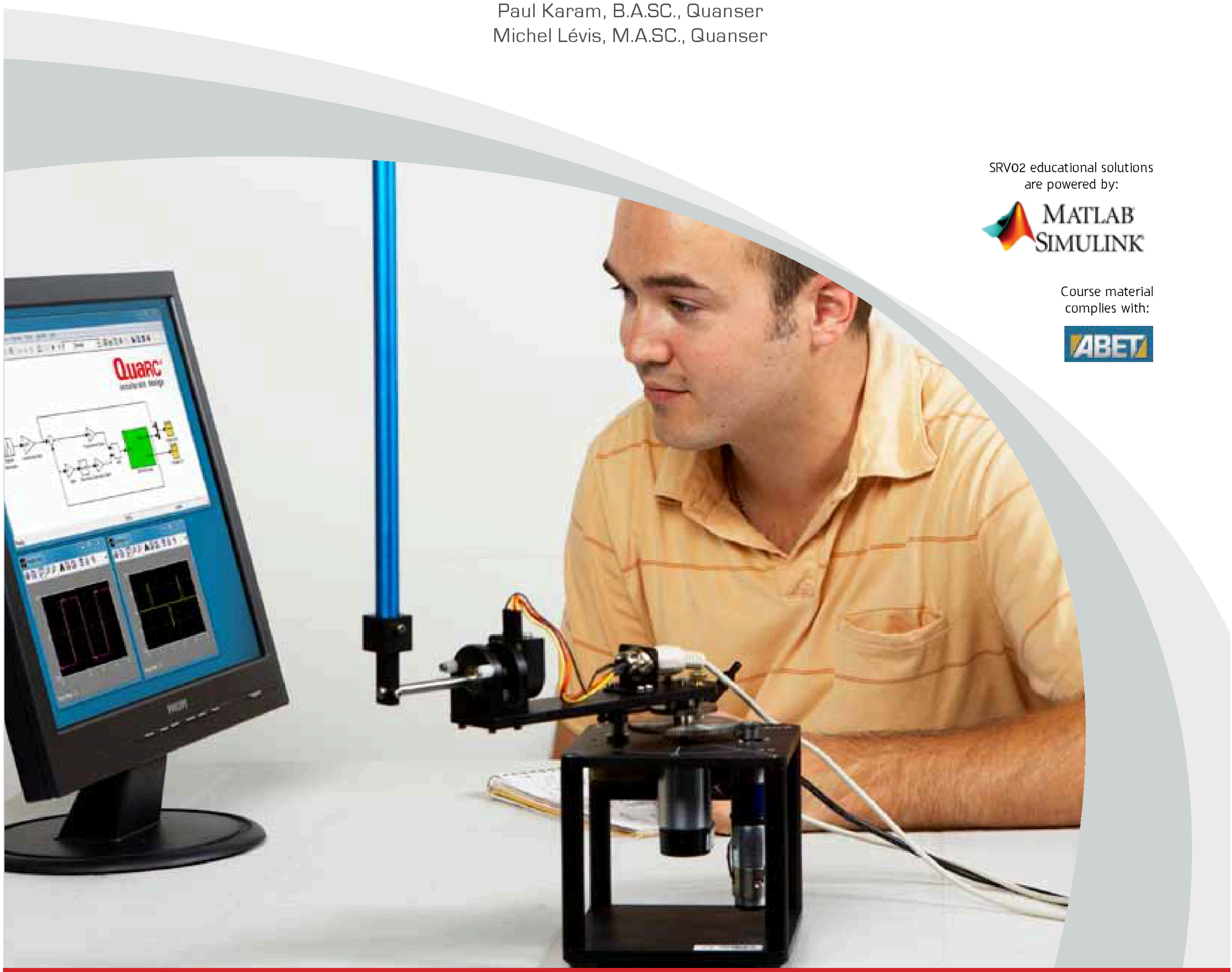
QUANSER
INNOVATE. EDUCATE.

STUDENT WORKBOOK

Inverted Pendulum Experiment for MATLAB®/Simulink® Users

Standardized for ABET* Evaluation Criteria

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



SRV02 educational solutions are powered by:

MATLAB SIMULINK

Course material complies with:

ABET

CAPTIVATE. MOTIVATE. GRADUATE.

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

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

Printed in Markham, Ontario.

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

ACKNOWLEDGEMENTS

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

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

Dr. K. J. Åström, Lund University, Lund, Sweden for his contributions to energy-based control, and

Andy Chang, National Instruments, Austin, Texas, for his help in designing this lab.

CONTENTS

1 INTRODUCTION

The objective of this laboratory is to design and implement a state-feedback control system that will balance the pendulum in the upright, vertical position.

Topics Covered

- Linearizing nonlinear equations of motion.
- Obtaining the linear state-space representation of the rotary pendulum plant.
- Designing a state-feedback control system that balances the pendulum in its upright vertical position using Pole Placement.
- Simulating the closed-loop system to ensure the specifications are met.
- Introduction to a nonlinear, energy-based swing up control.
- Implementing the controllers on the Quanser SRV02 Rotary Pendulum plant and evaluating its performance.

Prerequisites

- Know the basics of Matlab®and Simulink®.
- Understand state-space modeling fundamentals.
- Some knowledge of state-feedback.

2 MODELING

2.1 Background

2.1.1 Model Convention

The rotary inverted pendulum model is shown in Figure 2.1. The rotary arm pivot is attached to the SRV02 system and is actuated. The arm has a length of L_r, a moment of inertia of J_r, and its angle, θ, increases positively when it rotates counter-clockwise (CCW). The servo (and thus the arm) should turn in the CCW direction when the control voltage is positive, i.e., $V_m > 0$.

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 $\frac{L_p}{2}$. The moment of inertia about its center of mass is J_p. The inverted pendulum angle, α, is zero when it is perfectly upright in the vertical position and increases positively when rotated CCW.

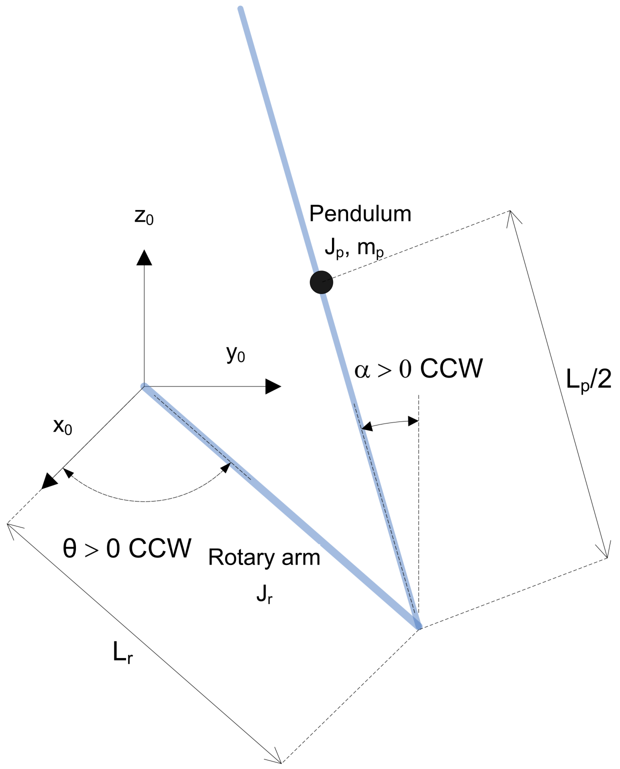


Figure 2.1: Rotary inverted pendulum conventions

2.1.2 Nonlinear Equations of Motion

Instead of using classical mechanics, the Lagrange method is used to find the equations of motion of the system. This systematic method is often used for more complicated systems such as robot manipulators with multiple joints.

More specifically, the equations that describe the motions of the rotary arm and the pendulum with respect to the servo motor voltage, i.e. the dynamics, will be obtained using the Euler-Lagrange equation:

$$\frac{\partial^2 L}{\partial t \partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i$$

The variables q_i are called *generalized coordinates*. For this system let

$$q(t)^{\top} = [\theta(t)\ \alpha(t)] \tag{2.1}$$

where, as shown in Figure 2.1, $\theta(t)$ is the rotary arm angle and $\alpha(t)$ is the inverted pendulum angle. The corresponding velocities are

$$\dot{q}(t)^{\top} = \left[\frac{\partial\theta(t)}{\partial t}\ \frac{\partial\alpha(t)}{\partial t}\right]$$

Note: The dot convention for the time derivative will be used throughout this document, i.e., $\dot{\theta} = \frac{d\theta}{dt}$. The time variable t will also be dropped from θ and α, i.e., $\theta = \theta(t)$ and $\alpha = \alpha(t)$.

With the generalized coordinates defined, the Euler-Lagrange equations for the rotary pendulum system are

$$\begin{aligned} \frac{\partial^2 L}{\partial t \partial \dot{\theta}} - \frac{\partial L}{\partial \theta} &= Q_1 \\ \frac{\partial^2 L}{\partial t \partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} &= Q_2 \end{aligned}$$

The Lagrangian of the system is described

$$L = T - V$$

where T is the total kinetic energy of the system and V is the total potential energy of the system. Thus the Lagrangian is the difference between a system's kinetic and potential energies.

The generalized forces Q_i are used to describe the non-conservative forces (e.g., friction) applied to a system with respect to the generalized coordinates. In this case, the generalized force acting on the rotary arm is

$$Q_1 = \tau - B_r\dot{\theta}$$

and acting on the pendulum is

$$Q_2 = -B_p\dot{\alpha}.$$

See [4] for a description of the corresponding SRV02 parameters (e.g. such as the back-emf constant, k_m). Our control variable is the input servo motor voltage, V_m. Opposing the applied torque is the viscous friction torque, or viscous damping, corresponding to the term B_r. Since the pendulum is not actuated, the only force acting on the link is the damping. The viscous damping coefficient of the pendulum is denoted by B_p.

The Euler-Lagrange equations is a systematic method of finding the equations of motion, i.e., EOMs, of a system. Once the kinetic and potential energy are obtained and the Lagrangian is found, then the task is to compute various derivatives to get the EOMs. After going through this process, the nonlinear equations of motion for the SRV02 rotary inverted pendulum are:

$$\begin{aligned} \left(m_p L_r^2 + \frac{1}{4}m_p L_p^2 - \frac{1}{4}m_p L_p^2 \cos(\alpha)^2 + J_r\right)\ddot{\theta} - \left(\frac{1}{2}m_p L_p L_r \cos(\alpha)\right)\ddot{\alpha} & \\ + \left(\frac{1}{2}m_p L_p^2 \sin(\alpha)\cos(\alpha)\right)\dot{\theta}\dot{\alpha} + \left(\frac{1}{2}m_p L_p L_r \sin(\alpha)\right)\dot{\alpha}^2 &= \tau - B_r\dot{\theta} \qquad (2.2) \\ -\frac{1}{2}m_p L_p L_r \cos(\alpha)\ddot{\theta} + \left(J_p + \frac{1}{4}m_p L_p^2\right)\ddot{\alpha} - \frac{1}{4}m_p L_p^2 \cos(\alpha)\sin(\alpha)\dot{\theta}^2 & \\ -\frac{1}{2}m_p L_p g \sin(\alpha) &= -B_p\dot{\alpha}. \qquad (2.3) \end{aligned}$$

The torque applied at the base of the rotary arm (i.e., at the load gear) is generated by the servo motor as described by the equation

$$\tau = \frac{\eta_g K_g \eta_m k_t (V_m - K_g k_m \dot{\theta})}{R_m}. \tag{2.4}$$

See [4] for a description of the corresponding SRV02 parameters (e.g. such as the back-emf constant, k_m).

Both the equations match the typical form of an EOM for a single body:

$$J\ddot{x} + b\dot{x} + g(x) = \tau_1$$

where x is an angular position, J is the moment of inertia, b is the damping, $g(x)$ is the gravitational function, and τ_1 is the applied torque (scalar value).

For a generalized coordinate vector q, this can be generalized into the matrix form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{2.5}$$

where D is the inertial matrix, C is the damping matrix, $g(q)$ is the gravitational vector, and τ is the applied torque vector.

The nonlinear equations of motion given in 2.2 and 2.3 can be placed into this matrix format.

2.1.3 Linearizing

Here is an example of how to linearize a two-variable nonlinear function called $f(z)$. Variable z is defined

$$z^{\top} = [z_1 \; z_2]$$

and $f(z)$ is to be linearized about the operating point

$$z_0{}^{\top} = [a \; b]$$

The linearized function is

$$f_{lin} = f(z_0) + \left(\frac{\partial f(z)}{\partial z_1}\right)\Big|_{z=z_0}(z_1 - a) + \left(\frac{\partial f(z)}{\partial z_2}\right)\Big|_{z=z_0}(z_2 - b)$$

2.1.4 Linear State-Space Model

The linear state-space equations are

$$\dot{x} = Ax + Bu \tag{2.6}$$

and

$$y = Cx + Du \tag{2.7}$$

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

$$x^{\top} = [\theta \; \alpha \; \dot{\theta} \; \dot{\alpha}] \tag{2.8}$$

and

$$y^{\top} = [x_1 \; x_2]. \tag{2.9}$$

In the output equation, only the position of the servo and link angles are being measured. Based on this, the C and D matrices in the output equation are

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.10}$$

and

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.11}$$

The velocities of the servo and pendulum angles can be computed in the digital controller, e.g., by taking the derivative and filtering the result though a high-pass filter.

2.2 Pre-Lab Questions

1. Linearize the first nonlinear inverted rotary pendulum equation, Equation 2.2. The initial conditions for all the variables are zero, i.e., $\theta_0 = 0$, $\alpha_0 = 0$, $\dot{\theta}_0 = 0$, $\dot{\alpha}_0 = 0$.

2. Linearize the second nonlinear inverted rotary pendulum equation, Equation 2.3, with initial conditions $\theta_0 = 0$, $\alpha_0 = 0$, $\dot{\theta}_0 = 0$, $\dot{\alpha}_0 = 0$.

3. Fit the two linear equations of motion found in the above exercises into the matrix form shown in Equation 2.5. Make sure the equation is in terms of θ and α (and its derivatives).

4. Solve for the acceleration terms in the equations of motion. You can either solve this using the two linear equations or using the matrix form. If you're doing it in the matrix form, recall that the inverse of a 2x2 matrix is

$$A^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\text{det}(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}, \tag{2.12}$$

   where det$(A) = ad - bc$.

   In any case, you'll have two equations of the form: $\ddot{\theta} = g_1(\theta, \alpha, \dot{\theta}, \dot{\alpha})$ and $\ddot{\alpha} = g_2(\theta, \alpha, \dot{\theta}, \dot{\alpha})$. Make sure you collect the terms with respect to the θ, α, $\dot{\theta}$, and $\dot{\alpha}$ variables.

5. Find the linear state-space of the rotary inverted pendulum system. Make sure you give the A and B matrices (C and D have already been given in Section 2.1).

2.3 In-Lab Exercises

2.3.1 Simulation: Model Analysis

1. Run the *setup_rotpen_student.m* script. The SRV02 and pendulum model parameters are automaticaly loaded using the *config_srv02.m* and *config_sp.m* functions. It then calls the *ROPTEN_ABCD_eqns_student.m* script to load the model in the Matlab workspace.
2. Open the *ROTPEN_ABCD_eqns_student.m* script. The script should contain the following code:

```
% State Space Representation
A = eye(4,4);
B = [0;0;0;1];
C = eye(2,4);
D = zeros(2,1);

% Add actuator dynamics
A(3,3) = A(3,3) - Kg^2*kt*km/Rm*B(3);
A(4,3) = A(4,3) - Kg^2*kt*km/Rm*B(4);
B = Kg * kt * B / Rm;

system = ss(A,B,C,D);
```

   The representative C and D matrices have already been included. You need to enter the state-space matrices A and B that you found in Section 2.2. The actuator dynamics have been added to convert your state-space matrices to be in terms of voltage. Recall that the input of the state-space model you found in Section 2.2 is the torque acting at the servo load gear (i.e., the pivot of the pendulum). However, *we do not control torque directly - we control the servo input voltage*. The above code uses the voltage-torque relationship given in Equation 2.4 in Section 2.1.2 to tranform torque to voltage.

3. Run the *ROTPEN_ABCD_eqns_student.m* script to load the state-space matrices in the Matlab workspace. Show the numerical matrices that are displayed in the Matlab prompt.
4. Find the open-loop poles of the system.

Before ending this lab... To do the pre-lab questions in Section 3.3, you need the A and B matrices (numerical representation) and the open-loop poles. Make sure you record these.

2.3.2 Implementation: Calibration

Experimental Setup

The *q_rotpen_model_student* Simulink diagram shown in Figure 2.2 is used to confirm that the actual system hardware matches the modeling conventions. It is also a good check that the system is connected properly. The QUARC blocks are used to interface with encoders of the system. For more information about QUARC, see Reference [3]. This model outputs the rotary arm and pendulum link angles and can apply a voltage to the DC motor.

IMPORTANT: Before you can conduct this experiment, you need to make sure that the lab files are configured according to your system setup. If they have not been configured already, then you need to go to Section 5.3 to configure the lab files first.

1. Run the *setup_rotpen.m* script to load your Rotary Pendulum model.
2. In the *q_rotpen_model_student* Simulink diagram, go to QUARC | Build to build the QUARC controller.
3. Turn ON the power amplifier.

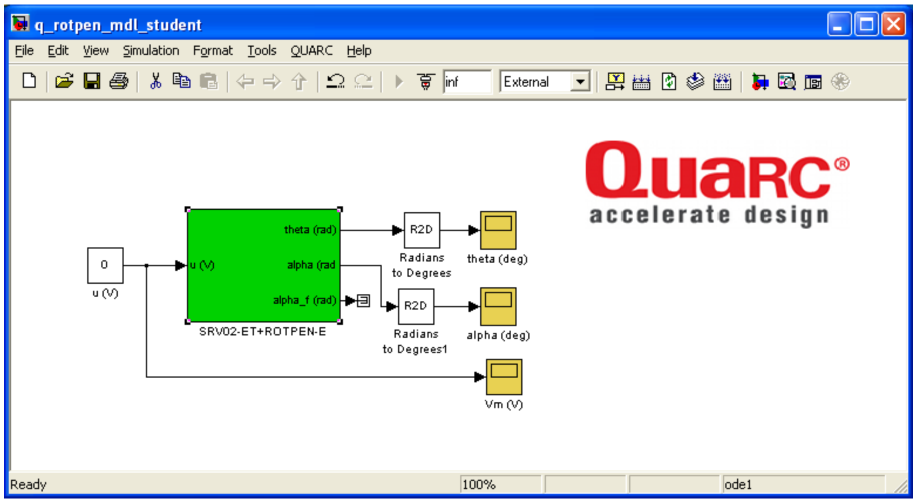


Figure 2.2: q_rotpen_mdl_student Simulink diagram used to confirm modeling conventions

4. Go to QUARC | Start to run the controller.
5. Rotate the arm and the pendulum in the counter-clockwise direction and examine the direction of their response. Does the direction of these measurements agree with the modeling conventions given in Section 2.1.1? Explain why or why not.
6. Go to the *SRV02-ET+ROTPEN-E* subsystem block, shown in Figure 2.3.

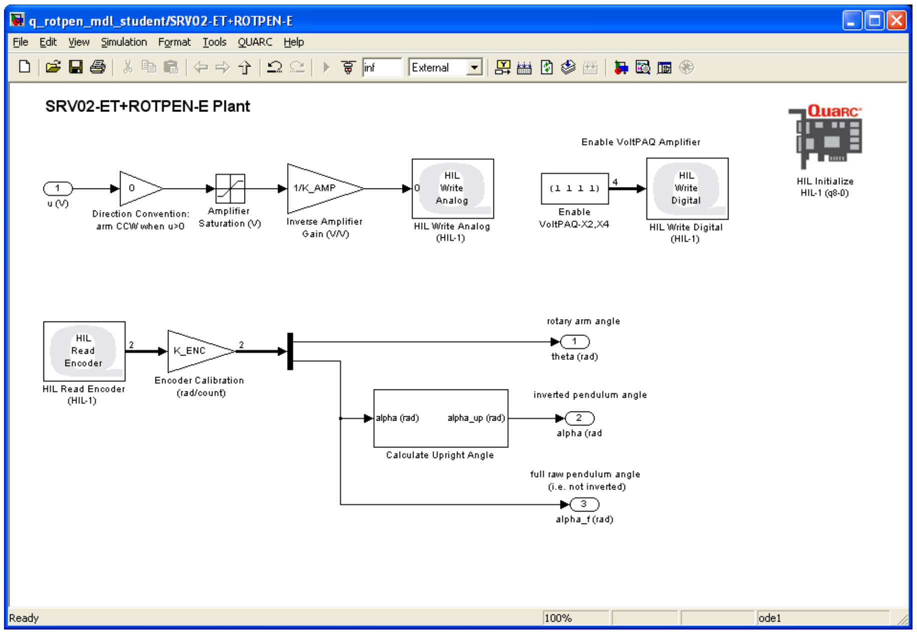


Figure 2.3: SRV02-ET+ROTPEN-E Subsystem - Student Version

7. The Source block called *u (V)* in *q_rotpen_mdl_student* Simulink diagram is the control input. When you set *u (V)* to 1 V, the rotary arm must move according to the model conventions that were defined in Section 2.1.1. As shown in Figure 2.3, the *Direction Convention* Gain block is currently set to 0. Change this value such that the model conventions are adhered to. Plot the rotary arm response and the motor voltage in a Matlab figure when 1 V is applied.

   Note: When the controller stops, the last 10 seconds of data is automatically saved in the Matlab workspace to the variables *data_theta* and *data_Vm*. The time is stored in *data_alpha(:,1)* vector, the pendulum angle is stored the *data_alpha(:,2)* vector, and the control input is in the *data_Vm(:,2)* structure.

8. Click on the STOP button to stop running the QUARC controller.
9. Shut off the power amplifier.

2.4 Results

Fill out Table 1 with your answers from your modeling lab results - both simulation and implementation.

Description	Symbol	Value	Units
State-Space Matrix	A		
State-Space Matrix	B		
State-Space Matrix	C		
State-Space Matrix	D		
Open-loop poles	OL		

Table 1: Results

3 BALANCE CONTROL

3.1 Specifications

The control design and time-response requirements are:

Specification 1: Damping ratio: $\zeta = 0.7$.

Specification 2: Natural frequency: $\omega_n = 4$ rad/s.

Specification 3: Maximum pendulum angle deflection: $|\alpha| < 15$ deg.

Specification 4: Maximum control effort / voltage: $|V_m| < 10$ V.

The necessary closed-loop poles are found from specifications 1 and 2. The pendulum deflection and control effort requirements (i.e., specifications 3 and 4) are to be satisfied when the rotary arm is tracking a ± 20 degree angle square wave.

3.2 Background

In Section 2, we found a linear state-state space model that represents the inverted rotary pendulum system. This model is used to investigate the inverted pendulum stability properties in Section 3.2.1. In Section 3.2.2, the notion of controllabitliy is introduced. The procedure to transform matrices to their companion form is described in Section 3.2.3. Once in their companion form, it is easier to design a gain according to the pole-placement principles, which is discussed in Section 3.2.4. Lastly, Section 3.2.6 describes the state-feedback control used to balance the pendulum.

3.2.1 Stability

The stability of a system can be determined from its poles ([8]):

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

The poles are the roots of the system's characteristic equation. From the state-space, the characteristic equation of the system can be found using

$$\det(sI - A) = 0$$

where *det()* is the determinant function, s is the Laplace operator, and I the identity matrix. These are the *eigenvalues* of the state-space matrix A.

3.2.2 Controllability

If the control input u of a system can take each state variable, x_i where $i = 1 \ldots n$, from an initial state to a final state then the system is controllable, otherwise it is uncontrollable ([8]).

Rank Test The system is controllable if the rank of its controllability matrix

$$T = \left[B\ AB\ A^2B \ldots A^nB\right] \tag{3.1}$$

equals the number of states in the system,

$$\text{rank}(T) = n.$$

3.2.3 Companion Matrix

If (A, B) are controllable and B is $n \times 1$, then A is similar to a companion matrix ([1]). Let the characteristic equation of A be

$$s^n + a_n s^{n-1} + \ldots + a_1.$$

Then the companion matrices of A and B are

$$\tilde{A} = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \end{bmatrix} \tag{3.2}$$

and

$$\tilde{B} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{3.3}$$

Define

$$W = T\tilde{T}^{-1}$$

where T is the controllability matrix defined in Equation 3.1 and

$$\tilde{T} = [\tilde{B} \; \tilde{B}\tilde{A} \ldots \tilde{B}\tilde{A}^n].$$

Then

$$W^{-1}AW = \tilde{A}$$

and

$$W^{-1}B = \tilde{B}.$$

3.2.4 Pole Placement

If (A,B) are controllable, then pole placement can be used to design the controller. Given the control law $u = -Kx$, the state-space in Equation 2.6 becomes

$$\begin{aligned} \dot{x} &= Ax + B(-Kx) \\ &= (A - BK)x \end{aligned}$$

To illustate how to design gain K, consider the following system

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 3 & -1 & -5 \end{bmatrix} \tag{3.4}$$

and

$$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.5}$$

Note that A and B are already in the companion form. We want the closed-loop poles to be at $[-1 - 2 - 3]$. The *desired* characteristic equation is therefore

$$(s+1)(s+2)(s+3) = s^3 + 6s^2 + 11s + 6 \tag{3.6}$$

For the gain $K = [k_1\ k_2\ k_3]$, apply control $u = -Kx$ and get

$$A - KB = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 3 - k_1 & -1 - k_2 & -5 - k_3 \end{bmatrix}.$$

The characteristic equation of $A - KB$ is

$$s^3 + (k_3 + 5)s^2 + (k_2 + 1)s + (k_1 - 3) \tag{3.7}$$

Equating the coefficients between Equation 3.7 and the desired polynomial in Equation 3.6

$$\begin{aligned} k_1 - 3 &= 6 \\ k_2 + 1 &= 11 \\ k_3 + 5 &= 6 \end{aligned}$$

Solving for the gains, we find that a gain of $K = [9\ 10\ 1]$ is required to move the poles to their desired location.

We can generalize the procedure to design a gain K for a controllable (A,B) system as follows:

Step 1 Find the companion matrices $\tilde{A}$ and $\tilde{B}$. Compute $W = T\tilde{T}^{-1}$.

Step 2 Compute $\tilde{K}$ to assign the poles of $\tilde{A} - \tilde{B}\tilde{K}$ to the desired locations. Applying the control law $u = -Kx$ to the general system given in Equation 3.2,

$$\tilde{A} = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_1 - k_1 & -a_2 - k_2 & \cdots & -a_{n-1} - k_{n-1} & -a_n - k_n \end{bmatrix} \tag{3.8}$$

Step 3 Find $K = \tilde{K}W^{-1}$ to get the feedback gain for the original system (A,B).

Remark: It is important to do the $\tilde{K} \rightarrow K$ conversion. Remember that (A,B) represents the actual system while the companion matrices $\tilde{A}$ and $\tilde{B}$ do not.

3.2.5 Desired Poles

The rotary inverted pendulum system has four poles. As depicted in Figure 3.1, 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, specifications given in Section 3.1. Let the conjugate poles be

$$p_1 = -\sigma + j\omega_d \tag{3.9}$$

and

$$p_2 = -\sigma - j\omega_d \tag{3.10}$$

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 3.1.

3.2.6 Feedback Control

The feedback control loop that balances the rotary pendulum is illustrated in Figure 3.2. The reference state is defined

$$x_d = [\theta_d\ 0\ 0\ 0]$$

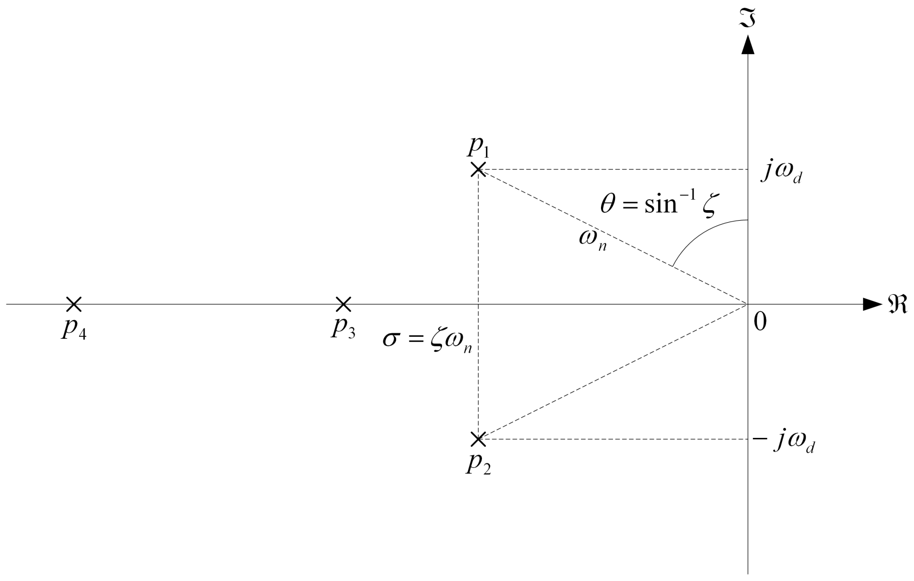


Figure 3.1: Desired closed-loop pole locations

where θ_d is the desired rotary arm angle. The controller is

$$u = K(x_d - x). \tag{3.11}$$

Note that if $x_d = 0$ then $u = -Kx$, which is the control used in the pole-placement algorithm.

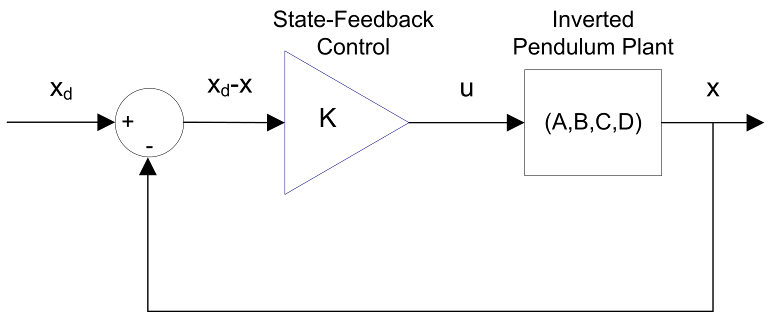


Figure 3.2: State-feedback control loop

When running this on the actual system, the pendulum begins in the hanging, downward position. We only want the balance control to be enabled when the pendulum is brought up around its upright vertical position. The controller is therefore

$$u = \begin{cases} K(x_d - x) & |x_2| < \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

where ϵ is the angle about which the controller should engage. For example if $\epsilon = 10$ degrees, then the control will begin when the pendulum is within ± 10 degrees of its upright position, i.e., when $|x_2| < 10$ degrees.

3.3 Pre-Lab Questions

1. Based on your analysis in Section 2.3, is the system stable, marginally stable, or unstable? Did you expect the stability of the inverted pendulum to be as what was determined?
2. Using the open-loop poles, find the characteristic equation of A.
3. Give the corresponding companion matrices $\tilde{A}$ and $\tilde{B}$. Do not compute the transformation matrix W (this will be done in the lab using QUARC®).
4. Find the location of the two dominant poles, p_1 and p_2, based on the specifications given in Section 3.1. Place the other poles at $p_3 = -30$ and $p_4 = -40$. Finally, give the desired characteristic equation.
5. When applying the control $u = -\tilde{K}x$ to the companion form, it changes $(\tilde{A}, \tilde{B})$ to $(\tilde{A} - \tilde{B}\tilde{K}, \tilde{B})$. Find the gain $\tilde{K}$ that assigns the poles to their new desired location.

3.4 In-Lab Exercises

3.4.1 Control Design

1. Run the *setup_rotpen_student.m* script to load the rotary pendulum the model you found in pervious modeling lab.

2. Using Matlab commands, determine if the system is controllable. Explain why.

3. Open the *d_pole_placement_student.m* script. As shown below, the companion matrices $\tilde{A}$ and $\tilde{B}$ for the model are automatically found (denoted as Ac and Bc in Matlab).

```
% Characteristic equation: s^4 + a_4*s^3 + a_3*s^2 + a_2*s + a_1
a = poly(A);
%
% Companion matrices (Ac, Bc)
Ac = [  0 1 0 0;
        0 0 1 0;
        0 0 0 1;
        -a(5) -a(4) -a(3) -a(2)];
%
Bc = [0; 0; 0; 1];
% Controllability
T = 0;
% Controllability of companion matrices
Tc = 0;
% Transformation matrices
W = 0;
```

   In order to find the gain K, we need to find the transformation matrix $W = T\tilde{T}^{-1}$ (note: $\tilde{T}$ is denoted as T_c in Matlab). Modify the *d_pole_placement_student.m* script to calculate the controllability matrix T, the companion controllabilty matrix Tc, the inverse of Tc, and W. Show your completed script and the resulting T, Tc, Tc^{-1}, and W matrices.

4. Enter the companion gain, $\tilde{K}$, you found in the pre-lab as *Kc* in *d_pole_placement_student.m* and modify it to find gain K using the transformation detailed in Section 3. Run the script again to calculate the feedback gain K and record its value in Table 2.

5. Evaluate the closed-loop poles of the system, i.e., the eigenvalues of $A - BK$. Record the closed-loop poles of the system when using the gain K calculated above. Have the poles been placed to their desired locations? If not, then go back and re-investigate your control design until you find a gain that positions the poles to the required location.

6. In the previous exercises, gain K was found manually through matrix operations. All that work can instead be done using a pre-defined *Compensator Design* Matlab command. Find gain K using a Matlab pole-placement command and verify that the gain is the same as generated before.

3.4.2 Simulating the Balance Control

Experiment Setup

The *s_rotpen_bal* Simulink diagram shown in Figure 3.3 is used to simulate the closed-loop response of the Rotary Pendulum using the state-feedback control described in Section 3 with the control gain K found in Section 3.4.1.

The *Signal Generator* block generates a 0.1 Hz square wave (with amplitude of 1). The *Amplitude (deg)* gain block is used to change the desired rotary arm position. The state-feedback gain K is set in the *Control Gain* gain block and is read from the Matlab workspace. The Simulink *State-Space* block reads the A, B, C, and D state-space matrices

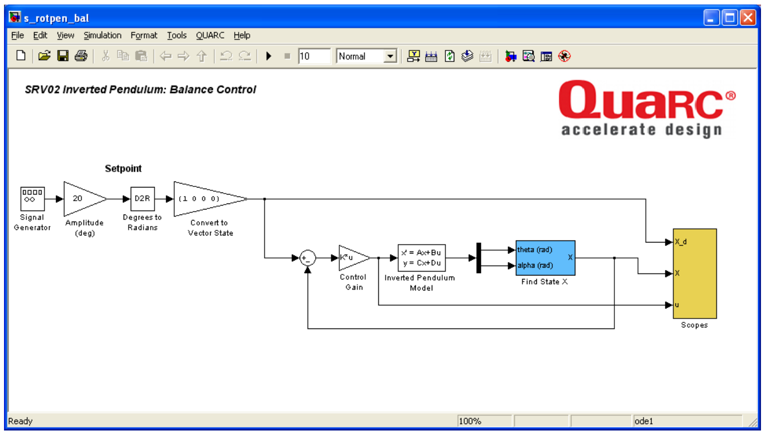


Figure 3.3: s_rotpen_bal Simulink diagram used to simulate the state-feedback control

that are loaded in the Matlab workspace. The *Find State X* block contains high-pass filters to find the velocity of the rotary arm and pendulum.

IMPORTANT: Before you can conduct this experiment, you need to make sure that the lab files are configured according to your system setup. If they have not been configured already, go to Section 5.4 to configure the lab files first. **Make sure the model you found in Section 2.3 is entered in ROTPEN_ABCD_eqns_student.m.**

1. Run *setup_rotpen.m*. Ensure the gain K you found in Section 3.4.1 is loaded.
2. Run the *s_rotpen_bal.mdl*. The response in the scopes shown in Figure 3.4 were generated using an arbitrary feedback control gain. Plot the simulated response of rotary arm, pendulum, and motor input voltage obtained using your obtained gain K in a Matlab figure and attach it to your report.

   Note: When the simulation stops, the last 10 seconds of data is automatically saved in the Matlab workspace to the variables *data_theta*, *data_alpha*, and *data_Vm*. The time is stored in the *data_theta(:,1)* vector, the desired and measured rotary arm angles are saved in the *data_theta(;,2)* and *data_theta(;,3)* arrays, the pendulum angle is stored the *data_alpha(:,2)* vector, and the control input is in the *data_Vm(:,2)* structure.
3. Measure the pendulum deflection and voltage used. Are the specifications given in Section 3.1 satisfied?
4. Close the Simulink diagram when you are done.

3.4.3 Implementing the Balance Controller

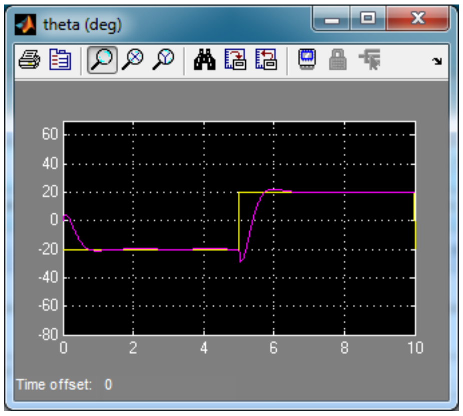
In this section, ths state-feedback control that was designed and simulated in the previous sections is run on the actual SRV02 Rotary Pendulum device.

Experiment Setup

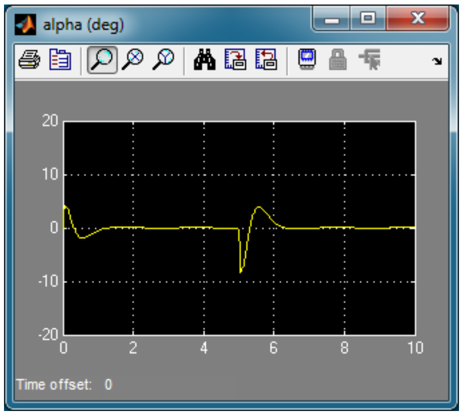
The *q_rotpen_bal_student* Simulink diagram shown in Figure 3.5 is used to run the state-feedback control on the Quanser Rotary Pendulum system. The *SRV02-ET+ROTPEN-E* subsystem contains QUARC blocks that interface with the DC motor and sensors of the system. The feedback developed in Section 3.4.1 is implemented using a Simulink *Gain* block.

IMPORTANT: Before you can conduct this experiment, you need to make sure that the lab files are configured according to your system setup. If they have not been configured already, then go to Section 5.5 to configure the lab files first.

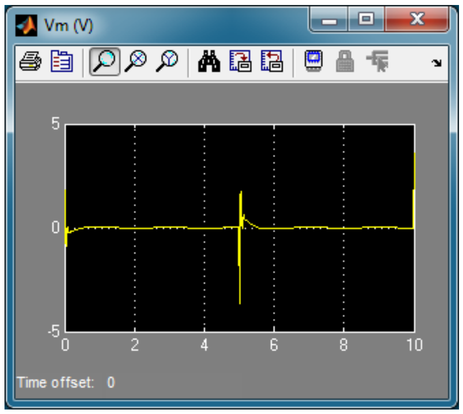
1. Run the *setup_rotpen.m* script.
2. Make sure the gain K you found in Section 3.4.1 is loaded.



(a) Rotary Arm Angle



(b) Inverted Pendulum Angle



(c) Motor Voltage

Figure 3.4: Balance control simulation using default gain

3. Open the q_rotpen_bal_student Simulink diagram.
4. Turn ON the power amplifier.
5. Go to QUARC | Build to build the controller.
6. Go to QUARC | Start to run the controller.

   Ensure the modifications you made in the Modeling Laboratory (Section 2.3) have been applied. Verify that the model conventions still hold (e.g., motor turns in expected way when a positive voltage is supplied).
7. As shown in Figure 3.5, the Simulink diagram is incomplete. Add the blocks from the Simulink library to implement the balance control. When implementing the control, keep in mind the following: unlike in the simulation, where the pendulum is already upright, the pendulum begins in the hanging down position. Thus when the controller starts, the inverted pendulum angle reads ± 180 and it goes up to zero when brought to the upright position. You will need to add a switch logic to implement Equation 3.12.
8. Ensure the pendulum is in the hanging down position and is motionless. Go to QUARC | Build and QUARC | Run to start the QUARC controller. Once it is running, manually bring up the pendulum to its upright vertical position. You should feel the voltage kick-in when it is within the range where the balance control engages. Once it is balanced, introduce the ± 20 degree rotary arm command by setting the *Amplitude (deg)* gain in the Simulink diagram to 20.
9. The response should look similar to your simulation. Once you have obtained a response, click on the STOP button to stop the controller. Be careful, as the pendulum will fall down when the controller is stopped. Similarly as in the simulation, the response data will be saved to the workspace. Use this to plot the rotary arm, pendulum, and control input responses in a Matlab figure (see Section 3.4.2 for more information on plotting).

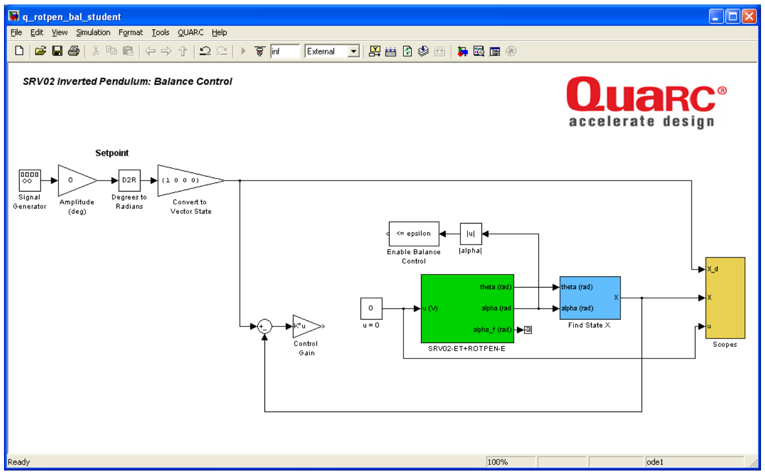


Figure 3.5: q_rotpen_bal_student Simulink diagram can be used to run balance controller

10. Measure the pendulum deflection and voltage used. Are the specifications given in Section 3.1 satisfied for the implementation?

11. Shut off the power amplifier.

3.5 Results

Fill out Table 2 with your answers from your control lab results - both simulation and implementation.

Description	Symbol	Value	Units
Pre Lab Questions			
Desired poles	DP		
Companion Gain	$\tilde{K}$		
Simulation: Control Design			
Transformation Matrix	W		
Control Gain	K		
Closed-loop poles	CLP		
Simulation: Closed-Loop System			
Maximum deflection	$\vert\alpha\vert_{max}$		deg
Maximum voltage	$\vert V_m\vert_{max}$		V
Implementation			
Control Gain	K		
Maximum deflection	$\vert\alpha\vert_{max}$		deg
Maximum voltage	$\vert V_m\vert_{max}$		V

Table 2: Results

4 SWING-UP CONTROL

4.1 Background

In this section a nonlinear, energy-based control scheme is developed to swing the pendulum up from its hanging, downward position. The swing-up control described herein is based on the strategy outlined in [9]. Once upright, the control developed in Section 3 can be used to balance the pendulum in the upright vertical position.

4.1.1 Pendulum Dynamics

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

$$J_p\ddot{\alpha} + \frac{1}{2}m_p g L_p \sin(\alpha) = \frac{1}{2}m_p L_p u \cos(\alpha). \tag{4.1}$$

The pivot accleration, u, is the linear acceleration of the pendulum link base. The acceleration is proportional to the torque of the rotary arm and is expressed as

$$\tau = m_r L_r u \tag{4.2}$$

where m_r is the mass of the rotary arm and L_r is its length, as shown in Section 2. The voltage-torque relationship is given in Equation 2.4.

4.1.2 Energy Control

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

The potential and kinetic energy of the pendulum is

$$E_p = \frac{1}{2}m_p g L_p(1 - \cos(\alpha)) \tag{4.3}$$

and

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

The pendulum parameters are described in Section 2 and their values are given in [6]. In the potential energy calculation, we assume the center of mass to be in the center of the link, i.e., $\frac{L_p}{2}$. Adding the kinetic and potential energy together give us the total pendulum energy

$$E = \frac{1}{2}J_p\dot{\alpha}^2 + \frac{1}{2}m_p g L_p(1 - \cos\alpha). \tag{4.4}$$

Taking its time derivative we get

$$\dot{E} = \dot{\alpha}\left(J_p\ddot{\alpha} + \frac{1}{2}m_p g L_p \sin\alpha\right). \tag{4.5}$$

To introduce the pivot acceleration u and eventually, our control variable, solve for $\sin\alpha$ in Equation 4.1 to obtain

$$\sin(\alpha) = \frac{1}{m_p g L_p}(-2J_p\ddot{\alpha} + m_p L_p u \cos(\alpha)).$$

Substitute this into $\dot{E}$, found in Equation 4.5, to get

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

One strategy that will swing the pendulum to a desired reference energy E_r is the proportional control

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

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

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

$$u = sat_{u_{max}}(\mu(E - E_r)\text{sign}(\dot{\alpha}\cos\alpha)) \tag{4.6}$$

where μ is a tunable control gain and $sat_{u_{max}}$ function saturates the control signal at the maximum acceleration of the pendulum pivot, u_{max}. Taking the sign of $\dot{\alpha}\cos\alpha$ allows for faster switching.

In order to translate the pivot acceleration into servo voltage, first solve for the voltage in Equation 2.4 to get

$$V_m = \frac{\tau R_m}{\eta_g K_g \eta_m k_t} + K_g k_m \dot{\theta}.$$

Then substitute the torque-acceleration relationship given in Equation 4.2 to obtain the following

$$V_m = \frac{R_m m_r L_r u}{\eta_g K_g \eta_m k_t} + K_g k_m \dot{\theta}. \tag{4.7}$$

4.1.3 Self-Erecting Control

The energy swing-up control can be combined with the balancing control in Equation 3.11 to obtain a control law which performs the dual tasks of swinging up the pendulum and balancing it. This can be accomplished by switching between the two control systems.

Basically the same switching used for the balance control in Equation 3.12 is used. Only instead of feeding 0 V when the balance control is not enabled, the swing-up control is engaged. The controller therefore becomes

$$u = \begin{cases} K(x_d - x) & |x_2| < \epsilon \\ sat_{u_{max}}(\mu(E - E_r)\text{sign}(\dot{\alpha}\cos\alpha)) & \text{otherwise} \end{cases} \tag{4.8}$$

4.2 Pre-lab Questions

1. Evaluate the potential energy of the pendulum when it is in the downward and upright positions.
2. Compute the maximum acceleration deliverable by the SRV02. Assume the maximum equivalent voltage applied to the DC motor is 5 V such that
$$V_m - K_g k_m \dot{\theta} = 5. \tag{4.9}$$
The SRV02 motor parameters are given in [4].
3. Find the controller acceleration when the pendulum is initially hanging down and motionless. From a pracitcal viewpoint, what does this imply when the swing-up control is activated?
4. Assume the pendulum is starting to swing from the downward position in the positive direction. Calculate the acceleration the swing-up controller will generate when $\mu = 20$. Does this saturate the controller?

4.3 In-lab Exercises

In this section, you will be implementing the energy-based swing-up controller described in Section 4.1.

Experiment Setup

The *q_rotpen_swingup_student* Simulink diagram shown in Figure 4.1 is used to run the swing-up control on the Quanser Rotary Pendulum system. Similarly with the q_rotpen_balance_student Simulink diagram, the *SRV02-ET+ROTPEN-E* subsystem contains QUARC blocks that interface with the system hardware and the feedback is implemented using a Simulink *Gain* block. The balance and swing-up control are not completed.

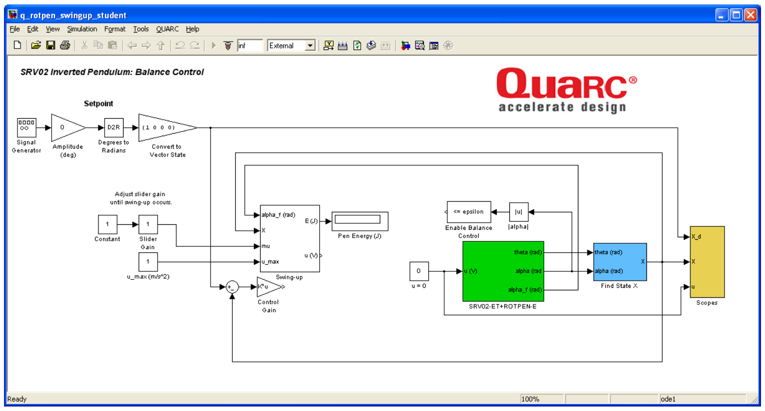


Figure 4.1: q_rotpen_swingup_student Simulink diagram can be used to run the swing-up controller

IMPORTANT: Before you can conduct this experiment, you need to make sure that the lab files are configured according to your system setup. If they have not been configured already, then go to Section 5.6 to configure the lab files first.

1. Run the *setup_rotpen.m* script.
2. Make sure the gain K you found in Section 3.4.1 is loaded.
3. Open the *q_rotpen_swingup_student* Simulink diagram.
4. Turn ON the power amplifier.
5. **Ensure the modifications you made in the Balance Control Laboratory in Section 3.4 have been applied to *q_rotpen_swingup_student*.** Run the controller and verify that the balance control runs well.
6. Open the *Swing-Up* subsystem. As shown in Figure 4.2, it is incomplete.

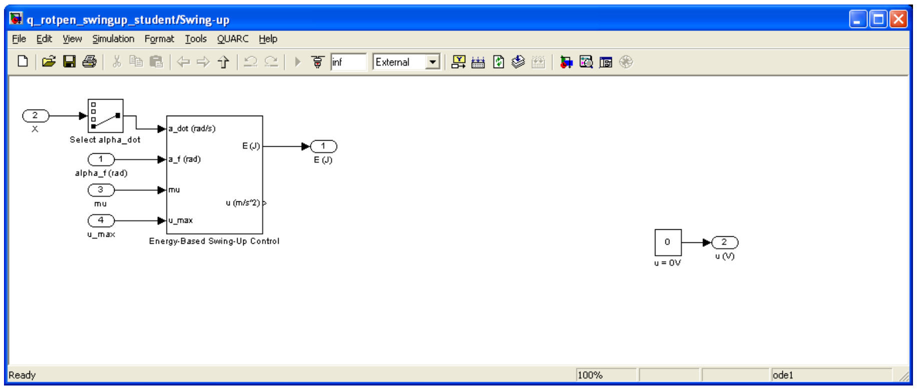


Figure 4.2: Incomplete Swing-Up Control subsystem.

7. Go into *Energy-Based Swing-Up Control* | *Pendulum Energy* block. The incomplete diagram is shown in Figure 4.3. Modify the *Pendulum Energy* diagram to measure the total energy of the pendulum. Use the pendulum parameters already loaded in Matlab, i.e., from the *config_sp* function, and any of the blocks from the Simulink library you require.

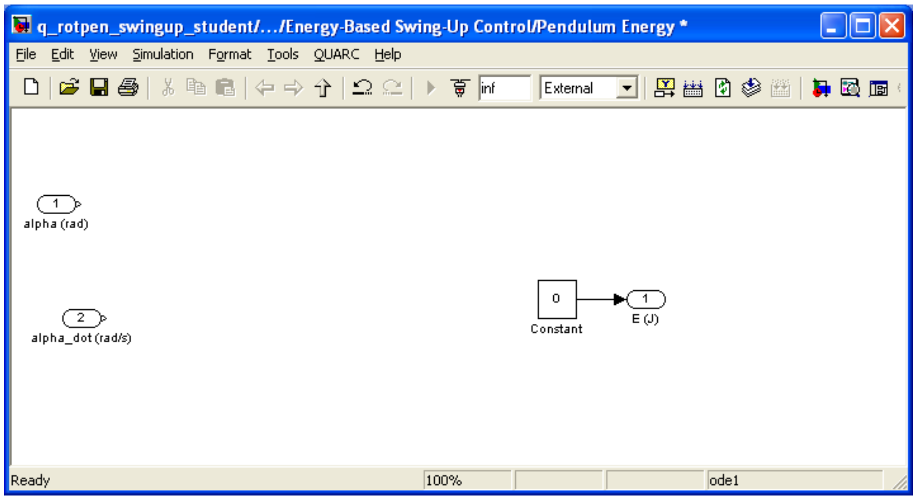

q_rotpen_swingup_student/.../Energy-Based Swing-Up Control/Pendulum Energy *
File Edit View Simulation Format Tools QUARC Help
inf
External
1
alpha (rad)
2
alpha_dot (rad/s)
0
Constant
1
E (J)
Ready
100%
ode1

Figure 4.3: Incomplete Energy-Based Swing-Up Control subsystem.

8. Go to QUARC | Build to build the controller.

9. Go to QUARC | Start to run the controller.

10. Run the controller and rotate the pendulum up to the upright position. While the inverted pendulum is balancing, record the total energy reading displayed in *Pen Energy (J)* numeric indicator. Is the value as expected?

11. Implement the energy-based swing-up controller by modifying the *Energy-Based Swing-Up Control* subsystem shown in Figure 4.4. Use the Source block with the variable *Er* as well as the inputs *u_max (m/s*^{2}*)* and *mu* that are already included. Make sure you are using the full pendulum angle α, i.e., not the upright based angle used in the feedback for the inverted pendulum balance control.

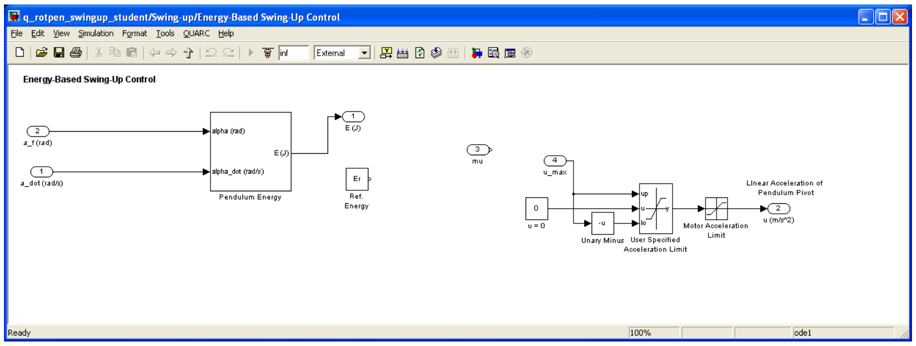

q_rotpen_swingup_student/Swing-up/Energy-Based Swing-Up Control
File Edit View Simulation Format Tools QUARC Help
inf
External
Energy-Based Swing-Up Control
2
a_f (rad)
1
a_dot (rad/s)
alpha (rad)
alpha_dot (rad/s)
E (J)
Pendulum Energy
1
E (J)
Er
Ref. Energy
3
mu
4
u_max
0
u = 0
-u
Unary Minus
up
u
lo
y
User Specified Acceleration Limit
Motor Acceleration Limit
LInear Acceleration of Pendulum Pivot
2
u (m/s^2)
Ready
100%
ode1

Figure 4.4: Incomplete Energy-Based Swing-Up Control subsystem.

12. Add the necessary modifications to convert the acceleration generated by the swing-up control to servo voltage. To do this, edit the *Swing-Up* subsystem shown in Figure 4.2. Use the SRV02 model parameters that are already defined in the Matlab workspace, i.e., using the *config_srv02* function, for any of the servo-based attributes you need.

13. Implement the self-erecting control in Equation 4.8, which includes both the swing-up and balance control. As in the Balance Control lab, plot the rotary arm, pendulum, and servo voltage response in a Matlab figure.

14. Set the reference energy, maximum acceleration, and proportional gain parameters in *q_rotpen_swingup_student* Simulink model to:

$$\begin{aligned} E_r &= E_p \\ u_{max} &= 1 \text{ m/s}^2 \\ \mu &= 1 \end{aligned}$$

Make sure the reference energy is set to the pendulum potential energy. Then go to QUARC | Run to start the controller.

The pendulum should be moving back and forth slowly. Gradually increase the u_{max} and/or μ until the pendulum goes up. Do not increase the u_{max} above the maximum acceleration you found for the SRV02 in Section 4.2. When the pendulum swings up to the vertical upright position, the balance controller should engage and balance the link. Show the response of the arm and pendulum angles as well as the control voltage and record the swing-up parameters. Did the swing-up behave with the parameters you expected?

15. Click on the STOP button to stop running the controller. Be careful, as the pendulum will fall down when the controller is stopped.

16. Shut off the power amplifier.

4.4 Results

Fill out Table 3 with your answers from your swing-up control lab results.

Description	Symbol	Value	Unit
Pre Lab Questions			
Potential Energy	E_p		J
Maximum Acceleration of SRV02	u_{max}		m/s^2
Implementation			
Balance Control Gain	K		
Reference Energy	E_r		J
Control Maximum Acceleration	u_{max}		m/s^2
Swing-Up Proportional Gain	μ		

Table 3: Results

5 SYSTEM REQUIREMENTS

Required Software

- Microsoft Visual Studio
- Matlab®with Simulink®, Real-Time Workshop, and the Control System Toolbox.
- QUARC®2.1, or later.

See the QUARC®software compatibility chart at [5] to see what versions of MS VS and Matlab are compatible with your version of QUARC and for what OS.

Required Hardware

- Data-acquisition (DAQ) card that is compatible with QUARC. This includes Quanser Hardware-in-the-loop (HIL) boards such as:
  - Q2-USB
  - Q8-USB
  - QPID
  - QPIDe

  and some National Instruments DAQ devices (e.g., NI USB-6251, NI PCIe-6259). For a full listing of compliant DAQ cards, see Reference [2].
- Quanser SRV02-ET rotary servo. See Reference [4].
- Quanser Rotary Pendulum Module (attached to SRV02). See Reference [6].
- Quanser VoltPAQ power amplifier, or equivalent (e.g. Reference [7] for VoltPAQ User Manual).

5.1 Overview of Files

File Name	Description
Rotary Pendulum User Manual.pdf	This manual describes the hardware of the Rotary Pendulum system and explains how to setup and wire the system for the experiments.
Rotary Pendulum Workbook (Student).pdf	This laboratory guide contains pre-lab questions and lab experiments demonstrating how to design and implement a self-erecting controller on the Quanser SRV02 Rotary Pendulum plant using QUARC® .
setup_rotpen.m	The main Matlab script that sets the SRV02 motor and sensor parameters, the SRV02 configuration-dependent model parameters, and the Rotary Pendulum sensor parameters. Run this file only to setup the laboratory.
config_srv02.m	Returns the configuration-based SRV02 model specifications Rm, kt, km, Kg, eta_g, Beq, Jeq, and eta_m, the sensor calibration constants K_POT, K_ENC, and K_TACH, and the amplifier limits VMAX_AMP and IMAX_AMP.
config_sp.m	Returns the pendulum model parameters.
calc_conversion_constants.m	Returns various conversions factors.
ROTPEN_ABCD_eqns_student.m	Contains the incomplete state-space A, B, C, and D matrices. These are used to represent the Rotary Inverted Pendulum system.
d_pole_placement_student.m	Use this script to find the feedback control gain K.
q_rotpen_mdl.mdl	Simulink file used with QUARC®to read angles and drive DC motor. Used to validated the Rotary Inverted Pendulum model conventions.
s_rotpen_bal.mdl	Simulink file that simulates the Rotary Inverted Pendulum system when using a state-feedback control.
q_rotpen_bal_student.mdl	Simulink file that implements a closed-loop state-feedback controller on the actual ROTPEN system using QUARC® . The balance control is not complete.
q_rotpen_swingup_student.mdl	Simulink file that implements the self-erecting controller on the actual ROTPEN system using QUARC® . This model is incomplete - both the swing-up and the balance control need to be finished.

Table 4: Files supplied with the SRV02 Rotary Inverted Pendulum Control Laboratory.

5.2 Hardware Setup

Follow these steps to get the system hardware ready for this lab:

1. Make sure the SRV02 is in the *high-gear configuration*.
2. Install the Rotary Inverted Pendulum module on top of the SRV02 gear as shown in [6].
3. Connect the Quanser Inverted Pendulum to the amplifier (e.g. VoltPAQ) and DAQ device as described in [6].

5.3 Setup for Modeling Lab

Before performing the in-lab exercises in Section 2.3.2, the q_rotpen_mdl_student Simulink diagram and the setup_rotpen.m script must be configured.

Follow these steps to get the system ready for this lab:

1. Setup the SRV02 with the Rotary Pendulum module as detailed in [6].
2. Load the Matlab software.
3. Browse through the *Current Directory* window in Matlab and find the folder that contains the QUARC ROTPEN file *q_rotpen_mdl_student.mdl*.
4. Open the *q_rotpen_mdl_student.mdl* Simulink diagram, shown in Figure 2.2.
5. **Configure DAQ:** Ensure the HIL Initialize block in the *SRV02-ET+ROTPEN-E* subsystem is configured for the DAQ device that is installed in your system. By default, the block is setup for the Quanser Q8 hardware-in-the-loop board. See Reference [2] for more information on configuring the HIL Initialize block.
6. Open the setup_rotpen.m file. This is the setup script used for the ROTPEN Simulink models.
7. **Configure setup script**: When used with the Rotary Pendulum, the SRV02 has no load (i.e., no disc or bar) and has to be in the high-gear configuration. Make sure the script is setup to match this setup:
   - EXT_GEAR_CONFIG to 'HIGH'
   - LOAD_TYPE to 'NONE'
   - ENCODER_TYPE and TACH_OPTION parameters are set according to the SRV02 system that is to be used in the laboratory.
   - K_AMP to the amplifier gain. For VoltPAQ-X1, set K_AMP to 1 or 3 depending how gain switch on amplifier is set.
   - AMP_TYPE to the amplifier you are using, e.g., VoltPAQ.
   - CONTROL_TYPE to 'MANUAL'.

5.4 Setup for Inverted Pendulum Control Simulation

Before going through the control simulation in Section 3.4.2, the s_rotpen_bal Simulink diagram and the setup_rotpen.m script must be configured.

Follow these steps to configure the lab properly:

1. Load the Matlab software.
2. Browse through the *Current Directory* window in Matlab and find the folder that contains the s_flexgage.mdl file.
3. Open s_rotpen_bal.mdl Simulink diagram shown in Figure 3.3.
4. Configure the setup_rotpen.m script according to your hardware. See Section 5.3 for more information. **IMPORTANT:** Make sure the model you found in Section 2.3 is entered in ROTPEN_ABCD _eqns_student.m.
5. Run the setup_rotpen.m script.

5.5 Setup for Inverted Pendulum Control Implementation

Before beginning the in-lab exercises given in Section 3.4.3, the q_rotpen_bal_student Simulink diagram and the setup_rotpen.m script must be setup.

Follow these steps to get the system ready for this lab:

1. Setup the SRV02 with the Rotary Pendulum module as detailed in [6] .
2. Load the Matlab software.
3. Browse through the *Current Directory* window in Matlab and find the folder that contains the *q_rotpen_bal.mdl* file.
4. Open the *q_rotpen_bal_student.mdl* Simulink diagram. The *student* based version is shown in Figure 3.5.
5. **Configure DAQ:** Ensure the HIL Initialize block in the *SRV02-ET+ROTPEN-E* subsystem is configured for the DAQ device that is installed in your system. By default, the block is setup for the Quanser Q8 hardware-in-the-loop board. See Reference [2] for more information on configuring the HIL Initialize block.
6. **Configure setup script:** Set the parameters in the *setup_rotpen.m* script according to your system setup. See Section 5.3 and Section 5.4 for more details.
7. Run the setup_rotpen.m script.

5.6 Setup for Swing-Up Control Implementation

Before beginning the in-lab exercises given in Section 4.3, the q_rotpen_swingup_student Simulink diagram and the setup_rotpen.m script must be setup.

Follow these steps to get the system ready for this lab:

1. Setup the SRV02 with the Rotary Pendulum module as detailed in [6] .
2. Load the Matlab software.
3. Browse through the *Current Directory* window in Matlab and find the folder that contains the *q_rotpen_swingup.mdl* file.
4. Open the *q_rotpen_swingup_student.mdl* Simulink diagram. The *student* based version is shown in Figure 4.1.
5. **Configure DAQ:** Ensure the HIL Initialize block in the *SRV02-ET+ROTPEN-E* subsystem is configured for the DAQ device that is installed in your system. By default, the block is setup for the Quanser Q8 hardware-in-the-loop board. See Reference [2] for more information on configuring the HIL Initialize block.
6. **Configure setup script:** Set the parameters in the *setup_rotpen.m* script according to your system setup. See Section 5.3 and Section 5.4 for more details.
7. Run the setup_rotpen.m script.

6 LAB REPORT

This laboratory contains three experiments, namely,

1. Modeling,
2. Balance Control, and
3. Swing-Up Control.

When you are writing your lab report, follow the outline corresponding to the experiment you conducted to build the *content* of your report. Also, in Section 6.4 you can find some basic tips for the *format* of your report.

6.1 Template for Content (Modeling)

I. PROCEDURE

1. *Model Analysis*
   - Briefly describe the main goal of the simulation.
   - Briefly describe the simulation procedure in steps 2 and 3 in Section 2.3.1.
2. *Calibration*
   - Briefly describe the main goal of the experiment.
   - Briefly describe the experiment procedure (Section 2.3.2)

II. RESULTS
Do not interpret or analyze the data in this section. Just provide the results.

1. State-space representation from Step 3 in Section 2.3.1.
2. Provide applicable data collected in this laboratory (from Table 1).

III. ANALYSIS
Provide details of your calculations (methods used) for analysis for each of the following:

1. Open-loop poles found in Step 4 in Section 2.3.1.
2. Measured arm response in Step 7 in Section 2.3.2.

IV. CONCLUSIONS
Interpret your results to arrive at logical conclusions for the following:

1. Whether the arm and pendulum angles match the model conventions in Step 5 of Section 2.3.2, *Sensor calibration*.
2. Whether the control voltage matches the model conventions in Step 7 of Section 2.3.2, *Actuator calibration*.

6.2 Template for Content (Balance Control Experiment)

I. PROCEDURE

1. *Control Design*
   - Briefly describe the main goal of the control design.
   - Briefly describe the controllability check procedure in Step 2 in Section 3.4.1.
   - Briefly describe the control design procedure in Step 3 in Section 3.4.1.
2. *Simulation*
   - Briefly describe the main goal of the simulation.
   - Briefly describe the simulation procedure in Step 2 in Section 3.4.2.
3. *Implementation*
   - Briefly describe the main goal of this experiment.
   - Briefly describe the experimental procedure in Step 8 in Section 3.4.3.

II. RESULTS

Do not interpret or analyze the data in this section. Just provide the results.

1. Matrices from Step 3 in Section 3.4.1, *Find transformation matrix*.
2. Response plot from Step 2 in Section 3.4.2, *Simulating balance control*.
3. Completed Simulink diagram in Step 7 in Section 3.4.2, *SimulatingInverted pendulum balance control block diagram*
4. Response plot from Step 9 in Section 3.4.3, *Implementing balance control*.
5. Provide applicable data collected in this laboratory (from Table 2).

III. ANALYSIS

Provide details of your calculations (methods used) for analysis for each of the following:

1. Step 2 in Section 3.4.1, *Controllability of system*.
2. Step 5 in Section 3.4.1, *Closed-loop poles*.
3. Matlab commands in Step 6 in Section 3.4.1, *Matlab commands used to generate the control gain*.
4. Step 3 in Section 3.4.2, *Balance control simulation*.
5. Step 10 in Section 3.4.3, *Balance control implementation*.

IV. CONCLUSIONS

Interpret your results to arrive at logical conclusions for each of the following:

1. Step 5 in Section 3.4.1, *Closed-loop system poles*.
2. Step 3 in Section 3.4.2, *Balance control simulation*.
3. Step 10 in Section 3.4.3, *Balance control implementation*.

6.3 Template for Content (Swing-Up Control Experiment)

I. PROCEDURE

1. *Implementation*
   - Briefly describe the main goal of this experiment.
   - Briefly describe the experimental procedure in Step 7 in Section 4.3.
   - Briefly describe the control states and parameters for the swing-up control in Step 11 in Section 4.3.
   - Briefly describe the parameters used to convert torque to voltage in Step 12 in Section 4.3.

II. RESULTS

Do not interpret or analyze the data in this section. Just provide the results.

1. Completed Simulink diagram in Step 11 in Section 4.3, *Implemented swing-up controller*.
2. Completed Simulink diagram in Step 13 in Section 4.3, *Implemented self-erecting controller*.
3. Response plot from Step 14 in Section 3.4.3, *Self-erecting inverted pendulum control implementation response*.
4. Provide applicable data collected in this laboratory (from Table 3).

III. ANALYSIS

Provide details of your calculations (methods used) for analysis for each of the following:

1. Step 10 in Section 4.3, *Potential energy of pendulum*.

IV. CONCLUSIONS

Interpret your results to arrive at logical conclusions for each of the following:

1. Step 14 in Section 4.3, *Swing-up control implementation*.

6.4 Tips for Report Format

PROFESSIONAL APPEARANCE

- Has cover page with all necessary details (title, course, student name(s), etc.)
- Each of the required sections is completed (Procedure, Results, Analysis and Conclusions).
- Typed.
- All grammar/spelling correct.
- Report layout is neat.
- Does not exceed specified maximum page limit, if any.
- Pages are numbered.
- Equations are consecutively numbered.
- Figures are numbered, axes have labels, each figure has a descriptive caption.
- Tables are numbered, they include labels, each table has a descriptive caption.
- Data are presented in a useful format (graphs, numerical, table, charts, diagrams).
- No hand drawn sketches/diagrams.
- References are cited using correct format.

REFERENCES

[1] Bruce Francis. Ece1619 linear systems course notes (university of toronto), 2001.

[2] Quanser Inc. *QUARC User Manual*.

[3] Quanser Inc. *SRV02 QUARC Integration*, 2008.

[4] Quanser Inc. *SRV02 User Manual*, 2009.

[5] Quanser Inc. *QUARC Compatibility Table*, 2010.

[6] Quanser Inc. *SRV02 Rotary Pendulum User Manual*, 2010.

[7] Quanser Inc. *VoltPAQ User Guide*, 2010.

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

[9] K. J. Åström and K. Furuta. Swinging up a pendulum by energy control. *13th IFAC World Congress*, 1996.

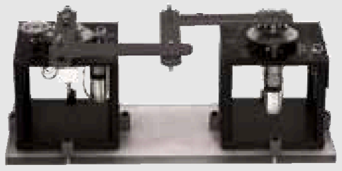
Ten modules to teach controls from the basic to advanced level

SRV02 Base Unit

Flexible Link

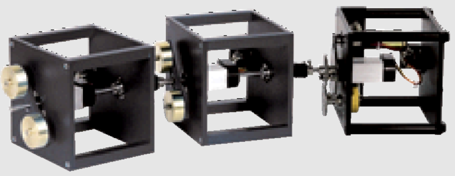
Ball and Beam

2 DOF Robot



2 DOF Inverted Pendulum

Multi-DOF Torsion



Flexible Joint

Gyro/Stable Platform

Double Inverted Pendulum

2 DOF Ball Balancer

With the SRV02 Base Unit, you can select from 10 add-on modules to create experiments of varying complexity across a wide range of topics, disciplines and courses. All of the experiments/workstations are compatible with LabVIEW™ and MATLAB®/Simulink®.

To request a demonstration or a quote, please email info@quanser.com.

QUANSER
INNOVATE. EDUCATE.

INFO@QUANSER.COM **+1-905-940-3575** **QUANSER.COM**

Solutions for teaching and research. Made in Canada.