



TÉCNICO LISBOA

MEEC - Mestrado em Engenharia Eletrotécnica e de Computadores

Controlo Multivariável, Não-linear e Ótimo (CMN-O)

2024/2025

Lab project

***Linear State Feedback Control
of an Inverted Pendulum***



J. Miranda Lemos, José A. Gaspar, A. Vale, C. Bispo

DEEC, Área Científica de Sistemas, Decisão e Controlo



The plant model, the design of the linearized controller and its implementation in SIMULINK has been done by **Samuel Balula**, in the framework of his M. Sc. dissertation, under the supervision of Prof. J. Miranda Lemos. Prof. **João Pedro Gomes** contributed with software for the interface between the optical encoder and the data acquisition boards. Prof. José Gaspar prepared the English version and reviewed the document.

Contact: João Miranda Lemos, jlm1@inesc-id.pt

Objectives

After completing this work, students should be able to:

1. Design a state feedback controller for a linear system described by a state model, including a state observer.
2. Test their design through simulations performed in SIMULINK.
3. Test their design in the real system using SIMULINK and replacing the block that simulates the plant (inverted pendulum) by blocks connected to A/D and D/A converters interconnecting SIMULINK computations with the physical system in real time.

Therefore, this work illustrates a situation of **fast prototyping**, in which a simple **cyber-physical system** (a system with interacting physical and computational parts) is created.

In this example, the interconnection between the “cyber” and the “physical” parts of the plant to achieve its objective (equilibrate the pendulum) is striking. Indeed, the pendulum will not stand up if the “cyber” part (the controller) is not working properly.

Bibliography

- Course slides. *Available in Fénix, on the course web page.*
- João Miranda Lemos (2019). *Controlo no Espaço de Estados*, Chap. 5.
- Franklin, Powell and Emami-Naeini. *Feedback Control of Dynamic Systems*, Pearson/Addison-Wesley (several editions). Chap. 7.
- Samuel Balula (2016). *Nonlinear control of an inverted pendulum*. M. Sc. Thesis, IST, Universidade de Lisboa. *Available in Fénix, on the course web page.*

Work to perform

After completing the work, each group of students must deliver a report with the answers to the questions indicated below. This report must contain:

- The assignment identification

- The identification of the students (numbers and names)
- The answers to the questions indicated below identified by their number
- The answers must be concise.

By submitting a report, the students confirm that it is original and corresponds to the work done by those signing it. Plagiarism will cause the grade to be zero, independently of the sanctions determined by the applicable law and IST/Univ. of Lisbon regulations.

Lab session planning

- Before **Lab Session 1** (“home” preparation):
 - Follow a tutorial to learn the basics of MATLAB and SIMULINK.
 - Write a MATLAB macro to serve as a basis to implement different controllers (that is, to compute the controller and observer gains) and determine the step and frequency responses of the controlled plants (see below).
 - Write a SIMULINK model to test the controlled system in simulation.
 - Perform simulations using different design configurations.
- **Lab Sessions 1 & 2:** Finalize the work you have done at “home” for the controller design and try different configurations in simulation. Document your results. Get insights on the performance impact of the different design options tested.
- **Lab Sessions 3 & 4:** Using a given SIMULINK block diagram and a MATLAB macro file, replace the controller and observer gains with the ones you have previously computed and test the controller on the real system.
- **Lab Sessions 5 & 6:** Try different configurations for your observer and controller. Document your work.

Each lab session lasts 1.5 hours.

Description of the plant to control

The plant to control, shown in Figure 1, consists of an inverted pendulum, connected at the base to a horizontal bar that can be rotated in the horizontal plane by a servo motor with a vertical axis. This plant is manufactured by QUANSER.



Figure 1: The plant to control: QUANSER rotary inverted pendulum.

The plant is provided with sensors that measure the angle of the motor shaft with respect to a reference direction (a potentiometer connected to the shaft), and the angle of the pendulum with respect to the vertical (an optical encoder).

The manipulated variable is the tension u applied to the motor through a power amplifier, and the process output to control, y , is the rotating angle of the pendulum with respect to the vertical axis; while keeping it balanced on its upright position.

The control objective

The control objective consists of moving the basis of the pendulum, using the motor, such that the upper tip of the pendulum is equilibrated in the upper-pointing position.

Plant model

The plant is a nonlinear system. The derivation of the nonlinear model can be found in Chapter 2 of Samuel Balula's thesis, although knowledge of the derivation specifics and corresponding linearization is not required for the students to successfully develop the work proposed here.

An equilibrium point consists of the pendulum pointing upwards, with all the variables still. This equilibrium corresponds to all the variables being zero. If the variables denote increments with respect to this equilibrium, a linearized model that represents the plant with reasonably good accuracy is given by

$$\dot{x} = Ax + Bu, \quad (1)$$

$$y(t) = Cx(t) + Du(t). \quad (2)$$

Matrices A , B , C and D are in the file: **IP_MODEL.mat**.

The state is the vector of dimension 5 given by (see the schematic on Figure 2)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \alpha \text{ [rad]} \\ \dot{\alpha} \text{ [rad/s]} \\ \beta \text{ [rad]} \\ \dot{\beta} \text{ [rad/s]} \\ i \text{ [A]} \end{bmatrix}$$

which α is the angle of the horizontal bar, β is the angle of the pendulum with respect to the vertical, and i is the motor current.

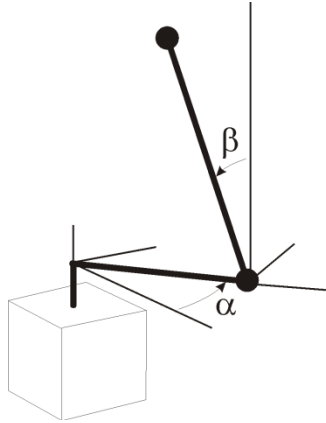


Figure 2: Pendulum schematic.


Work to perform (🏠 to do at home; 🧪 to do at the lab)

1. 🏠 **Characterize the model.** What are the eigenvalues of matrix A ? What can you say about these eigenvalues?


Suggestion: Run the command `load IP_MODEL.mat`. To know which variables are in the MATLAB workspace use the command `who`. Use the MATLAB function `eig` to compute the eigenvalues of matrix A .

2. 🏠 **Characterize the open loop system in terms of controllability.** If the system is controllable, then we can design a state feedback controller such that the closed-loop eigenvalues can be placed anywhere in the complex plane. The pair (A, B) is controllable if the rank of the controllability matrix is equal to the dimension of the state. To check that the model is controllable, use MATLAB macro to compute the controllability matrix.


Suggestion: Use the function `ctrb` to find the controllability matrix, and the function `rank` to find the rank (number of linearly independent rows or columns) of a matrix.

3.  Characterize the open loop system in terms of observability. A state realization is observable if the observability matrix has rank equal to the dimension of the state. Start by considering that the only output is the pendulum angle (x_3). In this case, what is the matrix C ? Repeat then the study for the case in which you measure x_1 and x_3 . To check that the model is observable, use MATLAB macro to compute the observability matrix.

Suggestion: Use the function **obsv** to find the observability matrix, and the function **rank** to find the rank (number of linearly independent rows or columns) of a matrix.

4.  Plot the Bode diagram of the open-loop system. Write a MATLAB macro to plot the Bode diagram of the open loop system in a range of frequencies that you find convenient. Comment on the diagram shape considering the model.

Suggestion: Use the MATLAB functions **ss** to build the state model, **ss2tf** to obtain the transfer function from the state model and **bode** to plot the Bode diagrams.

5.  Compute the Regulator Gain. Write a MATLAB macro to find the vector of gains of the controller. In this phase of the design process assume that all the components of the state are available.

The state feedback controller is defined by

$$u(t) = -Kx(t). \quad (3)$$

When coupled with (1), the control law (3) yields the closed-loop model

$$\dot{x} = (A - BK)x. \quad (4)$$

If the pair (A, B) is controllable (a condition checked in 2.), then, it is always possible to find the vector of controller gains K such that the closed loop dynamics $A - BK$ has its eigenvalues at the specified chosen positions, wherever they might be. Therefore, to design the controller, one approach might be to specify the closed-loop eigenvalues and then compute K accordingly. However, here we will follow a different path, although intimately connected to it. The controller will be obtained as the solution to an optimization problem. Thus, we compute K such as to minimize the quadratic cost

$$J = \int_0^{\infty} (x^T Q_r x + R_r u^2) dt. \quad (5)$$

The solution to the problem of minimizing J defined by (5) amounts to finding the positive definite matrix P that verifies the Algebraic Riccati Equation (ARE)

$$A^T P + P A^T - P B R_r^{-1} B^T P + Q_r = 0, \quad (6)$$

and then computing K from

$$K = R_r^{-1} B^T P. \quad (7)$$

Use MATLAB function `lqr` to compute the state feedback vector of gains K . Try different values for Q_r (a positive semidefinite matrix) and R_r (a positive scalar). You may compute the closed loop poles that result from a given choice for the pair (Q_r, R_r) by computing the eigenvalues of $A - BK$. Use the MATLAB function `eig` for that.

6. Build a SIMULINK block diagram to simulate the controlled system.

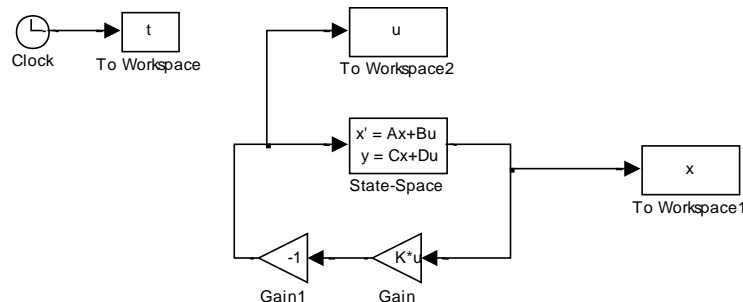


Figure 3: SIMULINK block diagram to test the state feedback controller.


```

% Furuta pendulum - State feedback test
%
load('IP_MODEL.mat'); %%Load Matrices A, B, C, D

Qr = diag([10,0,1,0,0]); %Weight Matrix for x
Rr = 1; %Weight for the input variable

K = lqr(A, B, Qr, Rr); %Calculate feedback gain

%-----
% Simulate controller

x0=[0.1 0 0 0 0]';
D=[0 0 0 0 0]';
T=2; % Time duration of the simulation

sim('statefdbk',T);


gg=plot(t,x);
set(gg,'LineWidth',1.5)
gg=xlabel('Time (s)');
set(gg,'FontSize',14);
gg=ylabel('\beta (rad)');
set(gg,'FontSize',14);

%-----
% End of file

```

Figure 4: A MATLAB macro to perform simulations with the SIMULINK block diagram of Figure 3 to test the controller gains

Figure 3 shows a block diagram used to test the controller gains. This block diagram is called by the macro shown in Figure 4, that also contains the call to function `lqr`, used to compute K . Notice that, since the reference is zero, the excitation must come from the initial conditions that must be non-zero.

7.  Compute the Estimator Gain. Write a MATLAB macro to find the vector of gains of the observer (state estimator).

The observer is designed to estimate the state x , i.e., the \hat{x} . It consists of a replica of the plant model excited by the same input u , plus the difference

between what one expects the plant output y to be (given by $C\hat{x}$) and the observed output, y . In mathematical terms, the observer is defined by the differential equation

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \quad (8)$$


The observer adds poles to the closed loop system. These poles are the eigenvalues of $A - LC$.

```
G = eye(size(A)); %Gain of the process noise
Qe = eye(size(A))*10; %Variance of process errors
Re = eye(2); %Variance of measurement errors

L = lqe(A, G, C, Qe, Re); %Calculate estimator gains
```

Figure 5: Software to design the vector of observer gains

The vector of observer gains is designed such that the estimation error converges to zero. A way to do this is to employ the Kalman filter, for which the gain is computed using the MATLAB function `lqe`. Figure 5 shows the code to use this function.

8.  Build a SIMULINK block diagram to simulate the controlled system, using the controller and observer that you have designed. Show the simulation results that you have obtained.

When using the observer, the feedback is not from the state (assumed to be unavailable for measure) but from its estimate, according to

$$u(t) = -K\hat{x}(t). \quad (9)$$

Combining the observer equation (8), and the control equation (9), it is possible to write the following state model for the controller

$$\dot{\hat{x}} = (A - BK - LC)\hat{x} + Ly, \quad (10)$$

$$u(t) = -K\hat{x}(t). \quad (11)$$

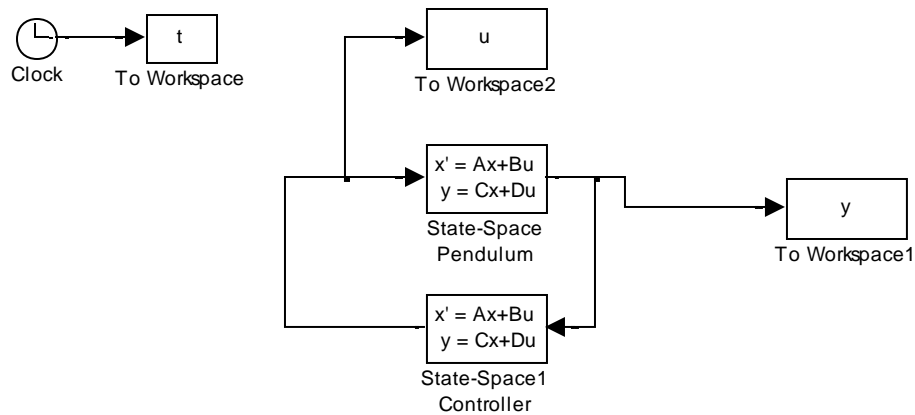


Figure 6: Interconnection of plant and controller, represented by state models.

According to (10,11), the controller is represented by a state model with input y (the plant output), and output u (the plant input). Therefore, the new dynamics matrix is $A - BK - LC$ (the controller “A” matrix), the input matrix is L (the controller “B” matrix), and the output equation is $-K$ (the “C” matrix of the controller). We may therefore use the state-space block of SIMULINK to define the controller in a compact way, as in Figure 6.

9. ✂ Lab sessions 1 & 2. Simulate the controller and simulation setup that you have developed at “home” (points 1 to 8). Try different weights for the cost functions and compare the results obtained. Explain how you evaluate the performance of your controller in a systematic (numeric) way.

10. ✂ Lab session 3 & 4. Using the gains that you have computed above, conduct now a test on the real system.

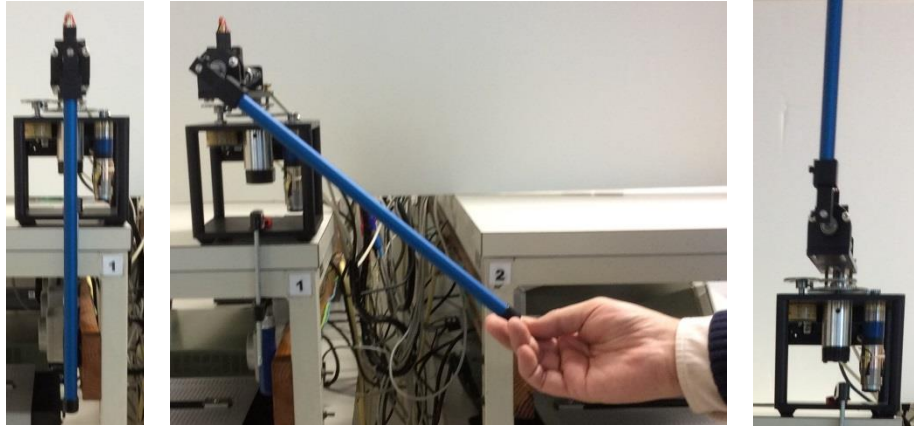


Figure 7: Pendulum setup.

Perform the following steps:

- **Attention:** recent MATLAB releases can be used to develop the work, but at the laboratory it is mandatory to use **MATLAB 2015a** installed on the desktops available on the workbenches. Simulink files and .MAT files can be exported to MATLAB2015a version. For instance, run the following command in the console to export a Simulink model:


```
save_system('model_name', 'model_name_2015a.slx', 'ExportToVersion', 'R2015a')
```
- Check what is the data acquisition board installed in the desktop of your workbench. For that sake, look on the back side of the computer on your workbench, and locate the data acquisition board.
 - If the connection is **silver**, then the board is the **NI-PCI6221**
 - If the connection is **blue**, then the board is the **NI-MIO16E4**
- Look at the software files provided to you under this course. You must select the respective files according to the acquisition board installed in the desktop.
- Open the Simulink model “**IP_TEST_MIO16E4_2015a.slx**” or “**IP_TEST_PCI6221_2015a.slx**”, accordingly, to observe the sensors data and play experiment the motor.

- Using the MATLAB editor, edit the respective m-file. Observe that it has an instruction to load the file that contains the plant model, then a few lines that configure parameters used by the data acquisition board, and finally a block of lines to design the controller and the observer. You may change the cost weights (for the controller, and for the observer) to change the controller design. Run this macro file.
- Open the Simulink model “`IP_LQG_PCI6221_2015a.slx`” or “`IP_LQG_MIO16E4_2015a.slx`”, accordingly, as depicted in Figure 9. Run the SIMULINK block diagram. To conduct an experiment, proceed as follows:
 - Place your pendulum in the vertical position, and such that it is perfectly still, as in the left picture of Figure 7.
 - Click the button ▶ to start the experiment. Wait for SIMULINK to compile the block and to appear “0” in the yellow box connected to the system (see Figure 9). When the time starts counting, wait to appear “1” in the same yellow box. Then, the sensors are calibrated, and you have now less than 6 seconds to gently raise the pendulum to the vertical upward position (as in Figure 7 center) and keep it there. The number of the yellow box switch to “2”. Finally, when the value switch to “3”, the controller starts working and you may remove your hand while the pendulum is equilibrated automatically (Figure 7 right). Use the button ■ to stop the program or (preferably) wait for the time-out. Remark that there are start/stop buttons for MATLAB and for SIMULINK.

Observe, register, and store the results. Compare the time responses obtained using the real system with the ones obtained in simulation. Comment on the differences.

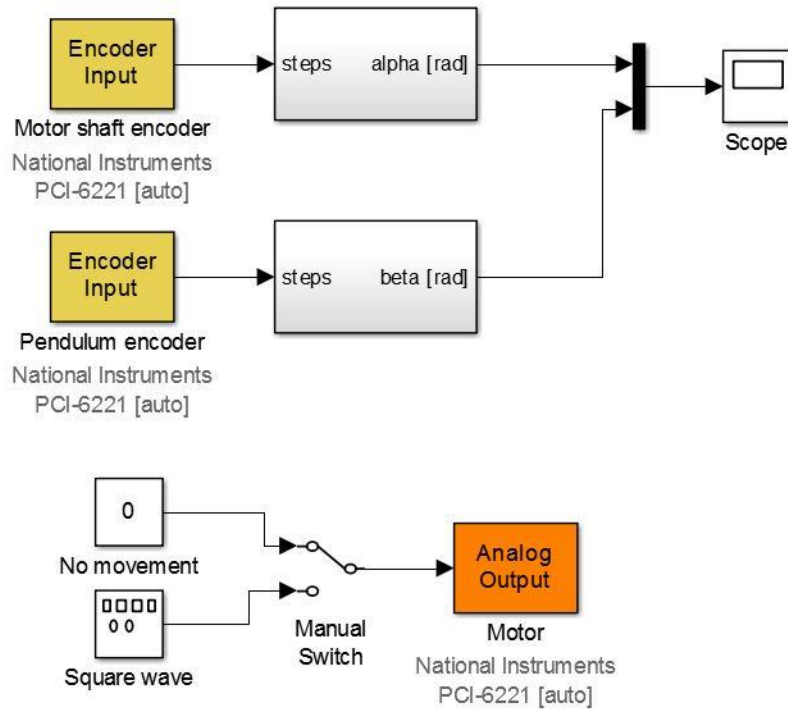


Figure 8: Simulink model to observe the sensors data and control the motor.

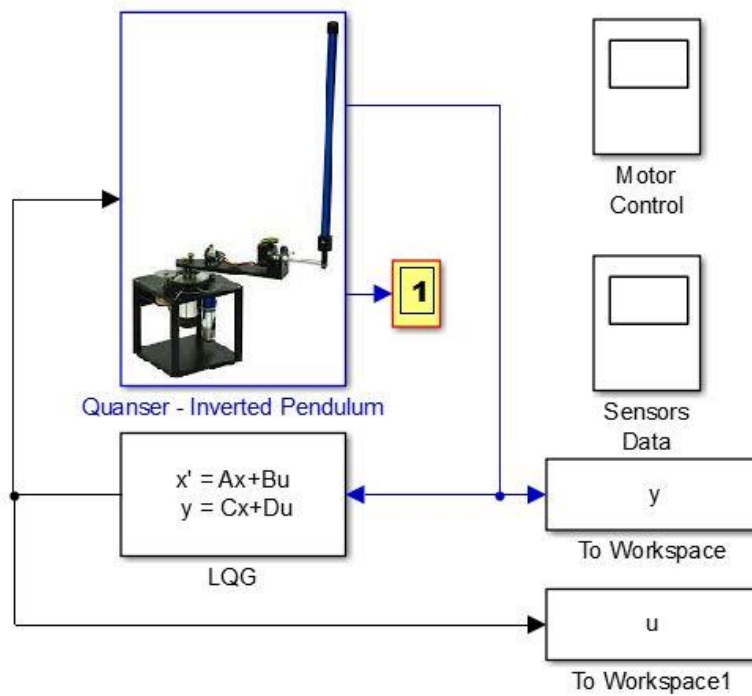


Figure 9: Simulink model to control the real inverted pendulum.

11. ✂ Lab session 5 & 6. Try other weights for the observer and/or controller. Register and comment on your results. Choose the best design and justify your decision. How can you improve the results by changing the controller in order to make the angle α stand still?

Suggestions: Compare the different options you have considered using an objective figure of merit, such as the tracking error mean square value or other.

