing through the coil, J_m is the moment of inertia of the motor and $\ddot{\theta}$ is the angular acceleration of the motor.

The current flowing through the motor can be related to the motor voltage input by:

$$V = I_m R_m + E_{emf} = I_m R_m + K_m \dot{\theta} \tag{2}$$

where $\dot{\theta}$ is the angular velocity of the motor, R_m is the resistance of the motor windings and K_m is the back EMF constant (in $\frac{\text{V}}{\text{rad/s}}$).

The torque is related to the applied force via

$$K_g T_m = F_a \cdot r \tag{3}$$

where r is the radius of the motor gear and K_g is the gearbox gear ratio. The motor's angular velocity is related to the cart's linear velocity via

$$K_g \dot{x} = \dot{\theta} \cdot r \;\Rightarrow\; K_g \ddot{x} = \ddot{\theta} \cdot r \tag{4}$$

3.4 Step Response of a Dynamical System

Figure 7 shows the typical step response of a SISO (single-input, single-output) dynamical system.

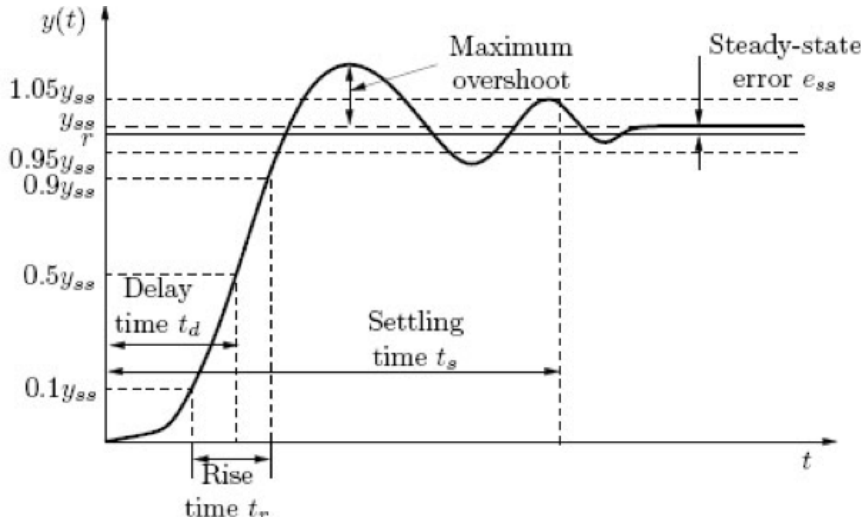

Figure 7: Typical step response of a control system

Recall the following quantities from class:

- **Steady-state value:** The *steady-state value* of the response $y(t)$ is defined as $y_{ss} := \lim_{t\to\infty} y(t)$.
- **Steady-state error:** For a control system, we want the output, $y(t)$, to follow a desired reference signal, $r(t)$. Thus we can define the error as $e(t) := r(t) - y(t)$. Consequently, the *steady-state error* is given by $e_{ss} := \lim_{t\to\infty} e(t)$.
- **Maximum overshoot:** Let y_{max} denote the maximum value of $y(t)$. The *maximum overshoot* of the step response $y(t)$ is defined by $y_{max} - y_{ss}$. It is often represented as a percentage of the steady-state value: percent maximum overshoot $= \frac{y_{max} - y_{ss}}{y_{ss}}$. The maximum overshoot is often used to measure the relative stability of a system. A system with a large overshoot is usually undesirable.

- **Delay time:** The *delay time* t_d is defined as the time required for the step response to reach 50% of its steady-state value.
- **Rise time:** The rise time t_r is defined as the time required for the step response to rise from 10% to 90% of its steady-state value.
- **Settling time:** The settling time t_s is defined as the time required for the step response to stay within 5% of its steady-state value.

4 Pre-Lab

4.1 Equations Governing the Cart Dynamics

Derive the following equation of motion for the cart system shown in Figure 5:

$$\left(m_c r^2 R_m + R_m K_g^2 J_m\right) \ddot{x} + \left(K_t K_m K_g^2\right) \dot{x} = \left(r K_t K_g\right) V \tag{5}$$

Table 1 lists the parameters that appear in (5).

Parameter	**Unit**	**Description**
V	Volt	input voltage
m_c	kg	mass of the car
r	meter	radius of the motor gears
R_m	Ω	resistance of the motor windings
K_t	N·m/A	torque motor constant
K_m	Vs/rad	back EMF constant
K_g	-	gearbox ratio
J_m	$\mathrm{kg\,m^2}$	moment of inertia of the motor

Table 1: Parameters of the cart system

In order to derive the equation (5), follow the steps below:

1. Using the free body diagram in Figure 5, apply Newton's second law to the cart.
2. Combine the motor dynamics, equations (1) - (4), to obtain the relationship between the input voltage V and the applied force F_a. Substitute this relationship into your equation from Step 1. This is the final model of your plant.
3. Is this system linear? If not, linearize the system. If so, leave as is.

4.2 Derive System Models

- **Transfer Function:** Apply the Laplace transform to your linear system and solve for the transfer function $H(s) = \frac{X(s)}{V(s)}$.
- **State Space Model:** Using cart position and velocity as states x_1, x_2, respectively, and the cart position as the system output y, derive a state space representation (i.e. matrices A, B, C and D) for your linear system.

- **SS to TF:** Using the following equation, derive a transfer function from your state space matrices and verify that it matches the transfer function you got directly from taking the Laplace transform of the equation of motion.

$$G(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D \tag{6}$$

Hint: As you should know, the inverse of a 2×2 matrix is given by

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{7}$$

4.3 MATLAB Step Response

Use the values for the parameters given in Table 2 to create a Simulink block diagram of the cart system in a simple negative feedback loop with a gain K as the controller. It is your choice whether you want to use the state space or transfer function representation of the system. Use a step function as input.

Parameter	**Value**
m_c	$0.57 + 0.37 = 0.94\,\text{kg}$
r	$6.36 \cdot 10^{-3}\,\text{m}$
R_m	$2.6\,\Omega$
K_t	$7.67 \cdot 10^{-3}\,\text{Nm/A}$
K_m	$7.67 \cdot 10^{-3}\,\text{Vs/rad}$
K_g	3.71
J_m	$3.9 \cdot 10^{-7}\,\text{kg m}^2$

Table 2: Parameters of the cart system

Vary the value of K until you achieve a percent maximum overshoot $< 4.0\%$ and rise time $t_r < 1.5s$. You only need to find a single value that works, not a range of values. Include your block diagram (and any code you used) as well as your final value of K and plots verifying these design conditions are met. Make sure you describe your process of finding a suitable value of K.

Note: You will find the MATLAB function `find(cond,N)` to be very useful for this. This returns at most the first N indices that match the condition `cond`. Type `doc find` to read about the other various uses for this function if you wish. For example, for an array of output values `out` and time values `time`, you can use the following code to find the time of the first value of out that exceeds the value of 0.1:

```
>> find(out >= 0.1,1)
ans =
   110
>> time(110)
ans =
   1.0900
```

To get more precise time and output values, we recommend that you set Simulink to a small, fixed-step interval (e.g. 0.01s). This is done by choosing "Fixed-step" as the "Type" and "ode3" as the "Solver" under the solver options in the model configuration parameters. You can access these settings by choosing "Simulation" → "Model Configuration Parameters" in the Simulink window.

5 Lab

5.1 Cart Dynamics

Confer with your group to agree on a system representation (either state space or transfer function) to use in this lab.

5.2 The Quanser Hardware

The GSI will explain how to properly use the Quanser system. Make sure you understand the system's functionality so you can implement your controller easily. **Make sure to observe safety precautions at all times**. In particular, make sure to turn off the amplifier after you are finished using Quanser hardware. Any group left the lab without turning off the amplifier will *lost 10 points* from their lab report. This penalty will stack for every offense.

5.3 Using the Actual Hardware: Find Encoder-Distance Conversion

The GSI will cover how to interface with the actual cart hardware by building a Simulink subsystem. Encoder values for the position of the cart will be read in encoder counts, however, our input defined in the system equations were in meters. In feedback, the two values you compare must be in the same units, so we need a conversion factor.

- Build a simple Simulink file that read the position encoder count. You can set the final time in Simulink to "inf" for infinity to run the program indefinitely. You can use QuaRC → Stop to stop the program. If you use a Scope, it will update in real time. You will need to use the following blocks in the Simulink library:
  - under QUARC Targets / Data Acquisition / Generic / Configuration: HIL Initialize block. Use the following options: Board type: **q4** (green board) OR **q2-usb** (black box).
  - under QUARC Targets / Data Acquisition / Generic / Immediate I/O: HIL Read Encoder block. Use the following options: channels: Encoder input #0 and #1.
  - You can use the "To Workspace" block in the Commonly Used Blocks / Sinks section to save the positions and angular positions to variables in the Matlab workspace. In order to have the output as an array variable, you can change the "Save Format" option to "Array".
  - You can also use the "Clock" block to get the simulation time.

  In the Simulink window, use the QuaRC / Build command to compile the model before running it. You also need to set Simulation / Mode to "external". This can also be set using the drop-down menu next to the simulation time field.
- In order to run the model, you first have to connect to the real time target.

- Once you start running the QuaRC program, manually move the cart along the track and watch the encoder values update. Using a ruler, move the cart manually a certain distance (note down by how much!) and let it sit for a while. Then move the cart again and repeat this a couple of times.
- Using the plateaus in your plot (from letting the cart sit at a position for a while), calculate the encoder count increment from moving the cart one cm. Average your values over all moves for better accuracy. *Hint:* Make sure to choose a high enough value in the "Limit data points to last" field in the Data Import/Export section of the Model parameters window if you are using "Out" blocks to output data to the workspace.
- Include the plot of the cart encoder values (not converted) vs. time in your lab report and document your calculation of the encoder counts/m.

The Quanser manual gives the position encoder resolution to be 4096 counts/revolution. Given that the radius of the position pinion is $r_{pp} = 0.01482975\,\text{m}$, what is the "actual" encoder resolution in counts/m? How close was your estimated encoder resolution?

5.4 Using the Actual Hardware: Cart Step Response

- Go back to your Simulink model of the cart system from the pre-lab. Now change the step function to be of height 0.15, corresponding to the cart moving 0.15 m. Again, try to find a value of K so that percent maximum overshoot $< 4.0\%$ and $t_r < 1.5s$. Report this value in your lab report. How different is the value you found here from the value of K you found in the pre-lab for a step size of 1? Why is that?
- Once the simulated step response looks fine, you can move over to the actual hardware. Replace your system block (ss or tf) in the feedback loop with the hardware subsystem, with the counts/m conversion included. Make sure you include a saturation block to limit the input voltage to the motor. Use $6V$ as the saturation value. **Important: Check in with the GSI before running the QuaRC program on the actual hardware**.

  For the hardware subsystem you will need the following blocks:
  - Data acquisition / Generic / Immediate I/O:HIL Write Analog.
  - A Saturation block (in Commonly Used Blocks).

  Plot the initial hardware response and compare with the plot from the Simulink model. How close was the actual to the predicted? What might have caused any discrepancies? *Hint:* take a look at the control signal (i.e. the motor voltage).
- Now change your value of K until you achieve a percent maximum overshoot of $< 4.0\%$ and $t_r < 1.5s$. Report your new K value and plot the hardware response. Why is the new K different? Show your modified hardware step response to the GSI before the end of the lab session.
- Finally, find a value of K that achieves a $t_r < 0.3s$ and maximum overshoot $< 12\%$ using the hardware (do not use a K value over 60). Is this possible in your Simulink model from the pre-lab? Discuss any discrepancies and make some hypotheses as to why they occur.