BACHELOR THESIS:

Design and Implementation of a Furuta Pendulum Device for Benchmarking Non-Linear Control Methods

A Thesis submitted by Samuel Barrios Valero for the degree of
Electronic and Automatic Engineering in the
University of Zaragoza

Supervised by:
Péter Galambos
Francisco Pérez Cebolla

Acknowledgement

Foremost, I would like to express my sincere gratitude to my advisor Dr. Péter Galambos for the continuous support in the preparation of thesis, for his patience, motivation and enthusiasm. He has given me the opportunity to work in such a great place as it the FabLab of the Antal Bejczy Center for Intelligent Robotics of the Óbuda University. I could not have imagined having a better advisor and mentor for my Bachelor study or working in a more modern and equipped laboratory.

Besides my advisor, I would like to thank my workmate László Szücs for his design of the Furuta pendulum. He has been a continuous support and a friend. I have nothing but good words about him and his great help throughout the project development.

I am also grateful to Tivadar Garamvölgyi for his invaluable help in the design and manufacturing of the Furuta pendulum. This project would not have been possible without his collaboration and work.

Last but not least I would like to thank to the other people of the FabLab for allowing me to realize my thesis in such a great place. It has been a real pleasure working with you.

Abstract

Furuta pendulum as an academic benchmark example for evaluating non-linear control algorithms. The main aim of this dissertation is to study this physical system, showing its dynamic model and several strategies for its control. An assortment of swing-up and upright control approaches is reported with its design and simulations.

Besides, this document describes the project development which is being done at the FabLab of the Óbuda University, whose objective is to design and manufacture a demonstration device that is capable to test and display various control strategies. Requirements and specifications of the design, used tools and future work are described.

The dissertation is structured in eight different chapters: (1) History of the Furuta pendulum, describing the origin of this system; (2) State-of-the-art in non-linear control, giving a background for the different control strategies; (3) Dynamic model of the Furuta pendulum; (4) Swing-up by energy control, based on the work of Åström and Furuta; (5) Stabilizing local control, via full state feedback; (6) Hybrid control, which sums up the previous approaches; (7) Development project, which describes the work realized in the FabLab and (8) Conclusion, discussing the knowledge extracted from the development of this thesis.

Contents

List of Figures

1 History

Inverted pendulums are a family of devices that constitute a very comprehensive and interesting testing bench for non-linear control engineering. The most studied member of this family is called inverted control on a vehicle, which is commonly referred as cart. It consists of a pendulum or rod freely rotating on one end by a joint located on a cart that moves on a horizontal straight guide under the action of a force F, which is the control input with which it is intended to act on the position of the rod. Initially, in the 60s of the last century, this system could be seen in the control laboratories of the most prestigious universities. The demonstration consisted of manually set the pendulum in the vertical position, release it and autonomously feeding back the position the pendulum continued in the inverted position by applying the proper action to the cart. The issue of control is local and its interest lies in that it was stabilize an unstable position in open loop which, as we know, is a very remarkable control problem. This issue because of its local character can be solved with linear methods, and this has been done since the 60s. It is important to note that in linear systems closed loop stabilization of unstable points in open loop offers no particular problems. They appear when the system is non-linear. The drawback with this version of the pendulum, when global problems are raised, is that the cart path is limited, so if one of the ends of the horizontal support is reached the system stops working. To avoid this limitation Katsuhisa Furuta, from the Tokyo Institute of Technology, proposed in the 70s the rotary inverted pendulum known since then as Furuta pendulum [1].

It consists of a driven arm which rotates in the horizontal plane and a pendulum attached to that arm which is free to rotate in the vertical plane. The pendulum is an under-actuated 2 degrees of freedom system, extremely non-linear due to the gravitational forces and the coupling arising from the Coriolis and centripetal forces. As [2] reports, the pendulum shows two different and interesting control problems: The first one is to swing the pendulum up from the rest state (down) to the upright position. The second one comes once the pendulum is close to the desired upright position. At low speed, a stabilization or balancing strategy is needed there. Other control problems which are also quite interesting are the stabilization of autonomous oscillations or the control through bifurcation analysis.

2 State-of-the-art in Non-linear Control

2.1 Linear Control (PI or PID)

The PID controller is by far the most dominating form of feedback in use today. More than 90% of all control loops are PID. In fact, most loops are PI because derivative action is not used very often. A strength of the PID controller is that it also deals with the important practical issues such as actuator saturation or integrator windup. However, the PID controller being linear is not suited for strongly non-linear systems, such as an inverted pendulum. Nevertheless, as [3] introduces, in order to improve the performance of linear PID controllers, many approaches haven been developed to enhance the adaptability and robustness by adopting the self-tuning method, general predictive control, fuzzy logic and neural networks strategy. Among these approaches, non-linear PID (N-PID) control is one effective and simple method for industrial application. It has application in nonlinear systems, where N-PID control is used to accommodate the non-linearity and achieve consistent response across a range of conditions.

2.2 Sliding Mode Control

As mentioned in [4], the sliding mode approach is an efficient tool to design robust controllers for complete high-order non-linear dynamic systems. The research in this area began 40 years ago in the Soviet Union. The major advantage of sliding mode is low sensitivity to plant parameter variations and disturbances which eliminates the necessity of exact modelling. Sliding mode control enables the decoupling of the overall system into independent partial components of lower dimension and, as a result, reduces the complexity of feedback design. Sliding mode control implies that control actions are discontinuous state functions which may easily be implemented by conventional power converters with on-off as the only admissible operation mode.

2.3 Feedback Linearization

According to [5], feedback linearisation is perhaps the most important non-linear control design strategy developed during the last few decades. The main objective of the approach is to algebraically transform non-linear system dynamics into linear by using state feedback and a non-linear coordinate transformation based on a differential geometric analysis of the system. By eliminating non-linearities in the system, conventional control techniques can be applied. The linearisation is carried out by model-based state transformation and feedback rather than by linear approximations of the dynamics, as used in Jacobian linearisation, where the resulting linear model is only locally valid. Differential geometry has proved to be a successful mean of analysing and designing non-linear control systems, equivalently to that of linear algebra and Laplace transform in relation to linear systems. Feedback linearisation is a strong research field with rigorous mathematical formulations.

2.4 Parallel Distributed Control (PDC)

There has been a rapidly growing interest in fuzzy controllers in recent years. Fuzzy logic has many varieties to be implemented for control purposes. As [6] reports, one of them is parallel distributed compensation (PDC). The PDC offers a procedure to design a fuzzy controller from a given TakagiSugeno (TS) fuzzy model. Most of the non-linear systems can be transformed into the TS fuzzy model. The main idea of the PDC technique is to partition the dynamics of a non-linear system into a number of linear subsystems, design a number of local controllers for each linear subsystem, and finally generate the overall controller (compensator) by the fuzzy blending of such local controllers.

2.5 Tensor Product Model Transformation

As mentioned in [7], the Tensor Product (TP) model transformation based control approach is applied to stabilize a system with structural non-linearities. It can decompose numerically any given non-linear Quasi Linear Parameter Varying (qLPV) model into a composition of several Linear Time Invariant (LTI) systems. It is a numerical transformation that can be executed quasi automatically without deep analytical manipulations and stability analysis of delayed differential equations. This method establishes a gateway between the various delayed system representations and the tensor product (TP) type convex polytopic models which allows the direct use of LMI-based multi-objective synthesis techniques [8].

3 Dynamic model of the Furuta Pendulum

3.1 Fundamentals

3.1.1 Definitions

Following the modelling realized in [9] , let's use a right hand coordinate system to define the inputs, states, and the Cartesian coordinate systems 1 and 2. The inertia tensors are diagonal form due to the coordinate axes of Arm 1 and Arm 2 are the principal axes

$$J_1 = \begin{bmatrix} J_{1xx} & 0 & 0 \\ 0 & J_{1yy} & 0 \\ 0 & 0 & J_{1zz} \end{bmatrix} \tag{1}$$

$$J_2 = \begin{bmatrix} J_{2xx} & 0 & 0 \\ 0 & J_{2yy} & 0 \\ 0 & 0 & J_{2zz} \end{bmatrix} \tag{2}$$

The angular rotation of Arm 1, θ_1, is measured in the horizontal plane where a counterclockwise direction (when viewed from above) is positive. The angular rotation of Arm 2, θ_2, is measured in the vertical plane where a counterclockwise direction (when viewed from the front) is positive, when Arm 2 is hanging down in the stable equilibrium position θ_2 The torque the servomotor applies to Arm 1, τ_1, is positive in a counterclockwise direction (when viewed from above). A disturbance torque, τ_2, is experienced by Arm 2, where a counterclockwise direction (when viewed from the front) is positive. L_1 and L_2 are the length of the horizontal and vertical respectively. Similarly, l_1 and l_2 are the position of the mass centre of each arm relative to its beginning. Finally, m_1 and m_2 are the masses of each arm.

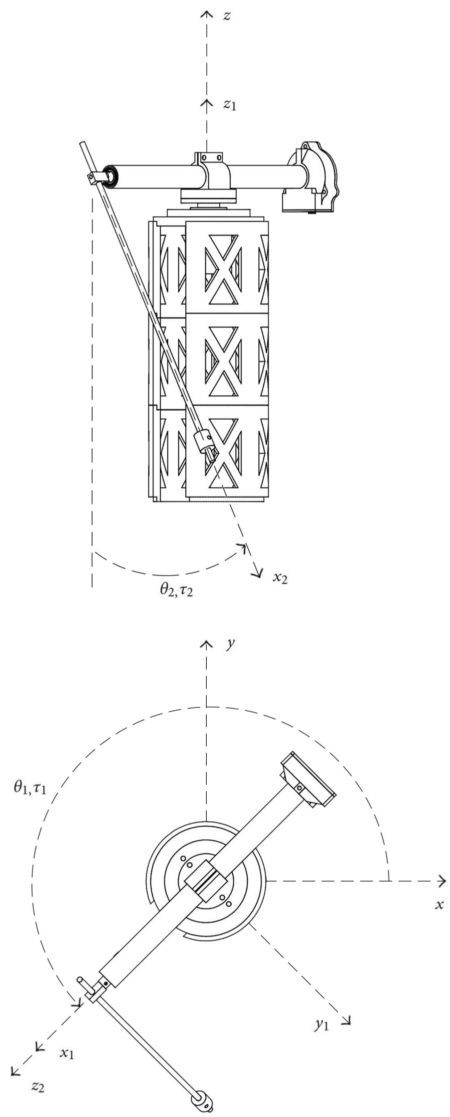

Figure 1: Schematic of the Furuta pendulum

3.1.2 Assumptions

The following assumptions where used for the model:

- The motor shaft and the first arm are assumed to be rigidly coupled and infinitely stiff.

- Arm 2 is assumed to be infinitely stiff.
- The coordinate axes of Arm 1 and Arm 2 are the principal axes. Therefore the inertia tensors are diagonal.
- Backlash effects are not being modelled.
- Motor and pendulum have only viscous friction. Other forms, such as Coulomb damping, have been disregarded.

3.2 Lagrangian formulation

3.2.1 Rotation matrices

Let's define first two rotation matrices which are used in the Lagrange formulation. The rotation matrix from the base to Arm 1 is a Z axis basic rotation:

$$R_1 = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The rotation matrix from Arm 1 to Arm 2 is composed by two different matrices: a Y axis basic rotation to match the frame 1 and the frame 2, followed by a Z axis basic rotation for θ_2, given by

$$R_2 = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \sin\theta_2 & -\cos\theta_2 \\ 0 & \cos\theta_2 & \sin\theta_2 \\ 1 & 0 & 0 \end{bmatrix} \tag{4}$$

3.2.2 Velocities

The angular velocity of Arm 1 is

$$\omega = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \tag{5}$$

Let's consider the base frame at rest, such that the joint between the frame and Arm 1 is at rest as well and the velocity is given by

$$v_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

The total linear velocity of the centre mass of Arm 1 is

$$v_{1c} = v_1 + \omega_1 \times \begin{bmatrix} l_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \dot{\theta}_1 l_1 \\ 0 \end{bmatrix} \tag{7}$$

The angular velocity for Arm 2 is given by

$$\omega_2 = R_2\omega_1 + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} -\cos(\theta_2)\dot{\theta}_1 \\ \sin(\theta_2)\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{8}$$

The velocity of the joint between Arm 1 and Arm 2 in reference frame 1 is

$$\omega_1 \times \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} \tag{9}$$

which in reference frame 2 gives

$$v_2 = R_2(\omega_1 \times \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix}) = \begin{bmatrix} \dot{\theta}_1 L_1 \sin\theta_2 \\ \dot{\theta}_1 L_1 \cos\theta_2 \\ 0 \end{bmatrix} \tag{10}$$

The total linear velocity of the centre of mass of Arm 2 is given by

$$v_{2c} = v_2 + \omega_2 \times \begin{bmatrix} l_2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 L_1 \sin\theta_2 \\ \dot{\theta}_1 L_1 \cos\theta_2 + \dot{\theta}_2 l_2 \\ -\dot{\theta}_1 l_2 \sin\theta_2 \end{bmatrix} \tag{11}$$

3.2.3 Energies

The potential energy of Arm 1 is

$$E_{p1} = 0 \tag{12}$$

and the kinetic energy is

$$E_{k1} = \frac{1}{2}(v_{1c}^T m_1 v_{1c} + \omega_1^T J_1 \omega_1) = \frac{1}{2}\dot{\theta}_1^2(m_1 l_1^2 + J_{1zz}) \tag{13}$$

The potential energy of Arm 2 is

$$E_{p2} = g m_2 l_2 (1 - \cos\theta_2) \tag{14}$$

and the kinetic energy is

$$\begin{aligned} E_{k2} &= \frac{1}{2}(v_{2c}^T m_2 v_{2c} + \omega_2^T J_2 \omega_2) \\ &= \frac{1}{2}\dot{\theta}_1^2(m_2 L_1^2 + (m_2 l_2^2 + J_{2yy})\sin^2\theta_2 + J_{2xx}\cos^2\theta_2) \\ &\quad + \frac{1}{2}\dot{\theta}_2^2(J_{2zz} + m_2 l_2^2) + m_2 L_1 l_2 \cos(\theta_2)\dot{\theta}_1\dot{\theta}_2 \end{aligned} \tag{15}$$

The total potential and kinetic energies are given, respectively, by

$$E_p = E_{p1} + E_{p2} \tag{16}$$

$$E_k = E_{k1} + E_{k2} \tag{17}$$

3.2.4 Lagrangian

The Lagrangian is the difference between kinetic and potential energies

$$L = E_k - E_p \tag{18}$$

From this, we obtain the Euler-Lagrange equation

$$\frac{d}{dt}(\frac{\partial L}{\partial q_i}) + b_i\dot{q}_i - \frac{\partial L}{\partial q_i} = Q_i \tag{19}$$

where $q_i = \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}$ is a generalised coordinate, $b_i = \begin{bmatrix}b_1\\ b_2\end{bmatrix}$ is a generalised viscous damping coefficient and $Q_i = \begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix}$ is a generalised torque.

Evaluating the terms of the Euler-Lagrange equation for $q_i = \theta_1$ and θ_2 gives

$$\begin{gathered}\frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}_1}) = \ddot{\theta}_1(J_{1zz} + m_1 l_1^2 + m_2 L_1^2\\ +(m_2 l_2^2 + J_{2yy})\sin^2\theta_2 + J_{2xx}\cos^2\theta_2)\\ +m_2 L_1 l_2 \cos(\theta_2)\ddot{\theta}_2 - m_2 L_1 l_2 \sin(\theta_2)\dot{\theta}_2^2\\ +\dot{\theta}_1\dot{\theta}_2\sin(2\theta_2)(m_2 l_2^2 + J_{2yy} - J_{2xx})\end{gathered} \tag{20}$$

$$\begin{gathered}\frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}_2}) = \ddot{\theta}_1 m_2 L_1 l_2 \cos\theta_2\\ +\ddot{\theta}_2(J_{2zz} + m_2 l_2^2) - \dot{\theta}_1\dot{\theta}_2 m_2 L_1 l_2 \sin\theta_2\end{gathered} \tag{21}$$

$$-\frac{\partial L}{\partial \theta_1} = 0 \tag{22}$$

$$\begin{gathered}-\frac{\partial L}{\partial \theta_2} = -\frac{1}{2}\dot{\theta}_1^2\sin(2\theta_2)(m_2 l_2^2 + J_{2yy} - J_{2xx})\\ +\dot{\theta}_1\dot{\theta}_2 m_2 L_1 l_2 \sin\theta_2 + g m_2 l_2 \sin\theta_2\end{gathered} \tag{23}$$

3.2.5 Equations of motion

Substituting the previous terms into the Euler-Lagrange equation, the following simultaneous differential equations are obtained:

$$\begin{gathered}\ddot{\theta}_1(J_{1zz} + m_1 l_1^2 + m_2 L_1^2\\ +(m_2 l_2^2 + J_{2yy})\sin^2\theta_2 + J_{2xx}\cos^2\theta_2)\\ +m_2 L_1 l_2 \cos(\theta_2)\ddot{\theta}_2 - m_2 L_1 l_2 \sin(\theta_2)\dot{\theta}_2^2\\ +\dot{\theta}_1\dot{\theta}_2\sin(2\theta_2)(m_2 l_2^2 + J_{2yy} - J_{2xx}) + b_1\dot{\theta}_1 = \tau_1\end{gathered} \tag{24}$$

$$\begin{gathered}\ddot{\theta}_1 m_2 L_1 l_2 \cos\theta_2 + \ddot{\theta}_2(J_{2zz} + m_2 l_2^2) - \frac{1}{2}\dot{\theta}_1^2\sin(2\theta_2)\\ \times(m_2 l_2^2 + J_{2yy} - J_{2xx}) + g m_2 l_2 \sin\theta_2 + b_2\dot{\theta}_2 = \tau_2\end{gathered} \tag{25}$$

3.3 Simplifications

Due to the Furuta pendulum has long arms, the moment of inertia along the axis of the arms can be obviated. Besides, the arms have rotational symmetry, such that the moments of inertia in two of the principal axes are equal. Based on previous information, the inertia tensors may approximated as follows:

$$J_1 = \begin{bmatrix} J_{1xx} & 0 & 0 \\ 0 & J_{1yy} & 0 \\ 0 & 0 & J_{1zz} \end{bmatrix} \approx \begin{bmatrix} 0 & 0 & 0 \\ 0 & J_1 & 0 \\ 0 & 0 & J_1 \end{bmatrix} \tag{26}$$

$$J_2 = \begin{bmatrix} J_{2xx} & 0 & 0 \\ 0 & J_{2yy} & 0 \\ 0 & 0 & J_{2zz} \end{bmatrix} \approx \begin{bmatrix} 0 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_2 \end{bmatrix} \tag{27}$$

We can get more simplifications making the following substitutions: The total moment of inertia of Arm 1 about the pivot joint (using the parallel axis theorem) is $\hat{J}_1 = J_1 + m_1 l_1^2$. In the same way, the total moment of inertia of Arm 2 about its pivot point is $\hat{J}_2 = J_2 + m_2 l_2^2$. Finally, the total moment of inertia the motor rotor experiences when the pendulum is in its equilibrium position (hanging vertically down) is given by $\hat{J}_0 = \hat{J}_1 + m_2 L_1^2$. Besides, due to the underactuated nature of the system, the second element of the generalised torque (τ_2) can be taken as 0. Finally, τ_1 can be expressed as Ku, where K is the equivalent gain from the motor to the control and u the input the control applies.
Substituting the previous definitions into the above equations of motion gives a more compact form

$$\begin{gathered} \ddot{\theta}_1(\hat{J}_0 + \hat{J}_2 \sin^2\theta_2) + \ddot{\theta}_2 m_2 L_1 l_2 \cos\theta_2 - m_2 L_1 l_2 \\ \times \sin\theta_2 \dot{\theta}_2^2 + \dot{\theta}_1 \dot{\theta}_2 \hat{J}_2 \sin(2\theta_2) + b_1 \dot{\theta}_1 = Ku \end{gathered} \tag{28}$$

$$\begin{gathered} \ddot{\theta}_1 m_2 L_1 l_2 \cos\theta_2 + \ddot{\theta}_2 \hat{J}_2 - \frac{1}{2}\dot{\theta}_1^2 \hat{J}_2 \sin(2\theta_2) \\ + b_2 \dot{\theta}_2 + g m_2 l_2 \sin\theta_2 = 0 \end{gathered} \tag{29}$$

As [10] proposes, these equations can be expressed in a more compact form using the following parameters

$$\begin{aligned} \alpha &= \frac{m_2 L_1 l_2}{\hat{J}_2} \qquad \beta = \frac{\hat{J}_0}{\hat{J}_2} \qquad \gamma = \frac{K}{m_2 g l_2} \\ \omega_0^2 &= \frac{m_2 g l_2}{\hat{J}_2} \qquad c_{p1} = \frac{b_1}{\hat{J}_2} \qquad c_{p2} = \frac{b_2}{\hat{J}_2} \end{aligned} \tag{30}$$

And substituting into (28) and (29) the following equations are obtained

$$\begin{gathered} \ddot{\theta}_1(\beta + \sin^2\theta_2) + \ddot{\theta}_2 \alpha \cos\theta_2 - \dot{\theta}_2^2 \alpha \sin\theta_2 \\ + 2\dot{\theta}_1 \dot{\theta}_2 \sin\theta_2 \cos\theta_2 + \dot{\theta}_1 c_{p1} = \gamma \omega_0^2 u \end{gathered} \tag{31}$$

$$\ddot{\theta}_1\alpha\cos\theta_2 + \ddot{\theta}_2 - \dot{\theta}_1^2\sin\theta_2\cos\theta_2 + \omega_0^2\sin\theta_2 + \dot{\theta}_2 c_{p2} = 0 \tag{32}$$

which can be rewritten in the standard matrix form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = Mu \tag{33}$$

where $\ddot{\theta}_1$ and $\ddot{\theta}_2$ can be easily solved. Thus, we obtain

$$\begin{aligned}
&\begin{bmatrix} \beta + \sin^2\theta_2 & \alpha\cos\theta_2 \\ \alpha\cos\theta_2 & 1 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} \\
&+ \begin{bmatrix} \dot{\theta}_2\sin\theta_2\cos\theta_2 + c_{p1} & -\dot{\theta}_2\alpha\sin\theta_2 + \dot{\theta}_1\sin\theta_2\cos\theta_2 \\ -\dot{\theta}_1\sin\theta_2\cos\theta_2 & c_{p2} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \\
&+ \begin{bmatrix} 0 \\ \omega_0^2\sin\theta_2 \end{bmatrix} = \begin{bmatrix} \gamma\omega_0^2 u \\ 0 \end{bmatrix}
\end{aligned} \tag{34}$$

From which we can solve both accelerations by applying

$$\ddot{q} = D^{-1}(q) \times (Mu - C(q,\dot{q})\dot{q} - g(q)) \tag{35}$$

Thus, the two accelerations are obtained in their explicit form

$$\begin{aligned}
\ddot{\theta}_1 = \frac{1}{\Delta}(&\gamma\omega_0^2 u - 2\dot{\theta}_1\dot{\theta}_2\sin\theta_2\cos\theta_2 - \dot{\theta}_1 c_{p1} + \dot{\theta}_2^2\alpha\sin\theta_2 \\
&-\dot{\theta}_1^2\alpha\sin\theta_2\cos^2\theta_2 + \dot{\theta}_2 c_{p2}\alpha\cos\theta_2 + \alpha\omega_0^2\sin\theta_2\cos\theta_2)
\end{aligned} \tag{36}$$

$$\begin{aligned}
\ddot{\theta}_2 = \frac{1}{\Delta}(&-\alpha\gamma\omega_0^2 u\cos\theta_2 + 2\dot{\theta}_1\dot{\theta}_2\alpha\sin\theta_2\cos^2\theta_2 + \dot{\theta}_1 c_{p1}\alpha\cos\theta_2 \\
&-\dot{\theta}_2^2\alpha^2\sin\theta_2\cos\theta_2 + (\beta + \sin^2\theta_2)\dot{\theta}_1^2\sin\theta_2\cos\theta_2 \\
&-(\beta + \sin^2\theta_2)\dot{\theta}_2 c_{p2} - \beta\omega_0^2\sin\theta_2 - \omega_0^2\sin^3\theta_2)
\end{aligned} \tag{37}$$

where $\Delta = \beta + \sin^2\theta_2 - \alpha^2\cos^2\theta_2$ is the determinant of the matrix $D(q)$. Notice that (36) and (37) are a fourth-order non-linear system strongly coupled and the coordinate $q_1 = \theta_1$ is cyclic. Remind that, a coordinate is said to be cyclic when it does not appear in the Lagrangian (19).

3.4 Linearisation via Taylor Series

A linearised model of the system can be obtained using Taylor Series and neglecting the quadratic and higher order terms. Thus, a function $f(x_1, \ldots, x_n)$ can be approximated around a steady-state operating point $x_s = \{x_{1s}, \ldots, x_{ns}\}$ as follows

$$f(x_1, \ldots, x_n) \approx f(x_s) + \left.\frac{\partial f}{\partial x_1}\right|_{xs}(x_1 - x_{1s}) + \cdots + \left.\frac{\partial f}{\partial x_n}\right|_{xs}(x_n - x_{ns}) \tag{38}$$

which can be also expressed as

$$\bar{f}(\bar{x}_1, \ldots, \bar{x}_n) = \left.\frac{\partial f}{\partial x_1}\right|_{xs}\bar{x}_1 + \cdots + \left.\frac{\partial f}{\partial x_n}\right|_{xs}\bar{x}_n \tag{39}$$

where the bar symbol indicates that the variables are linear deviation variables around the steady-state operating point.
Thus, the upright position of the pendulum defines a steady-state operating point given by

$$\begin{aligned} \theta_{1s} &= 0 \quad \dot{\theta}_{1s} = 0 \\ \theta_{2s} &= \pi \quad \dot{\theta}_{2s} = 0 \end{aligned} \tag{40}$$

and applying (39) to (36) and (37), the following linear space state equations are obtained:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} 0 & 0 & \Delta & 0 \\ 0 & 0 & 0 & \Delta \\ 0 & \alpha\omega_0^2 & -c_{p1} & -c_{p2}\alpha \\ 0 & \beta\omega_0^2 & -c_{p1}\alpha & -c_{p2}\beta \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} 0 \\ 0 \\ \gamma\omega_0^2 \\ \alpha\gamma\omega_0^2 \end{bmatrix} [u] \tag{41}$$

where $\Delta = \beta - \alpha^2$. Note that for avoiding clutter of symbols between bars and dots, the deviation variable notation has been omitted even though the above equations express deviation around the steady-state operating point.
We can observe that the matrix A has a column of zeros. This with the fact that the variable θ_1 is cyclic indicates that the order of the system can be reduced. Thus, we obtain

$$\begin{bmatrix} \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} 0 & 0 & \Delta \\ \alpha\omega_0^2 & -c_{p1} & -c_{p2}\alpha \\ \beta\omega_0^2 & -c_{p1}\alpha & -c_{p2}\beta \end{bmatrix} \begin{bmatrix} \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} 0 \\ \gamma\omega_0^2 \\ \alpha\gamma\omega_0^2 \end{bmatrix} [u] \tag{42}$$

3.5 Error function model

Due to the control approaches applied to the Furuta pendulum aim to keep it in the upright position, it is interesting to obtain an error model which can measure the difference between the actual angle and the upright position one. In other words, the objective is to match $\theta_2 = 0$ to the upright position. A simple way to get it is to substitute θ_2 in (36) and (37) for $\theta_2 + \pi$. Thus, an angle shift is introduced so that the downward position is given by $\theta 2 = \pi$ and the upright position is given by $\theta_2 = 0$. Thereby, with the above mentioned and applying the trigonometric identities $\cos(\theta_2 + \pi) = -\cos(\theta_2)$ and $\sin(\theta_2 + \pi) = -\sin(\theta_2)$, (36) and (37) can be transformed into the following error functions:

$$\begin{aligned} \ddot{\theta}_1 = &\frac{1}{\Delta}(\gamma\omega_0^2 u - 2\dot{\theta}_1\dot{\theta}_2 \sin\theta_2 \cos\theta_2 - \dot{\theta}_1 c_{p1} - \dot{\theta}_2^2 \alpha \sin\theta_2 \\ &+ \dot{\theta}_1^2 \alpha \sin\theta_2 \cos^2\theta_2 - \dot{\theta}_2 c_{p2} \alpha \cos\theta_2 + \alpha\omega_0^2 \sin\theta_2 \cos\theta_2) \end{aligned} \tag{43}$$

$$\begin{aligned} \ddot{\theta}_2 = &\frac{1}{\Delta}(\alpha\gamma\omega_0^2 u \cos\theta_2 - 2\dot{\theta}_1\dot{\theta}_2 \alpha \sin\theta_2 \cos^2\theta_2 - \dot{\theta}_1 c_{p1} \alpha \cos\theta_2 \\ &- \dot{\theta}_2^2 \alpha^2 \sin\theta_2 \cos\theta_2 + (\beta + \sin^2\theta_2)\dot{\theta}_1^2 \sin\theta_2 \cos\theta_2 \\ &- (\beta + \sin^2\theta_2)\dot{\theta}_2 c_{p2} + \beta\omega_0^2 \sin\theta_2 + \omega_0^2 \sin^3\theta_2) \end{aligned} \tag{44}$$

where $\Delta = \beta + \sin^2\theta_2 - \alpha^2 \cos^2\theta_2$ is the determinant of the matrix $D(q)$.

4 Swing-up by energy control

4.1 Preliminaries

The control law proposed in [11] by Åstrom and Furuta is based on controlling the energy of the pendulum regardless of the horizontal arm. The simplified model of two dimensions is given by

$$m_2 l_2^2 \ddot{\theta}_2 - m_2 g l_2 \sin\theta_2 - m_2 g l_2 u \cos\theta_2 = 0 \tag{45}$$

where u is the acceleration of the pivot of the pendulum. The model given in (45) is based on several assumptions: friction has been neglected and it has been assumed that the pendulum is a rigid body. It has also been assumed that there is no limitation on the velocity of the pivot.
The energy of the uncontrolled pendulum ($u = 0$) is

$$E = \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 g l_2 \cos\theta_2 \tag{46}$$

Being $\theta_2 = \pi$ the downward position and $\theta_2 = 0$ the upright equilibrium (3.5), in order to swing the pendulum up its energy must be increased from $-m_2 g l_2$ to $m_2 g l_2$. We set $E_0 = m_2 g l_2$. Now, to perform energy control it is necessary to understand how the energy is influenced by the acceleration of the pivot. Differentiating with respect to time we find

$$\frac{dE}{dt} = m_2 l_2^2 \ddot{\theta}_2 \dot{\theta}_2 - m_2 g l_2 \dot{\theta}_2 \sin\theta_2 = m_2 g l_2 \dot{\theta}_2 u \cos\theta_2 \tag{47}$$

where (45) has been used to obtain the last equality. (47) implies that the energy can be controlled in an easy way, due to the system behaves like an integrator with varying gain. To increase energy the acceleration of the pivot u should be positive when the quantity $\dot{\theta}_2 \cos\theta_2$ is positive, and similarly to decrease it the acceleration should be negative when the mentioned amount s negative.

4.2 Energy control

A control strategy is easily obtained by the Lyapunov method. As [12] mentions, given an autonomous dynamical system

$$\dot{x} = f(x, u) \tag{48}$$

where $x \in \mathbb{R}^n$ is the state-vector, $u \in \mathbb{R}^m$ is the control vector and we want to feedback stabilize it to $x = 0$ in some domain $D \subset \mathbb{R}^n$. A control-Lyapunov function is a function $V : D \to \mathbb{R}$ that is continuously differentiable, positive definite and such that

$$\forall x \neq 0, \exists u : \frac{dV}{dt}(x, u) = \nabla V(x) \cdot f(x, u) < 0 \tag{49}$$

Which in words it says that for each state x we can find a control u that will reduce the "energy" V. Intuitively, if in each state we can always find a way to reduce the energy, we should eventually be able to bring the energy to zero. The function $V = (E - E_0)^2/2$ is proposed in [11]. In order to fulfil the condition in (49), let's differentiate V with respect to the time

$$\frac{dV}{dt} = m_2 l_2 u \dot{\theta}_2 \cos\theta_2 (E - E_0) \tag{50}$$

where applying the proposed control law

$$u = -k(E - E_0)\dot{\theta}_2 \cos\theta_2 \tag{51}$$

where $k > 0$ is an adjustable gain, we find that

$$\frac{dV}{dt} = -m_2 l_2 k((E - E_0)\dot{\theta}_2 \cos\theta_2)^2 < 0 \tag{52}$$

Thus, the Lyapunov function decreases as long as $\dot{\theta}_2 \neq 0$ and $\cos\theta_2 \neq 0$. Since the pendulum cannot maintain a stationary position with $\theta_2 = \pm\pi/2$, strategy (51) drives the energy towards its desired value E_0.

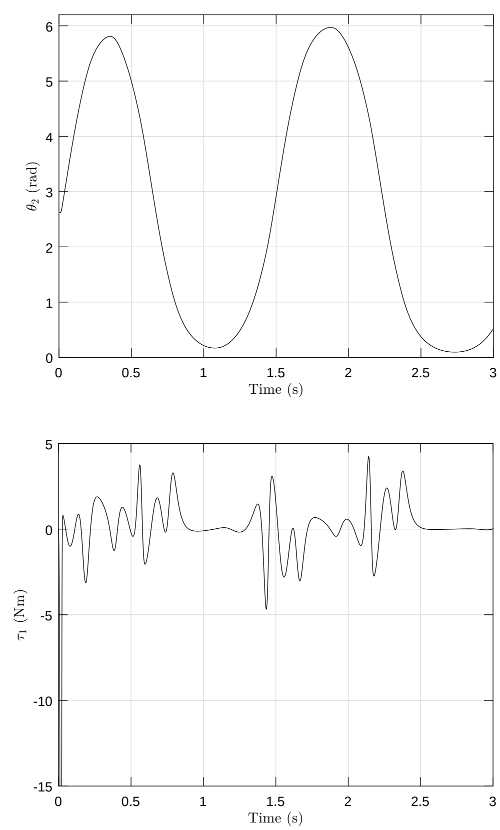

Figure 2: Input and response of the system to a Swing-up strategy using a linear control law

In the Figure 2 we can see the evolution of the output θ_2 and the input τ_1 when the swing-up strategy (51) is applied. Note that the parameters used for this and the following simulations are the ones in [9], with a higher m_2 so that the features of the control may be appreciated in a better way. Thus, we can observe a high initial action due to the high difference of energies $E - E_0$, which quickly decreases as the pendulum approaches the upright position.
To change the energy as fast as possible the magnitude of the control signal should be as large as possible. This is achieved saturating the signal to a value u_{max}, that is

$$u = -u_{max} sgn((E - E_0)\dot{\theta}_2 \cos\theta_2) \tag{53}$$

or

$$u = \begin{cases} u_{max} & \text{if } (E - E_0)\dot{\theta}_2 \cos\theta_2 < 0 \\ -u_{max} & \text{if } (E - E_0)\dot{\theta}_2 \cos\theta_2 > 0 \end{cases} \tag{54}$$

which drives the function $V = (E - E_0)^2/2$ to zero and E towards E_0.

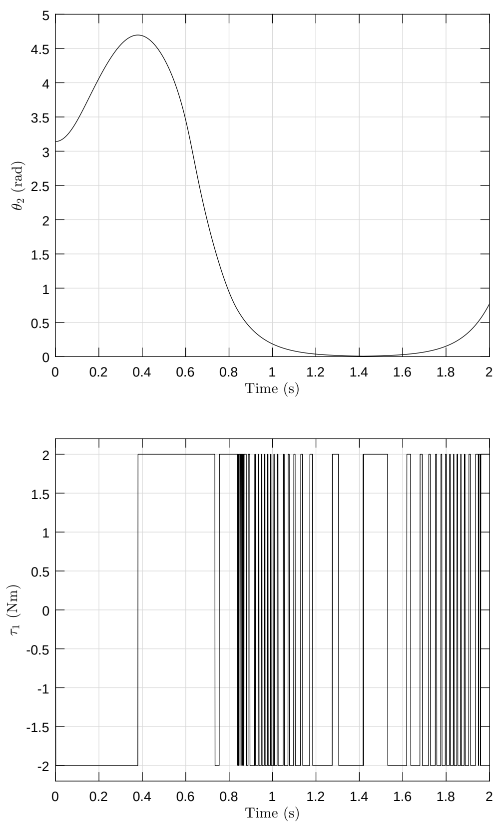

Figure 3: Input and response of the system to a Swing-up strategy using a saturated control variation

However, as it is shown in the Figure 3 the problem is that control law (53) may result into chattering. This can be avoided with the control law

$$u = -sat(k(E - E_0)sign(\dot{\theta}_2 \cos\theta_2)) \tag{55}$$

or

$$u = \begin{cases} k(E - E_0) & \text{if } \dot{\theta}_2 \cos\theta_2 < 0 \wedge k(E - E_0) < u_{max} \\ u_{max} & \text{if } \dot{\theta}_2 \cos\theta_2 < 0 \wedge k(E - E_0) \geq u_{max} \\ & \\ -k(E - E_0) & \text{if } \dot{\theta}_2 \cos\theta_2 > 0 \wedge -k(E - E_0) > -u_{max} \\ -u_{max} & \text{if } \dot{\theta}_2 \cos\theta_2 > 0 \wedge -k(E - E_0) \leq -u_{max} \end{cases} \tag{56}$$

where sat in (55) denotes a linear function which saturates at u_{max}, as can be seen in (56).

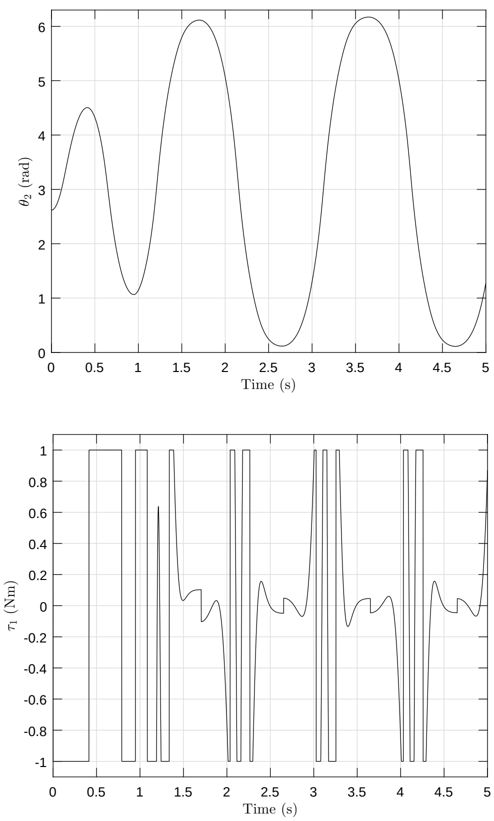

Figure 4: Input and response of the system to a Swing-up strategy using a hybrid variation

Thus, as Figure 4 shows, strategy (55) behaves like a linear controller (51) for small errors and like (53) for large errors, which supposes an intermediate solution between the two ones above cited.
Notice that the function sign is not defined when its argument is zero. If the value is defined as zero the control signal will be zero when the pendulum is at rest or when it is horizontal. If the pendulum starts at rest in the downward position strategies (51), (53) and (55) all give $u = 0$ and the pendulum will remain in the downward position. For this reasons it is convenient to set a default value (e.g. $\pm u_{max}$ or $\pm k(E - E_0)$). Thus, the system does not remain stuck in the downward position.

5 Stabilizing local control

In this section a control strategy for the stabilization of the pendulum around the upright equilibrium position is deduced. Note that there is no known strategy which reaches the equilibrium position starting from any point. In other words, this strategy has a local area of application around the desired operation point.

5.1 Full State Feedback with pole placement

As [13] reads, given a linear time-invariant (LTI) system on the form

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{57}$$

A natural control law is to use a state feedback

$$u = -Kx(t) \tag{58}$$

The closed-loop system under (58) is then

$$\dot{x} = (A - BK)x(t) \tag{59}$$

The closed-loop dynamics is completely determined by $(A - BK)$, and the stability of the closed-loop system as well as the rate of regulation of x to zero is determined by the eigenvalues of $(A - BK)$, which can be also called the poles of the closed-loop system. In particular, the system (57) is stable if and only if all eigenvalues of $(A - BK)$ lie in $Re(s) < 0$. The problem of finding a K to achieve a prescribed set of eigenvalues for $(A - BK)$ and set the dynamic behaviour of the system is called the pole assignment problem.
Before applying any strategy the controllability of the system must be checked. For that, let's define the controllability matrix

$$\zeta = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \tag{60}$$

where n is equal to the number of state variables (in other words, the number of rows of x(t)). Thus, the system will be controllable if and only if ζ is invertible, which is equivalent to

$$rank(\zeta) = n \tag{61}$$

If the system is controllable the next step is to select the desired dynamical behaviour of the system or in other words the closed-loop poles. For that, let's define the closed-loop characteristic polynomial as

$$D(s) = (s - p_1)(s - p_2) \dots (s - p_n) \tag{62}$$

where $\{p_1, p_2, \dots, p_n\}$ are the closed-loop poles and n the number of state variables.
Finally, the last step is to apply the Ackerman's formula, which is given by

$$K = \upsilon^T \zeta^{-1} D(A) \tag{63}$$

where $\upsilon^T = [0, 0, \dots, 0, 1]$ is a vector with dimension n, ζ is the controllability matrix and $D(A)$ is the closed-loop characteristic polynomial evaluated in A.
Let's now apply the strategy above related to our system. The chosen poles are $s_1 = -10 - 10j$,$s_2 = -10 + 10j$ and $s_3 = -1$, so $D(s) = s^3 + 21s^2 + 220s + 200$. Thus, simulating with the same parameters used in the Swing-up section we obtain

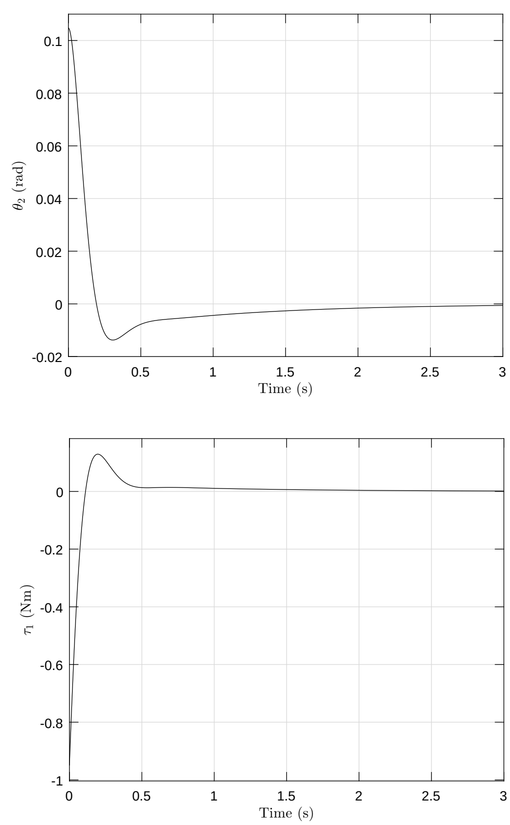

Figure 5: Upright control using Full State Feedback with pole placement

Where an initial value of $\theta_2 = 6° \approx 0.11\,rad$ has been set. We can observe how the action τ_1 decreases as θ_2 approaches to the upright position. Notice that the system presents an small overshoot and although it takes several seconds to achieve a zero error, a half degree difference is acquired in less than 0.5 seconds.

6 Hybrid control

A global solution to lead the pendulum from its stable position ($\theta_2 = \pi$) to the upright position ($\theta_2 = 0$) is carried out by a hybrid control. It gathers the two different approaches described above: the swing-up by energy control and the stabilizing local control. Thus, establishing a threshold which determines which strategy is applied in each moment, we are able to lift the pendulum and keep it upright.

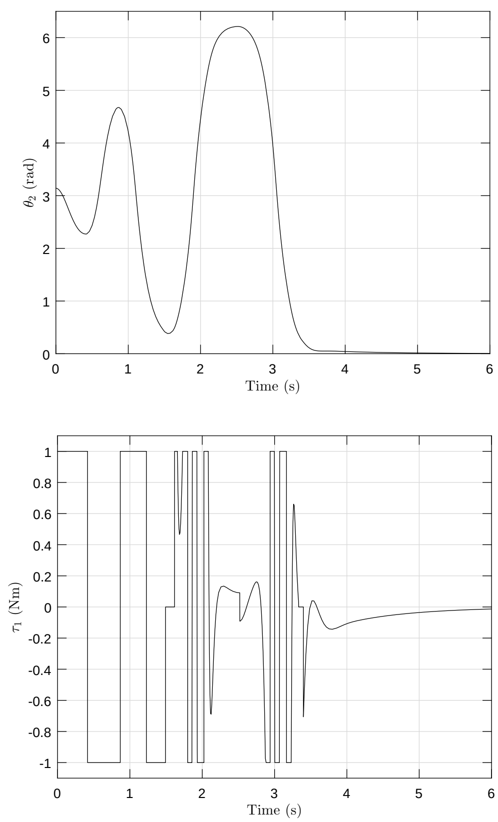

Figure 6: Input and response of the system to a hybrid control

For example, for the Figure 6 simulation two different thresholds have been chosen: while $\theta_2 > 25°$ the swing-up strategy will be applied. Once the pendulum goes through the 25° threshold it will move with its remaining energy.

Finally, when the error is 15° or lower, the stabilizing local control keeps the pendulum in a vertical position. For the swing-up strategy the third variation has been chosen due to it provides a low lifting time without chattering.

7 Development Project

7.1 Goal and motivation

The Furuta pendulum is a well known academic benchmark example for evaluating non-linear control algorithms. The purpose of this work is the physical realization of such a device focusing not only the control theoretical aspects but also for practical problems of design and implementation. The goal is to design and manufacture a demonstration device that is capable to test and display various control strategies.

7.2 Requirements

Since the physical system is going to be a teaching presentation object it must be mobile. The model needs to be light and easy to transport. Besides, it has to be interesting for several demonstrations and tests, which means that its characteristics must be variable. This results in that the main parameters must be able to be changed, for example the length of the arms or the mass of the weight. Finally, each part should fit into each other in a way that there is not backslash effect or it is as small as possible for the correct demonstration. Of course an exception for this is the bearing in the direction that they are able to rotate.

7.3 Specifications

The design realized by László Szücs consists basically of a cylindrical body, an horizontal holder and the two arms. The body is composed of three detachable cylinders. These sub-parts have been holed so that their weight is minimized without affecting its stiffness and strength. They fit into each other so the big cylindrical body can be easily transportable, and have an opening so that the motor and wiring is accessible.

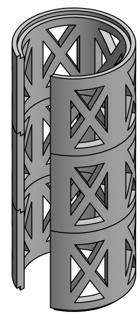

Figure 7: Cylindrical body of the Furuta pendulum

The horizontal holder consists of a bell part, a circular linkage, a support for the arm and a coupler. The binding of this part and the cylindrical structure is carried by a holed circular cap which fits into the cylinder. The function of the bell part is to house the coupling between the motor shaft and the rest of the structure, which is carried by a plastic coupler and a brass axis fixed to the circular linkage. To ensure proper rotation of the shaft, the coupling between the bell and the circular linkage is made with a bearing. Besides, the support holds an aluminium pipe inside which is the horizontal arm, which can rotate by two bearings arranged at both sides of the pipe. Finally, in one end of the pipe there is a plastic holder for the encoder which measures the inclination of the pendulum. It is covered by an acrylic glass piece so that the rotation of the encoder wheel can be seen from the outside. Moreover, at the other end of the pipe there is the joint of the two arms, carried by two small brass pieces (one cylindrical and one cubic) which allow a stable connection between the arms and also ensures an opportunity to take the device apart for transportation.

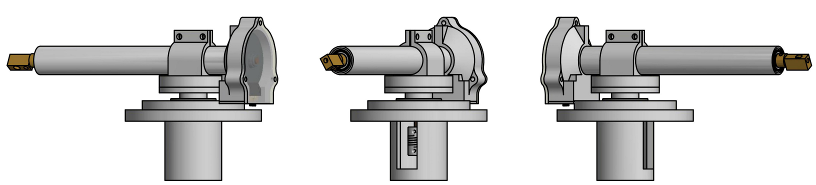

Figure 8: Horizontal holder of the Furuta pendulum

Both arms are made of carbon fibre. As it is said above, the horizontal one rotates inside of the pipe. On the other hand, the vertical one holds a brass mass which can be moved along the whole arm.

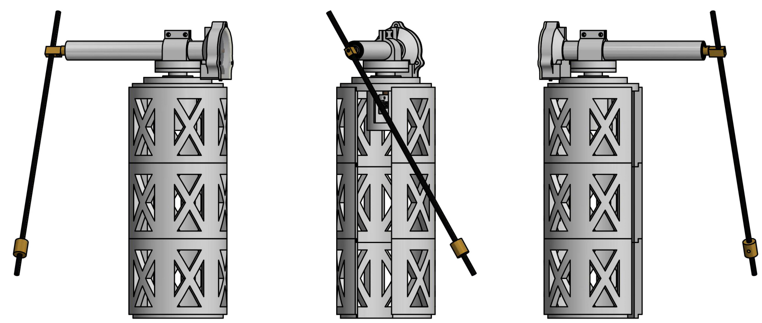

Figure 9: Full design of the Furuta pendulum

7.4 Tools

The project has been carried out using the following devices:

- The manufacture of the plastic (polylactide) structure was realized by a 3D printer (MakerBot Replicator 2) as it allows the fabrication in the laboratory in a simple and fast way, without the need of resort to external suppliers. Besides, the rapid prototyping makes parts can be checked in situ and if they don't fit the requirements, the design can be changed and improved at the time.

- The movement of the horizontal arm is carried out by a Maxon EC45 brushless DC motor. It is a ⌀45 mm and 150 Watt motor, with a 24 Volts input, 183 mNm of nominal torque and 952 mNm of stall torque. It includes a Hall sensor, a 500 PPR encoder (HEDS-9140) and a brake, which was removed due to the additional current supply it needs and the heating involved. Besides, the torque used is small which involves that the security requirements are minimum and an emergency brake is not a must.

- Another encoder (HEDS-9040) measures the inclination of the pendulum (the vertical arm). It has a higher resolution that the one inside the motor (2000 PPR versus 500 PPR). This is because the angle of the pendulum is the controlled variable and requires a higher precision. Besides, the measurement of the exact angle of the horizontal arm is not necessary, only its variation which via derivation by time gives the velocity and acceleration of the arm.

- A Trinamic's TMCM-1640. It is a small low cost controller and driver module for universal brushless DC motors applications. The board can be used in standalone operation or remote controlled via serial interface using the Trinamic Motor Control Lenguage (TMCL). In our project, the module is responsible for carrying out the torque control. It is connected via USB to a computer where the control algorithm for the pendulum is executed in a MATLAB environment. The adaptation of language between the board (TMCL) and MATLAB has been performed developing an Object Oriented library (Appendix A).

- An Arduino Micro board performs the reading of the two encoders. It is connected also via USB to the computer and sends the angle of both arms. With this measure, the velocity and acceleration of the arms can be obtained and with them, the torque needed for the pendulum control.
- A Simulink simulation model was realized to perform simulations of various control techniques and to adjust the parameters of them. The model allows switching between the different swing-up techniques and choose between swinging the pendulum up, control its vertical position or perform a hybrid control. A scheme of the Simulink model can be found in Appendix B.

7.5 Future work

The next steps in the development of the project are to conduct a thorough measurement of the parameters of the real system, performing various measures and focusing mainly on the moments of inertia, motor torque constant and friction coefficients. After that, the different control strategies explained along this document will be implemented and tested, so that the results may be compared with the theoretical ones obtained by the simulations.

Besides, there are some features which can be improved in the near time. One of them is the utilization of two different boards. The Trinamic's board has only one encoder slot, which makes necessary adding another device. Besides, executing a command in that board, such as setting a torque or reading the encoder, is expensive in time. For that reason, the distribution of tasks in parallel is a must. Thus, the Arduino board is responsible of reading the two encoders whereas the Trinamic's one sets the torque for the control of the pendulum. A possible solution for this could be the utilization of a more powerful board (for example a Raspberry Pi). This new board would perform the control of the system by itself, which means read the encoders, execute the control algorithm and set the proper torque. Every device would be connected to this board so that the time spent on serial communication would be minimized. Finally, the connection with a computer would be only used for showing information about the system, plotting graphics or setting the system's operation mode.

On the other hand, another feature that could be improved is the cable connection of the system. As there are two boards, two different cabling can be differentiated: the connection between the motor and Trinamic's board, where the cables run inside the cylindrical structure in a clean and unproblematic way. Otherwise, the connection between the Arduino board and the two encoders is where the problem resides. The connection with the motor encoder is carried out inside the structure as well, but the connection with the encoder in the vertical arm the cables hang around the structure. This means that while the system is running several cables are rotating with the arm, which can generate electrical noise as well as curls and pulls in the cables. A possibility to solve this situation is to connect the encoder to the board via wireless connection. Along with the other solution proposed above, it would allow the encapsulation of the system, so it was much more compact, robust and easy to transport (which was one of the main requirements).

8 Conclusion

A detailed dynamical model of the Furuta pendulum is provided. To this model several control strategies have been applied, both swing-up and vertical position stabilization. This two different control strategies result in a hybrid control, which allows to raise the pendulum from its stable downward position to a vertical position. To discuss these techniques, several simulations have been made and attached, that allow to check the behaviour of the system to such controls strategies.
Besides, the development project realized in the FabLab of the Óbuda University has been described. The motivation of the project as well as the design, used tools and the work that will be performed in the future time have been discussed.

A TMCL-MATLAB library

```
classdef driver

    properties (Access = private)

        port;

    end

    methods (Access = private)

        function receivedValue = decryption(value)

            byte1 = dec2hex(value(5),2);
            byte2 = dec2hex(value(6),2);
            byte3 = dec2hex(value(7),2);
            byte4 = dec2hex(value(8),2);

            bit1 = hex2dec(byte1(1));
            bit2 = hex2dec(byte1(2));
            bit3 = hex2dec(byte2(1));
            bit4 = hex2dec(byte2(2));
            bit5 = hex2dec(byte3(1));
            bit6 = hex2dec(byte3(2));
            bit7 = hex2dec(byte4(1));
            bit8 = hex2dec(byte4(2));

            valuehelp8 = bit8;
            valuehelp7 = bit7 * 16;
            valuehelp6 = bit6 * 256;
            valuehelp5 = bit5 * 4096;
            valuehelp4 = bit4 * 65536;
            valuehelp3 = bit3 * 1048576;
            valuehelp2 = bit2 * 16777216;
            valuehelp1 = bit1 * 268435456;

            receivedValue = valuehelp1 + valuehelp2 + valuehelp3 ...
                + valuehelp4 + valuehelp5 + valuehelp6 + ...
                valuehelp7 + valuehelp8;
        end

    end
    methods

        function obj = driver(comnumber)

            obj.port = serial(comnumber);
            set(obj.port, 'BaudRate', 1000000);
            set(obj.port, 'InputBufferSize', 9);
            fopen(obj.port);

        end

        function answer = setBaudRate(obj, value)
```

```

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 9 65 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);
            answer = fread(obj.port);
        end

        function BaudRate = getBaudRate(obj)

           message = uint8([1 10 65 0 0 0 0 0 76]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            BaudRate = decryption(msgback);

        end

        function setEncoderSteps(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 250 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function steps = getEncoderSteps(obj)

            message = uint8([1 6 250 0 0 0 0 0 1]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            steps = decryption(msgback);

        end

        function setVelocityRamp(obj,value)

```

```
            if value == 0
                message = uint8([1 5 146 0 0 0 0 0 152]);
            else
                message = uint8([1 5 146 0 0 0 0 1 153]);
            end
            fwrite(obj.port, message);

        end

        function velocityRamp = getVelocityRamp(obj)

            message = uint8([1 6 146 0 0 0 0 0 153]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            velocityRamp = decryption(msgback);

        end

        function setHallSensorInvert(obj, value)

            if value == 0
                message = uint8([1 5 254 0 0 0 0 0 4]);
            else
                message = uint8([1 5 254 0 0 0 0 1 5]);
            end
            fwrite(obj.port, message);

        end

        function hallSensorInvert = getHallSensorInvert(obj)

            message = uint8([1 6 254 0 0 0 0 0 5]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            hallSensorInvert = decryption(msgback);

        end

        function setMotorPoles(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 253 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

```

```
        function poles = getMotorPoles(obj)

            message = uint8([1 6 253 0 0 0 0 0 4]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            poles = decryption(msgback);

        end

        function setMotorCurrentP(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 172 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function currentP = getMotorCurrentP(obj)

            message = uint8([1 6 172 0 0 0 0 0 179]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            currentP = decryption(msgback);

        end

        function setMotorCurrentI(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 173 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function currentI = getMotorCurrentI(obj)
```

```
            message = uint8([1 6 173 0 0 0 0 0 180]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            currentI = decryption(msgback);

        end

        function setMotorSpeedP(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 234 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function speedP = getMotorSpeedP(obj)

            message = uint8([1 6 234 0 0 0 0 0 241]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            speedP = decryption(msgback);

        end

        function setMotorSpeedI(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 235 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function speedI = getMotorSpeedI(obj)

```

```
            message = uint8([1 6 235 0 0 0 0 0 242]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            speedI = decryption(msgback);

        end

        function setMotorMaxCurrent(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 6 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function maxCurrent = getMotorMaxCurrent(obj)

            message = uint8([1 6 6 0 0 0 0 0 13]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            maxCurrent = decryption(msgback);

        end

        function setMotorCurrent(obj, value)

            if value < 0
                value = 4294967296 + value;
            end
            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 155 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

```

```
        function current = getMotorCurrent(obj)

            message = uint8([1 6 150 0 0 0 0 0 157]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            current = decryption(msgback);
            if current > 2147483647
                current = current - 4294967296;
            end
        end

        function rotateLeft(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 2 0 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function rotateRight(obj, value)

            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 1 0 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function setMotorSpeed(obj, value)

            if value < 0
                value = 4294967296 + value;
            end
            binvalue = dec2bin(value,32);
            value1 = uint8(bin2dec(binvalue(1:8)));
            value2 = uint8(bin2dec(binvalue(9:16)));
```

```
            value3 = uint8(bin2dec(binvalue(17:24)));
            value4 = uint8(bin2dec(binvalue(25:32)));
            message = [uint8([1 5 2 0]) value1 value2 value3 ...
                value4 0];
            checksum = sum(message);
            while checksum > 255
                checksum = checksum - 256;
            end
            checksum = uint8(checksum);
            message(9) = checksum;
            fwrite(obj.port, message);

        end

        function speed = getMotorSpeed(obj)

            message = uint8([1 6 3 0 0 0 0 0 10]);
            fwrite(obj.port, message);
            msgback =  fread(obj.port);
            speed = decryption(msgback);
            if speed > 2147483647
               speed = speed - 4294967296;
            end

        end

        function position = getMotorPosition(obj)

            message = uint8([1 6 1 0 0 0 0 0 8]);
            fwrite(obj.port, message);
            msgback = fread(obj.port);
            position = decryption(msgback);
            if position > 2147483647
                position = position - 4294967296;
            end

        end

        function stopMotor(obj)

            message = uint8([1 3 0 0 0 0 0 0 4]);
            fwrite(obj.port, message);

        end

    end

end
```

B Simulink model scheme

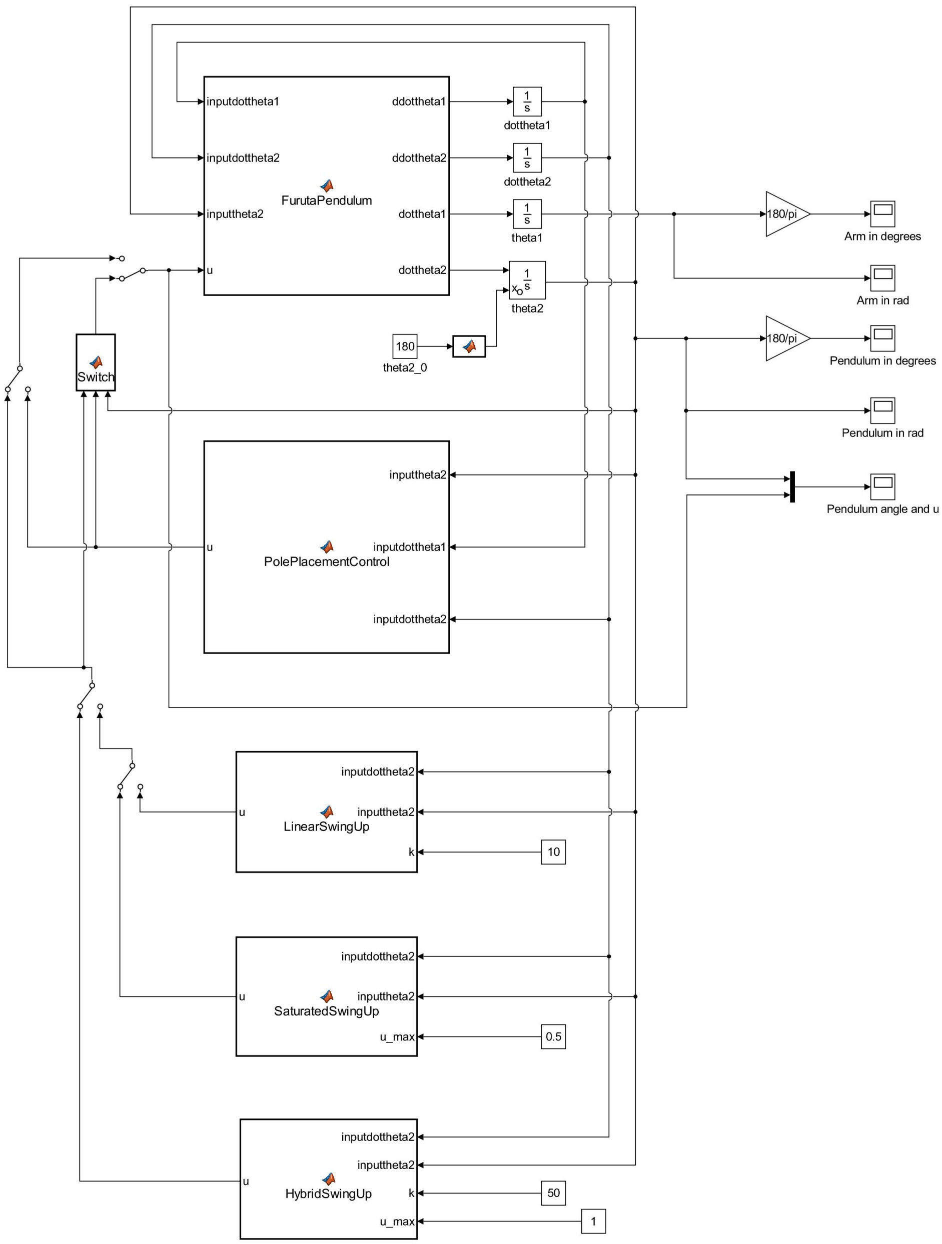

Figure 10: Simulink model of the Furuta pendulum and its different control strategies

Bibliography

[1] J. Aracil and F. Gordillo, "El péndulo invertido: un desafío para el control no lineal," *RIAII*, vol. 2, no. 2, pp. 8–19, 2005.

[2] J. A. Acosta, "Furuta's Pendulum: A Conservative Nonlinear Model for Theory and Practise," *Mathematical Problems in Engineering*, 2010.

[3] Y. Su, D. Sun, and B. Duan, "Design of an enhanced nonlinear PID controller," *Mechatronics*, vol. 15, no. 8, pp. 1005–1024, 2005.

[4] V. I. Utkin, "Sliding mode control," *Variable structure systems: from principles to implementation*, vol. 66, p. 1, 2004.

[5] F. R. Garces, V. M. Becerra, C. Kambhampati, and K. Warwick, *Strategies for feedback linearisation: a dynamic neural network approach.* Springer Science & Business Media, 2012.

[6] M. S. Sadeghi, B. Safarinejadian, and A. Farughian, "Parallel distributed compensator design of tank level control based on fuzzy takagi–sugeno model," *Applied Soft Computing*, vol. 21, pp. 280–285, 2014.

[7] P. Grof and Y. Yam, "Furuta pendulum–a tensor product model-based design approach case study," in *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pp. 2620–2625, IEEE, 2015.

[8] P. Galambos and P. Baranyi, "TP model transformation: A systematic modelling framework to handle internal time delays in control systems," *Asian Journal of Control*, vol. 17, no. 2, pp. 486–496, 2015.

[9] B. S. Cazzolato and Z. Prime, "On the dynamics of the furuta pendulum," *Journal of Control Science and Engineering*, vol. 2011, p. 3, 2011.

[10] J. Acosta, J. Aracil, and F. Gordillo, "Estudio comparativo de diferentes estrategias de control para el péndulo de furuta," *XXI Jornadas de Automática*, no. 1.

[11] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[12] R. Freeman and P. V. Kokotovic, *Robust nonlinear control design: state-space and Lyapunov techniques.* Springer Science & Business Media, 2008.

[13] K. Ogata, *Modern Control Engineering.* Instrumentation and controls series, Prentice Hall, 2010.