POLYTECH NANCY

Lab 2
State-space model based 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

Acknowledgements

The contents of this lab has been largely inspired and adapted from initial versions of the different chapters provided by Quanser Inc[1]. This is fully acknowledged.

[1] www.quanser.com

Part 1

State feedback control of the rotary inverted pendulum

In this part, we will mainly use the QUBE-Servo 2 with the inverted pendulum.

1.1 Download of the files required for the lab

1. Download the zipped file `Lab_QUBE_Pendulum.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_Pendulum` and select `add to path the selected folder`.

4. Double-click on the folder `Lab_QUBE_Pendulum` so that it becomes your current folder. You should see the different `.slx` and `m.` files needed for this Lab.

Layout of the Lab

A balance system is a mechanical system in which the center of mass is balanced above a pivot point. Some common examples of balance systems are shown in Figure 1.1. The Segway personal transporter uses a motorized platform to stabilize a person standing on top of it. When the rider leans forward, the transportation device propels itself along the ground but maintains its upright position. Another example is a rocket, in which a gimbaled nozzle at the bottom of the rocket is used to stabilize the body of the rocket above it. Other examples of balance systems include humans or other animals (gorilla, kangaroos, etc) standing upright. This might seem easy but it is not and requires some learning process and skill. You can test your ability to balance a pen at the end of one of your finger to feel the difficulty.

(a) Segway (b) Saturn rocket (c) Cart–pendulum system

Figure 1.1: Examples of balance systems. (a) Segway personal transporter, (b) Saturn rocket and (c) inverted pendulum on a cart.

While PID control is applicable to many control problems and often perform satisfactorily, it can perform poorly in some applications. This is in particular the case when the system has multiple inputs and multiple outputs or when the system is unstable in open loop.
This lab considers the two previous situations: it addresses the control of an inverted pendulum balanced on a rotary arm. We investigate how the inverted pendulum can be stabilized/balanced using full state feedback, and explore pole placement and optimal LQR as techniques for choosing the feedback gains. The objective is to illustrate the various stages of design that lead to both pole placement and LQR control to stabilize/balance the rotary inverted pendulum.
The lab is structured into the following sections:

1. a refresher about model-based state feedback control by the pole placement method. The methodology is illustrated by using a simple second-order simulation example and tested in Simulink;

2. the design of a pole placement controller based on the physics based linearized state-space model and the evaluation of its control performance in simulation by using Simulink;

3. the implementation and test of the designed pole placement based state feedback control to stabilize/balance the physical rotary inverted pendulum.

4. the design of an optimal LQR controller based on the physics based linearized state-space model and the evaluation of its control performance in simulation by using Simulink;

5. the implementation and test of the designed LQR based state feedback control to stabilize/balance the physical rotary inverted pendulum.

1.2 Control performance requirements

The performance requirements and time-domain specifications for balancing the pendulum and for tracking a rotary arm setpoint are described in Table 1.1.

Requirement	Assessment criteria	Level
Balance the inverted pendulum	Position setpoint tracking	balanced & stable with no steady-state error
	Motor input voltage	limited to [-10V ; +10 V]
	Percent Overshoot	$D_1 = 6.8$ %
	Settling time at 5 %	$T_s^{5\%} = 1.5$ s
	Disturbance rejection	Rejection of impulse-type flick

Table 1.1: Performance requirements for the state-feedback control of the inverted pendulum

Thus, as the rotary arm goes back and forth to track the reference while balancing the pendulum, it should have a percent overshoot and settling time matching these requirements.

1.3 Pole placement control - A refresher from a simulation example

1.3.1 A video to start with

To get a refresher about pole placement (or full) state feedback control, watch at the beginning of the lab, all together, the video from Brian Douglas on this topic:
What is Pole Placement (Full State Feedback) | State Space - Part 2
`www.youtube.com/watch?v=FXSpHy8LvmY`
Listen carefully to the different comments given by Brian Douglas !

1.3.2 Background

Pole placement is a method of calculating the feedback gain matrix used to assign closed-loop poles to specified locations, thereby ensuring system stability. Closed-loop pole locations have a direct impact on time response characteristics such as rise time, settling time, and transient oscillations.
The standard state-space representation of a multi-input multi-output (MIMO) continuous linear time-invariant (LTI) system with n state variables, r input variables, and m output variables is

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{1.1}$$

where $x \in \mathcal{R}^{n\times 1}$ is the vector of state variables, $u \in \mathcal{R}^{r\times 1}$ is the control input vector, $y \in \mathcal{R}^{m\times 1}$ is the output vector, $A \in \mathcal{R}^{n\times n}$ is the state matrix, $B \in \mathcal{R}^{n\times r}$ is the input matrix, $C \in \mathcal{R}^{m\times n}$ is the output matrix, and $D \in \mathcal{R}^{m\times r}$ is the feedforward matrix.
The schematic of a standard state-feedback control is shown in Figure 1.12 where K is a matrix of control gains. Note that here we feedback all of the system states x, rather than using the system outputs y for feedback.

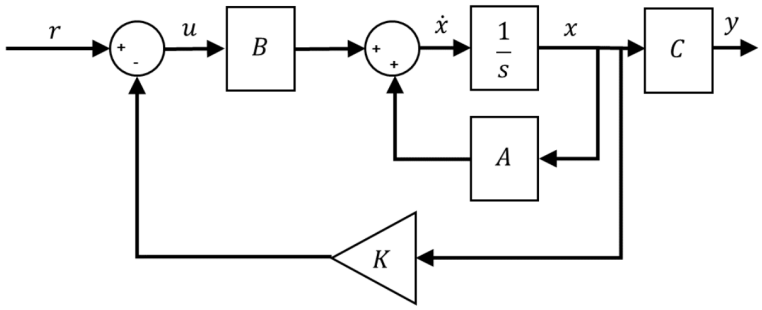


Figure 1.2: Block diagram of standard state-feedback control

Let $u(t)$ be a state feedback control law of the form

$$u(t) = -Kx(t) + r(t) \tag{1.2}$$

where $K \in \mathcal{R}^{m \times n}$ is the feedback gain vector. Applying this to the first equation of (1.1) gives the closed-loop system equation

$$\dot{x}(t) = Ax(t) - BKx(t) + Br(t) = (A - BK)x(t) + Br(t). \tag{1.3}$$

1.3.3 Controllability

If the system is unstable, we might be able to design a state-feedback controller to stabilize it. Even if the system is stable, we may still want to regulate the performance of the system according to some design specifications (e.g., final state, rate of convergence, and settling time). These are possible if the system is controllable.
By controllable, we mean that for any initial state vector x_a and any desired final state x_b, there exists a control input u that can steer the state of the system from x_a to x_b in finite time [1]. Otherwise, we say that the system is uncontrollable. Note that the definition does not require u to be bounded.
To check if the system is controllable, we can compute the rank of the Controllability matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{1.4}$$

where n is the dimension of the state vector x. Then, we say that the linear system described by (1.1) or, equivalently, the pair (A, B) is controllable if and only if $\text{rank}(\mathcal{C}) = n$. Otherwise, we conclude that the linearized system (or, in other words, the pair (A, B)) is uncontrollable.
We will use the MATLAB command `ctrb` to generate the controllability matrix and the MATLAB command `rank` to test the rank of the matrix.

```
Co = ctrb(A,B);
controllability = rank(Co)
```

1.3.4 Pole placement

If (A, B) is controllable, we can use the state-feedback to place the poles at desired locations, e.g. in the left half of the s-plane. Thus we can find a matrix gain K for the closed-loop system in (1.3) to obtain a desired characteristic equation

$$\prod_{i=1}^{n}(s - \lambda_i) = s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0 = 0 \tag{1.5}$$

where $\lambda_i \in \mathbb{C}$ for $i \in \{1, \ldots, n\}$ and $a_j \in \mathbb{R}$ for $j \in \{0, 1, \ldots, n-1\}$.
We will use the MATLAB command `place` to generate the matrix gain K
`P=[`$\lambda_1 \ldots \lambda_n$`]`

```
K=place(A,B,P)
```

Caution
Pole placement can be badly conditioned if you choose unrealistic pole locations. In particular, you should avoid:

- Placing multiple poles at the same location.
- Moving poles that are weakly controllable or observable. This typically requires high gain, which in turn makes the entire closed-loop eigenstructure very sensitive to perturbation.

1.3.5 State feedback control design of a simulation second-order system

To illustrate how to design K to obtain desired pole location, i.e. desired characteristic equation, we consider the design in the case of a simulation second-order system.

1.3.5.1 Analytical determination of the feedback gain

Consider a LTI system described by the following second-order ordinary differential equation:

$$\ddot{y}(t) + \dot{y}(t) - 2y(t) = \dot{u}(t) + u(t) \tag{1.6}$$

1. Show that the transfer function of the system is

$$G(s) = \frac{Y(s)}{U(s)} = \frac{s+1}{s^2 + s - 2} \tag{1.7}$$

2. Determine the poles and show that the open-loop system is unstable.

3. Let us transform the transfer function model into state-space.
$G(s)$ can also be decomposed as

$$\frac{Y(s)}{U(s)} = \frac{Y(s)}{Z(s)} \times \frac{Z(s)}{U(s)} \tag{1.8}$$

with

$$\frac{Y(s)}{Z(s)} = s + 1 \tag{1.9}$$

and

$$\frac{Z(s)}{U(s)} = \frac{1}{s^2 + s - 2} \tag{1.10}$$

Show that (1.9) and (1.10) are respectively equivalent to

$$y(t) = \dot{z}(t) + z(t) \tag{1.11}$$

$$\ddot{z}(t) + \dot{z}(t) - 2z(t) = u(t) \tag{1.12}$$

4. Assume $x_1(t) = \dot{z}(t)$ and $x_2(t) = z(t)$, such that

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{1.13}$$

show that the state space model in canonical form can be written as

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{1.14}$$

with

$$A = \begin{bmatrix} -1 & 2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{1.15}$$

5. Show that the characteristic equation of the open loop system is

$$|\lambda I - A| = \lambda^2 + \lambda - 2 = 0. \tag{1.16}$$

6. Determine the eigenvalues of A and show that the open loop space-space model is unstable.

7. We use here the pole placement method such that the closed loop poles have the values at -1 and -2.
The desired closed loop characteristic equation is therefore:

$$(\lambda + 1)(\lambda + 2) = \lambda^2 + 3\lambda + 2 = 0 \tag{1.17}$$

The control law is:

$$u(t) = k_r r(t) - K\, x(t) \tag{1.18}$$

where k_r is a scalar and K is a 1×2 gain matrix.

$$K = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \tag{1.19}$$

Show by plugging (1.18) into (1.14) that the closed-loop state-space model becomes:

$$\begin{cases} \dot{x}(t) = A_{CL}x(t) + B_{CL}r(t) \\ y(t) = Cx(t) \end{cases} \tag{1.20}$$

with

$$A_{CL} = A - BK = \begin{bmatrix} -1 - k_1 & 2 - k_2 \\ 1 & 0 \end{bmatrix}, \quad B_{CL} = k_r B = \begin{bmatrix} k_r \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{1.21}$$

8. Show that the characteristic equation of the closed-loop model is :

$$\lambda^2 + (1 + k_1)\lambda - 2 + k_2 = 0 \tag{1.22}$$

Hint: determine the eigenvalues of A_{CL} by calculating $|\lambda I - A_{CL}| = 0$.

9. Show that equating the coefficients of the polynomial (1.22) with the desired polynomial in (1.17) leads to

$$\begin{cases} k_1 = 2 \\ k_2 = 4 \end{cases} \tag{1.23}$$

1.3.5.2 Determination of the feedback gains with Matlab

Verify all your analytical calculations above by writing the following code in a Matlab script:

```
Num=[1 1];
Den=[1 1 -2];
roots(Den)
[A,B,C,D]=tf2ss(Num,Den)
eig(A)
Co = ctrb(A,B);
controllability = rank(Co)
P=[-1 -2];
K=place(A,B,P)
Acl=A-B*K
eig(Acl)
Sys_cl=ss(Acl,B,C,D);
kr=1/dcgain(Sys_cl)
```

1.3.5.3 Simulation of the full state feedback control with Simulink

We will now set up a Simulink model to implement the full state feedback.
Different options are possible to implement the state feedback depending how the open loop state space model is built. We use 3 different possibilities here.
Reproduce the Simulink diagram given in Figure 1.3.

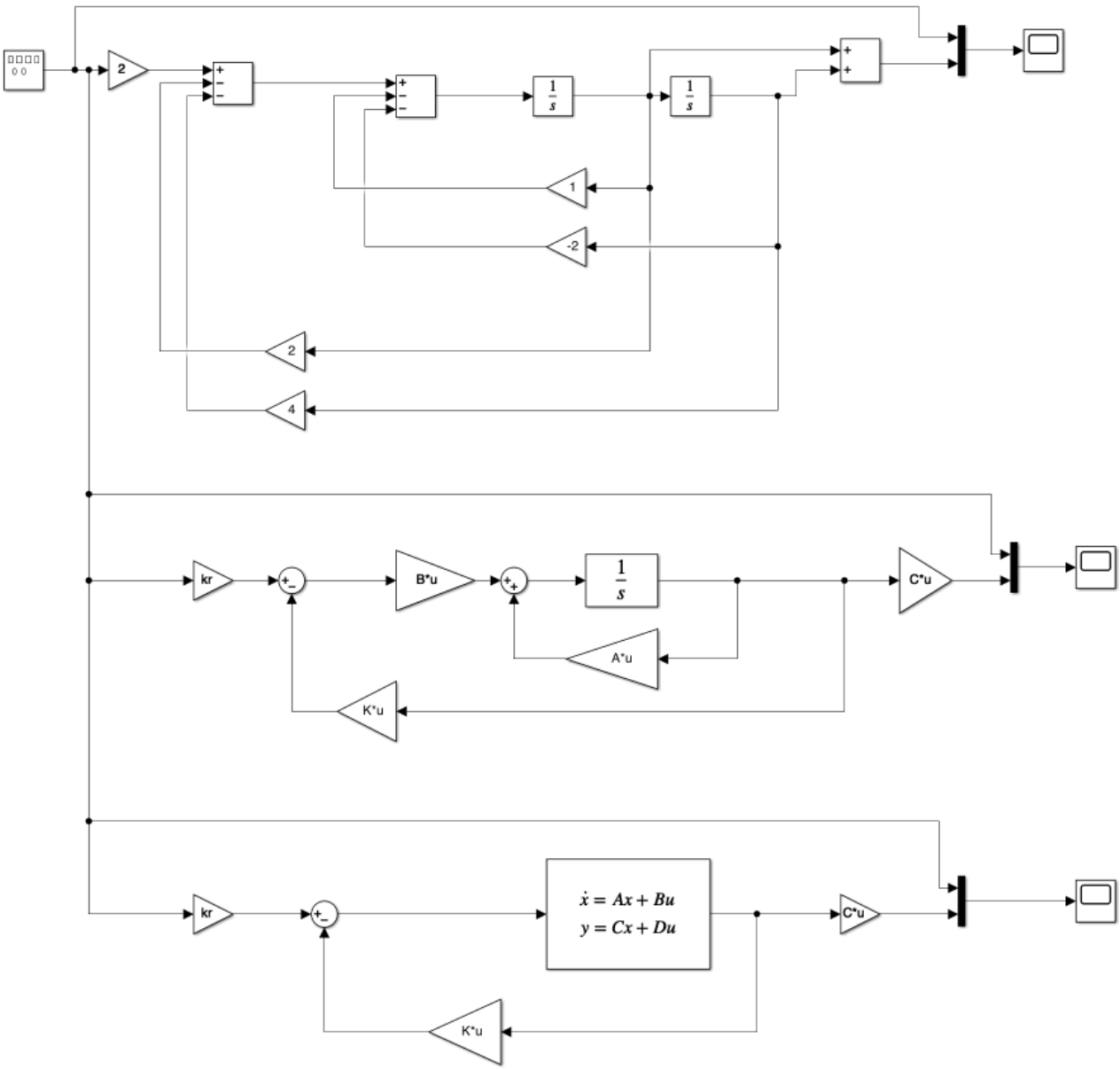


Figure 1.3: Different implementations of the full state feedback control with Simulink

Configure the signal generator and the configuration parameters as shown in Figure 1.4 and Figure 1.5 respectively.

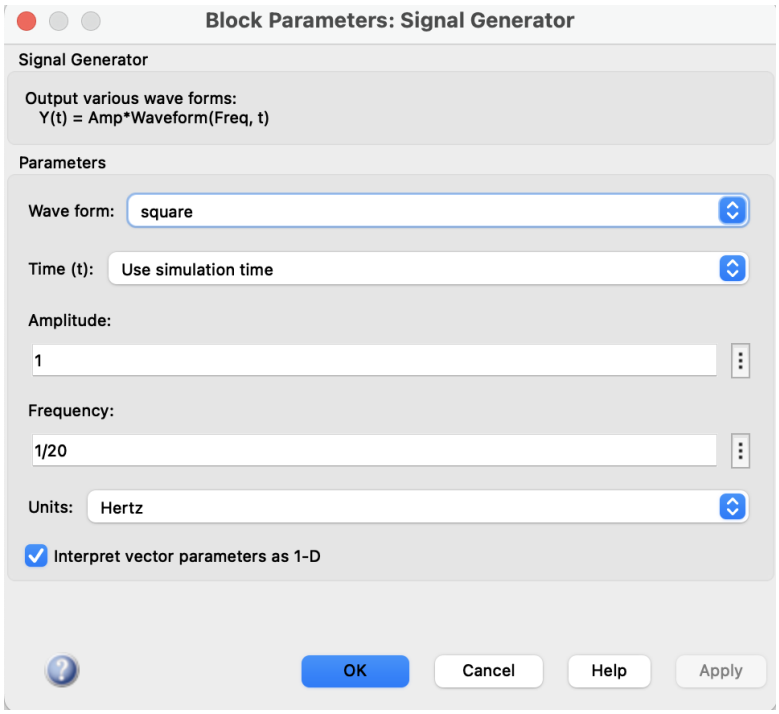


Figure 1.4: Setting of the signal generator block

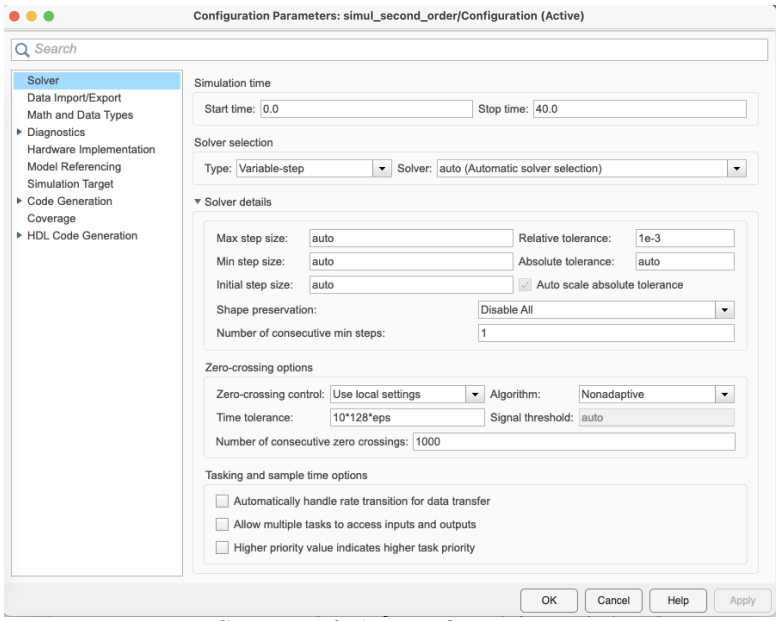


Figure 1.5: Setting of the configuration parameters

Note that when you use the state-space model block of Simulink, you need first to define the state-space block with an artificial identity C matrix to assess the full states to be feedback and apply the true C matrix to the states to get the output variable.

Build and run the Simulink model. The response of the three scopes should be identical.

1.4 Pole placement-based balance control for the inverted pendulum

1.4.1 State-space model of the inverted pendulum

A schematic diagram of the rotary pendulum model is shown in Figure 1.6. The arm has a length of L_r, a moment of inertia of J_r, and its angle θ increases positively when it rotates counter-clockwise (CCW). The servo (and thus the arm) should turn in the CCW direction when the control voltage is positive ($V_m > 0$).

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

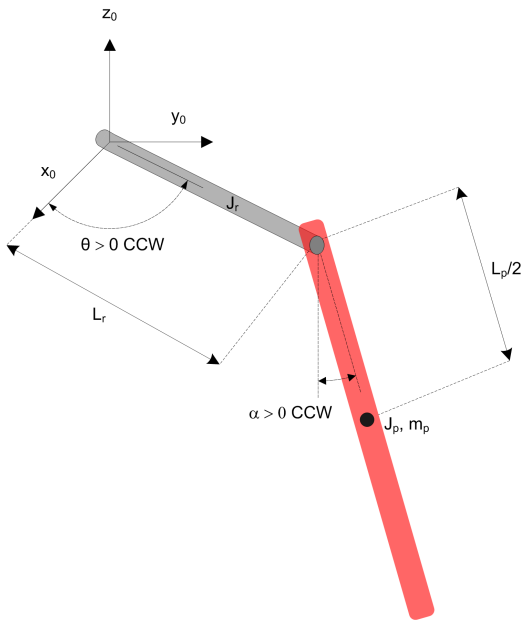


Figure 1.6: Schematic diagram of the rotary pendulum

1.4.1.1 State-space model of the non-inverted pendulum

The design of the state-space controller is based on a model of the system. There are two approaches available to determine a mathematical model of a dynamical system:

- Data-driven modelling where a model is determined by using the system response data to an excitation input (step input for example) ;
- Physics-informed modelling where the equations of motion for the system are developed from the Physics.

We will used the second approach here. This rotary inverted pendulum system has been widely studied and as a consequence the physics-informed model has been established and is available.

However, when the QUBE-Servo 2 is used with the pendulum is the up position, the system is unstable and therefore no experimental open-loop test can be recorded to test the quality of the

derived model. As the model of the inverted and non-inverted pendulum are very close (they only differ with some minus signs for some coefficients) we will first deal with the non-inverted pendulum.
If the state vector x of the rotary pendulum system is chosen as

$$x(t) = \left[\theta(t); \alpha(t); \dot{\theta}(t); \dot{\alpha}(t)\right] \tag{1.24}$$

the linearized state-space model of the rotary pendulum-down system is:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{1.25}$$

with

$$A = \frac{1}{J_T} \begin{bmatrix} 0 & 0 & J_T & 0 \\ 0 & 0 & 0 & J_T \\ 0 & \frac{1}{4} m_p^2 L_p^2 L_r g & -\left(J_p + \frac{1}{4} m_p L_p^2\right) D_r & \frac{1}{2} m_p L_p L_r D_p \\ 0 & -\frac{1}{2} m_p L_p g \left(J_r + m_p L_r^2\right) & \frac{1}{2} m_p L_p L_r D_r & -\left(J_r + m_p L_r^2\right) D_p \end{bmatrix} \tag{1.26}$$

$$B = \frac{1}{J_T} \begin{bmatrix} 0 \\ 0 \\ J_p + \frac{1}{4} m_p L_p^2 \\ -\frac{1}{2} m_p L_p L_r \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{1.27}$$

1.4.1.2 Experimental model validation from step response

It is important that you verified your model for the pendulum-down case with the steps below.

1. In Matlab, run the `setup_non_inverted_pend_ss_model.m` script. This creates, in the Matlab workspace, the QUBE-Servo 2 rotary pendulum-down state-space model matrices A, B, C, and D based on numerical values of the different physical parameters. The A and B matrices should be displayed in the Command Window. Ensure that the generated matrices match the solution.

```
A =
         0         0         1         0
         0         0         0         1
         0    149.3   -14.93    4.915
         0   -261.6    14.76   -8.614
B =
         0
         0
     49.73
    -49.15
```

2. Open the Simulink model `test_non_inverted_pend_ss_model_2_square_wave.slx` whose contents is shown in Figure 1.7. It applies for 20s a $0 - 1V, 1$ Hz square wave to the physical inverted pendulum system and to the physics-based state-space model.

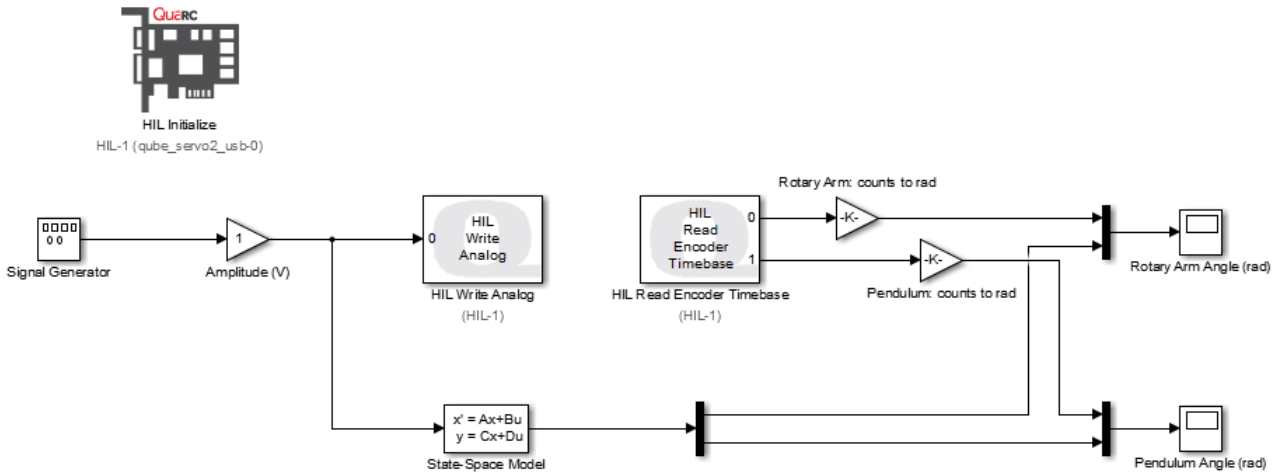


Figure 1.7: Applies a square wave voltage and displays the measured responses and the simulated model pendulum responses

3. Place manually the pendulum close to the 0° position and click on `Monitor & Tune` to build and run the test.
   The scope response should be similar to Figure 1.8. Does your model represent the actual pendulum-down well?
   The model of the arm response should display the same characteristics of the measured arm response, but an offset may be observed due to possible un-modelled dynamics such as the disturbance introduced by the encoder cable.

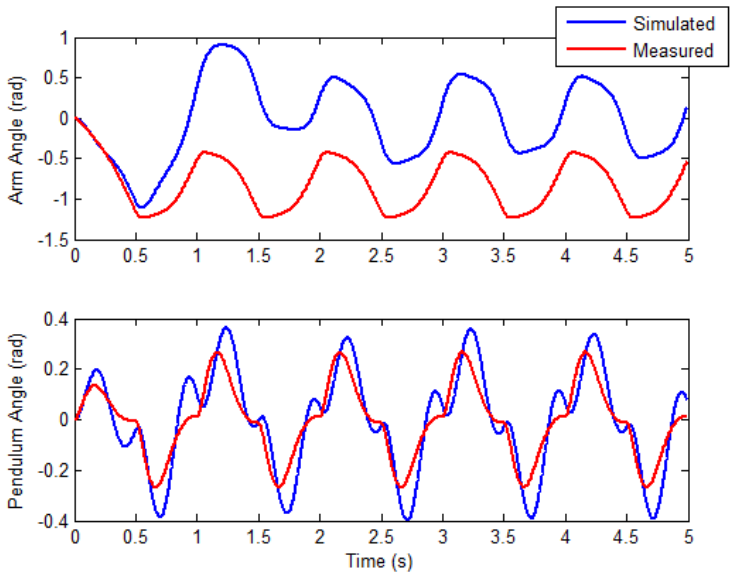


Figure 1.8: Step response of the pendulum system.

4. In the Matlab script `setup_non_inverted_pend_ss_model.m`, the rotary arm viscous damping coefficient D_r is set to $0.0015 N.m.s/rad$, and the pendulum viscous damping coefficient is set to D_p to $0.0005 N.m.s/rad$.
   These parameters were found experimentally to reasonably accurately reflect the viscous damping of the system due to effects such as friction, when subject to a step response. However, the viscous damping of each inverted pendulum can vary slightly from system

to system. If your model does not accurately represent your specific pendulum system, try modifying by trial and error the damping coefficients D_r and D_p to obtain a more accurate model.

1.4.1.3 Time-domain specifications for higher order systems

The rotary inverted pendulum system has four poles. However, if two of the closed loop poles are chosen to be closer to the imaginary axis (typically by a factor equal or greater than four) than the remaining poles, the conjugate poles are considered to be dominant and the system behavior can be approximated by a second-order system. As depicted in Figure 1.2, poles p_1 and p_2 are the complex conjugate dominant poles and are chosen to satisfy the natural frequency, ω_n, and damping ratio, ζ, second-order specifications. Let the conjugate poles be

$$p_1 = -\sigma + j\omega_d \tag{1.28}$$

$$p_2 = -\sigma - j\omega_d \tag{1.29}$$

where $\sigma = \zeta\omega_n$ and $\omega_d = \omega_n\sqrt{1-\zeta^2}$ is the damped natural frequency. The remaining closed-loop poles, p_3 and p_4, are placed along the real axis to the left of the dominant poles, as shown in Figure 1.9.

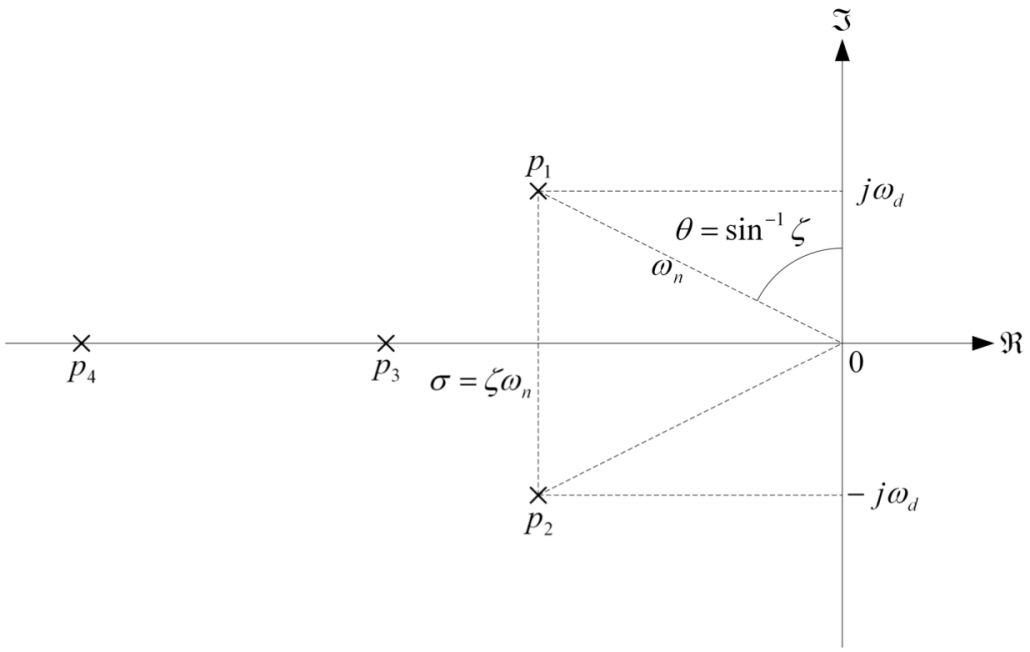


Figure 1.9: Desired closed-loop pole locations

1.4.2 Pole placement control design for the inverted pendulum

1. In Matlab, run the `setup_inverted_pend_ss_model.m` script. This creates, in the Matlab workspace, the QUBE-Servo 2 rotary pendulum-up state-space model matrices A, B, C, and D based on numerical values of the different physical parameters. The A and B matrices should be displayed in the Command Window. Ensure that the generated A and B matrices match the form given below.

```
A =
         0         0         1         0
```

```
         0         0         0         1
         0    149.3    -14.93    -4.915
         0    261.6    -14.76    -8.614
B =
         0
         0
     49.73
     49.15
```

2. Ensure that the open-loop poles of the inverted pendulum are: $\{0, -28.64, 10.67, -5.57\}$ Note that the system is unstable because of the positive pole at 10.67.

3. Evaluate the controllability of the rotary inverted pendulum system. Use the `ctrb` function to compute the controllability matrix and explain whether or not the system is controllable.

4. The percent overshoot and settling time specifications given translates into the following natural frequency and damping ratio requirements:

$$\zeta = 0.65$$
$$\omega_n = 4\,\text{rad/s}$$

   Based on these specifications, find the location of the two dominant poles p_1 and p_2.

5. Find the desired characteristic equation if the other poles are placed at $p_3 = -40$ and $p_4 = -45$.

6. Use the pole placement design command in Matlab to find the state-feedback gain, K, required to place the closed-loop poles to the desired locations. Give the Matlab commands used and the obtained matrix gain K.

1.4.3 Implementation of the balance control to the rotary inverted pendulum

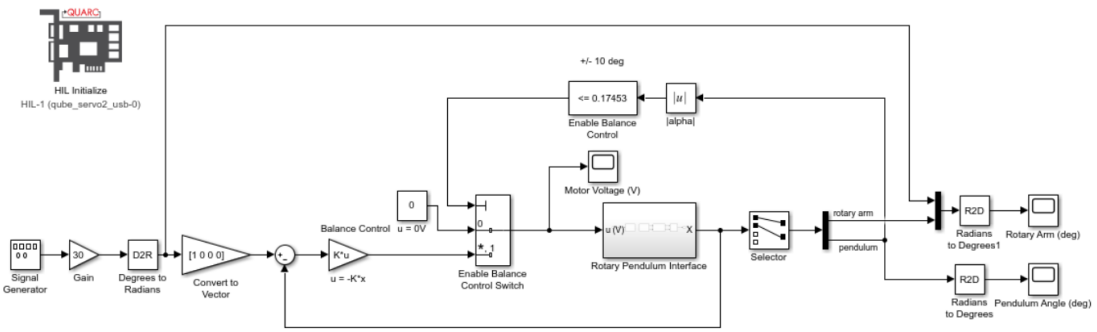


Figure 1.10: Simulink model used with pole placement-based balance controller

1. Open the `balance_inverted_pend_poleplace.slx` Simulink model whose contents is shown in Figure 1.10.

2. Make sure the variable K is set in the Matlab workspace.

3. Set the Signal Generator block to the following:

   - Type = Square
   - Amplitude = 1
   - Frequency = 0.125 Hz

4. Set first the Gain block that is connected to the Signal Generator to 0.

5. Click on `Monitor & Tune` to build and run the file.

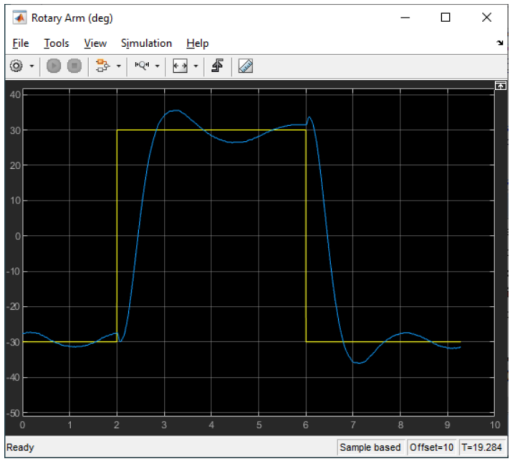
6. When the QUBE-Servo 2 turns green, gently and manually rotate the pendulum in the upright position until the controller engages. Observe the scopes. The balance state-space control should work fine. There i so magic here even if this is quite amazing.

7. Once the pendulum is balanced, set the Gain to 30 to make the arm angle go between ±30 degrees. The scopes shown in Figure 1.11 show an example response when using a state feedback gain of $K = [-2 \quad 35 \quad -1 \quad 3]$.
   Attach your response of the rotary arm, pendulum, and controller voltage using the control gain found in the previous Section.

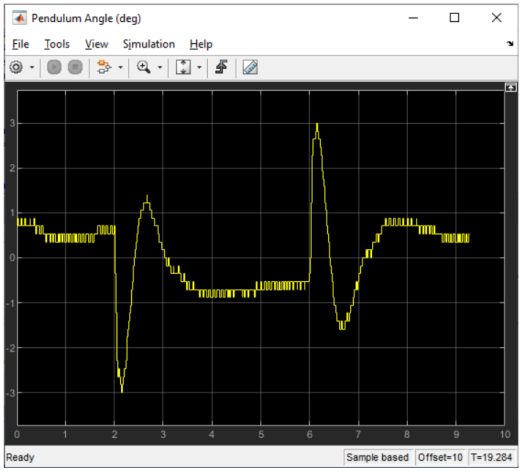
8. Does the rotary arm and pendulum response match the settling time and percent overshoot specifications given in Table 1.1? If not, give one reason why there is a discrepancy. Modify the desired closed-loop poles to get better results.

9. Stop the controller.

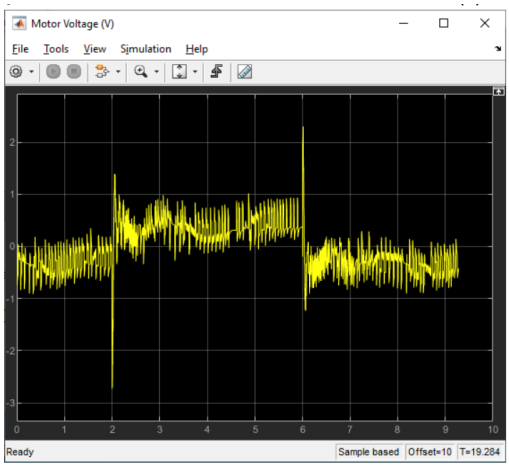
Note that the encoders only provide measurements for the arm and pendulum angular positions. We need to also measure or estimate the arm and pendulum velocities in order to perform full state feedback. In this lab, we will estimate the two angular speeds from the angular position measures by using high-pass filters of the form $\frac{50s}{s+50}$ already implemented in the Simulink files provided (double-click on the Rotary Pendulum Interface block to see the high-pass filters). Computing any variable time-derivatives by using simple numerical approximations should be AVOIDED. Next year, you will build an observer and estimate the arm and pendulum velocities from the measurements of arm and pendulum position alone.



(a) Rotary Arm



(b) Pendulum



(c) Motor input voltage

Figure 1.11: QUBEServo 2 example rotary pendulum response using default gain

1.5 Optimal LQR-based balance control for the inverted pendulum

1.5.1 A video to start with

If you need a refresher about LQR control, watch Brian Douglas' video:
What Is Linear Quadratic Regulator (LQR) Optimal Control? | State Space, Part 4
`www.youtube.com/watch?v=E_RDCFOlJx4&list=PLn8PRpmsu08podBgFw66-IavqU2SqPg_w&index=4`
Listen carefully to the different comments given by Brian Douglas !

1.5.2 LQR control

Linear Quadratic Regulator (LQR) theory is a technique that is ideally suited for finding the controller parameters to balance the inverted pendulum.
Given that the equations of motion of the inverted pendulum system can be described in state-space form (1.1). The LQR algorithm computes a control law u such that the performance criterion or cost function

$$J = \int_0^\infty \left(x_{ref} - x(t)\right)^T Q \left(x_{ref} - x(t)\right) + u(t)^T R u(t) dt \tag{1.30}$$

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

$$x = \begin{bmatrix} \theta & \alpha & \dot{\theta} & \dot{\alpha} \end{bmatrix}^T. \tag{1.31}$$

Since there is only one control variable, R is a scalar. The reference signal r is set to:

$$r = \begin{bmatrix} \theta_r & 0 & 0 & 0 \end{bmatrix}^T,$$

The control strategy used to minimize the cost function J is thus given by

$$u = K(r - x) = k_{p,\theta}(\theta_r - \theta) - k_{p,\alpha}\alpha - k_{d,\theta}\dot{\theta} - k_{d,\alpha}\dot{\alpha}. \tag{1.32}$$

This control law is a full state-feedback control and is illustrated in Figure 1.12.

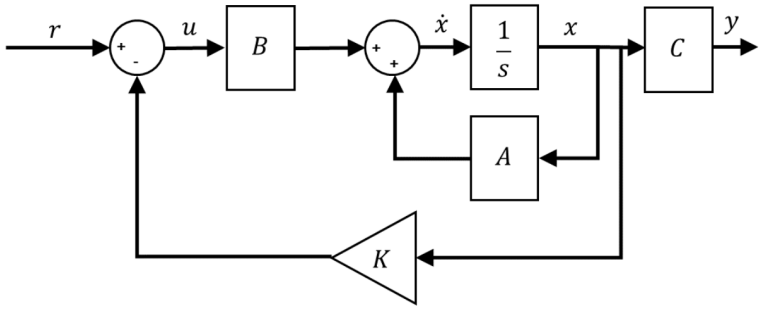


Figure 1.12: Block diagram of standard state-feedback control

1.5.3 Design and implementation of the LQR control

The goal is to build a QUARC controller similarly to Figure 1.13 that balances the pendulum on the QUBE-Servo 2 rotary pendulum system using a generated control gain K.

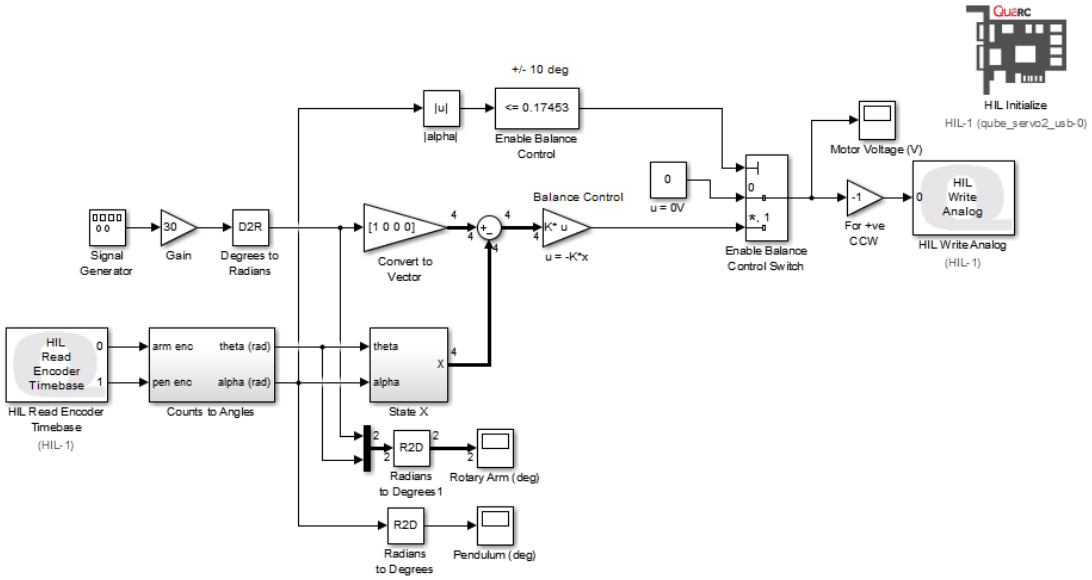


Figure 1.13: Simulink model used with QUARC run optimized balance controller

The LQR theory has been packaged in the Matlab *Control Design Module*. Given the model of the system, in the form of the state-space matrices A and B, and the weighting matrices Q and R, the LQR function computes the feedback control gain automatically.
In this experiment, as the state-space model is already available, the effect of changing the Q weighting matrix while R is fixed to 1 on the cost function J will be explored.

1.5.3.1 LQR control design

1. In Matlab, run the `setup_qube2_rotpen.m` script. This loads the QUBE-Servo 2 rotary pendulum state-space model matrices A, B, C, and D. The A and B matrices should be displayed in the Command Window:

```
A =

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

B =

         0
         0
   49.7275
   49.1493
```

2. Use the Matlab `eig` command to find the open-loop poles of the system. You should observe that the inverted rotary pendulum system is unstable because there is one pole in the right-hand plane.

3. Find and use the Matlab command to compute the controllability matrix and evaluates the controllability of the rotary inverted pendulum system. Look through the documentation to find the appropriate command if necessary.

4. Use the Matlab `lqr` function with the loaded model and the following weighting matrices

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

   to compute the optimal gain vector K.
   Using the following Matlab commands:

```
Q = eye(4,4);
R = 1;
K = lqr(A,B,Q,R);
```

   You should get the following control gain vector:

$$K = \begin{bmatrix} -1.00 & 34.24 & -1.23 & 3.08 \end{bmatrix}.$$

5. Use the Matlab `eig` command to find the poles of the closed-loop system state matrix $A - BK$. You should observe that the controlled inverted pendulum system is now stable as all poles are now in the left-hand plane.

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

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

   Using the following Matlab commands:

```
Q = diag([5 1 1 1]);
K = lqr(A,B,Q,R)
```

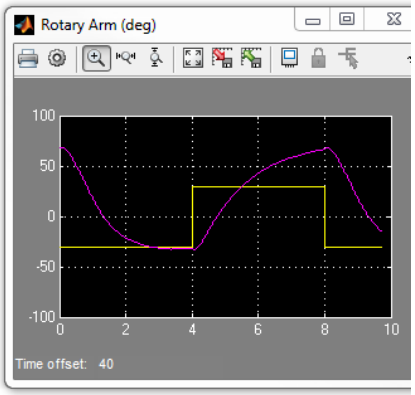
   You should get the following new gain vector:

$$K = \begin{bmatrix} -2.24 & 37.62 & -1.50 & 3.38 \end{bmatrix}$$

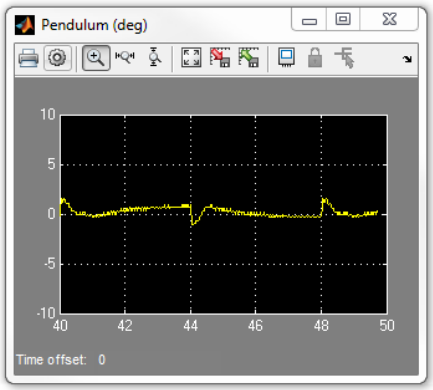
   Interestingly, changing the q_{11} element primarily increases the rotary arm proportional gain, $k_{p,\theta}$, but it also affects all the other gains. This makes sense when looking at the cost function in (1.30). Increasing q_{11} makes the LQR algorithm work harder to reduce the $x_1^2 = \theta^2$ term (as well as other states affected by this change), which augments the gain $k_{p,\theta}$.

1.5.3.2 LQR-based balance control

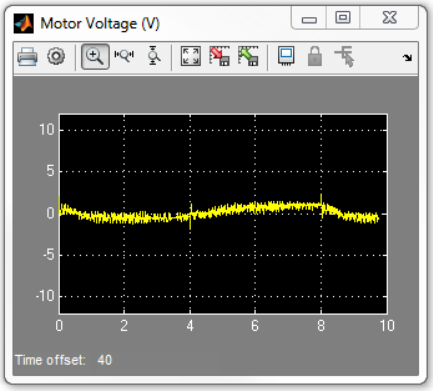
1. Run the `setup_qube2_rotpen.m` script in Matlab.
2. Open and understand the different elements in the Simulink model `q_qube2_bal_lqr.mdl`, shown in Figure 1.13
3. Load the gain K obtained in (4). Make sure it is set as variable `K` in the Matlab workspace.
4. Set the `Signal Generator` block to the following:
   - Type = `Square`
   - Amplitude = 1
   - Frequency = 0.125 Hz
5. Set the `Gain` block that is connected to the `Signal Generator` to 0.
6. Click on `Monitor & tune` to build and run the QUARC controller.
7. Manually rotate the pendulum in the upright position until the controller engages.
8. Once the pendulum is balanced, set the `Gain` to 30 to make the arm angle go between ±30°. The scopes should read something similar as shown in Figure 1.14. Attach your response of the rotary arm, pendulum, and controller voltage.



(a) Rotary Arm



(b) Pendulum



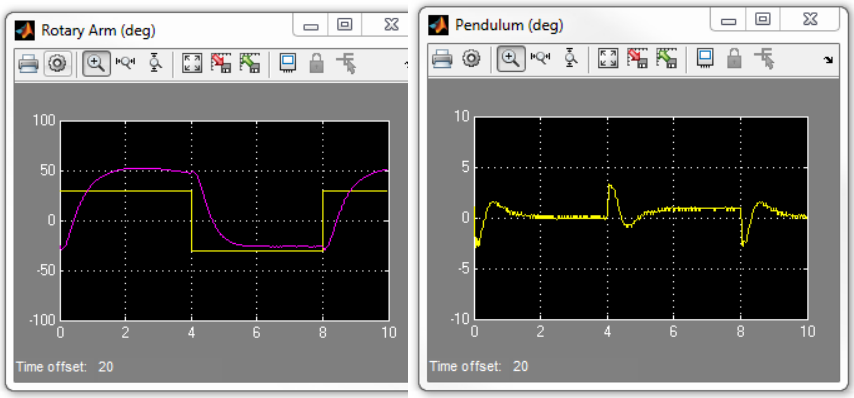
(c) Motor Voltage

Figure 1.14: QUBE Servo 2 rotary pendulum response

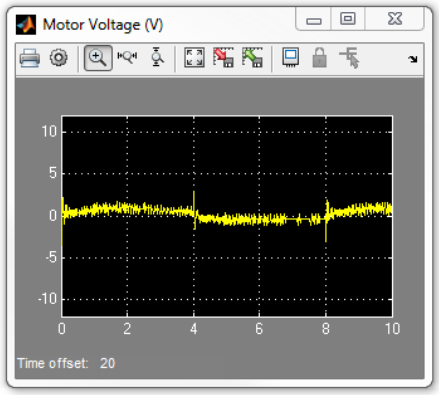
9. In Matlab, generate the gain using $Q = \texttt{diag}(\begin{bmatrix}5 & 1 & 1 & 1\end{bmatrix})$. The `diag` command specifies the diagonal elements in a square matrix.

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

11. Examine and describe the change in the *Rotary Arm (deg)* and *Pendulum (deg)* scopes. As shown in Figure 1.15, the arm response should become faster, i.e. peak time decreases, mainly due to the increased arm proportional gain. Because the rotary arm responds faster, the pendulum tends to deflect more from its vertical position.



(a) Rotary Arm (b) Pendulum



(c) Motor Voltage

Figure 1.15: QUBE Servo 2 rotary pendulum response when increasing q_{11} to 5

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

    Based on the analysis done in section 1.5.3.1, the elements in Q that are most likely to affect the pendulum gains are q_{22} and q_{44}. When going through the procedure outlined below, it will be observed that changing q_{22} has little affect on the generated gain K. Leaving q_{44} as the main variable to adjust.

    The procedure to find out which Q element affect the pendulum the most effectively is as follows:

    i - Adjust Q matrix element.

ii - Compute the gain K.

iii - Verify if the new gain has changed significantly. If not, then adjust another element or increase more. If it has changed, go to next step.

iv - Implement the gain on QUARC controller.

v - Examine the pendulum response.

13. List the resulting LQR Q matrix and control gain K used to yield the desired performance results.

    The gain generated using $Q = \texttt{diag}\left(\begin{bmatrix}5 & 1 & 1 & 20\end{bmatrix}\right)$ should be:

    $$K = \begin{bmatrix}-2.24 & 53.86 & -1.70 & 6.53\end{bmatrix}.$$

    The response when increasing the LQR gain element $q_{44} = 20$ is shown in Figure 1.16. As shown, the pendulum deflection is decreased from $\pm 4.5°$, as depicted in Figure 1.15, down to $\pm 1.9°$. Having larger proportional and derivative gains acting on the pendulum decreases the amount it deflects when the rotary arm rotates back-and-forth.

    Responses when increasing the LQR gain element $q_{44} = 20$ are shown in Figure 1.16.

14. Stop the QUARC controller.

1.6 Bonus - Automatic swing-up and balance control of the inverted pendulum

1.6.1 Non-linear energy-based swing-up control of the inverted pendulum

1.6.1.1 Energy control

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

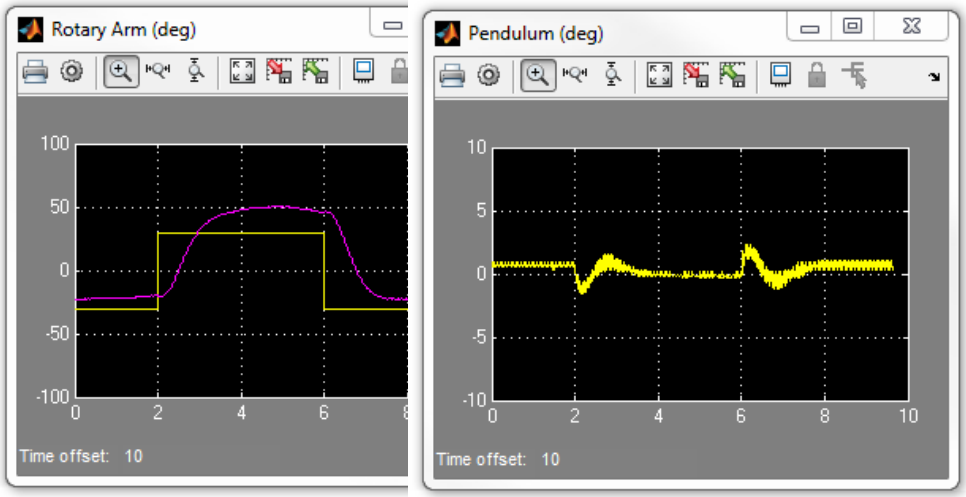
$$J_p\ddot{\alpha} + \frac{1}{2}M_p g L_p \sin\alpha = \frac{1}{2}M_p L_p u \cos\alpha. \tag{1.33}$$

Here, u is the linear acceleration of the pendulum.
The potential energy of the pendulum is

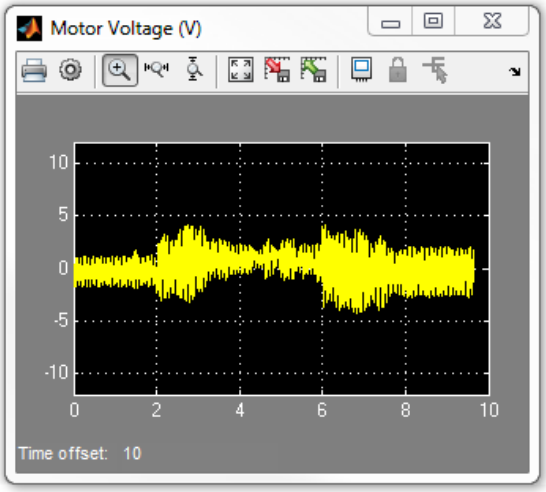
$$E_p = \frac{1}{2}M_p g L_p \left(1 - \cos\alpha\right),$$

and the kinetic energy is

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



(a) Rotary Arm (b) Pendulum



(c) Motor Voltage

Figure 1.16: QUBE Servo 2 rotary pendulum response with less pendulum deflection

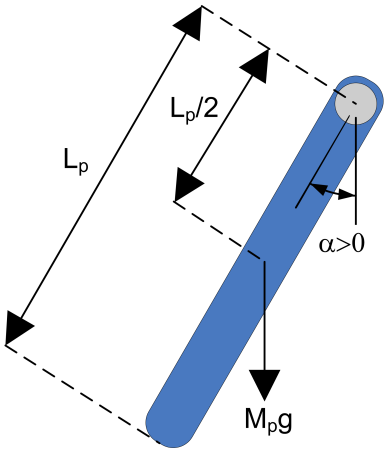


Figure 1.17: Free-body diagram of pendulum

The pendulum angle α and the lengths of the pendulum are illustrated in the free body diagram in Figure 1.17.
The potential energy is zero when the pendulum is at rest at $\alpha = 0$ and equals $M_p g L_p$ when the pendulum is upright at $\alpha = \pm\pi$. The sum of the potential and kinetic energy of the pendulum is

$$E = \frac{1}{2} J_p \dot{\alpha}^2 + \frac{1}{2} M_p g L_p \left(1 - \cos\alpha\right). \tag{1.34}$$

Differentiating (1.34) yields

$$\dot{E} = \dot{\alpha} \left(J_p \ddot{\alpha} + \frac{1}{2} M_p g L_p \sin\alpha \right). \tag{1.35}$$

Recalling (1.33) and rearranging terms as

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

eventually yields

$$\dot{E} = \frac{1}{2} M_p u L_p \dot{\alpha} \cos\alpha.$$

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

$$u = (E_r - E)\dot{\alpha}\cos\alpha. \tag{1.36}$$

By setting the reference energy to the pendulum potential energy ($E_r = E_p$), the control law will swing the link to its upright position. Notice that the control law is nonlinear because the proportional gain depends on the cosine of the pendulum angle α. Further, the control changes sign when $\dot{\alpha}$ changes sign and when the angle is ± 90 degrees.
For the system energy to change quickly, the magnitude of the control signal must be large. As a result the following swing-up controller is implemented in the controller as

$$u = \mathrm{sat}_{u_{max}} \left(\mu(E_r - E)\mathrm{sign}(\dot{\alpha}\cos\alpha)\right) \tag{1.37}$$

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

1.6.1.2 Hybrid swing-up control

The energy swing-up control in (1.36) (or (1.37)) can be combined with the balancing control law to obtain a control law that swings up the pendulum and then balances it.
Similarly as described in the Balance Control laboratory experiment, the balance control is to be enabled when the pendulum is within ±20 degrees. When it is not enabled, the swing-up control is engaged. Thus the switching can be described mathematically by:

$$u = \begin{cases} u_{bal} & \text{if } |\alpha| - \pi \leq 20^\circ \\ u_{swing\_up} & \text{otherwise} \end{cases} \tag{1.38}$$

1.6.2 Implementation of the swing-up and balance control

The Simulink model that swings up and balances the pendulum on the QUBE Servo 2 rotary pendulum system is shown in Figure 1.18. The *Swing-Up Control* subsystem implements the energy control described in section 1.6.1.

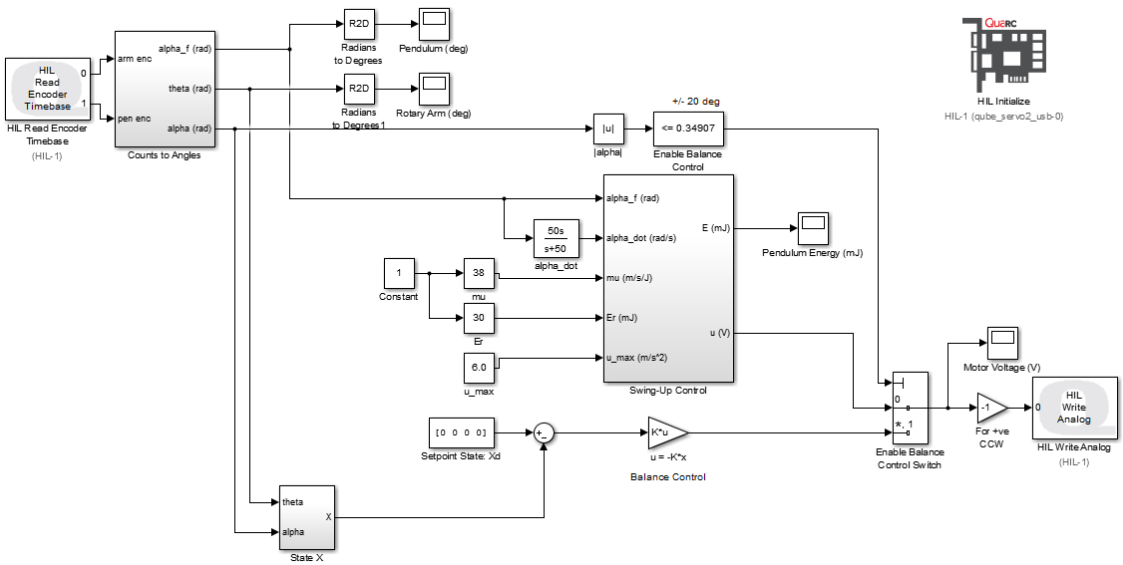


Figure 1.18: Simulink model used with QUARC to run swing-up controller

1.6.2.1 Energy control

1. Open the `q_qube_swingup.mdl` Simulink model.

2. Run the `setup_qube_rotpen.m` Matlab script. This loads the pendulum parameters that is used by the Simulink model.

3. To turn the swing-up control off, set the `Slider Gain` block called *mu* to 0.

4. Click on `Monitor & tune` to build and run the QUARC controller.

5. Manually rotate the pendulum at different levels and examine the pendulum angle and energy in the *Pendulum (deg)* and *Pendulum Energy (mJ)* scopes.

6. What do you notice about the energy when the pendulum is moved at different positions? Record the energy when the pendulum is being balanced (i.e. fully inverted in the upright

vertical position). Does this reading make sense in terms of the equations developed in section 1.6.1?

The pendulum energy should increase proportionally with the pendulum angle. When being balanced, the energy read is $30mJ$. As discussed in section 1.6.1, the reading corresponds to the potential energy of the pendulum: $E_r = M_p g L_p$.

7. Click on the `Stop` button to bring the pendulum down to the initial downward position.

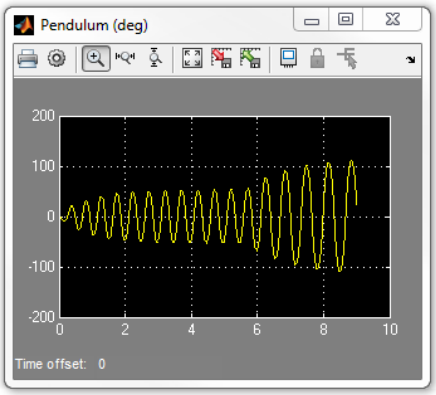
8. Set the swing-up control parameters (i.e. the `Constant` and `Gain` blocks connected to the inputs of the `Swing-Up Control` subsystem) to the following:

   - mu $= 50m/s/J$
   - Er $= 10.0mJ$
   - u_max $= 6m/s^2$

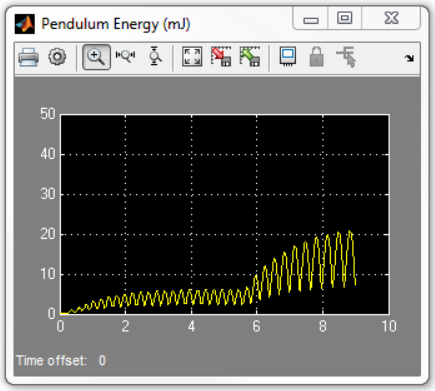
9. If the pendulum is not moving, gently perturb the pendulum with your hand to get it going.

10. Vary the reference energy, Er, between $10.0mJ$ and $20.0mJ$. 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.

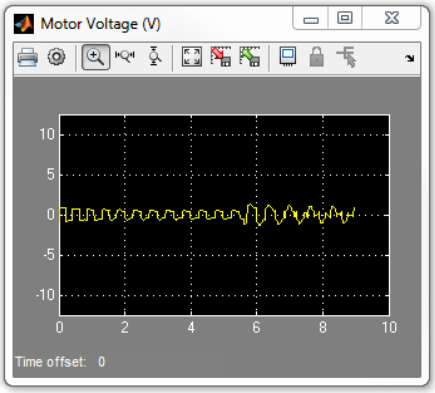
    Figures 1.19 show the responses when increasing the reference energy from 10 to $20mJ$.



(a) Pendulum Angle



(b) Pendulum Energy



(c) Motor Voltage

Figure 1.19: Pendulum response when changing reference energy

11. Fix Er to $20.0mJ$ and vary the swing-up control gain mu between 20 and $60m/s/J$.

    You should observe that, as the mu gain increases, the amplitude of the pendulum swings become larger. Recall from swing-up controller given in (1.37) that mu is the proportional gain.

12. Stop the QUARC controller.

1.6.3 Hybrid swing-up control

1. Open the `q_qube_swing_up.mdl` Simulink model.

2. Run the `setup_qube_rotpen.m` Matlab script. This loads the pendulum parameters that is used by the Simulink model.

3. Set the swing-up control parameters to the following:

   - mu = $20m/s/J$
   - u_max = $6m/s^2$

4. Based on your observations in the previous subsection 1.6.2.1, what should the reference energy E_r be set to? Set it to this value.

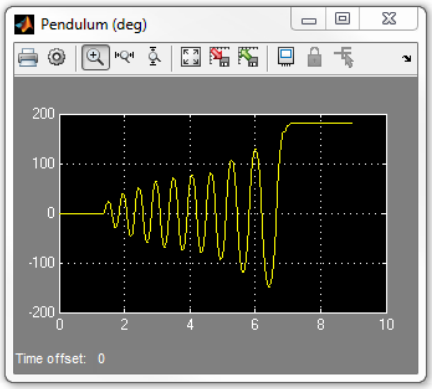
5. Make sure the pendulum is hanging down motionless and the encoder cable is not interfering with the pendulum.

6. Click on `Monitor & tune` to build and run the QUARC controller.

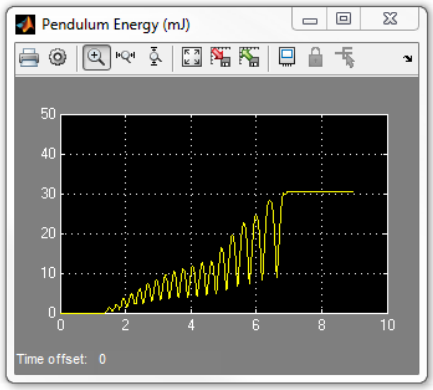
7. The pendulum should begin going back and forth. If not, manually perturb the pendulum with your hand. **Click on the `Stop` button in the Simulink tool bar if the pendulum goes unstable.**

8. Gradually increase the swing-up gain, μ, denoted as the mu `Slider Gain` block, until the pendulum swings up to the vertical position. Capture a response of the swing-up and record the swing-up gain that was required. Show the pendulum angle, pendulum energy, and motor voltage.

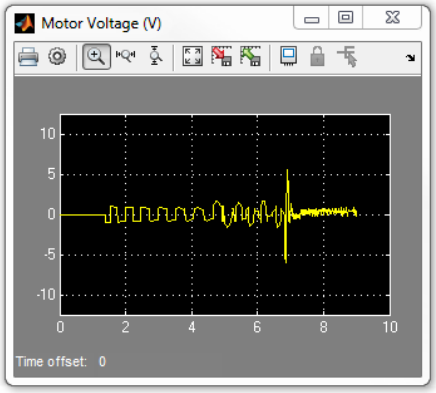
   The responses shown in Figure 1.20 are using hybrid swing-up control detailed in section 1.6.1 using $mu = 38m/s/J, Er = 30mJ$, and $u\_max = 6m/s^2$. The pendulum catch angle is set to $\pm 20°$.



(a) Pendulum angle



(b) Pendulum energy



(c) Motor voltage

Figure 1.20: Sample hybrid swing-up control response

9. Stop the QUARC controller.
10. Power *OFF* the QUBE Servo 2.

1.7 Final note

The techniques used to model, balance, and swing up an inverted pendulum have tremendous carry-over to other applications. State-space modelling is a mainstay to modeling complex MIMO systems. State-feedback control is sometimes used in multi degree-of-freedom robot manipulators, quadrotor systems, aerospace devices.