POLYTECH NANCY

Control Engineering Labs

Semester 6

(a) QUBE-Servo 2 with Inertia Disc Module

(b) QUBE-Servo 2 with Pendulum Module

Hugues Garnier
Floriane Collin

March 2023

Contents

Introduction to the Control Engineering Labs

About this manual

The purpose of this manual is to provide you with the information necessary for conducting the laboratory experiments of the Control Engineering course. You should by no means consider this manual as the sole source of information for the design and implementation process. You should rely on lectures slides, problem solving sessions, as well as your own knowledge of control engineering theory, to develop and implement your control strategies.

Objectives of the Control Engineering labs

The objectives of the labs can be summarized as follows:

- to learn how to investigate the properties of physical systems both through simulation and experimentation;
- to learn how to design and analyze feedback control systems using classical control techniques based on transfer function model and block-diagrams;
- to learn how to perform identification experiments on different physical systems;
- to learn how to simulate classical on-off and PID control in Matlab/Simulink;
- to learn how to implement classical digital feedback control strategies on different experimental systems;
- to develop practical skills in an experimental environment.

Pre-lab questions

The pre-lab questions of the first lab are related to the problem solving session n°4. Solutions to all questions should be available upon request from the lab assistant. These solutions will most likely require to be adapted to your experimental conditions.
When the lab starts, it is assumed that you have a full understanding of these solutions, and have a clear idea of the tasks that will have to perform during the lab.

Doing your own work

While you are encouraged to discuss theory and methods with other classmates, when it comes time to analyze, design, and implement your system, it must be done on your own. As a future engineer, you must always fully understand the problem you are trying to solve. Also, remember that with experimentations, there is often no "right" answer. You should answer the question based on your observations for your simulation and experimental results from your particular system components, which may be quite different than others.

Grading

When working in an experimental environment, results are always important, but so is your ability to communicate the information with others. Any work you will submit should be neat, well-organized and easy to understand. Your report is a demonstration of your ability to understand the course you are studying as well as a reflection of your organizational skills.

Steps in the design of a control system

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

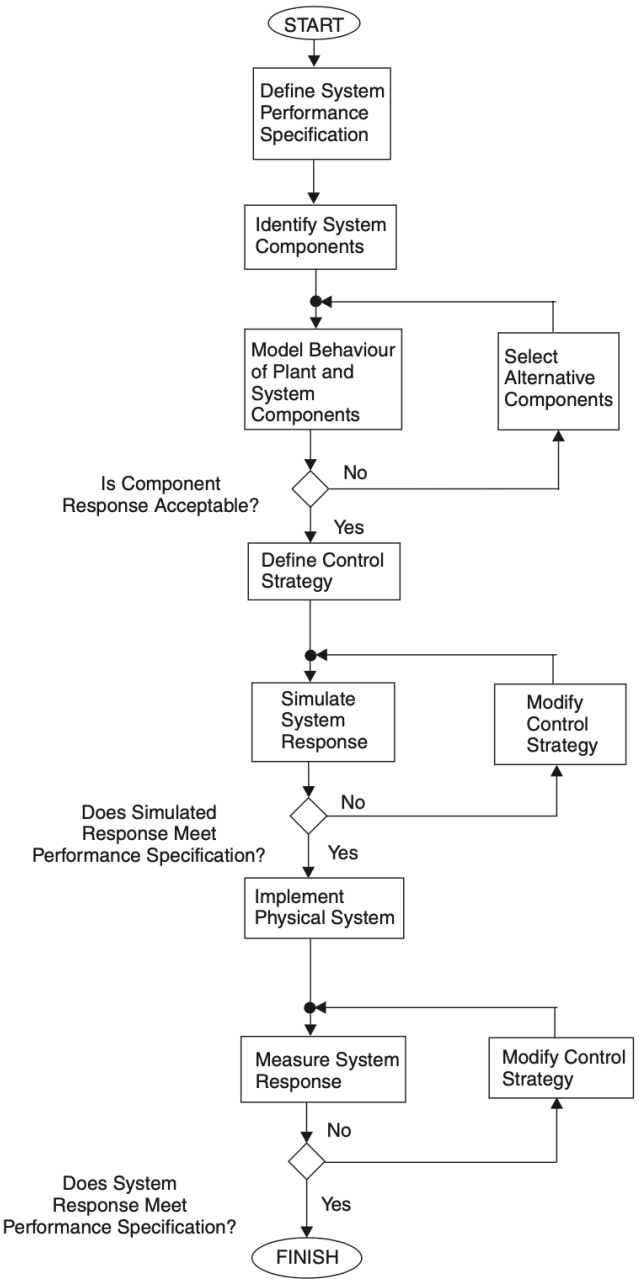


Figure 1: Steps in the design of a control system (From *S. Burns, Advanced Control Engineering, Butterworth-Heinemann, 2001)*

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

Use of Matlab/Simulink

MATLAB for MATrix LABoratory is a powerful software that can deal with complex problems in various scientific fields in an interactive environment. MATLAB is supplemented by various special application toolboxes, which contain additional functions and programs. One such toolbox is the Control System Toolbox which is also available in student editions.
Simulink is a very useful Graphical User Interface (GUI) tool of MATLAB for modeling closed-loop block-diagram control systems, and simulating their time response to specified inputs.
If you are not familiarized with Simulink, take some time before the lab sessions to learn about its use by reading Appendix A and/or some tutorial introduction available such as Simulink-on-ramp from the Mathworks website.

Lab 1

Digital model-based control designs of the speed of a rotating disk

You are not asked to submit a report for this lab which will not therefore be marked. You are, however, encouraged to write a personal report.
This lab aims at illustrating the various stages of two traditional model-based designs that lead to the implementation of a digital speed control of a rotating disk.

The objectives are the following:

1. to identify a linear low-order model from real data coming from a step test;
2. to design and tune two different digital controllers for the rotating disk speed control based on the identified linear model and evaluate its performance in simulation by using Simulink;
3. to implement and evaluate the best designed speed controllers on the physical QUBE-servo 2 platform equipped with the inertia rotating disk.

1.1 Pre-lab questions

The pre-lab questions of this lab are related to the tutorial session n°4. Solutions to all questions should be available upon request from the lab assistant. These solutions will require to be adapted to the experimental conditions (mainly due to a slightly different identified model and the presence of non-linear dry friction effects in the motor).

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

1.2 The QUBE-servo 2 from Quanser

One of the most common devices for actuating a control system is DC servo-motors.

1.2.1 The QUBE servo-motor

The Quanser QUBE-Servo 2, pictured in Figure 1.1, is a compact rotary servo system that can be used to perform a variety of classic servo control and inverted pendulum based experiments. The QUBE can be controlled by a computer via USB connection.
The system is driven using a direct-drive 18V brushed DC motor. The motor is powered by a

built-in Pulse Width Modulation (PWM) amplifier with integrated current sense. Two add-on modules are supplied with the system:

- an inertia disk;
- a rotary pendulum.

The two modules can be easily attached or interchanged using magnets mounted on the QUBE-Servo 2 module connector.
Single-ended rotary encoders are used to measure the angular position of the DC motor and pendulum, while the angular velocity of the motor is measured using an integrated software-based tachometer.

(a) QUBE-Servo 2 with Inertia Disc Module (b) QUBE-Servo 2 with Pendulum Module

Figure 1.1: QUBE-Servo 2 with different modules

During this lab, we will mainly use the QUBE-Servo 2 with the inertia disk to design a servo-control of the angular velocity. The input and output of the servo-control system are:

- input: voltage of the motor in V;
- output: speed of the rotating disk in rad/s.

1.2.2 Download of the files required for the lab

1. Download the zipped file *Lab1.zip* from the course website and save and unzip it in your `Documents/Matlab/Control_Lab/` folder.
2. Start Matlab.
3. Important ! By clicking on the browse for folder icon , **change the current folder of Matlab so that it becomes your `Documents/Matlab/Control_Lab/` folder** that contains the files needed for this lab.
4. In the Current Folder window of Matlab, click right on the folder `Documents/Matlab/Control_Lab/Lab1` and select `add to path the selected folder`.
5. Double-click on the folder `Lab1` so that it becomes your current folder. You should see the different .slx and m. files needed for this Lab.

1.2.3 Identification of the physical system components

Prior to operating any experimental system, it is imperative to familiarize yourself with the various components of the system. Part of the process involves the identification of the individual hardware components, including its sensors and actuators. Find out from the Qube description in the sub-section 1.2.1

1. the actuator;
2. the system;
3. the angular velocity sensor.

1.2.4 Control performance requirements

A schematic of the feedback speed control of the rotating disk is represented in Figure 1.2.

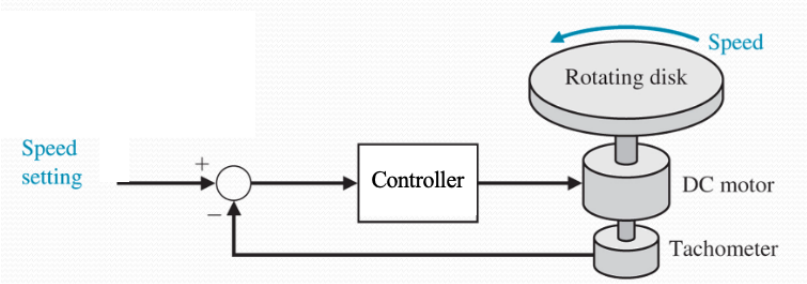


Figure 1.2: Schematic of the feedback speed control of the rotating disk

The performance requirements for the rotary speed digital control are described in Table 1.1.

Requirement	Assessment criteria	Level
Control of the rotary speed	Speed setpoint tracking	No steady-state error
	Motor input voltage	limited to [-10V ; +10 V]
	Settling time at 5 %	10 ms
	Peak overshoot	less than or equal to 5%
	Disturbance rejection	Rejection of load effects

Table 1.1: Performance requirements for rotary speed control

1.3 Linear model identification from step response test in open loop

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

1.3.1 Servo motor model

The motor voltage-to-angular velocity transfer function takes the form of a simple first-order model

$$\frac{\Omega(s)}{U(s)} = \frac{K}{1+Ts} \tag{1.1}$$

where $\Omega(s) = \mathcal{L}[\omega(t)]$ is the inertia disk angular velocity (or rotary speed) and $U(s) = \mathcal{L}[u(t)]$ is the applied motor voltage. K in rad/(V-s) is the model steady-state gain, T in s is the model time-constant.

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

1.3.2 Recording of the experimental step response

1. Open the file `Qube_step_resp_speed.slx` in Simulink.

2. Select `HARDWARE` tab in the Simulink menu bar.

3. Click on `Monitor & Tune` to run the step test that lasts **2.5 s** only.

4. Click on the two scopes to observe the angular velocity (in rad/s) response and the step input of amplitude 2V sent to the motor voltage. The relevant input/output variables will be saved in the `data_step_Qube_speed.mat` file for further use.

1.3.3 Identification of a linear first-order model

1. Open the file `speed_response_plot.m` in Matlab and run it. You should get two plots similar those given in Figure 1.3.

2. From your angular velocity response plot, determine the parameters of a first-order model. The values could slightly differ from the ones given in the tutorial session.

3. Open the file `test_your_model.m` in Matlab.

4. At the beginning of the file, modify the default values of the steady-state gain K and time-constant T parameters, then run the program.

5. Compare the measured and model angular velocity responses.

6. Refine the two model parameters, if necessary, to improve the fit between the measured and simulated angular velocity responses.

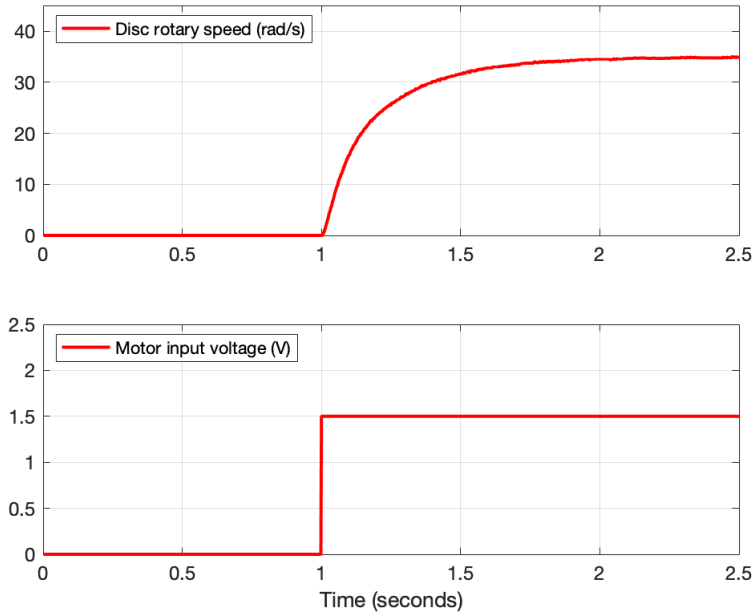


Figure 1.3: Angular velocity response of the inertia disk to a step input sent to the motor voltage

Figure 1.3 can be reproduced with Matlab by executing the `speed_response_plot.m` file.

1.4 Digital speed controller design via the approximation of a continuous-time PI controller

1.4.1 Simulation of the rotary speed digital PI controller in Simulink

A variation of the classical PI control will be used as shown in Figure 1.4. Unlike the standard PI where the proportional action is applied to the error, it is applied to the output.

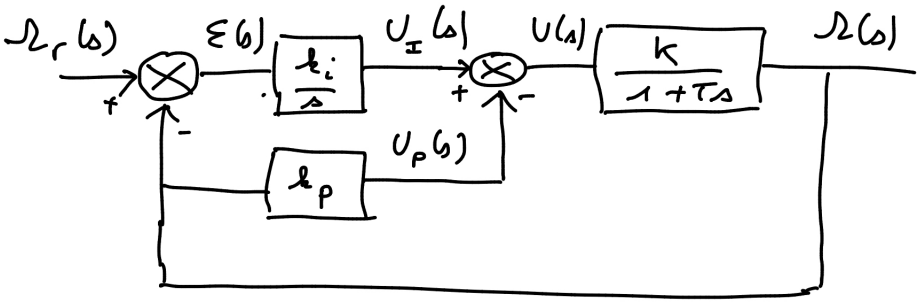


Figure 1.4: Block-diagram of the **continuous-time** PI feedback configuration for the rotary speed servo system with proportional effect on the output.

We seek for a discretised version of the continuous-time PI controller by using the Tustin approximation.

1. Recall the theoretical values for k_p and k_i in terms of K, T, ζ and ω that make the step response of the PI-based feedback system to have a percent overshoot of 4.3 % and a settling time at 5% of 0.05s (see the solution discussed in the tutorial session).

2. The sampling period will be initially set to $T_s = 0.005$s.

3. Open the `Qube_PI_gains_4_speed_control.m` Matlab script. It will automatically compute the k_p and k_i PI gains based on the solution derived during the tutorial session.

4. Modify if necessary the K and T parameter values of your identified first-order model at the beginning of the .m file.

5. Run the Matlab file and look at the simulation results. Verify that the requirement specifications are met?

6. Open the file `Simul_digital_PI_speed_control.slx` in Simulink. Observe the presence of the zero-order hold block.

7. Add the digital PI controller in the Simulink model. Do not forget to indicate the appropriate sampling time value of T_s in the discretised version of the Integral.

8. Enter the K and T parameter values of your first-order model by clicking on the blue *Qube model* block.

9. Set your values for k_p and k_i.

10. In the Simulink window, go to the Hardware panel. Click on Hardware Settings and configure the Solver item as shown in Figure 1.5.

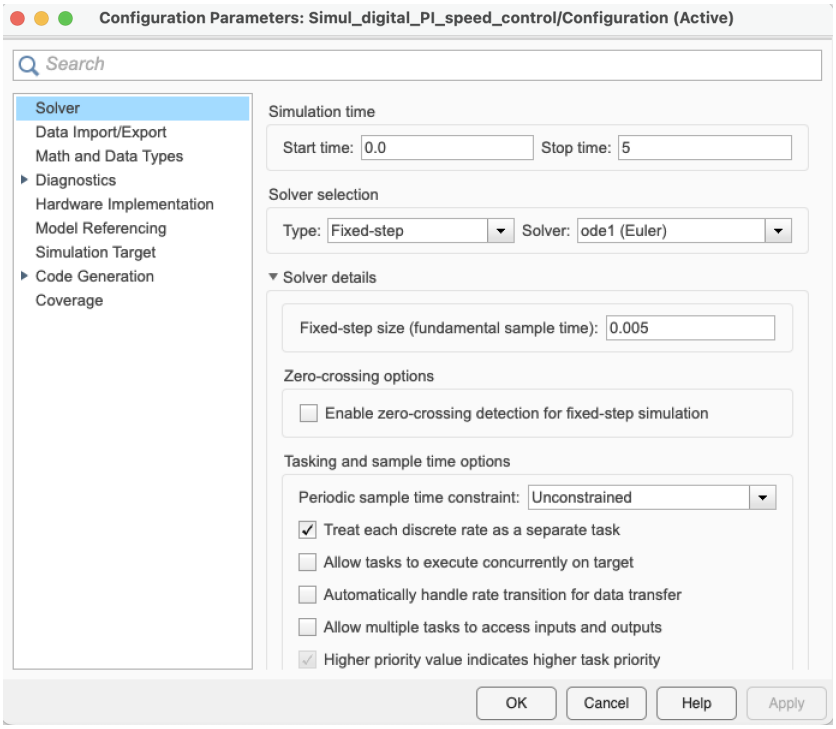


Figure 1.5: Configuration of the Solver in the Hardware Settings for the simulation

11. Select `SIMULATION` tab in the Simulink menu bar and click on run to simulate the closed-loop response.

12. Observe the simulated angular velocity response in servo control as well as the DC-motor input voltage. Determine the maximum and minimum values of the input voltage.

13. Increase the value of the sampling period and run again the Simulink file. Examine the influence of the sampling period on the digital PI-based closed-loop performance.

14. Close the file.

1.4.2 Practical implementation of the rotary speed digital PI controller on the physical QUBE-Servo 2

We will now proceed with the implementation and test of the PI feedback digital control on the QUBE-Servo 2.
The following procedure shows how to run a Simulink model in QUARC. The Simulink/QUARC model applies a square wave speed reference, or setpoint, of ±5 rad/s about a constant of 25 rad/s (i.e. speed reference goes between 20 rad/s and 30 rad/s).

1. Mount the inertia disk on the QUBE-Servo 2.

2. Connect the QUBE-Servo 2 USB to the PC and power it, as illustrated in the Quick Start Guide.

3. Open the `Qube_digital_PI_speed_control.slx` Simulink model.

4. Add the digital PI controller in the Simulink model.

5. Enter the numerical value of the two gains k_p and k_i of your best digital PI controller tuned in the previous section

6. In the Simulink window, go to the Hardware panel. Click on Hardware Settings and configure the Solver item as shown in Figure 1.6

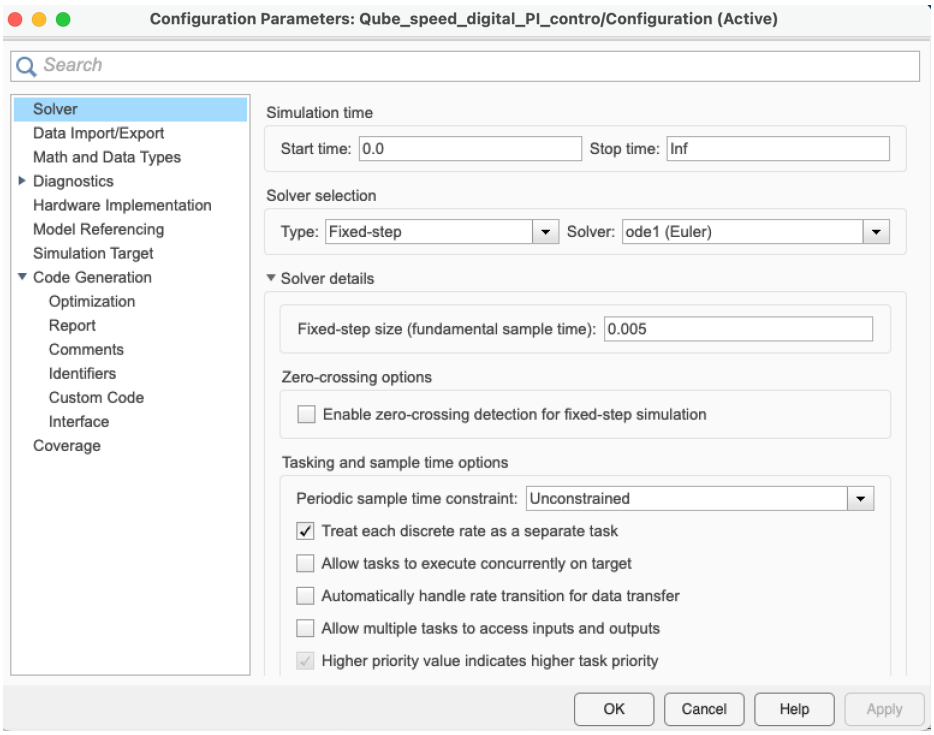


Figure 1.6: Configuration of the Solver in the Hardware Settings for practical implementation

7. In the Simulink window, go to the `Hardware` tab. Click on **Monitor & Tune** to generate the code and run the controller.

8. Observe the angular velocity response to the square setpoint signal along with the control signal.

9. Are the requirement specifications met?
   Due to unmodeled dynamics such as friction and amplifier nonlinearities, the experimental measured response can be more damped. As a result the measured response has less percent overshoot and settles faster (i.e. lower settling time) compared to the simulation.

10. If necessary, refine the value of k_p and k_i to get the best answer possible to the specification requirements (peak overshoot, steady-state error, settling time and amplitude of the motor voltage) with this digital PI feedback control.
    The icon can be useful to determine the peak-time and settling-time at 5%.

11. Increase the value of the sampling period (in the Solver item of the Hardware settings along with in the Integral term) and run again the Simulink file. Examine the influence of the sampling period on the digital PI-based closed-loop performance.

12. Conclude about the relevance of this digital PI-based rotary speed servo-control.

1.5 Digital speed controller design via direct design method in the z-domain

We now consider the design in the z-domain of a digital controller $C(z)$ so that the closed-loop step response respects the performance requirements.

1.5.1 Simulation of the rotary speed digital controller via a direct z-domain design method in Simulink

1. Recall that the digital controller $C(z)$ that fulfills the requirement specification takes the following form (see the solution discussed in the tutorial session):

$$C(z) = \frac{[\beta_1 z^2 + (\beta_0 + a_1\beta_1)z + a_1\beta_0]/b_1}{z^2 + (\alpha_1 - \beta_1)z + (\alpha_0 - \beta_0)}$$

   .

2. The sampling period will be initially set to $T_s = 0.005$s.

3. Open the `Qube_Cz_4_speed_control.m` Matlab script. It will automatically compute the digital controller coefficients of $C(z)$ based on the solution derived during the tutorial session.

4. Modify if necessary the K and T parameter values according to your identified first-order model at the beginning of the .m file.

5. Implement the designed digital controller in Simulink and verify that it fulfills the requirement specifications. You can modify the previous Simulink file used in the previous section.

1.5.2 Practical implementation of the rotary speed digital controller on the physical QUBE-Servo 2

Implement the designed digital controller in Simulink and verify that it fulfills the requirement specifications. You can modify the previous Simulink file used in the previous digital PI controller implementation section.

Modify the value of the sampling period and run again the Simulink file. Examine the influence of the sampling period on the digital PI-based closed-loop performance.

1.6 Summary and reflections

Summarize and reflect on what you have seen and learned in this lab.

Lab 2

Model-based state feedback control of a rotary inverted pendulum

In this lab, we will mainly use the QUBE-Servo 2 with the inverted pendulum.
The work done during this 4h00 lab will be noted in a report and marked. You are asked to follow the instructions given below to write your report.

Instructions for writing your lab report

A lab report is a scientific document. It should be self-contained: that is, someone who has never seen the lab instructions should be able to understand the problems that you are solving and how you are solving them. All the choices you have made should be clearly motivated. Comment and explain all your plots so as to make it easy to follow your way of thinking.

Your lab report can be written in French or in English but **do not mix both languages**. It should be organized as follows:

- a general introduction specifying the objectives of the lab
- For each assignment or exercise:
  - ▷ a brief presentation of the expected outcomes
  - ▷ a description of the obtained results in graphical and or numerical form
  - ▷ a critical analysis of the results
  - ▷ a short conclusion
- a general conclusion explaining what has been understood during the lab and any difficulties encountered.

Sending your report to the tutor

The report should be written in groups of two students and sent by email to the tutor in the form of a single pdf format file attached to the message by the deadline given by your instructor during the lab. Please indicate the following "subject" for your email when you send your report to the tutor: `Control_Lab2_Names_LabGroupNumber"`

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 2.1. 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, kangurus, 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 2.1: 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 as the technique for choosing the feedback gains. The objective is to illustrate the various stages of design that lead to the pole-placement 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 simulation example and tested in Simulink;

2. the design of a pole placement controller based on the physics informed 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.

2.1 Pole placement control - A refresher from a simulation example

2.1.1 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 !

2.1.2 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

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{2.1}$$

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 2.2 where K is a matrix of control gains. Note that here we feedback all of the system states, rather than using the system outputs for feedback.

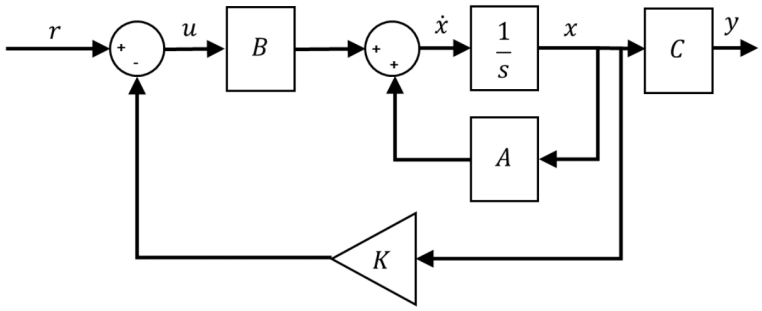


Figure 2.2: Block diagram of standard state-feedback control

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

$$u(t) = -Kx(t) + r(t) \tag{2.2}$$

where $K \in \mathcal{R}^{m\times n}$ is the feedback gain vector. Applying this to the first equation of (2.1) gives the closed-loop system equation

$$\dot{x}(t) = Ax(t) - BKx(t) + Br(t) = (A - BK)x(t) + Br(t). \tag{2.3}$$

2.1.3 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

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{2.4}$$

where n is the dimension of the state vector x. Then, we say that the linear system described by (2.1) 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)
```

2.1.4 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 (2.3) to obtain a desired characteristic equation

$$\prod_{i=1}^{n}(s - \lambda_i) = s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0 = 0 \tag{2.5}$$

where $\lambda_i \in \mathbb{C}$ and $a_i \in \mathbb{R}$ for $i \in \{0, 1, \ldots, n-1\}$.
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.

2.1.5 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.

2.1.5.1 Analytical determination of the feedback gains

Consider a linear time-invariant (LTI) system described by the following second-order ordinary differential equation:

$$\ddot{y}(t) + \dot{y}(t) - 2y(t) = \dot{u}(t) + u(t) \tag{2.6}$$

1. Show that the transfer function of the system is

$$G(s) = \frac{Y(s)}{U(s)} = \frac{s+1}{s^2 + s - 2} \tag{2.7}$$

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

$$\frac{Y(s)}{U(s)} = \frac{Y(s)}{Z(s)} \times \frac{Z(s)}{U(s)} \tag{2.8}$$

   with

$$\frac{Y(s)}{Z(s)} = s + 1 \tag{2.9}$$

   and

$$\frac{Z(s)}{U(s)} = \frac{1}{s^2 + s - 2} \tag{2.10}$$

   Show that (2.9) and (2.10) are respectively equivalent to

$$y(t) = \dot{z}(t) + z(t) \tag{2.11}$$
$$\ddot{z}(t) + \dot{z}(t) - 2z(t) = u(t) \tag{2.12}$$

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

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{2.13}$$

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

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{2.14}$$

   with

$$A = \begin{bmatrix} -1 & 2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{2.15}$$

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

$$|\lambda I - A| = \lambda^2 + \lambda - 2 = 0. \tag{2.16}$$

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:

$$(\lambda + 1)(\lambda + 2) = \lambda^2 + 3\lambda + 2 = 0 \tag{2.17}$$

   The control law is:

$$u(t) = k_r r(t) - K\, x(t) \tag{2.18}$$

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

$$K = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \tag{2.19}$$

   Show by plugging (2.18) into (2.14) that the closed-loop state-space model becomes:

$$\begin{cases} \dot{x}(t) = A_{CL}x(t) + B_{CL}r(t) \\ y(t) = Cx(t) \end{cases} \tag{2.20}$$

   with

$$A_{CL} = A - BK = \begin{bmatrix} -1 - k_1 & 2 - k_2 \\ 1 & 0 \end{bmatrix}, \quad B_{CL} = k_r B = \begin{bmatrix} k_r \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{2.21}$$

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

$$\lambda^2 + (1 + k_1)\lambda - 2 + k_2 = 0 \tag{2.22}$$

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

9. Show that equating the coefficients of the polynomial (2.22) with the desired polynomial in (2.17) leads to

$$\begin{cases} k_1 = 2 \\ k_2 = 4 \end{cases} \tag{2.23}$$

2.1.5.2 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)
```

2.1.5.3 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 2.3.

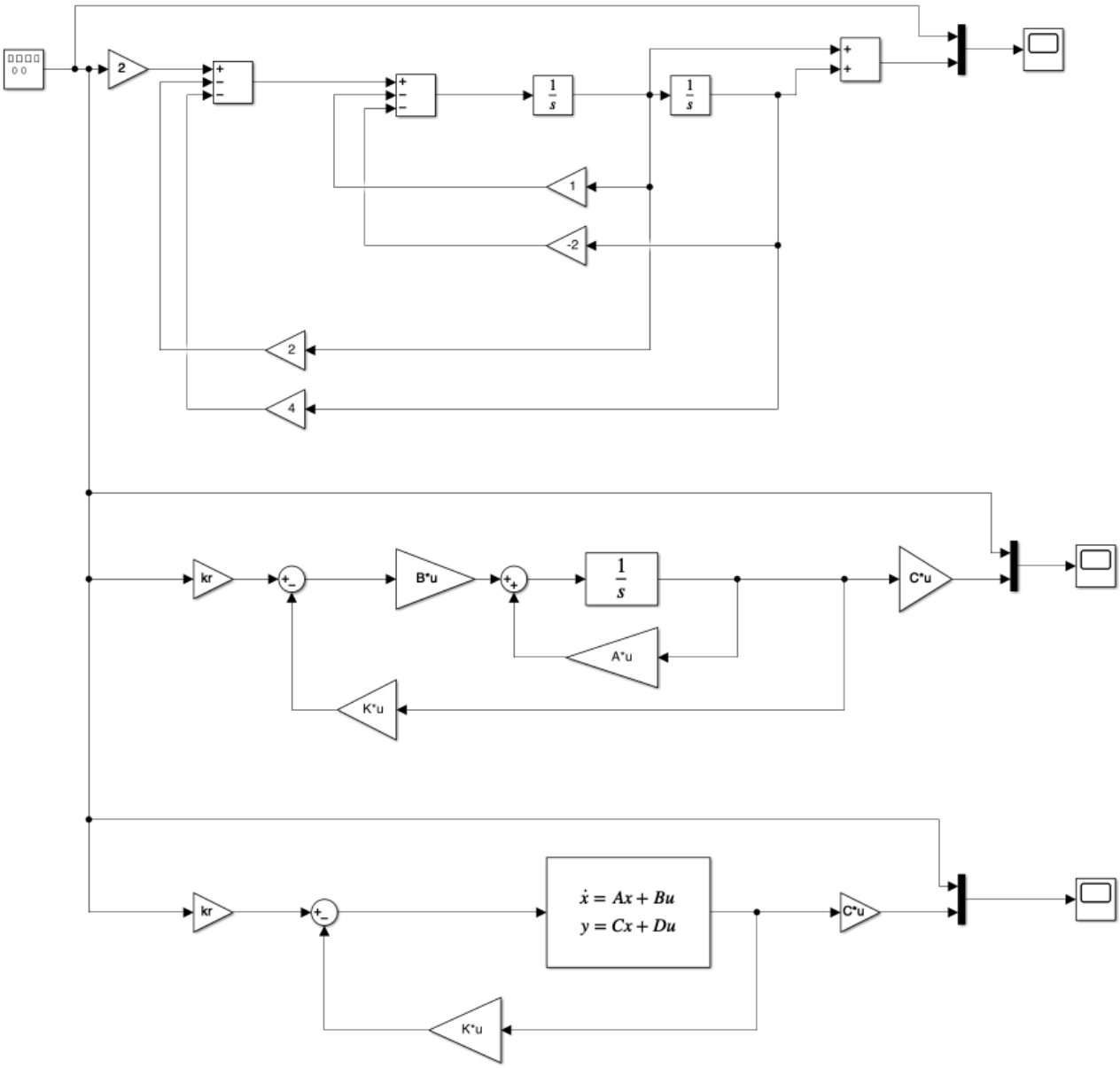


Figure 2.3: Different implementations of the full state feedback control with Simulink

Configure the signal generator and the configuration parameters as shown in Figure 2.4 and Figure 2.5 respectively.

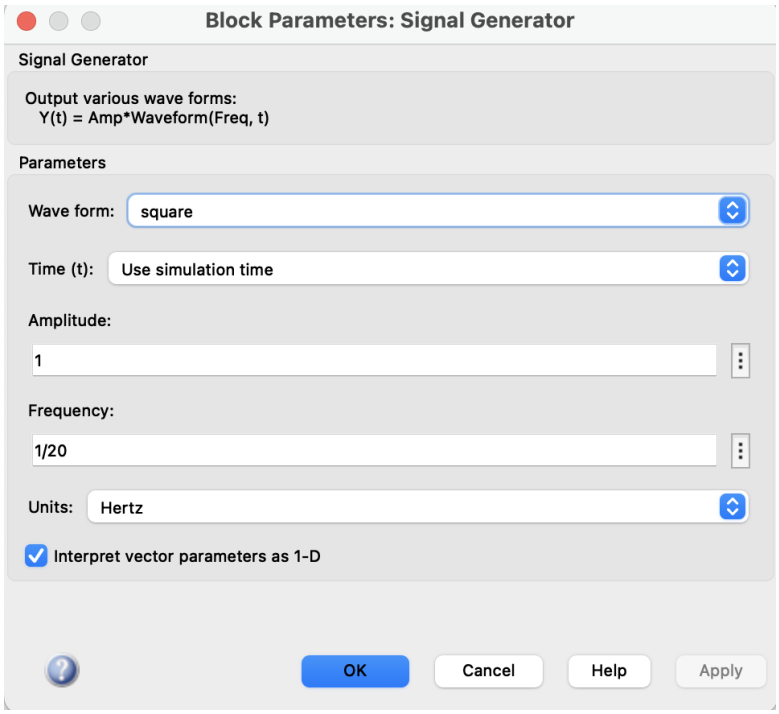


Figure 2.4: Setting of the signal generator block

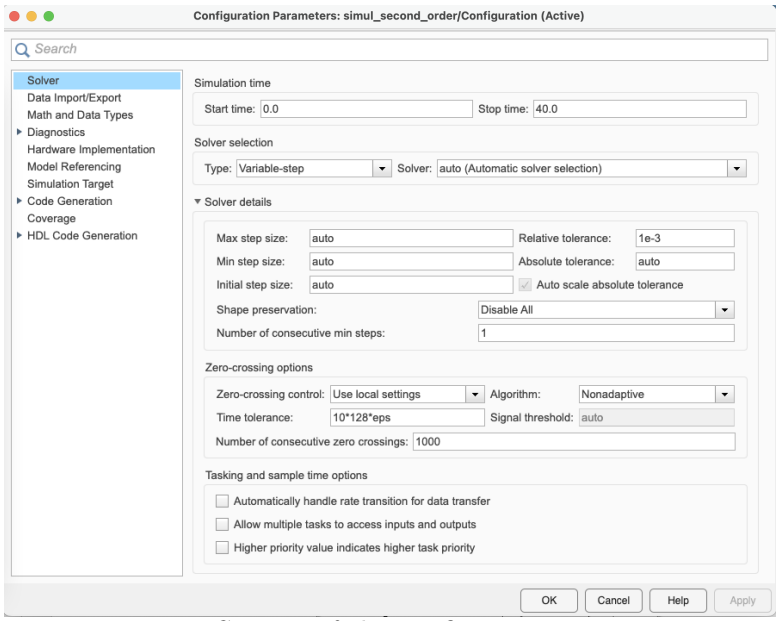


Figure 2.5: Setting of the configuration parameters

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.

2.2 Stabilization of the inverted pendulum by full state feedback

2.2.1 Download of the files required for the lab

1. Download the zipped file `Lab2.zip` from the course website and save and unzip it in your `Documents/Matlab/Control_Lab/` folder.

2. Start Matlab.

3. Important ! By clicking on the browse for folder icon  , **change the current folder of Matlab so that it becomes your `Documents/Matlab/Control_Lab/` folder** that contains the files needed for this lab.

4. In the Current Folder window of Matlab, click right on the folder `Documents/Matlab/Control_Lab/Lab2` and select `add to path the selected folder`.

5. Double-click on the folder `Lab2` so that it becomes your current folder. You should see the different `.slx` and `m.` files needed for this Lab.

2.2.2 State-space model of the inverted pendulum

The rotary pendulum model is shown in Figure 2.6. 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.

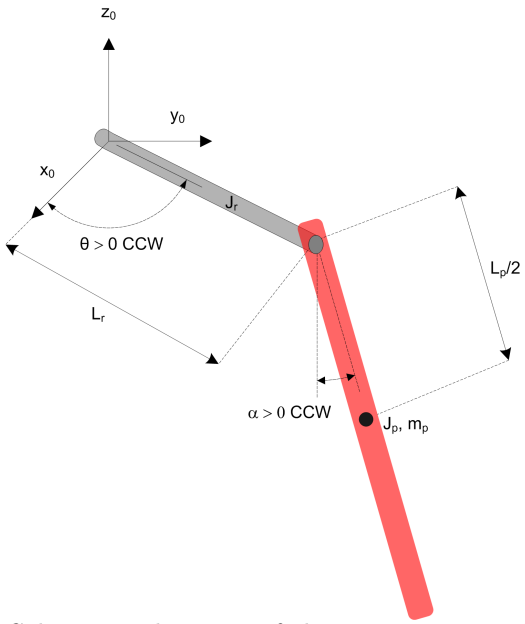


Figure 2.6: Schematic diagram of the rotary inverted pendulum

2.2.2.1 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 systems:

- 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 pendulum 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

$$x(t) = \left[\theta(t); \alpha(t); \dot{\theta}(t); \dot{\alpha}(t)\right] \tag{2.24}$$

the linearized state-space model of the non-inverted pendulum is:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{2.25}$$

with

$$A = \frac{1}{J_T}\begin{bmatrix} 0 & 0 & J_T & 0 \\ 0 & 0 & 0 & J_T \\ 0 & \frac{1}{4}m_p^2 L_p^2 L_r g & -\left(J_p + \frac{1}{4}m_p L_p^2\right) D_r & \frac{1}{2}m_p L_p L_r D_p \\ 0 & -\frac{1}{2}m_p L_p g\left(J_r + m_p L_r^2\right) & \frac{1}{2}m_p L_p L_r D_r & -\left(J_r + m_p L_r^2\right) D_p \end{bmatrix} \tag{2.26}$$

$$B = \frac{1}{J_T}\begin{bmatrix} 0 \\ 0 \\ J_p + \frac{1}{4}m_p L_p^2 \\ -\frac{1}{2}m_p L_p L_r \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.27}$$

2.2.2.2 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 2.7. 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.

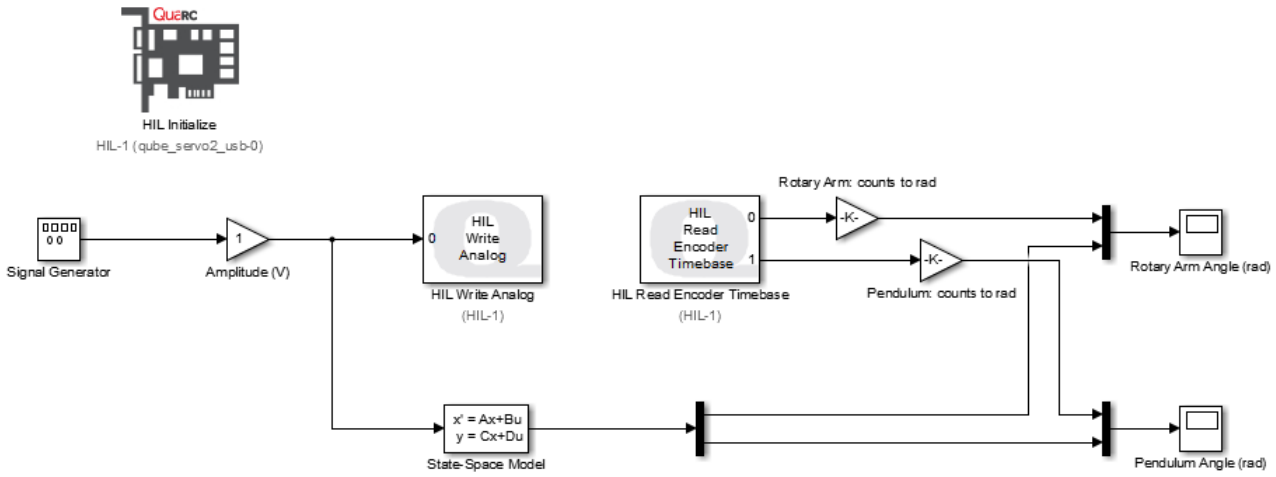


Figure 2.7: Applies a square wave voltage and displays the measured responses and the simulated model pendulum responses

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 2.8. 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.

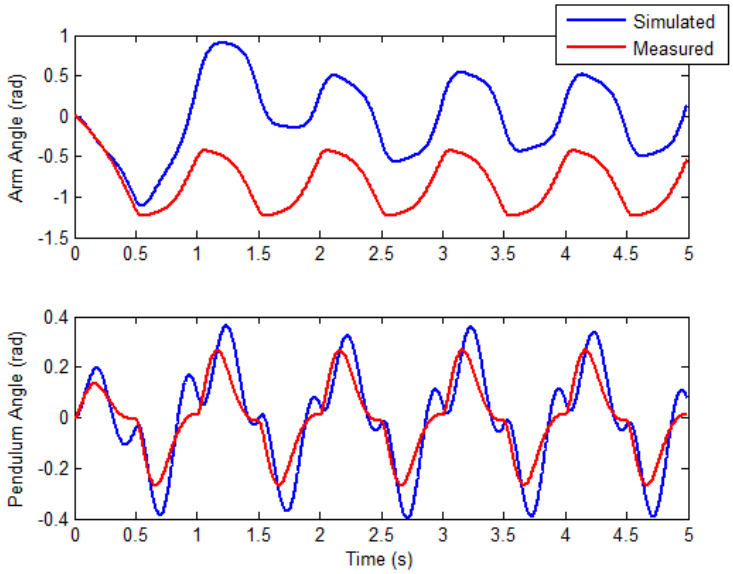


Figure 2.8: Step response of the pendulum system.

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.

2.2.3 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 2.1.

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 2.1: 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.

2.2.3.1 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?s 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

$$p_1 = -\sigma + j\omega_d \tag{2.28}$$

$$p_2 = -\sigma - j\omega_d \tag{2.29}$$

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 2.9.

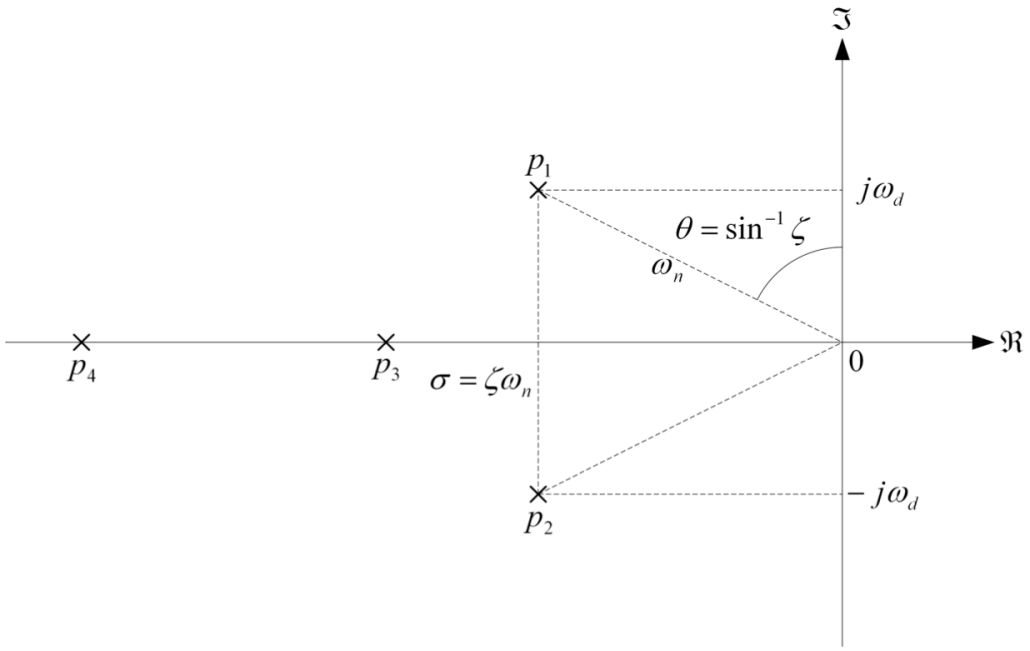


Figure 2.9: Desired closed-loop pole locations

2.2.4 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 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.

2.2.5 Implementation of the balance control to the rotary inverted pendulum

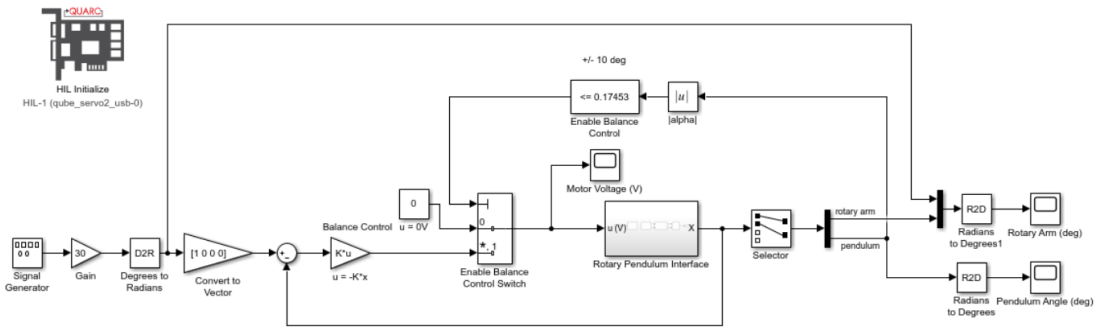


Figure 2.10: Simulink model used with pole placement-based balance controller

1. Open the `balance_inverted_pend_poleplace.slx` Simulink model whose contents is shown in Figure 2.10.

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.

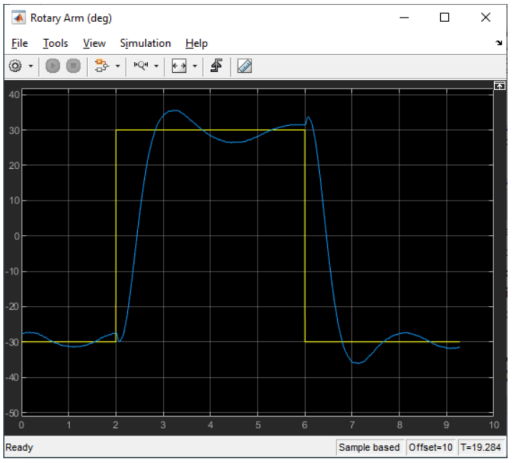
6. When the QUBE-Servo 2 turns green, gently and manually rotate the pendulum in the upright position until the controller engages. Observe the scopes. The balance state-space control should work fine. There i so magic here even if this is quite amazing.

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 2.14 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 2.1? If not, give one reason why there is a discrepancy. Modify the desired closed-loop poles to get better results.

9. Stop the controller.

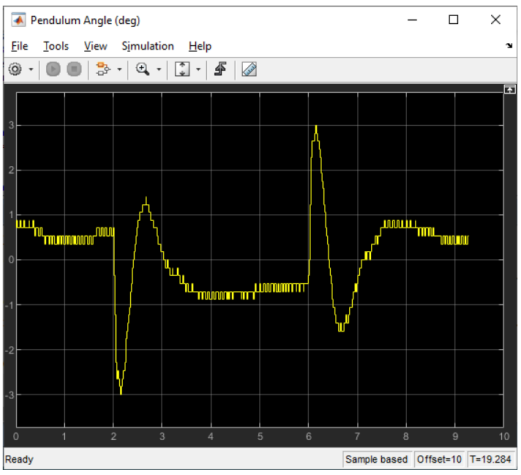
Note that the encoders only provide measurements for the arm and pendulum angular positions. We need to also measure or estimate the arm and pendulum velocities in order to perform full state feedback. In this lab, we will estimate the two angular speeds from the angular position measures by using high-pass 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.



0.3

(a) b

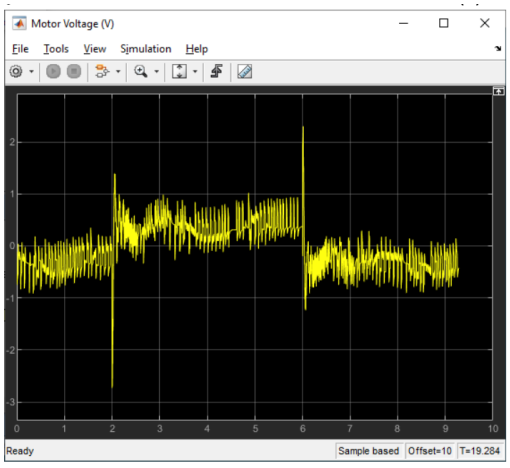
Figure 2.11: Rotary Arm



0.3

(a) b

Figure 2.12: Pendulum



0.3

(a) b

Figure 2.13: Motor input voltage

Figure 2.14: QUBEServo 2 example rotary pendulum response using default gain

2.3 Bonus - Implementation of the swing-up and balance control

We will here without studying the theory behing run a Simulink model that swings up and balances the pendulum on the QUBE Servo 2 rotary pendulum system

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. Open the `swingup_inverted_pend.slx` Simulink model.

3. Check that the swing-up control parameters are set to:

   - mu = 50 m/s/J
   - Er=30 J
   - u_max = 6 m/s^2

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

5. Click on `Monitor & tune` to build and run the controller. The pendulum should begin going back and forth until it swings up to the vertical position.
   If not, gently perturb manually the pendulum with your finger.
   Click on the `Stop` button in the Simulink tool bar if the pendulum goes unstable.

6. Stop the controller.

7. Repeat the experiment with your state feedback gain matrix K instead of the matrix gain set in the given Simulink file.

8. Power *OFF* the QUBE Servo 2.

2.4 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. We will further explore its use during the Control engineering labs next year.

English to French glossary

bandwidth	:	bande passante
crane	:	grue
closed-loop system	:	système bouclé
cut-off frequency	:	fréquence (ou pulsation) de coupure
damped frequency	:	pulsation amortie
damping ratio	:	coefficient d'amortissement
drag	:	traînée
feedback	:	contre-réaction
feedback system	:	système à contre-réaction
hoisting device	:	dispositif de levage
impulse response	:	réponse impulsionnelle
integral wind-up	:	emballement (de l'action) intégral
input	:	entrée
gain	:	gain
heading angle	:	angle de cap
linear time-invariant (LTI)	:	linéaire invariant dans le temps
motor shaft	:	arbre moteur
output	:	sortie
overdamped	:	sur-amorti
overshoot	:	dépassement
rise time	:	temps de montée
road grade	:	inclinaison de la route
robot arm joint	:	articulation d'un bras de robot
root locus	:	lieu des racines
setpoint	:	consigne
settling time	:	temps de réponse
steady-state gain	:	gain statique
steady-state response	:	réponse en régime permanent
steering	:	direction
step response	:	réponse indicielle
stream	:	courant

time-delay	:	retard pur
time-invariant	:	invariant dans le temps
transient response	:	réponse transitoire
throttle	:	accélérateur
undamped	:	non amorti
undamped natural frequency	:	pulsation propre non amortie
underdamped	:	sous-amorti