Project 2
Control of a Pendubot

Pr. J. Hendrickx, N. Bousselmi, F. Wielant

1 Goal of the project

Preliminary: this laboratory includes a few hours of test and trials with a real system in a lab room located in the Euler building. Due to fall season, the teams of students have to face and manage any disturbance in their planning induced by any sickness/quarantine period of one or several member(s) of the team.

The pendubot, short for PENDUlum roBOT, is an electro-mechanical system consisting of two rigid planar links (two degrees of freedom) interconnected by an axis of rotation. The first joint is actuated by a DC motor, and the second joint is unactuated. The second link is thus a simple pendulum whose motion can be controlled by action of the first link.

The pendubot is similar in spirit to the classical inverted pendulum on a cart. However, it is more challenging to design an observer and a controller for this system because it is underactuated.

The goal of this lab is to control this under-actuated open-loop unstable system. To do this, students will have to linearize and discretize the system before designing the control law.

2 Plant description

As briefly mentioned above, the pendubot is a system consisting of two aluminium alloy links coupled together at one end. One end of the first link is coupled to the shaft of a $800W$ (!) DC motor (itself coupled with an incremental encoder), while the other end is coupled trough a ball bearing (and an other incremental encoder) with the second link (see Fig. 1).

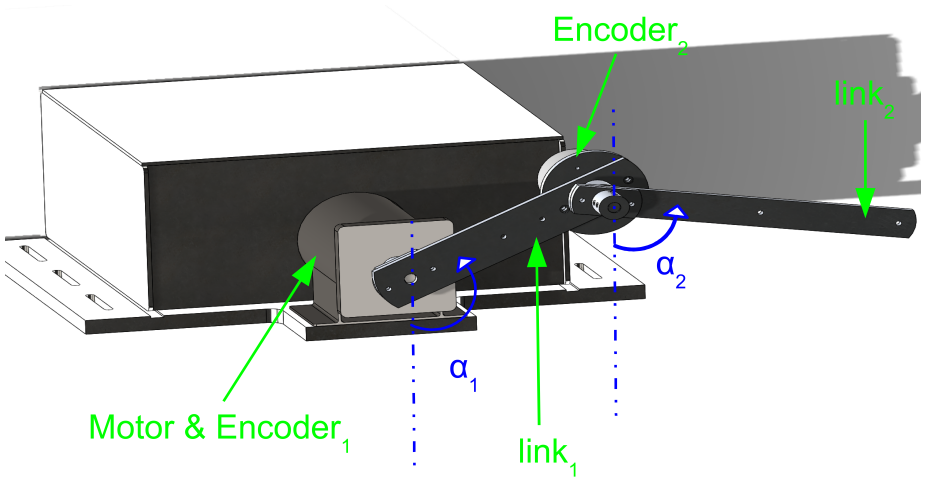

Figure 1: Overview of a pendubot system

The two links 1 and 2 form two angles with the vertical direction α_1 and α_2 respectively. The system can be completely described with for state variables: the two angles α_1 and α_2 in *radians* and the two angular velocities of the two links $\dot{\alpha_1}$ and $\dot{\alpha_2}$ in $radians.s^{-1}$, while the only command variable is the

torque of the DC motor: τ in Nm.

For the details of the physical model of the system, refer to appendix A.

3 Specifications

3.1 Objectives

According to the points below, there is two main objectives for this project.

3.1.1 Design two linear-based controllers

- Design two linear controllers (the first controller is mandatory, the second is a Bonus) that will stabilize the pendubot at a given angle:

$$\alpha_{1ref} = \frac{4\pi}{5} \quad ; \quad \alpha_{2ref} = \pi$$

  starting from a different initial angle $\alpha_{1ini} = \alpha_{2ini} = \pi$ (i.e. the first link is in vertical position above the axis, and the second link in the upper position). More precisely, the specifications for the two controllers are:

  - Controller *C1* (high-performance and fast controller):
    * Fast tracking (as fast as possible). This means that the required rise time t_r of the step response should be as short as possible.
    * Maximum overshoot of the step response: 20%.
    * Settling time should be reasonably short. There are no specific limits.
    * Static error as low as possible.
  - BONUS: Controller *C2* (robust and economic controller):
    * Slower but reasonable tracking (slower than for *C1*).
    * Maximum overshoot of the step response: 10% or less.
    * Settling time should be reasonably short. There are no specific limits.
    * Zero static error (an integrator is probably needed).
    * Energy consumption of the actuator as low as possible (e.g. by limiting or penalizing the input).

- Evaluate the performance and robustness of the designed controllers (students have to choose suitable criteria).

3.1.2 Design an advanced controller

Warning: This part needs a **considerable amount of work**. According the students' planning for the end of the semester, it is recommended to start to work on it as soon as possible (at least many weeks before the deadline of the project).

In a first time, propose a *C3* control scheme that can swing the system to $\alpha_{1ini} = \alpha_{2ini} = \pi$ from any angular position and speed, based on an energetic function, and make it running on the real system.

In a second time, adapt this controller to stabilize the system in any "stabilizable point".

As a reminder, for this point in particular, it's a good idea to make a brief review of the literature ("Google Scholar" is your friend). The goal is to get the best results and to use the Lyapunov control theory.

3.2 Methods

The controller *C1* and *C2* design procedure will consist of the following steps:

1. Linearize the nonlinear state-space model at the operating point. A linear model is obtained in this step.
2. Validate the linear model, i.e. compare the linear model with the original nonlinear model at the operating point. Compare, for example, some representative time responses.
3. Convert the continuous-time linear plant model to a discrete-time model. Choose an appropriate sampling time T_s. Comment the choice of the sampling time.
4. Validate the discrete-time model with respect to the continuous-time linear model. Compare some time and frequency responses of the two models. Think about the one step ahead prediction (for unstable system)...
5. Compare the real system with the discrete time model and the discrete time linearized model.
6. Design and test on MATLAB/Simulink the two linear closed-loop controllers *C1* and *C2*, using the obtained linear model. One controller has fast tracking, the other one is slower but more robust and economic. Use pole placement or optimal control as design method.
7. Test your controllers on the real system.
8. Compare performances of the controllers *C1* and *C2*. The performances to be compared are those ones defined in the specifications. The controllers should be compared on:
   - Not only the output responses but also the control action signals, to check control action amplitudes.
   - Using the nonlinear model only, check the closed-loop step response for operating points more and more distant from the original (equilibrium) operating point. It allows to analyze the controller robustness with respect to system nonlinearity.
   - Also, draw the Bode plot of the output noise transfer function $S = y/v$ (with $y_v = y + v$, where y is the pendubot actual output, y_v is the output measured, and v is the output noise). Compare this plot for the different controllers. How is this information related to robustness properties?

Regarding the *C3* controller, students are asked to justify their choice, and to try to analyze the stability of the closed loop system.

To guide the method, the student can first answer the following questions:

1. Give a function that is 0 for the desired point of the state-space and that is positive elsewhere.
2. If a control law can only decrease that function (i.e. dx/dt make an angle less than 90° with $gradV$), what are the equilibrium points (saddle-points identification)?
3. Why are we certain to go to a desired point?
4. How do we know that we will not go to an "irrelevant" zone (i.e. too much speed, too much turns, etc.)?
5. Students can of course iterate their answers to these question if their responses give not suitable results...
6. Switching to a local controller could be a good idea ...

4 Project report

Write a report describing all the results and conclusions. The report will consist of the following sections :

- *Section 1:* A brief introduction describing the content of the report. The students can consider this notice as a part of the report so it is not necessary to re-describe the physical system and specifications.
- *Section 2:*
  - A description of the model linearization.
  - A description of the continuous-time linear model and model validation obtained.
  - A description of the discrete-time linear model and model validation obtained.
  - The comparison with the real system (following suitable precise criteria).
- *Section 3:* A description of the two controllers *C1* and *C2*:
  - The design trials (give some advice and comments for the choice of the design parameters i.e. the weighting matrices, closed-loop poles, etc).
  - The precise control laws obtained and their implementations on the simulator.
  - The implementations of the two controllers on the real system.

  NOTE: If not actually totally implemented, students have to discuss how they would have designed and implemented the *C2* controller.
- *Section 4:*
  - Comparison of closed loop performances for the different controllers on the linear discrete-time model.
  - Comparison of closed-loop performances for the different controllers on the nonlinear model.
  - The analysis of the results obtained with the real system.
- *Section 5:* The *C3* controller and its test on the real system.
- *Section 6:* A brief conclusion. Students have to make comments on the advantages and disadvantages of the used design method.

Create a Matlab m-file related to the report, such that each section of the report corresponds to one section of the m-file. The file will contain:

- The calculations made in Matlab (controller designs, model conversion, simulations, ...).
- Plotted and displayed solutions and comparisons (controllers, time responses, frequency responses, ...).
- Relevant commentary explaining in brief the code and relating the code to the report.

5 Project equipment

5.1 Simulink nonlinear model

To test controllers on the original nonlinear model, the provided Simulink model has to be used. If the students prefer to use another simulator than the given MATLAB and Simulink one, they are of course free to choose the tool they want.

For the given simulator, five files are related to the Simulink model (the students need to put them into their working directory):

Pendubot.mdl Simulink model file which contains (see Figure 2):

- Diagram of the plant model in closed loop with controller.
- Window with online animation.
- Scopes with angle, rotation speed, and applied couple.

This file will be used to simulate the system, no modification should be made in this file except simulation parameters (start time, stop time, ...) and reference signal for arm position. The block "Controller" calls directly the function *Pendubot_reg.m* containing the control system algorithm. Initial conditions can be set by a double-click on the model block "Pendubot".

Pendubot_ini.m Initializing script for the Simulink model. It contains initialization of the sampling time value. The script has to be run before any simulation is made.

Pendubot_eqt.m Function with state equations implementing the nonlinear plant model. Do not modify this file.

Pendubota.m Function implementing the online animation. Do not modify this file.

Pendubot_reg.m Function implementing the designed control system algorithm. This function is to be modified. The function receives the actual reference signal for motor arm position α_{1d} and the state vector $[\alpha_1 - \pi, \alpha_2 - \pi, \dot{\alpha}_1, \dot{\alpha}_2]$ from the simulator and it has to send an actual value of the voltage u to be applied.

Example: of the function "Pendubot_reg.m" (this is not a solution)

```
function u=pendubot_reg(alpha1r,x)
%function u=pendubot_reg(alpha1r,x)
%
%INPUTS:
% alpha1r : reference value for first link angle [rad]
% x       : measured state vector x =   [
%                                       x1 = alpha_1 - pi
%                                       x2 = alpha_2 - pi
%                                       x3 = dalpha_1/dt
%                                       x4 = dalpha_2/dt
%                                       ]
%
%OUTPUT:
% u       : control signal (voltage) to be applied [Nm]
u=3*((alpha1r-pi)-x(1))-0.3*x(3);
```

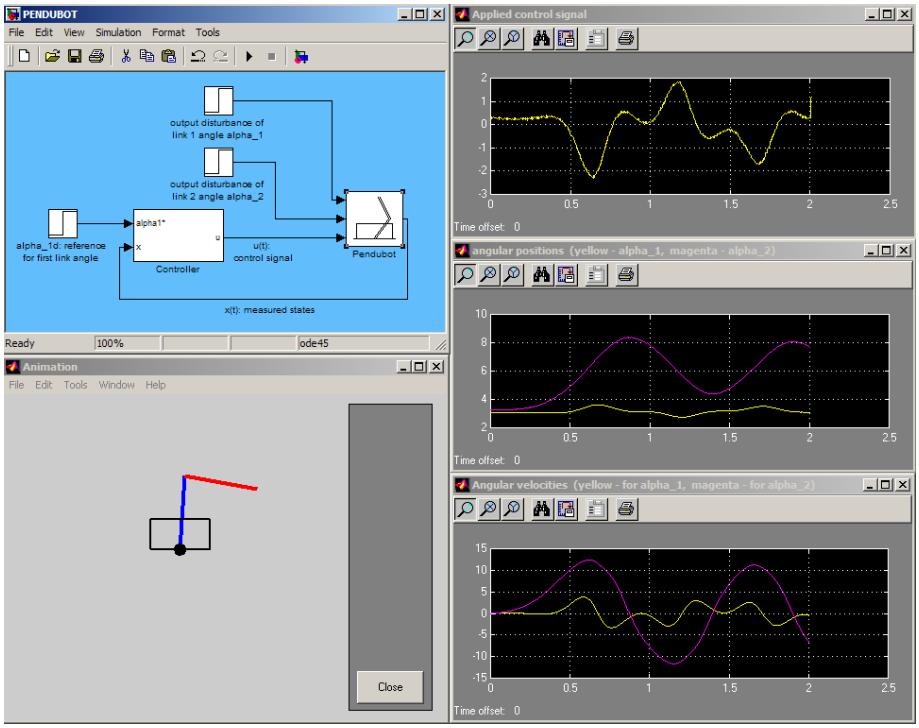

Figure 2: Simulink model Pendubot.mdl.

5.2 Real System

5.2.1 Global overview of the real system

When controllers are working in the MATLAB/Simulink environment, an other part of this project is to conduct tests on a real pendubot system. As shown on the Fig. 3, the system available for this project includes two pendubots (one traditional -with the wire-, and a second one using a camera to evaluate the two angles α_1 and α_2). Both can be used, but not simultaneously.

The global system comes with:

- an enclosure with the two pendubots and two electrical control panels;
- an other little electrical box named "*PenduBots Security Remote Control*";
- a black box named "*PenduBots Daq Adaptor*"
- a first PC running a specific real-time operating system, to compute the time critical commands and drive the data acquisition board, which deals with the Pendubots;
- a second PC running a Windows 10 OS, using for the development and for the monitoring. Students will only work on this computer.

5.2.2 Safety consideration

Given its characteristics, the pendubot is a **DANGEROUS process**. Students can not stay in the red zone around the pendubot cabinet (area delimited with red tape).

For **safety reason, and before any test session**, read carefully the *PenduBot Safety Manual* and ensure the complete understanding of all the instructions, or contact François Wielant for a briefing. If a tester does have any question, feel free to ask!
In the case of any malfunction, there is only one appropriate behavior:

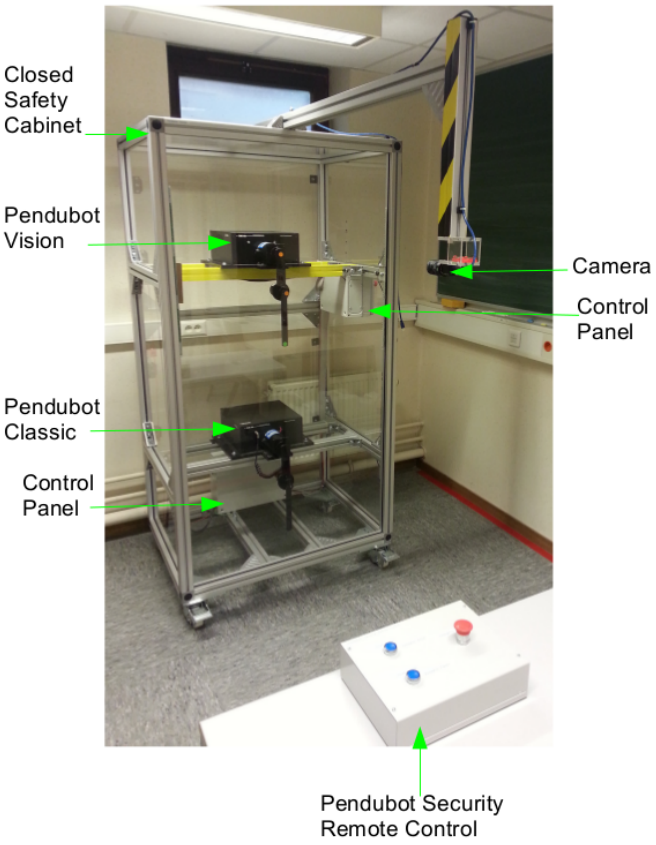

Figure 3: Pendubot system available for the this project.

1. **Press the Emergency STOP push-button**,
2. **Get out of the room**,

and call François Wielant.

5.2.3 User guide for the software manipulation

The software students will use to conduct their experiments is developed with the NI LabVIEW 2021 environment. A quick overview of this development system is shown by following the link: LabVIEW Environment.

To use the software and test several controllers, students can follow these steps:

1. Start the computer using the Win10 OS (login: *labo-user*, password: *BLAbla1234*) and start the computer running the RT OS (check its IP address, it must be 130.104.236.225, otherwise restart it).
2. On *Moodle*, download the software file `PenduBot_LINMA2671_2022.zip` and save it on the HDD. Extract the complete compressed folder in `D:\` and rename it `PenduBot_LINMA2671_2022_Studentsname` where `Studentsname` is the name of one student. The team of students will only store and work in this directory.
3. Start *LabVIEW 2021* and open `PendubotControl_LINMA2671_2022.lvproj` in the appropriate directory. Students will see the project explorer like shown on the Fig.4

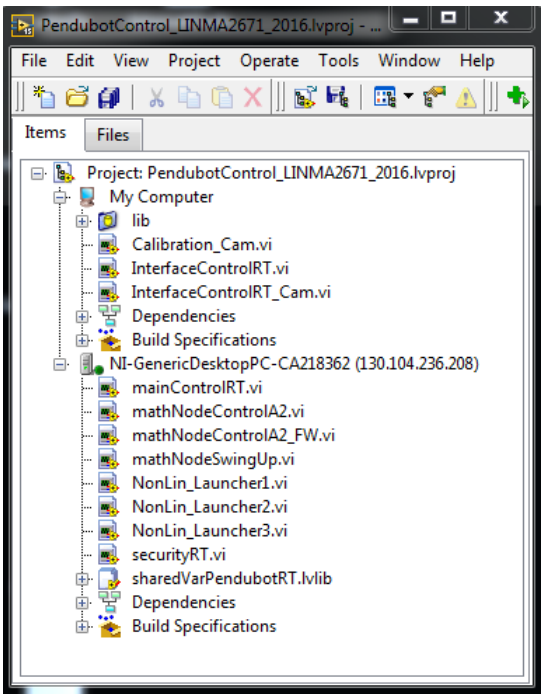

Figure 4: LabVIEW project explorer of the software.

4. In the project explorer, open (with a double left-click):

   - The code running on the RT computer: `mainControlRT.vi`,
   - The code running on the monitoring computer: `InterfaceControlRT.vi` or `InterfaceControlRT_Cam.vi` depending if you want use the classic version of the pendubot, of the one equipped with the camera (check the connections of the cables linked with the enclosure).

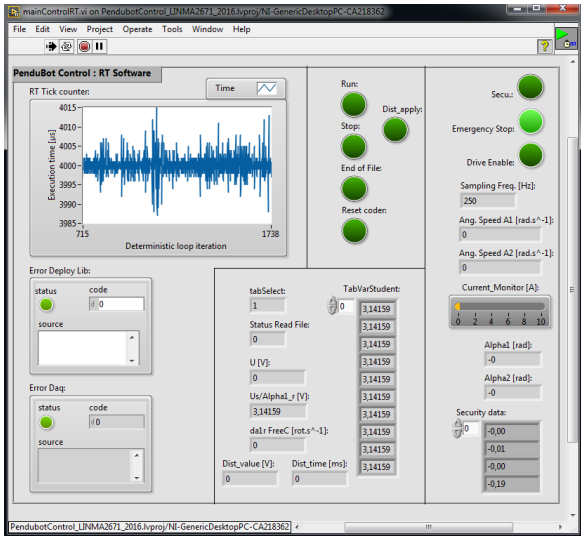

Figure 5: mainControlRT.vi

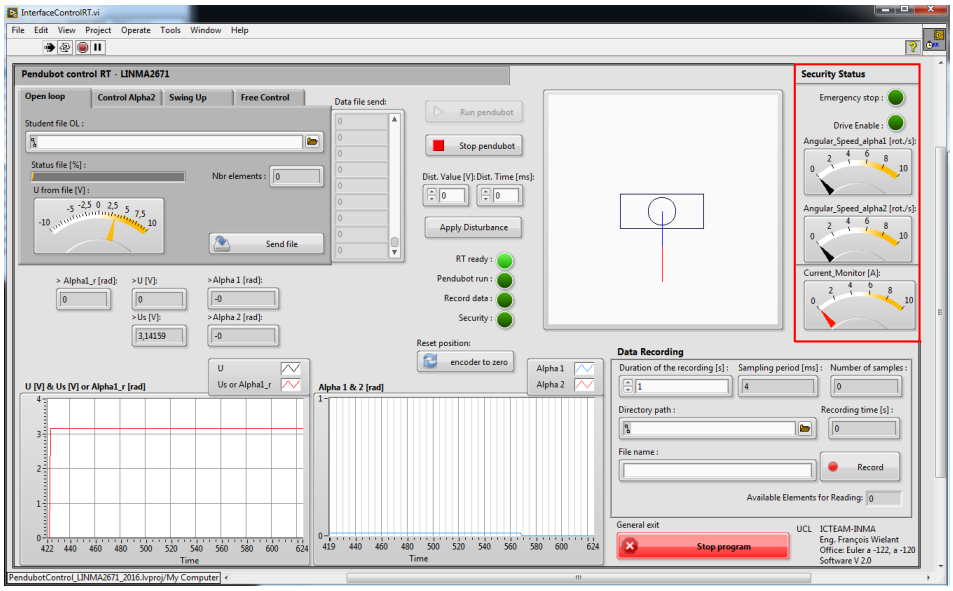

Figure 6: InterfaceControlRT.vi

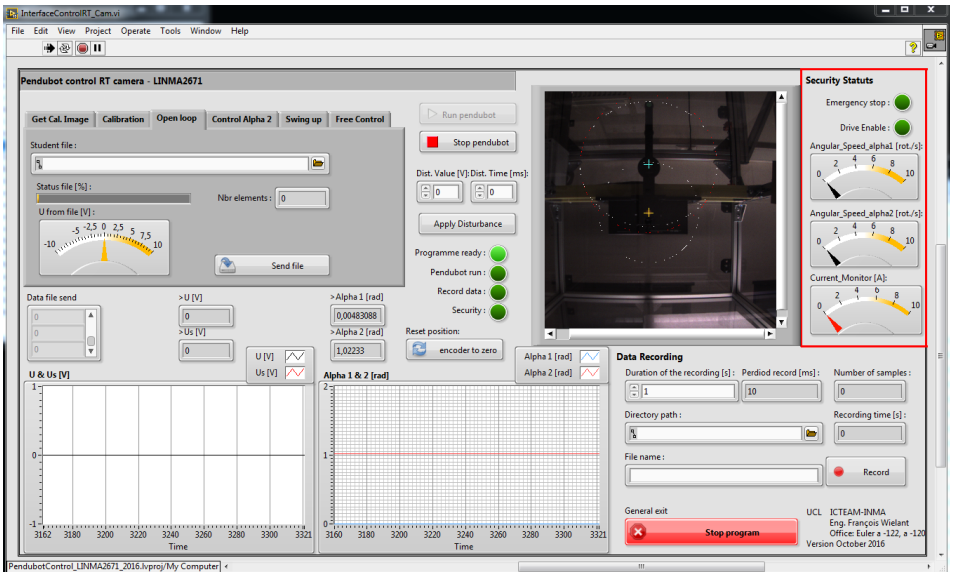

Figure 7: InterfaceControlRT_Cam.vi

5. Deploy and execute the two software by (in the same order):

   (a) Running the program `mainControlRT.vi` by clicking on its run button as presented on the figure Fig. 8. When the dialogue box `Deployment Progress` shows "`Deployment completed successfully`", click on the "`Close`" button.

Figure 8: `run` button.

   (b) Running the program `InterfaceControlRT.vi` or `InterfaceControlRT_Cam.vi` by clicking on its run button. When the dialogue box `Deployment Progress` shows "`Deployment completed successfully`", click on the "`Close`" button.

6. Now, students can conduct their experiments from the software `InterfaceControlRT.vi` or `InterfaceControlRT_Cam.vi`. This software includes 4 main tabs controls:

   - `Open loop`: to conduct test on the pendubot in open loop, the value of the *student file* (see B) are directly applied in the voltage control of the pendubot's motor driver;
   - `Control Alpha2` : to test the controller *C1* and *C2* (see 3), the pendubot is swung up, and after 5 seconds the student's controller is switched on. The controller of the students have to be coded in the `mathNodeControlA2.vi` (see C) and the data of the *student file* are the setpoints of α_1 in radians.
   - `Swing Up` : to test the controller *C3* that needs to be coded in the `mathNodeSwingUp.vi`, and the data of the *student file* are the septoints of α_1 in radians. For this particular point, an hidden function applies a open loop signal during 500 ms to transmit kinematic energy to the system (as mentioned before, there is 3 different signals, and student can choose between 1, 2 and 3 for the function input).
   - `Free control` : to set the angular speed of the first link of the pendubot with the slider button.
   - The tabs `Get Cal Image` and `Calibration` have not to be performed each time.

   To record the data of the test, please select the appropriate directory path (please do not record data in every directory of the computer).

   To run a basic test of a controller, students can execute the following steps:

   (a) Select a tab: *Control Alpha2* for example;
   (b) Select a *student file* with the setpoints of α_1;
   (c) Send this file to the RT computer by pressing the `Send file` button;
   (d) Set a duration of the recording;
   (e) Set the directory path of the file with the recorded data;
   (f) Set a file name of the file with the recorded data;
   (g) Press the `Record` Button;
   (h) Press the `Drive Enable` button on the `PenduBots Security Remote Control` box;
   (i) Press the `Run pendubot` button and the pendubot will be swing up with a mysterious controller, and after 5 seconds, the students controller will be switched on.

7. To exit the software, press the `Stop program` button on the `InterfaceControlRT.vi` software, and then stop the `mainControlRT.vi` by a click on the `Abort execution` button of LabVIEW (see Fig. 8).

6 Practical information

This project is initially proposed for a team of 2 students (one team of 3 students is allowed if the number of students is odd). An optimal way of working is probably not that everyone works simultaneously on the same topic...

For any question regarding the laboratory, do not hesitate to contact (send an e-mail with the questions before):

- François Wielant (Euler building, offices a.-120 and a.-122 or via Ms Teams);
- or Nizar Bousselmi (Euler building, office a.119).

A Plant description and model

A.1 Plant

As mentioned above, the pendubot (see Figure 9) is a system consisting of two rigid links coupled together at one end. The first link is motorized while the second is left free. The parameters in Figure 9 represent:

- l_1, l_2 : the length of the link 1 and 2 respectively,
- c_1, c_2 : the center of mass for the link 1 and 2 respectively,
- l_{c1}, l_{c2} : the distance to the center of mass for the link 1 and 2 respectively,
- α_1, α_2 : the angle between the link 1 and 2 respectively, and the vertical lower axis (the vertical axis represents 0 radian and positive angle is considered on the right of the axis),
- τ : input torque made by DC motor.

The two angles α_1, α_2 are considered to form a vector α:

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{1}$$

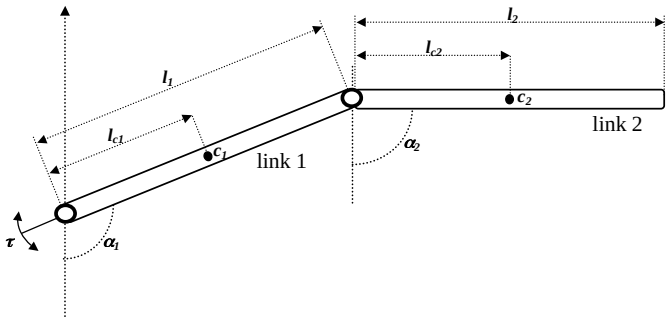

Figure 9: Pendubot system.

A.2 Physical model

In the mathematical model of the pendubot, the friction is usually neglected and the two links are assumed to be rigid bodies. The model is generally written using Euler-Lagrange formulation or equivalently by applying Newton's second law. The system is then described by two motion equations — since α is a vector — typically written as follows:

$$M(\alpha)\ddot{\alpha} + V_m(\alpha, \dot{\alpha})\dot{\alpha} + G(\alpha) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tau \tag{2}$$

where the input variable $\tau \in [\pm 3.9621]$ [Nm] is a torque created by a DC motor. In order to drive the motor, we use a control signal $u(t) \in [\pm 8]$ [V]. The relation between the control signal and the torque signal is given by:

$$\tau(t) = ku(t) \tag{3}$$

where k [V/Nm] is a fixed amplifier and DC motor torque constant gain. The matrices $M(\alpha)$ and $V_m(\alpha, \dot{\alpha})$ from (2) are the so-called inertia matrix and centripetal-Coriolis matrix, $G(\alpha)$ is called gravitational vector. The term $\dot{\alpha}$ represents the angle derivative with respect to time, i.e. angular

velocity, the term $\ddot{\alpha}$ is then the angular acceleration. For the system defined in Figure 9 they have the following form (they can differ depending on the choice of angles):

$$M(\alpha) = \begin{bmatrix} p_1 + p_2 + 2p_3\cos(\alpha_2 - \alpha_1) & p_2 + p_3\cos(\alpha_2 - \alpha_1) \\ p_2 + p_3\cos(\alpha_2 - \alpha_1) & p_2 \end{bmatrix} \tag{4}$$

$$V_m(\alpha, \dot{\alpha}) = p_3\sin(\alpha_2 - \alpha_1)\begin{bmatrix} \dot{\alpha}_1 - \dot{\alpha}_2 & -\dot{\alpha}_2 \\ \dot{\alpha}_1 & 0 \end{bmatrix} \tag{5}$$

$$G(\alpha) = \begin{bmatrix} p_4 g\sin(\alpha_1) \\ p_5 g\sin(\alpha_2) \end{bmatrix} \tag{6}$$

The five parameters p_1 - p_5 are defined in terms of the system mechanical properties as follows:

$$\begin{aligned} p_1 &= m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \\ p_2 &= m_2 l_{c2}^2 + I_2 \\ p_3 &= m_2 l_1 l_{c2} \\ p_4 &= m_1 l_{c1} + m_2 l_1 \\ p_5 &= m_2 l_{c2} \end{aligned} \tag{7}$$

with m_1 and m_2 the mass of link 1 and the mass of link 2. The values of these parameters are in our case:

$$\begin{aligned} p_1 &= 0.0148[kgm^2] \\ p_2 &= 0.0051[kgm^2] \\ p_3 &= 0.0046[kgm^2] \\ p_4 &= 0.1003[kgm] \\ p_5 &= 0.0303[kgm] \end{aligned}$$

and the coefficient g is considered to be $g = 9.81\ [m/s^2]$.

A.3 State space nonlinear model

Now the mathematical model can be reformulated in order to obtain a classical nonlinear state-space model form useful in control system theory:

$$\dot{x} = f(x) + g(x)u \tag{8}$$

A state space vector x can be specified as follows:

$$x = \begin{bmatrix} \alpha \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dot{\alpha}_1 \\ \dot{\alpha}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{9}$$

and the input signal u is considered to be the control signal $u(t)$ [V] from (3). Then, the function $f(x)$ is given by

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \\ f_4(x) \end{bmatrix} = \begin{bmatrix} \dot{\alpha} \\ M^{-1}(\alpha)(-V_m(\alpha, \dot{\alpha})\dot{\alpha} - G(\alpha)) \end{bmatrix} \tag{10}$$

and the function $g(x)$ is

$$g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ g_3(x) \\ g_4(x) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ M^{-1}(\alpha)\begin{bmatrix} k_A k_t \\ 0 \end{bmatrix} \end{bmatrix} \tag{11}$$

where α and $\dot{\alpha}$ are to be replaced by the corresponding states from (9).

This nonlinear state space model can be linearized around some given operating points using the Jacobian method.

A.4 Linearization method

Different approaches exist to linearize a non-linear mathematical model. It is sometimes possible to neglect non-linear parts, or some approximations are used to replace non-linear terms with the linear ones. For a general linearization in one operating point Taylor series expansions can be used. For the pendubot system, the linearization technique that is typically used is the so-called *Jacobian linearization*. The Jacobian linearization allows to linearize a system of differential equations at an equilibrium point.

Definition of Jacobian linearization: Consider a general system of nonlinear differential equations:

$$\begin{aligned} \dot{x} &= f(x,u) \mid x \in \Re^n, u \in \Re \\ y &= h(x,u) \mid y \in \Re \end{aligned} \tag{12}$$

where x is a vector of states with dimension n, while u and y are respectively some input and output variables. Suppose that x_e is an equilibrium point in state space and u_e is an input variable for which a linearization of (12) is needed.

First a new set of state variables z, inputs v, and outputs w is defined as follows:

$$\begin{aligned} z &= x - x_e \\ v &= u - u_e \\ w &= y - h(x_e) \end{aligned} \tag{13}$$

Then the system (12) linearized at the equilibrium point x_e, u_e has the following form:

$$\begin{aligned} \dot{z} &= Az + Bv \\ w &= Cz + Dv \end{aligned} \tag{14}$$

where the state space matrices A, B, C, and D are Jacobian matrices evaluated at the equilibrium point:

$$\begin{aligned} A &= \tfrac{\partial f(x,u)}{\partial x}|_{(x_e,u_e)} \quad B = \tfrac{\partial f(x,u)}{\partial u}|_{(x_e,u_e)} \\ C &= \tfrac{\partial h(x,u)}{\partial x}|_{(x_e,u_e)} \quad D = \tfrac{\partial h(x,u)}{\partial u}|_{(x_e,u_e)} \end{aligned} \tag{15}$$

Remark: For the function

$$g(x) = \begin{bmatrix} g_1(x_1, x_2, ..., x_n) \\ g_2(x_1, x_2, ..., x_n) \\ \vdots \\ g_m(x_1, x_2, ..., x_n) \end{bmatrix} \tag{16}$$

in the variable $x = [x_1, x_2, \ldots, x_n]^T$, the Jacobian matrix $\frac{\partial g(x)}{\partial x}$ is given by

$$\frac{\partial g(x)}{\partial x} = \begin{bmatrix} \frac{\partial g_1(x)}{\partial x_1} & \frac{\partial g_1(x)}{\partial x_2} & \cdots & \frac{\partial g_1(x)}{\partial x_n} \\ \frac{\partial g_2(x)}{\partial x_1} & \frac{\partial g_2(x)}{\partial x_2} & \cdots & \frac{\partial g_2(x)}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial g_n(x)}{\partial x_1} & \frac{\partial g_n(x)}{\partial x_2} & \cdots & \frac{\partial g_n(x)}{\partial x_n} \end{bmatrix} \tag{17}$$

B Using the *student file*

For testing, students have to prepare a basic single column text file which will be read by the software each sampling period (4 milliseconds). The detailed requirements are:

- "`.txt`" extension;
- No header;
- Single column file;
- "`,`" as decimal separator (the DAC has 16 bits of resolution);
- Nothing at the end of the line;
- In case of an inappropriate character, the program will sent a 0;
- The value of the line must be bounded between -10 and 10 (coerced otherwise).

Students have to know that this value corresponds to:

- In the *Open Loop* tab: an analog voltage (the software automatically maintains the value during the sampling period, as a zero order hold), which is converted by the driver of the motor in a current through the DC motor. It comes from electrical theory that there is a linear relation between the torque and the rotor's current in a DC motor. So in this case: the torque $\tau = k \times U$ (with k in $[Nm\ V^{-}1]$).
- In the *Control Alpha2* or *Swing Up* tabs: the setpoint of α_1 in radians. Remark, during a test, if the angle α_1 is greater or equal to 2π, a software safety reset the command voltage of the motor driver (that resets the motor torque to 0 Nm).

On Open loop part of the software interface, the data of this file is read and showed on the `Us [V]` indicator (the `U [V]` indicator is the value of `Us` applied on the input (bounded)).

C Coding the controller in LabVIEW

In the *LabVIEW project explorer*, open the `mathNodeControlA2.vi` or `mathNodeSwingUp.vi` (they have exactly the same structure), view the diagram of the *Virtual Instrument* (the name of a program developped with LabVIEW). The students can directly program their control law in the *mathscript node* which enable to execute a `.m` file in LabVIEW (see Fig. 10).

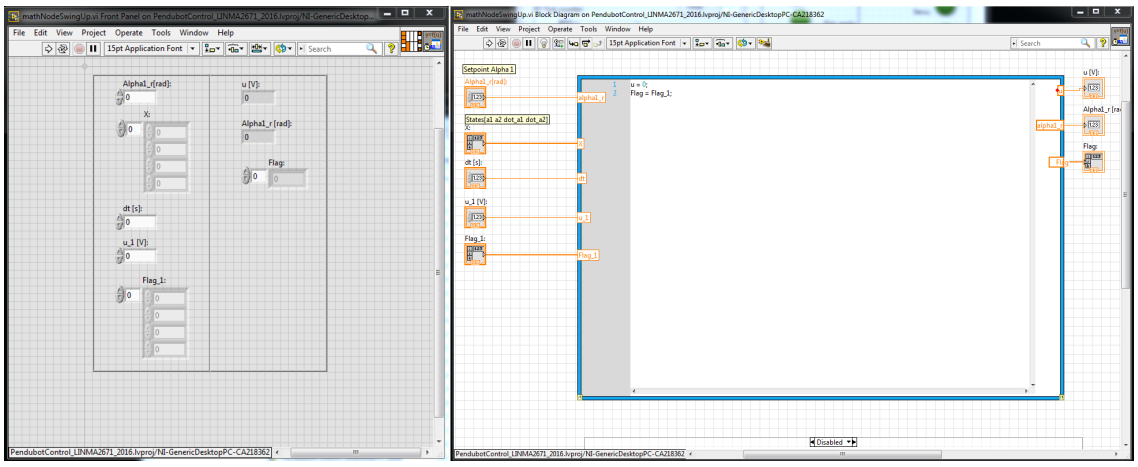

Figure 10: Programming the controller in the Mathscript Node.

For more details about the *MathScript Node* and, in this case, the *MathScript RT Module engine*, refer to the *MathScript* description.