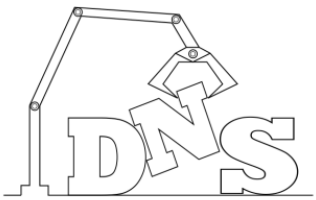
POLITECNICO DI TORINO

Mechatronic engineer master degree
Final project work

Model-based design of a fuzzy logic controller for an inverse pendulum

Supervisor:
Prof. Luigi Mazza
Co-supervisor:
Eng. Marco Pontin

Candidate:
Salvatore Di Natale 254192



A.Y. 2018/2019

Dreams: the source of innovation

Abstract

The main idea of this work is to design and realize a fuzzy controller to stabilize an inverse pendulum pneumatically actuated. This system is widely used and studied in academic and industrial applications. The fuzzy logic introduces a new design method in this classical system.
The system needs a double control, the position of the cart and the angle of the pendulum. This problem has been solved by using two nested feedback control loop. All the design has been done using the model-based approach.
The challenge of this work is to use a very low-cost controller, an Arduino board, to stabilize an inverted pendulum mechanically connected to a pneumatic actuator, so also the system is a low-cost application. The movement of the pneumatic actuator is controlled by four 2/2 digital pneumatic valves that are controlled by the Arduino board using the PWM technique.The Arduino board is programmed using the fuzzy logic and the control action is able to stabilize the cart in a given position.
After an introduction to the fuzzy logic (Chapter 2) and the description of the test bench (Chapter 3) I will provide a first design of a linear fuzzy controller starting from a PID controller already designed (Chapter 4). Then I will make the nonlinear model (Chapter 5) to design the nonlinear fuzzy controller (Chapter 6). In the chapter 7 I will provide the explanation of the program for the Arduino board. The experimental tests will not do due to time constraints, for this reason the program for the Arduino board is only a sketch and it should be tested and validated (I ensure that there are not compiler errors). So the purpose of this work is to give a new method (the fuzzy controller) to stabilize an inverse pendulum and its cart with low cost application following the model-based approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The inverse pendulum is widely studied in industrial application and for research reasons. It is a mechanical unstable system that consists of a rigid bar where at the upper end there is a concentrated mass. The bar is attached by a hinge in a mobile base. The base (generally a cart) can move freely (its moving can be rotative or linear) in order to guarantee the stabilization of the bar in the vertical position which is an unstable equilibrium point [10].
This type of problem is part of the main class of the under actuated mechanical system, i.e. all the systems that have less actuated degrees of freedom with respect to their real degrees of freedom.[8].
For example the inverted pendulum in [8] (very similar to the one that will be used here) has one input (u), the force, and two degrees of freedom (x and θ), one linear motion of the cart and one rotative motion around the joint of the pendulum (figure 1.1).

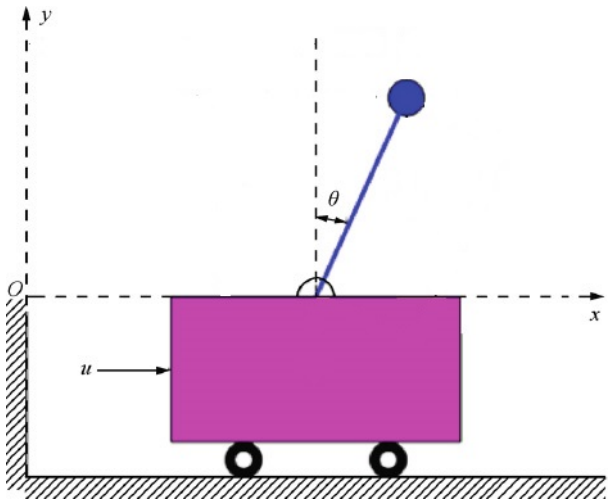


Figure 1.1: Example of an inverse pendulum [8]

Different types and model of the inverse pendulum exists. They differ by:

- The actuation of the base: it can be a cart or a roadless piston [3] or a cart connected with a pneumatic piston as in the case of this essay.

- The stabilization action: rotative [5], or linearly as in the case of this essay.

The common objective is to stabilize the bar of the pendulum in its unstable equilibrium point.
A very common example of application of the inverse pendulum is represented by the robot's class of the two wheeled moving robot [11]. From this type of robot the most common is the Segway® (figure 1.2).

Figure 1.2: A segway®

We can find other applications of the inverse pendulum in military environment, in fact similar algorithms are used to stabilize some kind of missiles and airplanes [1].
Recent studies [9] demonstrate the possibility to use the model of the inverse pendulum to develop a control action to stabilize an anthropomorphic robot. In figure 1.3 is possible to see a comparison between the inverse pendulum and a humanoid robot.

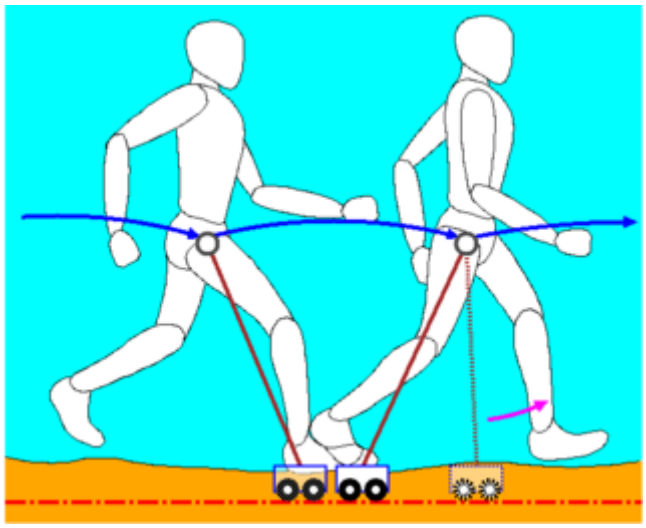
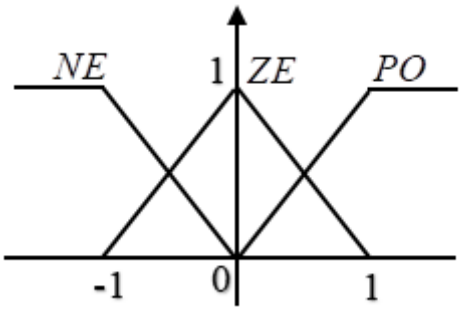
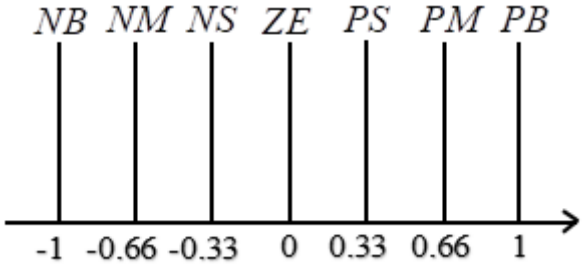


Figure 1.3: Comparison between the inverse pendulum and an humanoid robot

All of these applications are, in generally, expensive in terms of hardware of the controller and in terms of realization of the system. A low-cost choice is to realize the actuation of the cart of the pendulum with pneumatics as done in [3] and in [4].
Focusing on fuzzy logic a very interesting work is the [5]; where a fuzzy controller is used to stabilize a rotary inverted pendulum actuated with an electric motor. In this article the system is, first of all, modeled and simulated in MatLab® and then the controller is applied in a real system. The goal of the control action is to stabilize the pendulum in the vertical position and also to stabilize the arm in a given position.
Concerning the controller configuration, 4 inputs and one output are used. The four inputs are: the pendulum angle and its velocity, the arm angle and its velocity. The fuzzy sets of inputs and output are shown in figure 1.4(a) and 1.4(b).



(a) Membership functions of four standardized inputs [5]



(b) Membership functions of standardized output [5]

Figure 1.4: Membership functions of inputs and outputs

This configuration results in 81 rules visible in the appendix of the article.
The inputs and output are weighted with different gain. The overall control structure is shown in figure 1.5.

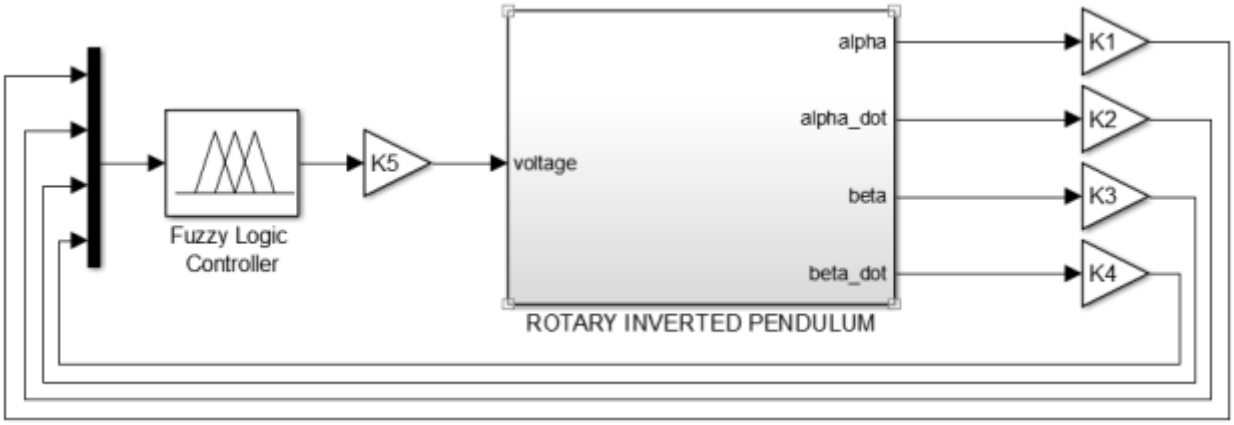


Figure 1.5: Overall control structure for the rotary inverted pendulum [5]

The gains K1,K2,K3,K4,K5 are chosen with trial and error. At the end of the article the experimental results shown that the system is stable with some oscillations and shows that the fuzzy controller is suitable to stabilize the inverse pendulum.

With my work I will design a fuzzy controller to stabilize a linear inverse pendulum pneumatically actuated. The challenge of this work is to use a very low-cost controller, an Arduino board, to stabilize an inverted pendulum mechanically connected to a pneumatic actuator. The movement of the pneumatic actuator is controlled by four 2/2 digital pneumatic valves that are controlled by the Arduino board using the PWM technique. The Arduino board is programmed using the fuzzy logic and the control action is able to stabilize the cart in a given position.
After an introduction to the fuzzy logic (Chapter 2) and the description of the test bench (Chapter 3) I will provide a first design of a linear fuzzy controller starting from a PID controller already designed (Chapter 4). Then I will make the nonlinear model (Chapter 5) to design the nonlinear fuzzy controller (Chapter 6). In the chapter 7 I will provide the explanation of the program for the Arduino board. The experimental tests will not do due to time constraints, for this reason the program for the Arduino board is only a sketch and it should be tested and validated (I ensure that there is not compiler errors). So the purpose of this work is to give a new method (the fuzzy controller) to stabilize an inverse pendulum and its cart with low cost application following the model-based approach.

Chapter 2

An introduction to fuzzy logic

The classical logic or Boolean logic provides only two values: true/false or one/zero. Now thinking at most of the reality conditions, a statement is rarely absolutely true or absolutely false. With the fuzzy logic a statement can be partially true or partially false.
A great example is a glass filled with water (figure 2.1). The glass can be full or empty; or it can be partially full or partially empty. With this very simple example is possible to see the difference between Boolean logic and Fuzzy logic.

Figure 2.1: Glass filled with water

Another example from [7] is an air conditioner. With the Boolean logic the temperature is hot or cold, but with the Fuzzy logic it can be "too cold" or "not so cold" or "too hot" and so on. Considering the statement "If the temperature is greater than 21°C it is hot" we can see the difference between the two logics in figure 2.2 where the dashed line represents the truth function of a statement with Boolean logic and the continuous line represents the truth function of a statement with Fuzzy logic.

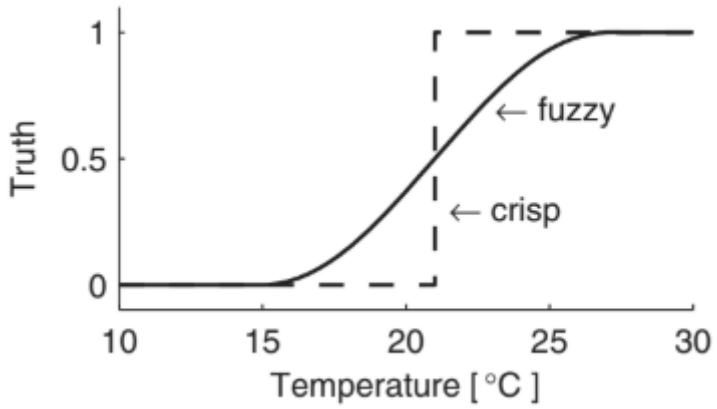


Figure 2.2: Comparison between Boolean and Fuzzy logics [7]

In figure 2.2 by the Fuzzy logic the temperature goes from cold to hot in a continuous fashion (at least in principle). That represents a very advantage in the control world because, for instance, by the Fuzzy logic it is possible to control the speed of the fans of the air conditioner in depending of the temperature of the room continuously and, hence, the control action should be more precise and more efficient.
A fuzzy controller is known to be a rule-based controller. It consists essentially in a series of if-then rules. Each rule is composed of an antecedent condition ("if") and of a consequent condition ("then"). The precedent condition is the result of the application of the fuzzy membership function to the input variables. The consequent condition is the result of the application of the membership functions to the output variables.
A membership function is the core of a fuzzy controller. It is a truth function for a linguistic variable (e.g. "high"). A clearer explanation is represented in figure 2.3.

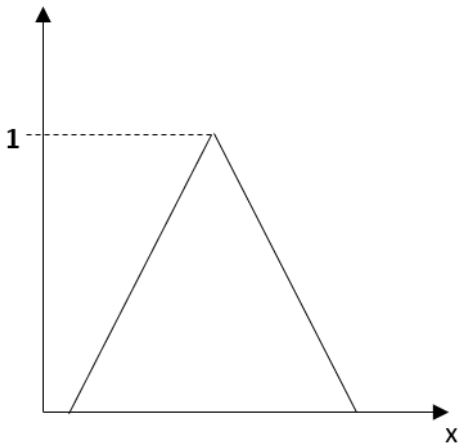


Figure 2.3: A membership function

In the figure the variable x is the input or output variable. In depending of its value the membership function can assume different values of truth (from 0 to 1). In general a variable has more than one membership functions with different names and the set of all membership functions is called fuzzy set. Recapping the example of the air conditioner a possible fuzzy set for the input variable (the temperature) can be the following one represented in figure 2.4.

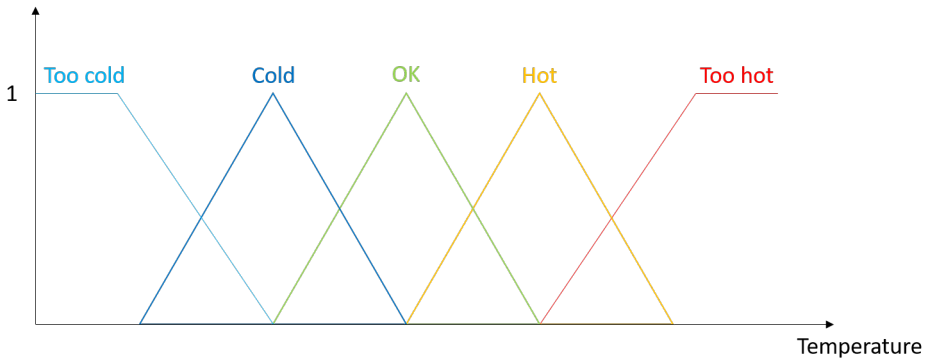


Figure 2.4: A possible fuzzy set for the input variable "temperature"

The range of the input variable is chosen by the designer and it is not the real range, but it is fuzzified, i.e. it is multiplied for a gain. Also, the values and the shape of each membership function is decided by the designer. The membership function can be triangular, trapezoidal, gaussian etc.
For the output variables is the same, the designer chooses a fuzzified range and the fuzzy set.
Once the output is computed accordingly to the rule-base it needs to be defuzzied. In fact, more than one rule can be activated. The output value will be the result of an algorithm. The most common one is the COG (Center of gravity) that consists in a weighted average between all the values given by each rule (if the rule is not activated its output is zero).
The Fuzzy controllers are widely used in industries for the following advantages:

- The control strategy consists of if-then rules and they are built using common words like "low", "high". In this way process operators can easily understand what the controller is doing, and designers can embed their experience directly. A fuzzy controller, in fact, is a human intuition controller.
- A fuzzy controller can accommodate multiple inputs and multiple outputs and they are connected in the if-then rules with the logic connectives AND/OR. All the rules are executed in parallel and the outputs are defuzzied according to a defuzzification method.

In principle a fuzzy controller is intuitively designed from experience. A sort of rigorous method exists to design it and to verify its stability. The method is depicted in [2] and it is:

- Design a PID controller
- Replace it with a linear fuzzy controller
- Make it nonlinear
- Fine-tune it

I will use this method in the following chapters to design my fuzzy controller.
Finally, concerning the tools that I will use, they are essentially two. A MatLab® tool called "Fuzzy logic toolbox" that is a very powerful tool to implement in Simulink® a fuzzy controller in a graphical view. This tool is useful to simulate the behavior of a control structure with a fuzzy controller. The tool provides a window in wich the controller can be set. The window can be open trought the command "fuzzy" (figure 2.5). In this window the designer can add IN/OUT variables and selected all the methods for the logic functions and for the defuzzification process.

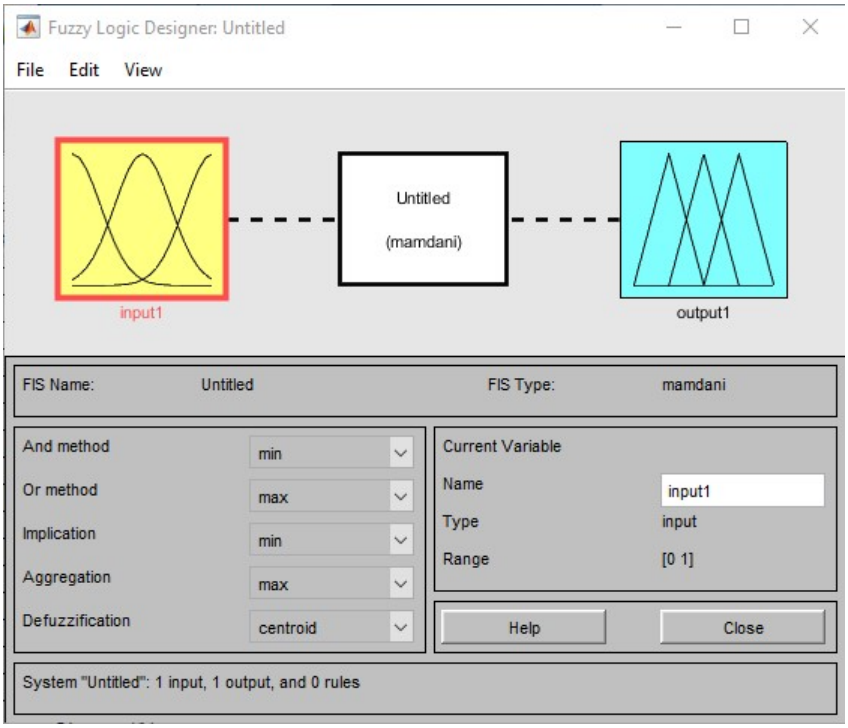
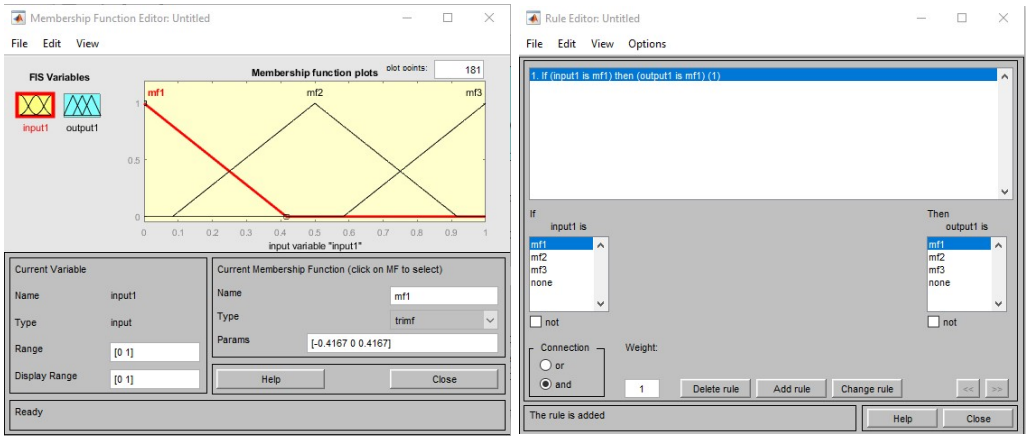


Figure 2.5: Main window of the Fuzzy logic toolbox of MatLab®

Through the menu "edit" is possible to access to the different area of the tool. They are two:

- Membership functions: In this window all the membership functions for each IN/OUT variable can be selected (shape and range). It is shown in figure 2.6(a).

- Rules: In this window all the rules are written in a linguistic way (If...AND—OR...then...). It is show in figure 2.6(b)



(a) Window of the fuzzy logic toolbox to set the membership function

(b) Window of the fuzzy logic toolbox to set the rules

Figure 2.6: The windows of the fuzzy logic toolbox accessible through the "edit" menu

Another tool is a C-library called "Fuzzy.h", it will be used in the Arduino idle to implement in a OOP fashion the algorithm of a fuzzy controller in the Arduino board. The library provides a class called "Fuzzy" where there are a set of functions to set very easily the controller. They will be explained in detail in a dedicated chapter.

Chapter 3

Description of the test bench

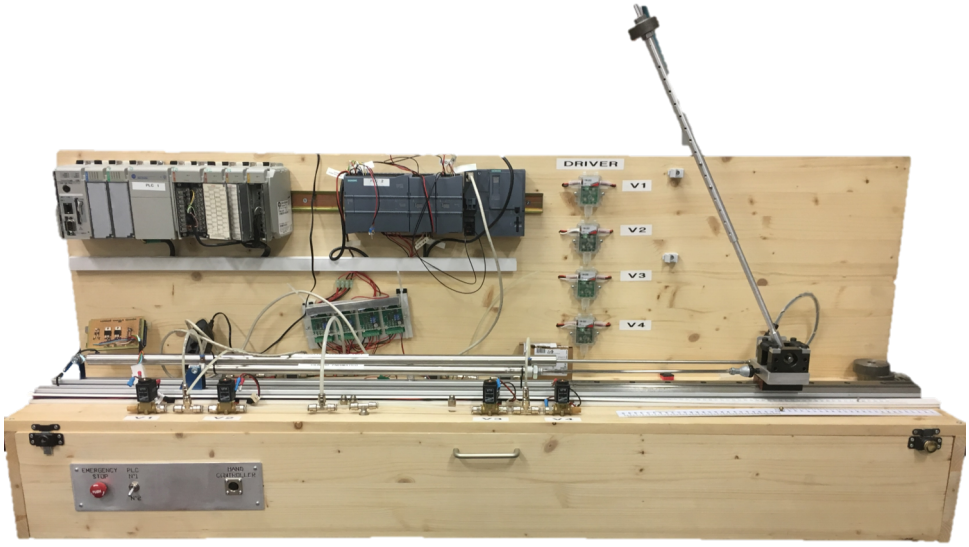


Figure 3.1: The test bench

The test bench in figure 3.1 is designed and realized by Marco Pontin [6] and Marino Alessandro. Its main features are the mounting and maintenance simplicity; it is possible also to modify some parameters in order to see their effects. The last feature is very important in a didactic device, that is the case of this bench.
The main components of the test bench can be classified in three categories: pneumatic, electric, mechanic.
The pneumatic components are:

- Pneumatic actuator
- 2/2 NC electrovalves
- Pipeline

The electric components are:

- Angular transducer
- Linear transducer
- Power suppliers
- Arduino board and two PLCs (the PLCs will not be used)
- PCB

The mechanic components are:

- Main structure
- Ball recirculating linear guide
- Cart/Pendulum group

An overall schematic view of the test bench is shown in figure 3.2. This figure is significant to better understand how the components are connected.

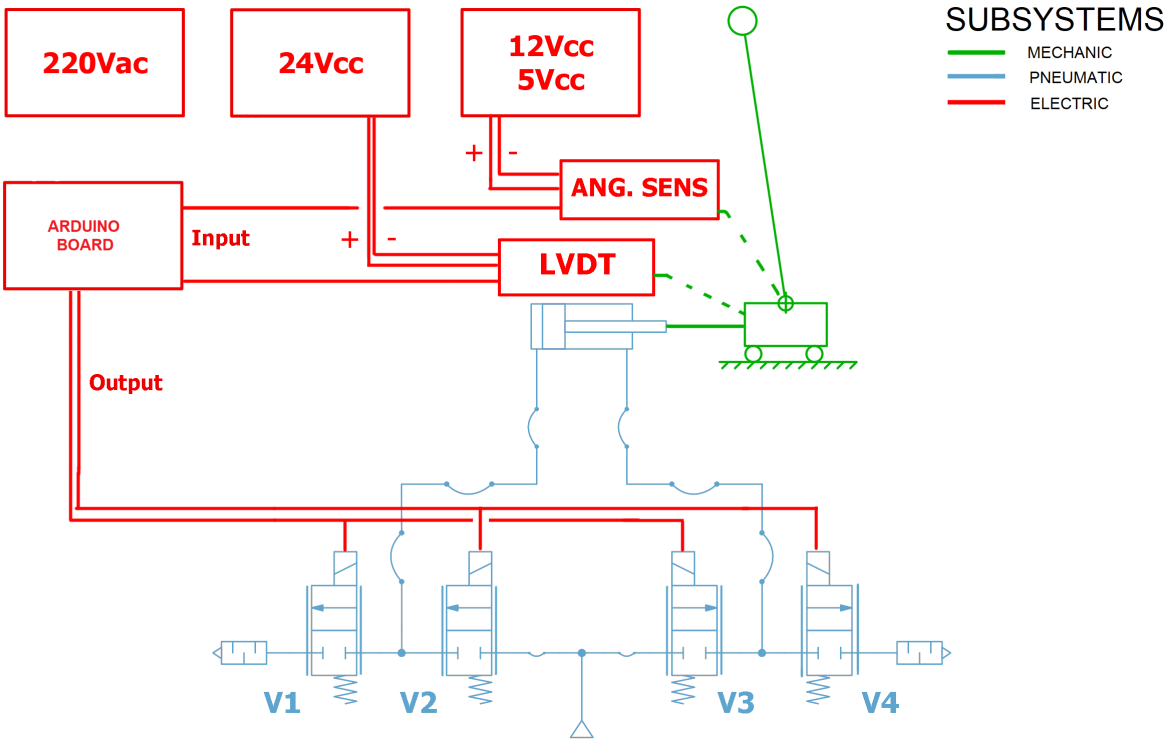


Figure 3.2: Schematic of the test bench [6]

In the following each component will be explained in detail, for more building or designing information please see [6].

3.1 Pneumatic components

The pneumatic actuator is a double effect linear actuator without dampers. It has a stroke of $500mm$. It is fixed to the main structure through a hinge and to the cart through a spherical joint. The two joints allow to compensate some misalignments and they guarantee that the force is always applied along the axis of the rod.
Four valves are collocated in the base of the bench. They are 2/2 normally closed proportional electrovalve (figure 3.3). Their opening or closing allows the piston of the actuator to perform a linear movement.

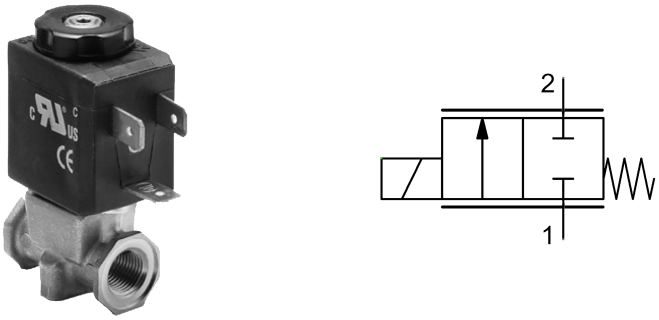


Figure 3.3: Valve AP-7211-LR2-U7 and its ISO schematic

Their opening is controllable by a driver that guarantee a linear proportional behavior. The driver perform a PWM feedback current control at 500Hz. By setting as input a voltage it is able to perform the control action to the solenoid of the valve, so the driver will be controlled with a PWM voltage signal provided by the Arduino board in the range $0 \div 5V$ at a greater frequency than the working one of the driver. So after a voltage amplifier that doubles the voltage provided by the Arduino board, it is sent to the driver: a voltage equal to $0V_{dc}$ corresponds to a 0% duty cycle and so the valve is totally closed, otherwise a voltage equal to $10V_{dc}$ corresponds to a 100% duty cycle and so the valve is totally open.

3.2 Electric component

With the angular transducer the Arduino can measure the angular position of the pendulum's bar. This is possible because the angle measurement is transformed into a voltage signal by this transducer. In particular it is a Hall's effect transducer.
On the same way of the angular transducer, with the position transducer the Arduino can measure the linear position of the cart. It is an LVDT type and it is fixed to the test

bench and to the cart through two hinge in order to compensate some misalignments. Its stroke is equal to the stroke of the pneumatic actuator ($500mm$) and the output voltage can assume a value in the range $0 \div 10V_{dc}$. Due to that range it is not possible to connect it directly to the analog input pin of the Arduino board because it accepts a maximum voltage of $5V_{dc}$. So the output voltage of the transducer needs to be halved; hence between the transducer and the analog input pin of the Arduino there is a voltage amplifier which will be described after.
To apply the control action to the plant an Arduino Mega 2560 (figure 3.4) board programmed with Fuzzy logic is used. The Arduino Mega 2560 is a microcontroller board. It is based on the microchip ATmega2560. It's ports map consists of 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack and a reset button. The board can be programmed with the Arduino Software (IDE) that provides high-level functions using the C-language.

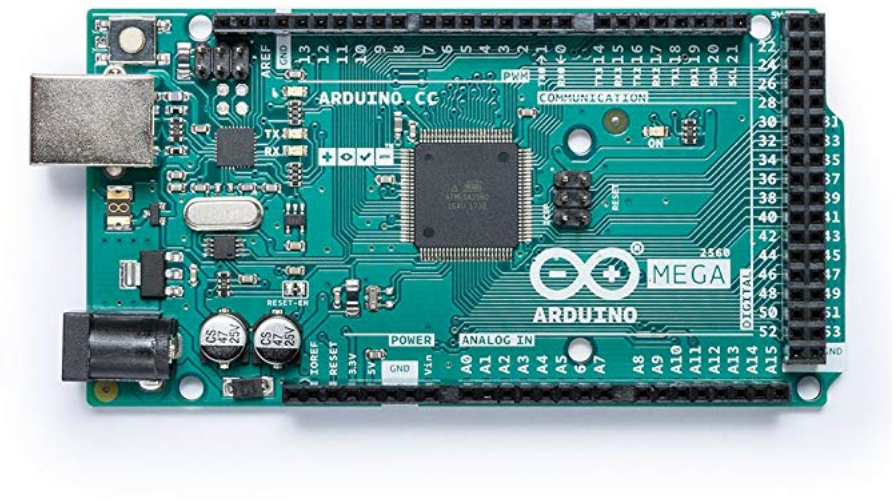


Figure 3.4: Arduino Mega 2560

The ATmega2560 on the Mega 2560 is already preprogrammed with a bootloader that allows to upload new code to it without the use of an external hardware programmer.
It has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader) and 8 KB of SRAM.

3.2.1 PCB

A PCB has been designed by me and realized by the online shop JLCPCB. The PCB works as interface between the sensors, valves'driver and the Arduino board. In fact the Arduino accepts as input and gives as output a voltage in the range $0 \div 5V$; but, as explained before, the components works with different range of voltages.
In particular the valves' drivers works with a voltage in the range $0 \div 10V$, so a voltage amplifier circuit that doubles the voltage given by the Arduino board is needed. Since the valves are controlled in couple, two of these circuits are needed. A schematic of them is shown in figure 3.5.

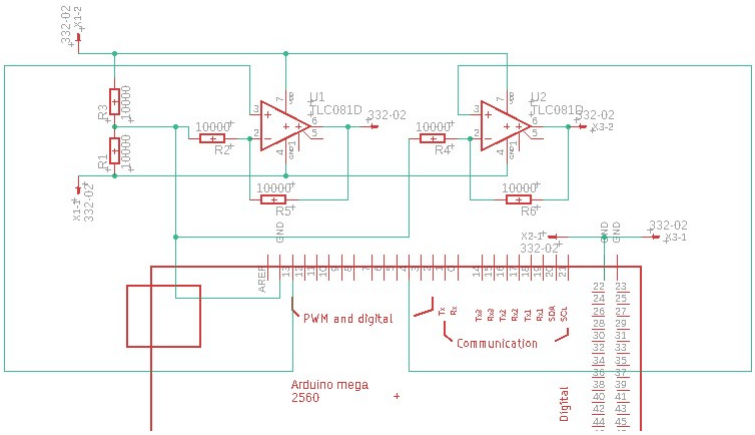


Figure 3.5: Schematic of a voltage amplifier circuit in the PCB

In figure 3.5 the operational amplifier is a TL081 provided by Texas Instrument and the resistor's values are chosen accordingly to the gain of the circuit that is:

$$A = 1 + \frac{R_2}{R_1} = 2 \Rightarrow \frac{R_2}{R_1} = 1 \tag{3.1}$$

So according to equation (3.1) the two resistors must be equal. The value can be anyone and it is $10k\Omega$. With referring to figure 3.5 there is also a voltage divider to supply the operational amplifier. That is necessary because the voltage supplier in the test bench gives $+24V$. In this way the $+24V$ can be divided into $+12V$ and $-12V$. Concerning the input of the circuits they are, obviously, the voltage gives by the Arduino board (the input in the non-inverting pin of the op-amps from the pins 13 and 4 of the Arduino board). The output of the op-amp will be connected to a pin in the PCB that will be connected to the valve's driver (pins X2-2 and X3-2).
The LVDT position sensor gives a voltage in the range $0 \div 10V$ so it needs to be halved. Since the input port of the Arduino has a very high impedance a simple voltage divider can be used to half the voltage. It's schematic is shown in figure 3.6 where the sensor is connected in the pins X5-2 (for the input in the port A8 in the Arduino board) and X5-1 (for the ground). Also in this case the resistors' values must be equal and they are $100k\Omega$.

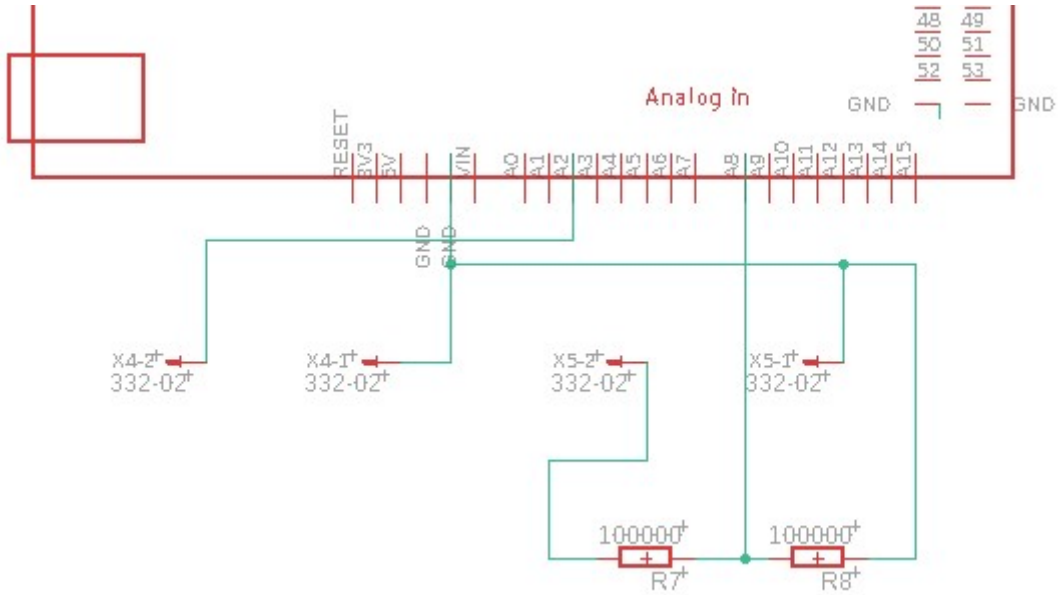


Figure 3.6: Schematic of a voltage divider circuit in the PCB

Referring to figure 3.6 there are also other two pins (X4-1 and X4-2) directly connected to the pin A2 and to the ground of the Arduino board. They are for the angular sensor that gives a voltage in the range $0 \div 5V$ so no modifications are needed and it can be connected directly to the pin of the Arduino board.
The total schematic of the PCB is shown in figure 3.7 where also a schematic of the shield for the Arduino Mega 2560 is included.

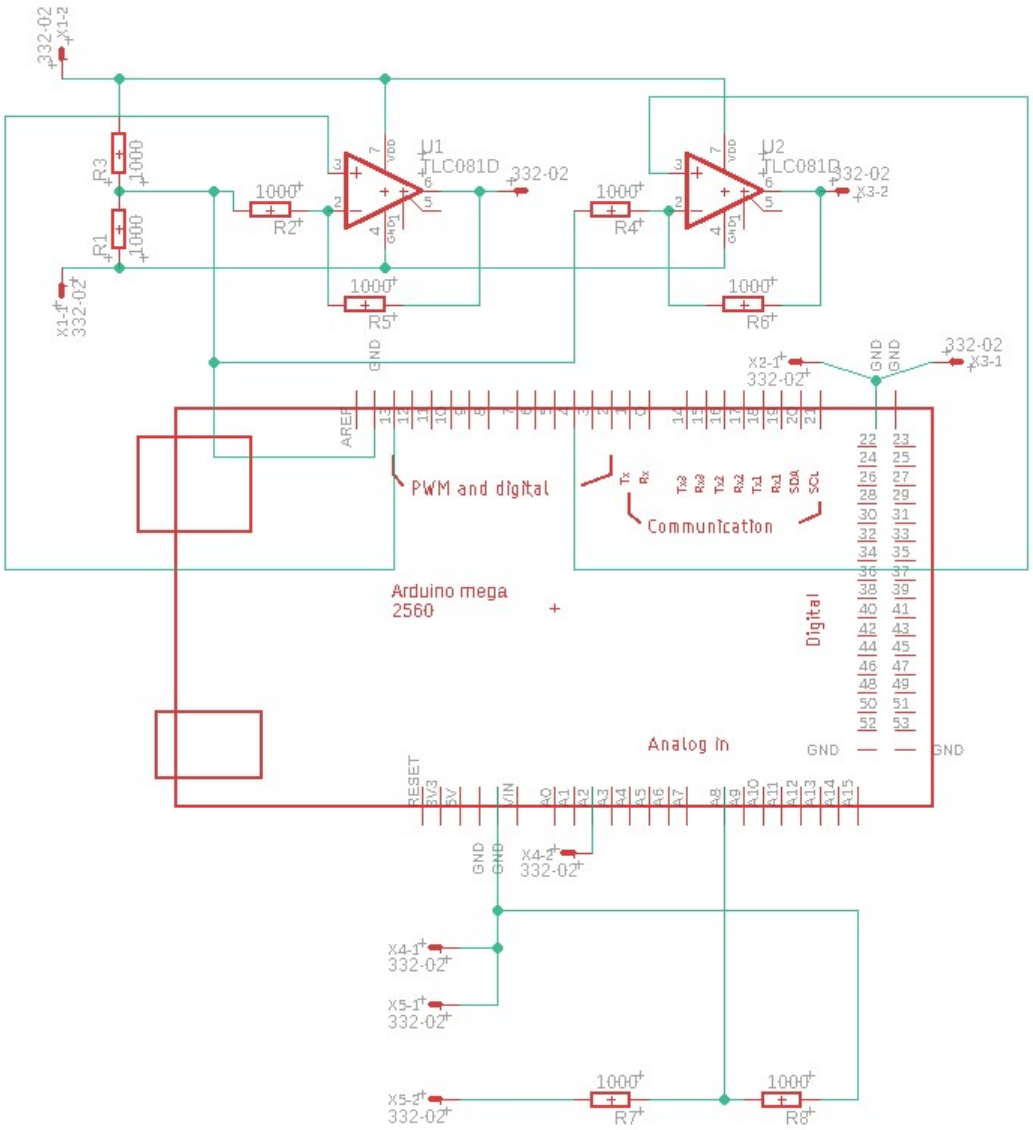


Figure 3.7: Schematic of the PCB

The realized PCB with all the components is shown in figure 3.8.

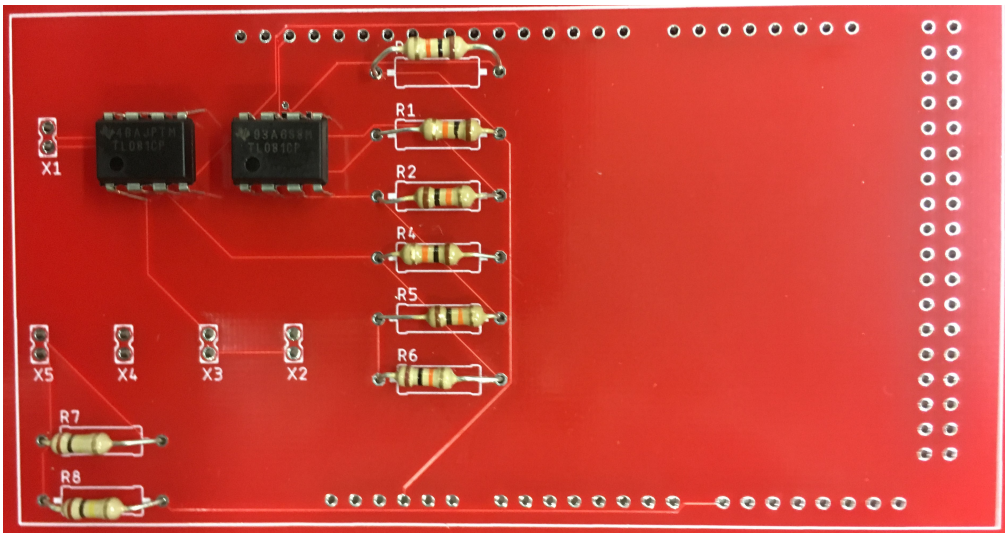


Figure 3.8: PCB

3.3 Mechanic components

The cart and the pendulum are joined together trough a hinge in the cart, in this way the pendulum can only rotate around one axis with respect to the cart.
The pendulum is built such that an user can modify easily its length and the concentrated mass at one end. In particular the bar is telescopic, in this way its length can be modified in the range $400 \div 700mm$. I will use a value of $500mm$ that is a middle value. The mass can be changed by changing a steel disk.
All the component of the pendulum except the concentrated mass is made of aluminium because in this way the bar is very light and its inertia moment can be neglected, so it is very near to an ideal pendulum with only a concentrated mass at one end.
The main goal of the cart is to allow the control action. Screwed holes allow to connect the cart to the pneumatic actuator and to the LVDT sensor. There are also two small plates to limit the angular stroke of the pendulum, in this way it can not fall in the bench.
A drawing of the joining between the cart and the pendulum is shown in figure 3.9 where the movement axes of the system are put in evidence.

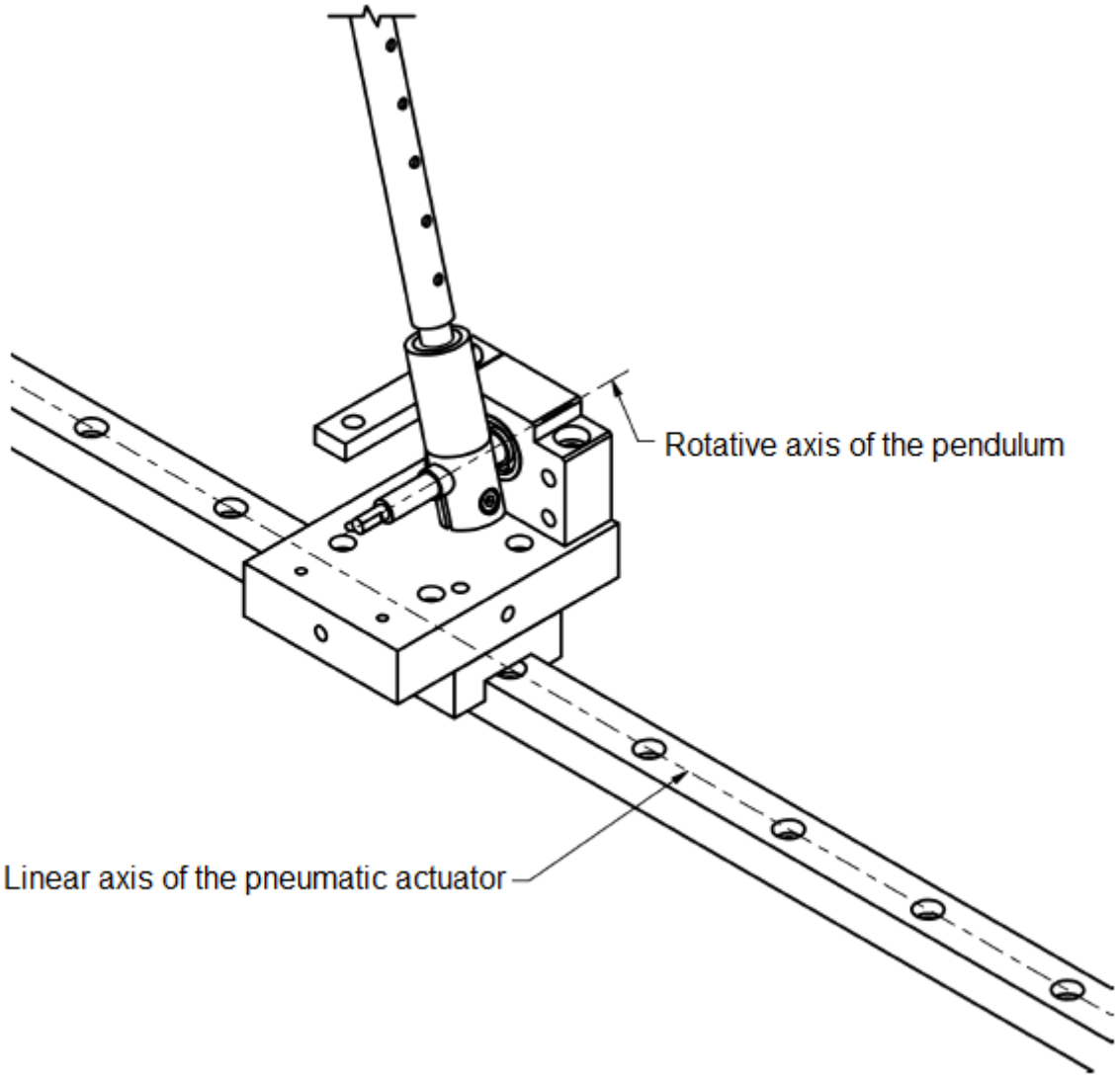


Figure 3.9: Drawing of the joining between cart and pendulum [6]

The degrees of freedom of this system are two and they are computed through the Grubler's equation:

$$dof = 3(m-1) - \sum_{i=1}^{m} l_i = 6 - 2 - 2 = 2 \tag{3.2}$$

Where m is the number of bodies and l_i is the number of degrees of freedom subtracted by the i-th cinematic couple.
The number of the degrees of freedom is correct because the system can perform only one linear movement (the movement of the cart) and only one rotative movement (the movement of the pendulum).
Finally, under the cart there is ball recirculating linear guide. Its goal is to allow the cart to slide easily and with a very low friction.The cart of the inverse pendulum is fixed to the slider through 4 threaded holes in the slider.

Chapter 4

First design approach

A procedure to design a fuzzy controller is explained in [2]. Essentially it consists in 4 steps:

1. Build and tune a conventional PID controller first
2. Replace it with an equivalent linear fuzzy controller
3. Make the fuzzy controller nonlinear
4. Fine-tune it

So according to this procedure I need a conventional PID controller. This controller is already designed in [6].
In this chapter I will exploit the linear control structure with the PID controllers and then I will design the fuzzy controllers.

4.1 Control structure with PID controllers

As already said the goal is to stabilize the angle of the pendulum and also the position of the cart. So the idea (from [6]) is to make two controllers with two nested feedback control loop, one to control the position of the cart (x) and one to control the angle of the pendulum (θ).
The block scheme of the overall control structure is shown in figure 4.1.

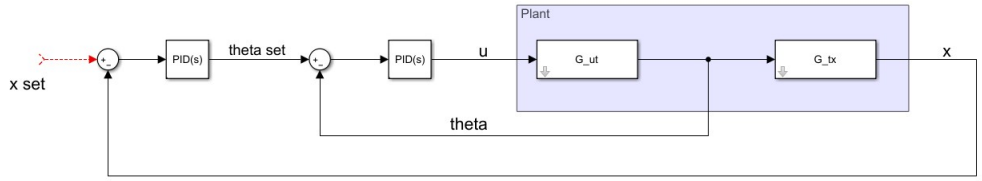


Figure 4.1: Block scheme of the overall control structure

4.1.1 Plant's transfer functions

In order to make the linear model of the plant I did the same considerations and hypothesys made in [6], in this way I can do a comparison between the response using the PID controllers and the response using the fuzzy controllers; hence I will provide a brief explanation of the used equations. For more details please see [6].
The overall system is divided into two subsystems: a mechanical system composed by the cart and the pendulum and a pneumatic system composed by the actuation valve and the pneumatic cylinder.
The reference conditions of the overall plant are a situation in which the piston is in the middle of its stroke (this corresponds to x=0) and the pendulum is in the vertical position ($\theta = 0$).
The starting point is the model of the mechanical system. A scheme of the system is visible in figure 4.2 where the arrows indicate the positive verse of the generalized coordinates (x and θ) and their derivatives.

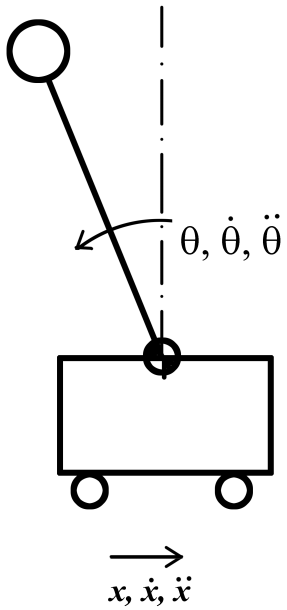


Figure 4.2: Scheme of the mechanical system: cart plus inverted pendulum [6]

Now the goal is to find a set of equations that give the generalized coordinates in depending of the input quantity. The input quantity is the force F that the pneumatic cylinder gives to the cart. To do that the Newtonian's equations is used with respect to the free body diagram represented in figure 4.3

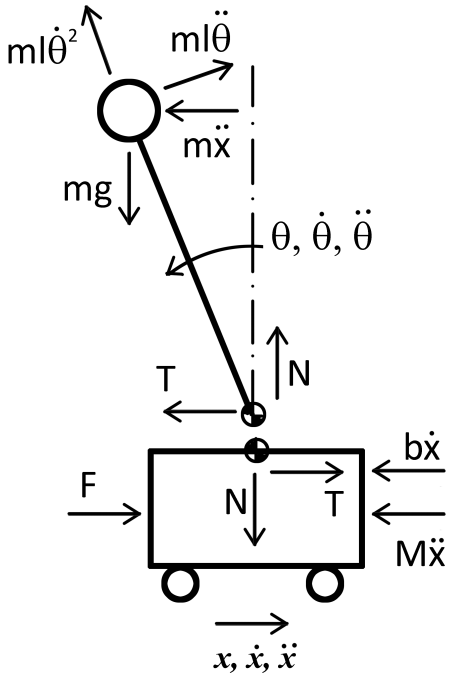


Figure 4.3: Free body diagram of the mechanical system [6]

Since this first design is based on linear assumptions, the following approximations are made:

$$\begin{aligned} \theta &\approx 0 \\ \dot{\theta}^2 &\approx 0 \\ \sin(\theta) &\approx \theta \\ \cos(\theta) &\approx 1 \end{aligned}$$

So the equations for the mechanical system are (the subscript c stands for cart):

$$M_c\ddot{x} + b_c\dot{x} - (m + M_c) = F \tag{4.1}$$
$$M_c\ddot{x} + b_c\dot{x} - mg\theta = F \tag{4.2}$$

The pneumatic system is modeled considering the air as ideal ($R = 287\frac{J}{kgK}, k = 1.4$) and isothermal transformation.
The 4 valves are modeled as a single 4/3 proportional valve, since they are used in pairs. A scheme of the valve is shown in figure 4.4.

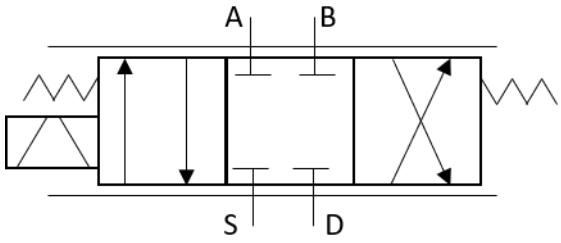


Figure 4.4: Schematic of the modeled pneumatic 4/3 valve

Referring to figure 4.4 the port S is the supply port, the port D is the discharge port, the ports A and B are connection ports for the actuator (A for the out-stroke and B for the in-stroke). The input command is in the range [-1 1].
A value equal to 1 corresponds to the position in which the port S is completely connected to the port A and the port D is completely connected to the port B.
A value equal to -1 corresponds to the position in which the port S is completely connected to the port B and the port D is completely connected to the port A.
A value equal to 0 corresponds to the central position in which all the ports are completely closed.
From now on when referring to pneumatic actuator the letter A corresponds to the rear chamber and letter B corresponds to the front chamber.
The equations are linearized around the working point defined by $p_A = p_B = p_s/2$ (where p_s is the supply pressure) and the reference condition (x=0).
So starting from the standard ISO equation for the flow rate in pneumatic valves and making some computations the equations for the pressure in the two chambers of the pneumatic actuator are:

$$\dot{p}_A = \frac{p_0}{2\rho_0} \cdot \frac{Kp_s u}{V_{A0}} - \frac{A_A \dot{x} p_s}{2V_{A0}} \tag{4.3}$$

$$\dot{p}_B = -\frac{p_0}{2\rho_0} \cdot \frac{Kp_s u}{V_{B0}} + \frac{A_B \dot{x} p_s}{2V_{B0}} \tag{4.4}$$

Where:

- p_0 is the air pressure of the environment
- ρ_0 is the air density of the environment
- $K = \frac{C_{eq}\rho_0}{1-b}$
- C_{eq} is the equivalent conductance of the valve considered as directly proportional to u
- b is the critical pressure ratio of the valve
- p_s is the supply pressure
- u is the actuation command

- V_{A0} is the volume of the rear chamber of the pneumatic cylinder in the reference condition

- V_{B0} is the volume of the front chamber of the pneumatic cylinder in the reference condition

- A_A is the rear area of the piston

- A_B is the front area of the piston

Collecting all the constant terms the equations (4.3) and (4.4) becomes:

$$\dot{p_A} = -C_A\dot{x} + K_A u \tag{4.5}$$
$$\dot{p_B} = C_B\dot{x} - K_B u \tag{4.6}$$

So, finally, the equation for the force is (the subscript P stands for piston):

$$F = p_A A_A - p_B A_B - b_p\dot{x} - M_P\ddot{x} \tag{4.7}$$

Now applying the Laplace transform of the equations (4.1) (4.2) (4.5) (4.6) (4.7) it is possible to retrieve the transfer functions between the output u and the input θ ($G_{u\theta}$) and between the first output θ and the input x ($G_{\theta x}$):

$$G_{u\theta} = \frac{(K_A + A_A + K_B A_B)s}{(M_C+M_P)ls^4 + (b_C+b_P)ls^3 - (m+M_C+M_P)gs^2 + (A_A C_A + A_B C_B)ls^2 - (b_C+b_P)gs - (A_A C_A + A_B C_B)g} \cdot \frac{ls^2 - g}{s^2} \tag{4.8}$$

$$G_{\theta x} = \frac{ls^2 - g}{s^2} \tag{4.9}$$

In this way the two transfer functions can be put in series as depicted in figure 4.1.
The numerical values of the variables are shown in the table 4.1.

Numerical data		
l	Pendulum length	$0.5m$
m	Concentrated pendulum mass	$0.2Kg$
M_C	Cart's mass	$1.8Kg$
M_P	Piston's mass	$0.4Kg$
b	Friction coefficient	$20.83N\frac{s}{m}$
A_A	Area of the rear chamber of the piston	$2.01 \cdot 10^{-4}m^2$
A_B	Area of the front chamber of the piston	$1.73 \cdot 10^{-4}m^2$
C_A	$\frac{A_A p_s}{2V_{A0}}$	1200000
C_B	$\frac{A_B p_s}{2V_{B0}}$	120000
K_A	$\frac{p_0}{2\rho_0} \cdot \frac{Kp_s}{V_{A0}}$	2177000
K_B	$\frac{p_0}{2\rho_0} \cdot \frac{Kp_s}{V_{B0}}$	2534000

Table 4.1: Experimental numerical values of the model's variables

4.1.2 PID controllers

From figure 4.1 the two controllers are inserted in the control structure in such a way that the first controller gives the θ_{set} to the second controller in depending of the difference of the actual position of the cart and the setting position. The second controller gives the input command to the plant in depending of the difference between the actual angle of the pendulum and the angle given from the first controller.
This choice of design can be better understood with the following example.
If, for example, the cart is in 0mm and we want to move it to 2mm, the first thing to do is to move the cart backward in order to move the pendulum with a certain negative angle. Then we have to move the cart forward to 2mm to re-stabilize the pendulum. So in this way we can move the cart in the wanted position.
The general transfer function of a PID controller is the following one:

$$C = k_p + k_i + \frac{k_d s}{T_f s + 1} \tag{4.10}$$

And considering the (4.10) the transfer functions of the two controllers are:

$$C_x = -0.000802 - \frac{4.68 \cdot 10^{-6}}{s} - \frac{0.0306s}{0.00875s + 1} \tag{4.11}$$

$$C_\theta = 2.09 + \frac{6.31}{s} + \frac{0.103s}{0.000875s + 1} \tag{4.12}$$

The (4.11) is the transfer function of the first controller or the cart controller and the (4.12) is the transfer function of the second controller or the pendulum controller. The two transfer functions are provided by [6].
The response is shown in figure 4.5 with x_{set} as a 0.05 step with step time at 1s.

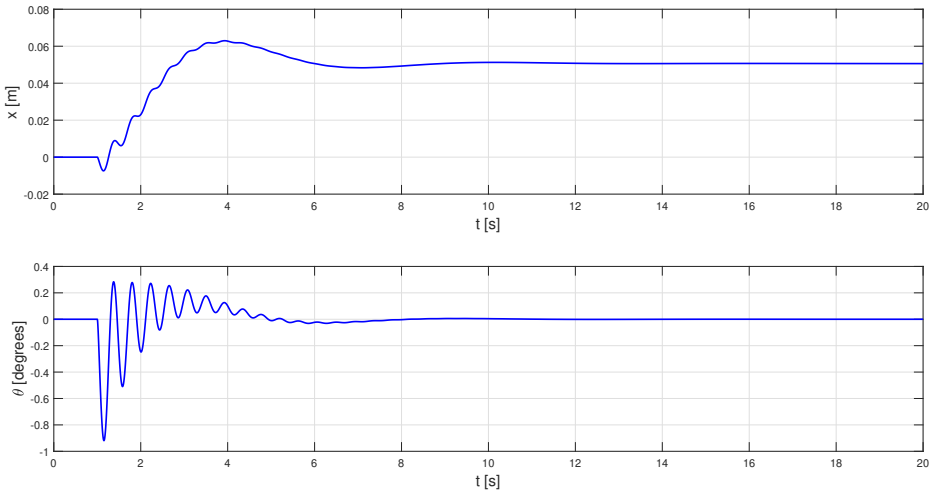


Figure 4.5: Step response with PID controllers

4.2 PD+I linear fuzzy controllers

The fuzzy controllers take as input the error, its derivative and its integral. According to [2] I choose a a PD+I fuzzy controller and not PID fuzzy controller because the second type requires three premise inputs and, if we make it simple, 3 linguistic terms per input. So the complete rule base consists of $3^3 = 27$ rules. It is difficult to set all the rule base and also to set the integral action. So a more simple solution is to combine a PD fuzzy controller (FPD) and a crisp integral action. This structure is shown in figure 4.6.

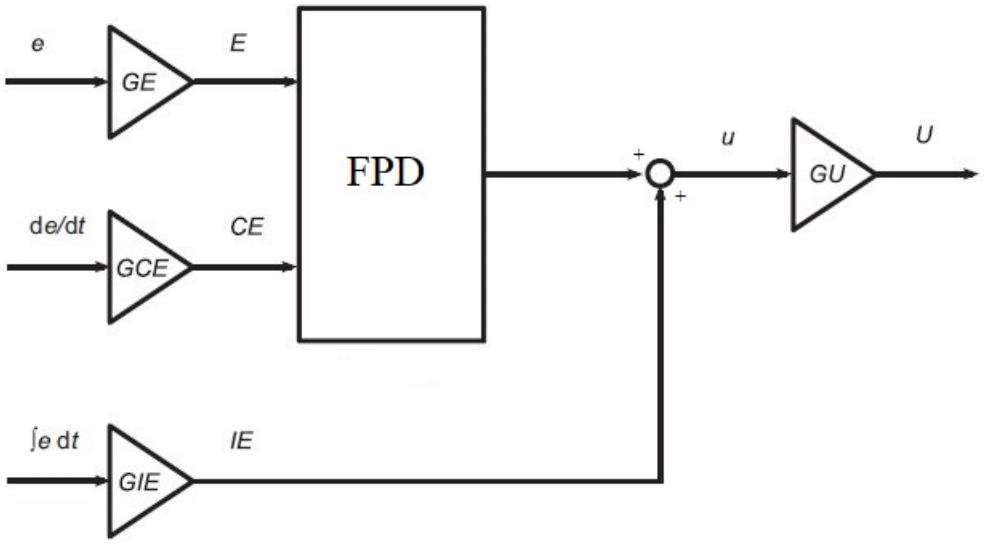


Figure 4.6: Block scheme of a PD+I fuzzy controller

According to figure 4.6 the general transfer function is:

$$u(t) = \left[f(GE \cdot e(t), GCE \cdot \dot{e}(t)) + GIE \cdot \int_0^t e(\tau) d\tau \right] \cdot GU \tag{4.13}$$

The function f is the function of the fuzzy controller. If it is linear and equal to the sum function it becomes:

$$f(GE \cdot e(t), GCE \cdot \dot{e}(t)) \approx GE \cdot e(t) + GCE \cdot \dot{e}(t) \tag{4.14}$$

This function can be obtained by setting the product as AND method and as Implication method, the probabilistic or as OR method, the COG as defuzzification method.
So, the equation of a linear fuzzy controller is:

$$u(t) = \left[GE \cdot e(t) + GCE \cdot \dot{e}(t) + GIE \cdot \int_0^t e(\tau) d\tau \right] \cdot GU \tag{4.15}$$

Now comparing the (4.15) with the (4.10), a set of equation can be found to get the gains of the fuzzy controller.

$$GU = \frac{K_p}{GE} \tag{4.16}$$

$$GCE = \frac{K_d}{GU} \tag{4.17}$$

$$GIE = \frac{K_i}{GU} \tag{4.18}$$

The free parameter is GE that is chosen considering the maximum error of the input variable (e_{max}) and the maximum value of its fuzzified value (f_{max}):

$$GE = \frac{f_{max}}{e_{max}} \tag{4.19}$$

So considering that the maximum error for the linear position is $0.25mm$ and for the angle is $20° = 0.35rad$ the corresponding gains are the following. The subscript "a" stands for angle and is inserted in the gains for the pendulum controller; the subscript "l" stands for linear and is inserted in the gains for the cart controller.

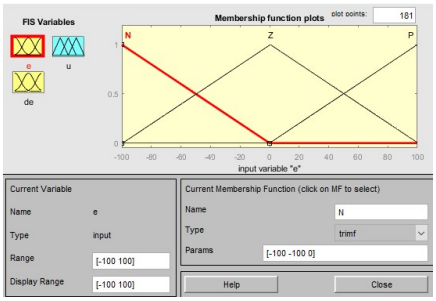
$$GE_a = 286.479 \qquad GE_l = 400$$
$$GU_a = 0.0073 \qquad GU_l = -2 \cdot 10^{-6}$$
$$GCE_a = 14.12 \qquad GCE_l = 1.53 \cdot 10^4$$
$$GIE_a = 865 \qquad GIE_l = 2.33$$

To set the fuzzy controller I used the "Fuzzy Logic Toolbox" of MatLab®.
As explained in [2] a very common first choice of the structure of the fuzzy controller is to set:

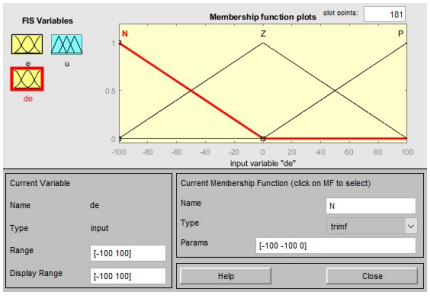
- For the input variable $e(t)$ the range [-100 100] and three triangular functions: "Negative" (N) centered in -100, "Zero" (Z) centered in 0, "Positive" (P) centered in 100. Figure 4.7(a)

- For the input variable $\dot{e}(t)$ the range [-100 100] and three triangular functions: "Negative" (N) centered in -100, "Zero" (Z) centered in 0, "Positive" (P) centered in 100. Figure 4.7(b)

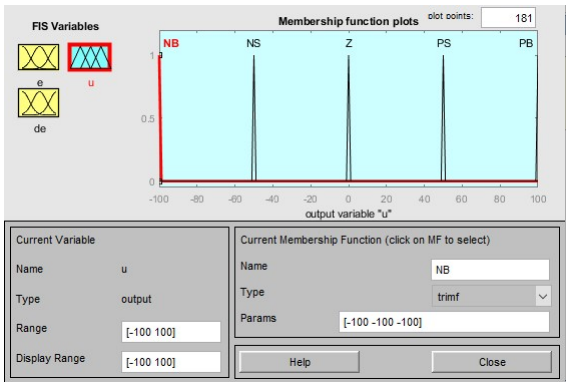
- For the output variable the range [-100 100] and five singletons: "Negative big" (NB) centered in -100, "Negative small" (NS) centered in -50, "Zero" (Z) centered in 0, "Positive small" (PS) centered in 50, "Positive big" centered in 100. Figure 4.7(c)



(a) Set membership functions of $e(t)$



(b) Set membership functions of $\dot{e}(t)$



(c) Set membership functions of $u(t)$

Figure 4.7: Set membership functions of $e(t)$ (a), $\dot{e}(t)$ (b) and $u(t)$ (c)

The main criterion to write the rules is to set the control action as big as is $e(t)$ and then as big as is $\dot{e}(t)$ accordingly to its sign. Following this criterion I write the rule base. It is shown in matrix form in the following table where in red are the linguistic value of the input and in blue the linguistic value of the output.

		$\dot{e}(t)$		
		N	Z	P
$e(t)$	N	NB	NS	Z
	Z	NS	Z	PS
	P	Z	PS	PB

Table 4.2: Rule table of the fuzzy PD+I controller

The table 4.2 corresponds to the following linguistic rules:

1. IF e is N AND $\dot{e}$ is N => u is NB

2. IF e is N AND $\dot{e}$ is Z => u is NS

3. IF e is N AND $\dot{e}$ is P => u is Z

4. IF e is Z AND $\dot{e}$ is N => u is NS

5. IF e is Z AND $\dot{e}$ is Z => u is Z

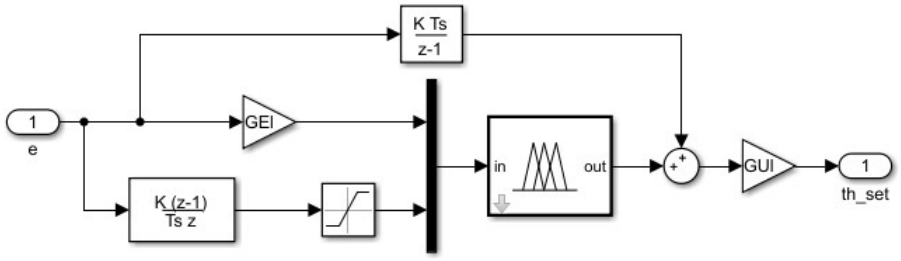
6. IF e is Z AND $\dot{e}$ is P => u is PS

7. IF e is P AND $\dot{e}$ is N => u is Z

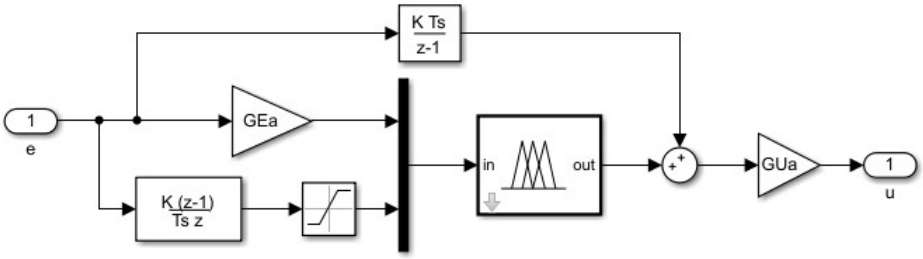
8. IF e is P AND $\dot{e}$ is Z => u is PS

9. IF e is P AND $\dot{e}$ is P => u is PB

Finally, the two controllers have to be inserted in the Simulink® scheme. In order to do that I have created two subsystems as shown in figure 4.8(a) for the cart controller and in figure 4.8(b) for the pendulum controller.
The integrator and derivative are discretized (since the fuzzy controller works in discrete time) and so the z-transform is used. The sampling time T_s is $0.01s$.



(a) Control structure of the cart controller



(b) Control structure of the pendulum controller

Figure 4.8: Control structure of the cart controller (a) and of the pendulum controller (b)

In figure 4.8 there are saturation blocks after the derivative block to keep the input variable in the range [-100 100]. This choice makes sense in reality because if the velocity is too high then it is too negative or to positive and so the fuzzy function gives a value of 1.
The overall block scheme is the same of figure 4.1, but this time instead the PID controllers there are the two subsystems with the fuzzy controllers (figure 4.9).

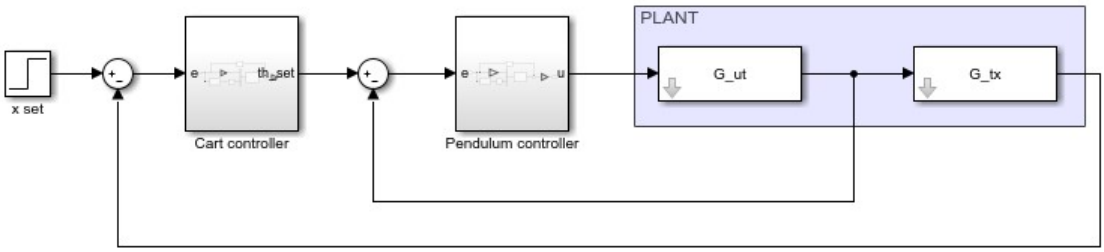
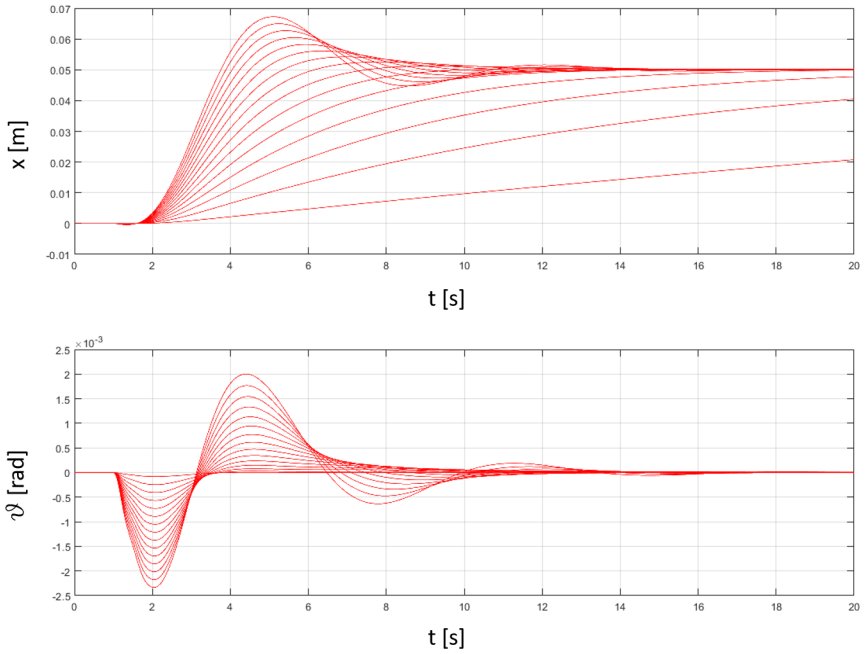
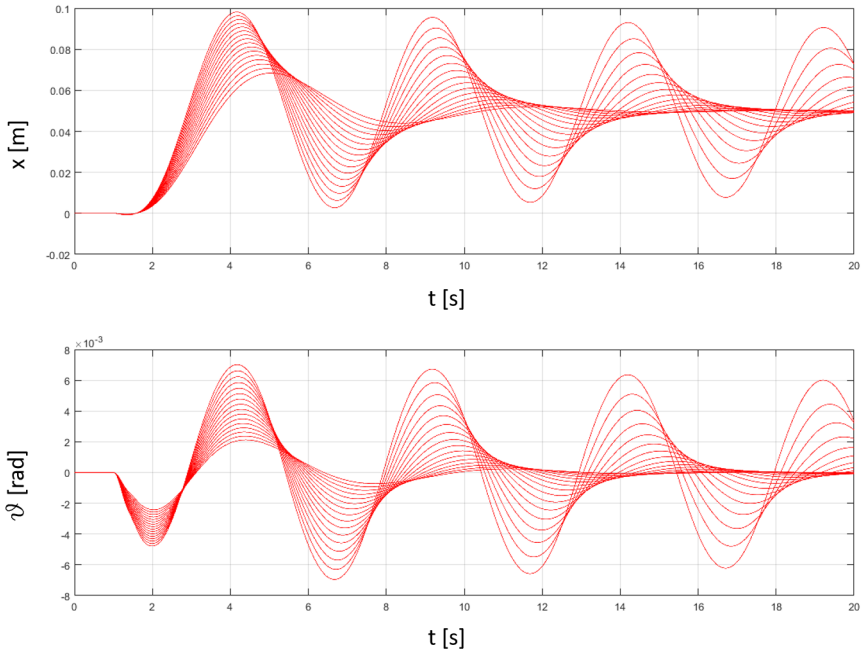


Figure 4.9: Overall control structure with fuzzy controllers

In figure 4.9 the x_{set} is a 0.05 step with step time at 1s. With this configuration the response compared to the response with the PID controllers results too slow.
I noted that increasing the gain GU_l and recomputing the other gains accordingly, the response becomes faster. So another gain k_l can be considered to increase the gain GU_l. In figure 4.10(a) is shown the step responses for different values of k in the range [1 30] with a step of 2 and in figure 4.10(b) the range is [30 60] with the same step.



(a) Step response with $k = [1 \quad 30]$



(b) Step response with $k = [30 \quad 60]$

Figure 4.10: Step response with $k = [1 \quad 30]$ (a) and with $k = [30 \quad 60]$ (b)

So, considering figure 4.10, I choose a value of k of 15.
With this value the step response compared with the one shown in figure 4.5 is shown in figure 4.11.

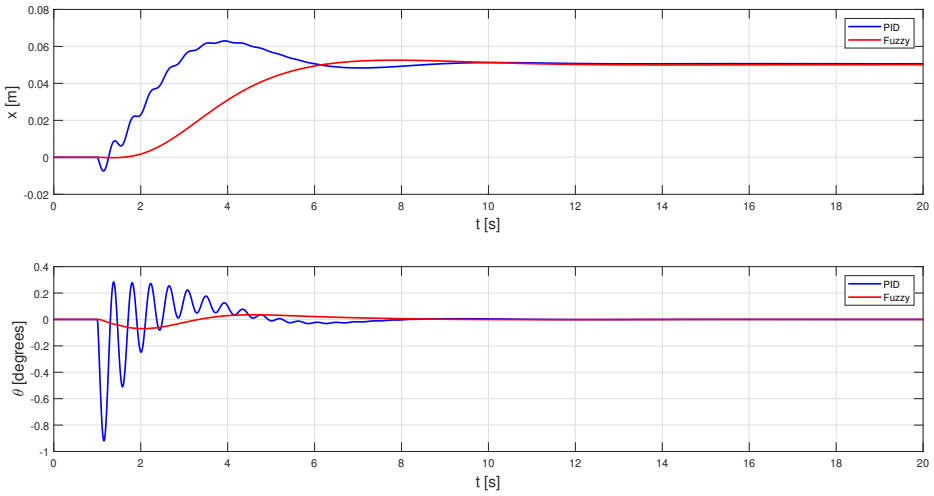


Figure 4.11: Step response with PID controllers compared with the one with fuzzy controllers

Considering the step response of the cart the obtained performances are (performances with PID controllers in brackets):

- Max overshoot: 5.1% (26%)
- Rising time: 6.15s (2.8s)
- Settling time ±10% : 5.24s (5.3s)

With the fuzzy controllers the system reaches the setting signal a little bit slower (consider both settling time and rising time) but the overshoot is significant lower. So I keep this control structure for the next step of design.

4.3 Stability analysis

The stability of this system can be verified by making the transfer function of the overall system and computing its poles. In order to compute it I will consider the block scheme in figure 4.9.
First of all I make the transfer function of the controllers. From [7] I have to find an analytical approximation of the fuzzy controllers. In this case it is easy because I choose

the fuzzy controllers as PID-like, so their transfer functions is the same of a PID controller with Laplace transform.

$$C_\theta(s) = 0.0073 \cdot \left(286.479 + \frac{865}{s} + \frac{14.12s}{0.01ts + 1}\right) \tag{4.20}$$

$$C_x(s) = -30 \cdot 10^{-6} \cdot \left(400 + \frac{0.155}{s} + \frac{1020s}{0.01s + 1}\right) \tag{4.21}$$

Since the system is linear I can use the block's algebra.
Starting from the internal loop, its transfer function is:

$$G_1 = \frac{C_\theta(s) \cdot G_{u\theta}(s)}{1 + C_\theta(s) \cdot G_{u\theta}(s)} \tag{4.22}$$

So the internal loop can be replaced with the transfer function G_1 (4.22).
Now the overall transfer function is simply a loop transfer function:

$$G_{tot} = \frac{C_x(s) \cdot G_1(s) \cdot G_{\theta x}(s)}{1 + C_x(s) \cdot G_1(s) \cdot G_{\theta x}(s)} \tag{4.23}$$

The poles of G_{tot} are:

$$\begin{aligned} p_1 &= -101 + 11.46i \\ p_2 &= -101 - 11.46i \\ p_3 &= -0.645 + 14.73i \\ p_4 &= -0.645 - 14.73i \\ p_5 &= -5.2155 \\ p_6 &= -0.271 + 1.017i \\ p_7 &= -0.271 - 1.017i \\ p_8 &= -0.46 \end{aligned}$$

No poles have real part greater than zero, so the system is stable.
Considering the Nichols plot of the open loop transfer function $L(s)$ I can measure the gain margin and the phase margin and also I can have another proof of the stability.
The open loop transfer function is:

$$L(s) = C_x(s) \cdot G_1(s) \cdot C_{\theta x}(s) \tag{4.24}$$

The Nichols plot of $L(s)$ is shown in figure 4.12.

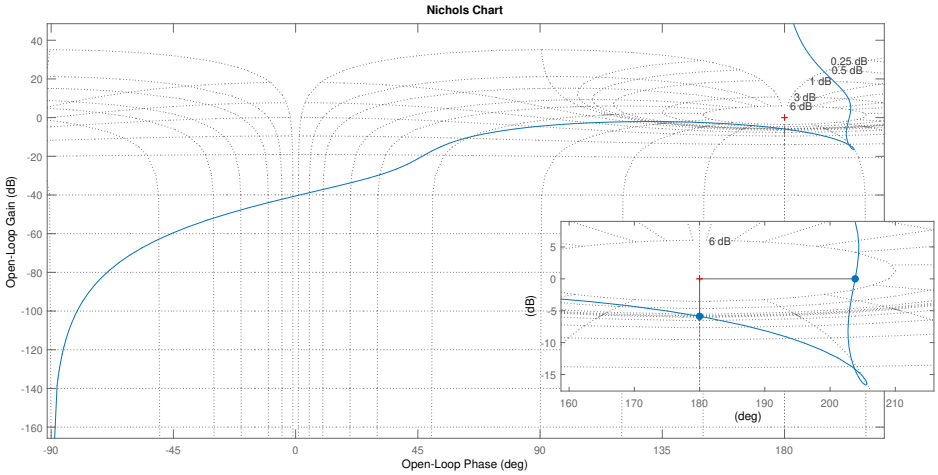


Figure 4.12: Nichols plot of the open loop transfer function

From figure 4.12 the system seems to be stable and the gain margin is $5.88dB$ and the phase margin is 23.85°.

Chapter 5

Nonlinear model of the plant

In order to make a more realistic simulation and to follow the design procedure explained in [2] I rebuilt the model of the plant without the linear approximations. The overall system is divided essentially into 3 subsystems as shown in figure 5.1 in which the pneumatic plant consists of the model of the pneumatic actuator and of the model of the 2/2 proportional electrovalves; the mechanical plant consists of the model of the cart and the pendulum and the control action consists of the 2 fuzzy controllers and it will be explained in the next chapter.

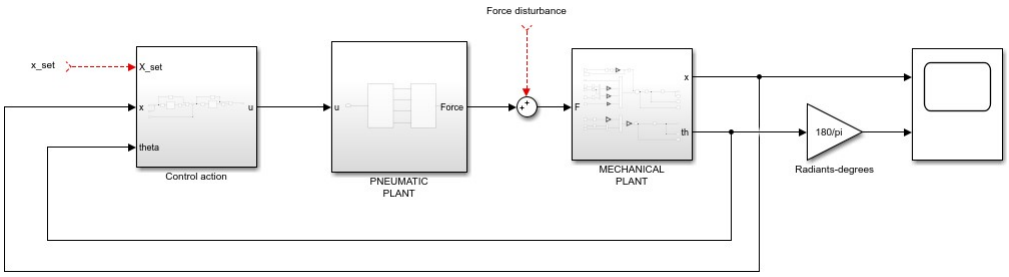


Figure 5.1: Block scheme of the overall nonlinear system

5.1 Mechanical plant

The mechanical plant is the same analyzed in the previous chapter, but this time without the linear approximation.
A scheme is shown in figure 5.2. It is graphically different from the one in the previous chapter, but the positive direction of the generalized coordinates and the geometry are, obviously, the same.

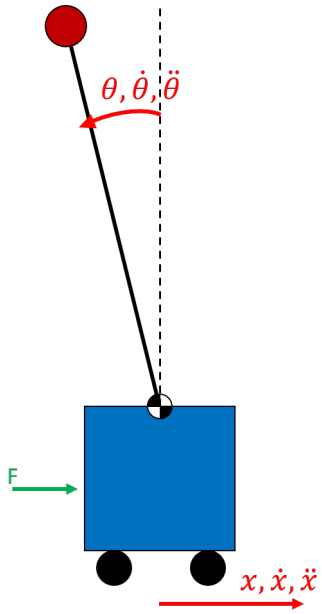


Figure 5.2: Scheme of the inverse pendulum

Now the system can be considered as composed of two bodies (the cart and the pendulum) connected through and ideal rotative joint. The free body diagram of the cart is shown in figure 5.3.

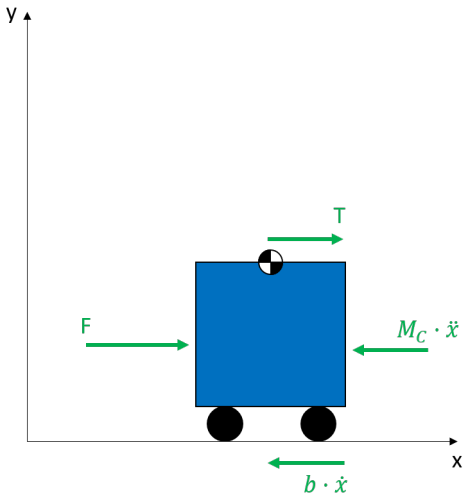


Figure 5.3: Free body diagram of the cart

From figure 5.3 the equilibrium equation is:

$$F - M_C \cdot \ddot{x} - b \cdot \dot{x} + T = 0 \tag{5.1}$$

The free body diagram of the pendulum is shown in figure 5.4 where as in the linear case the bar is considered as ideal and its moment of inertia is neglected.

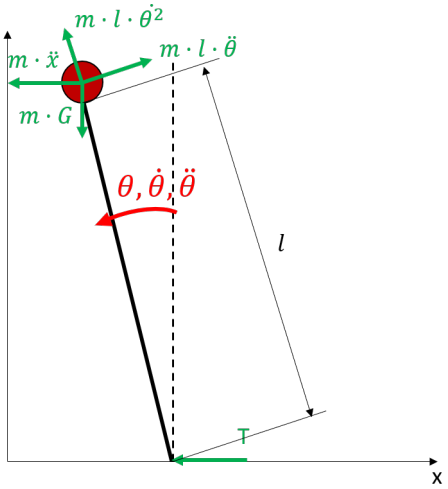


Figure 5.4: Free body diagram of the pendulum

From figure 5.4 the equilibrium equations are:

$$m \cdot l \cdot \ddot{\theta} \cdot \cos(\theta) - m \cdot \ddot{x} - m \cdot l \cdot \dot{\theta}^2 \cdot \sin(\theta) - T = 0 \tag{5.2}$$

$$m \cdot G \cdot l \cdot \sin(\theta) + m \cdot \ddot{x} \cdot l \cos(\theta) - m \cdot l^2 \cdot \ddot{\theta} = 0 \tag{5.3}$$

Making some computations the four equations above can be merged in order to obtain the linear and angular accelerations (the piston's mass M_P is considered as an additive mass of the cart):

$$\ddot{x} = (F - m \cdot l \cdot \dot{\theta}^2 \cdot \sin(\theta) + m \cdot G \cdot \sin(\theta) \cdot \cos(\theta) - b \cdot \dot{x}) \cdot \frac{1}{m + M_C + M_P - m \cdot \cos^2(\theta)} \tag{5.4}$$

$$\ddot{\theta} = (G \cdot \sin(\theta) + \ddot{x} \cos(\theta)) \cdot \frac{1}{l} \tag{5.5}$$

So the Simulink® block scheme for the equation (5.4) is shown in figure 5.5 and the Simulink® block scheme for the equation (5.5) is shown in figure 5.6.

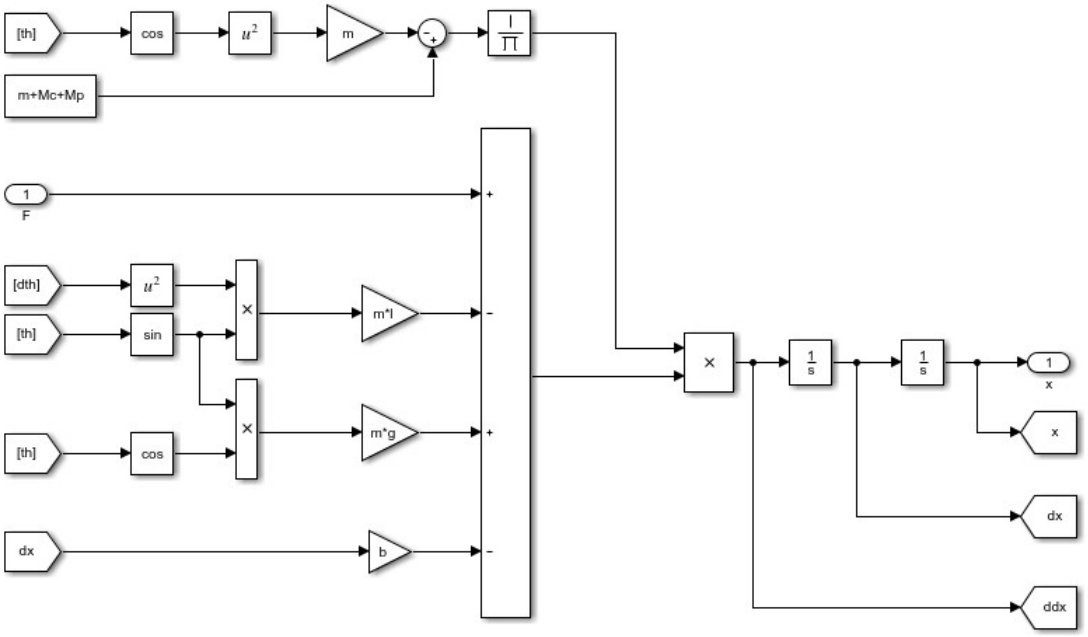


Figure 5.5: Simulink® block scheme for the equation (5.4)

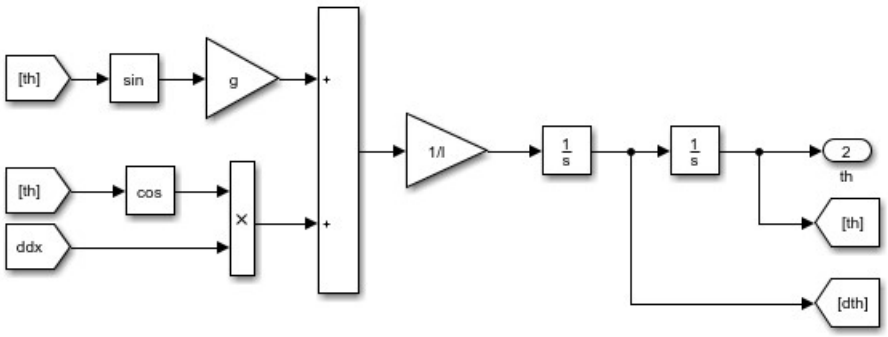


Figure 5.6: Simulink® block scheme for the equation (5.5)

The block schemes in figures 5.5 and 5.6 are both inserted in the subsystem "Mechanical Plant" in figure 5.1.

5.2 Pneumatic plant

The pneumatic plant can be divided into two subplants: the pneumatic actuator and the valves. In fact entering inside the subsystem in figure 5.1 called "Pneumatic plant" 2 subsystems are visible (figure 5.7). This modelling choice makes sense in reality because the valves impose a flow rate and in depending of this flow rate the pneumatic actuator gives a force to the cart of the inverse pendulum.
Now I will explain in two different sections the two subsystems.

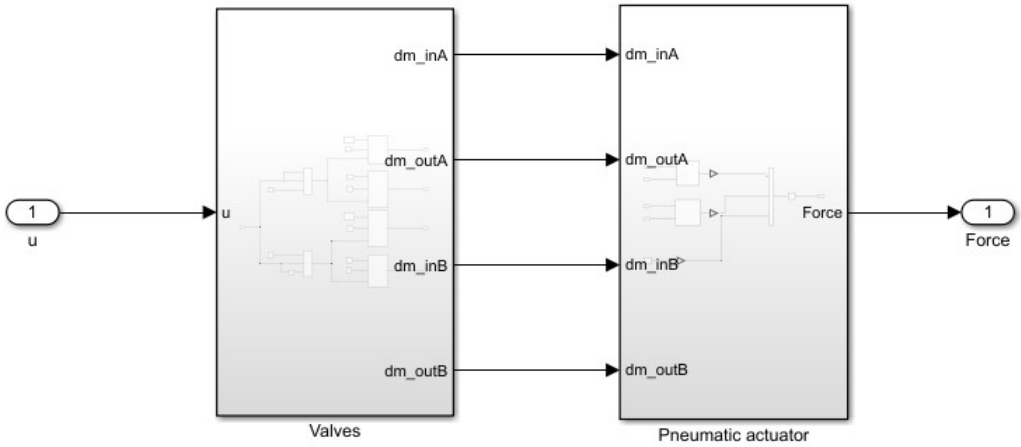


Figure 5.7: The sbsystems that compose the pneumatic plant

5.2.1 Nonlinear model of the pneumatic actuator

The pneumatic actuator is modeled without friction because it is considered in the model of the mechanical plant.
The starting point is the free body diagram of the piston and rod of the actuator. It is shown in figure 5.8.

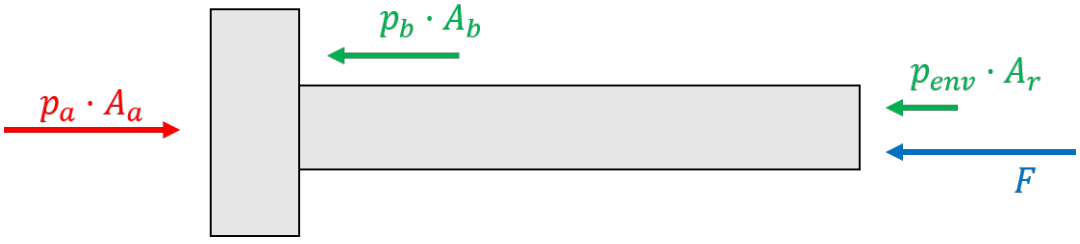


Figure 5.8: Free body diagram of the piston and rod of the actuator

So, considering figure 5.8, the equilibrium equation is:

$$F = p_a \cdot A_a - p_b \cdot A_b - p_{env} \cdot A_r \tag{5.6}$$

Now a pressure model is needed.
The flow rate that enters a chamber of the actuator is simply the time derivative of the mass:

$$\dot{m} = \frac{dm}{dt} \tag{5.7}$$

Considering the air as an ideal gas the mass can be expressed as following:

$$m = \frac{p \cdot V}{R \cdot T} \tag{5.8}$$

Where $R = 287.05 \frac{J}{kgK}$.
Hence substituting the (5.8) in the (5.7) and considering the temperature constant (isothermal transformation) at the environment temperature ($293K$):

$$\dot{m} = \frac{\dot{p} \cdot V}{R \cdot T} + \frac{p \cdot \dot{V}}{R \cdot T} \tag{5.9}$$

The flow rate in the equation (5.9) is a net flow rate. Conventionally for open systems the entering flow rate has a positive sign and the exiting flow rate has a negative sign. Following this one the flow rate balance in the chambers of the actuator is:

$$Chamber \quad A: \quad \dot{m}_{INa} - \dot{m}_{OUTa} = \frac{\dot{p_a} \cdot V_a}{R \cdot T} + \frac{p_a \cdot \dot{V_a}}{R \cdot T} \tag{5.10}$$

$$Chamber \quad B: \quad \dot{m}_{INb} - \dot{m}_{OUTb} = \frac{\dot{p_b} \cdot V_b}{R \cdot T} + \frac{p_b \cdot \dot{V_b}}{R \cdot T} \tag{5.11}$$

The chambers' volume can be computed by the following equations considering that a value of x equal to zero corresponds to the position of the piston in the middle of its stroke.

$$V_a = V_{A0} + A_a \cdot (\frac{1}{2}c + x) \tag{5.12}$$

$$V_a = V_{B0} + A_b \cdot (\frac{1}{2}c - x) \tag{5.13}$$

$$\dot{V_a} = A_a \cdot \dot{x} \tag{5.14}$$

$$\dot{V_b} = A_b \cdot \dot{x} \tag{5.15}$$

Where V_{A0} and V_{B0} are the death volumes of the chambers A and B respectively ($1cm^3$ for both) and c is the stroke of the actuator ($0.5m$).
Now making some computations the finally equations of the pressure of the two chambers of the actuator are:

$$\dot{p_a} = \frac{RT \cdot (\dot{m}_{INa} - \dot{m}_{OUTa})}{V_{A0} + A_a \cdot (\frac{1}{2}c + x)} - \frac{p_a A_a \dot{x}}{V_{A0} + A_a \cdot (\frac{1}{2}c + x)} \tag{5.16}$$

$$\dot{p_b} = \frac{RT \cdot (\dot{m}_{INb} - \dot{m}_{OUTb})}{V_{B0} + A_b \cdot (\frac{1}{2}c - x)} + \frac{p_b A_b \dot{x}}{V_{B0} + A_b \cdot (\frac{1}{2}c - x)} \tag{5.17}$$

Considering the equations (5.16) and (5.17) the block schemes are the ones shown in figure 5.9 and 5.10.

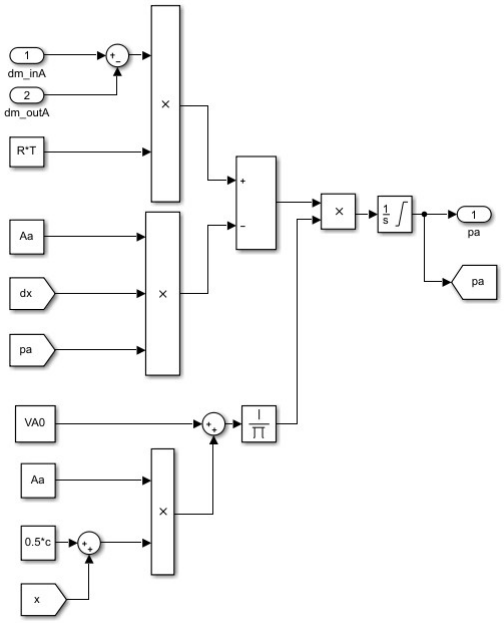


Figure 5.9: Block scheme of the pressure model for chamber A

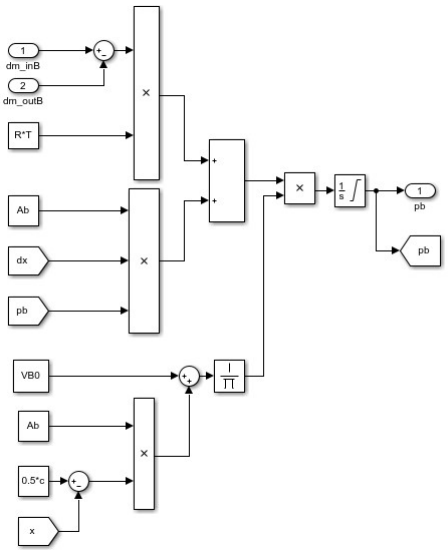


Figure 5.10: Block scheme of the pressure model for chamber B

The integrators of the pressures are limited in the range $[p_{env} \quad p_{supply}]$ to avoid to go

over the limit of the system.
Now two subsystems containing the models of the pressures can be built. Regarding the end strokes of the actuator they can be modeled as a spring with an infinity stiffness. In Simulink® an infinity value corresponds to a very huge number and it can be $1 \cdot 10^{10}$. The block scheme of the end strokes modeled as springs is shown in figure 5.11.

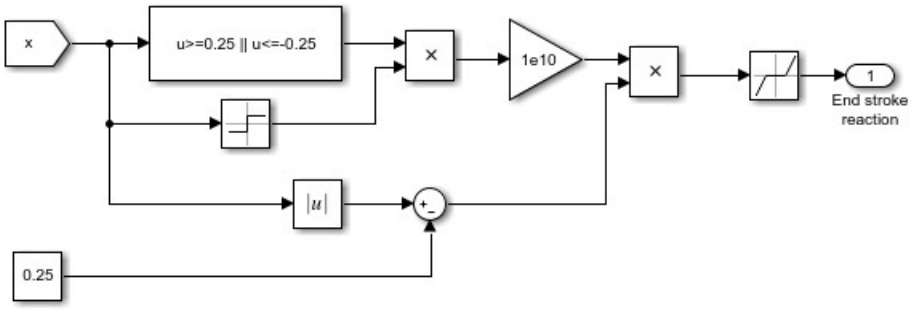


Figure 5.11: Block scheme of the end strokes of the actuator

The implemented logic is: if the piston reaches the stroke limits ($x = 0.25m$ *or* $x = -0.25m$) it goes against the end stroke that is an ideal spring with very huge stiffness. The reaction force using the Hooke's equation: $F = k \cdot \Delta x$. There are also a sign block to distinguish the left end stroke from the right one and a dead zone to neglect very low values which are computational machine error. The resulting force is negative summed to the force given by the actuator.
Now is possible to collect all the built subsystems and make the model of the pneumatic actuator. It is shown in figure 5.12. Also in this case there is a dead zone to avoid numerical machine errors.

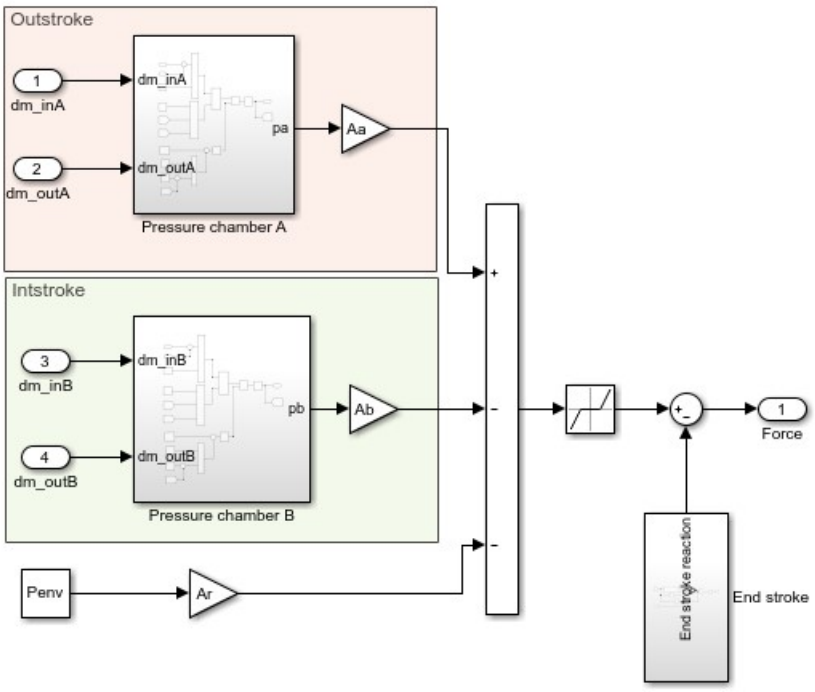


Figure 5.12: Block scheme of the pneumatic actuator

5.2.2 Model of the 2/2 NC proportional electrovalve

Like the linear model also in this case the valve is considered to be a zero-order component and so its conductance is directly proportional to the input command u. This modelling choice is not so far from reality, in fact the dynamic of the valve is very fast and it can be neglected.
A scheme of this valve is shown in figure 5.13

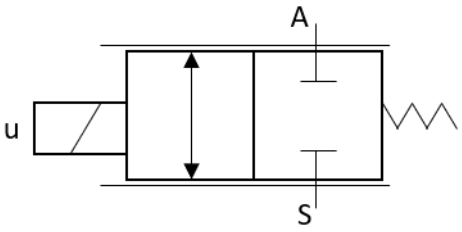


Figure 5.13: Scheme of a 2/2 NC proportional electrovalve

The range of the input command u per each valve is [0 1]. When it is zero the valve is in the closed position and when it is 1 the valve is in the open position. For values between 0 and 1 the valve is in an intermediate position.

To compute the flow rate the equation from standard ISO 6358 is used.

$$\begin{cases} \dot{m} = C \cdot u \cdot \rho_{ANR} \cdot p_{UP} & if \quad r \leq b \\ \dot{m} = C \cdot u \cdot \rho_{ANR} \cdot p_{UP} \cdot \sqrt{1 - \left(\frac{r-b}{1-b}\right)^2} & if \quad r > b \end{cases} \tag{5.18}$$

Two cases must be considered because if the ratio between the downstream pressure and the upstream pressure (r) is less or equal to the critical ratio (0.3) the flow rate is constant and equal to the critical flow rate.
The Simulink® block scheme of this model is shown in figure 5.14. In the scheme there is a saturation block after the computation of the pressure ratio (r) to allow it to stay only in the range [0 1]. This is done to avoid machine computational errors and to not go over the limits of the system.

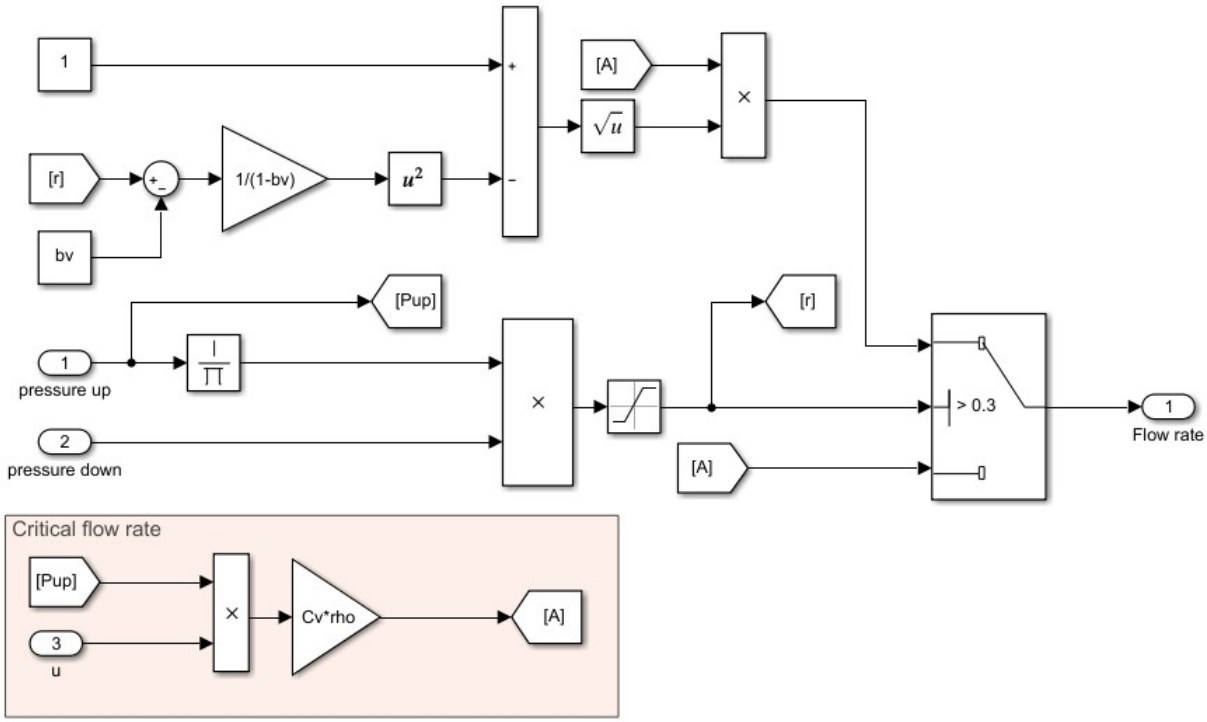


Figure 5.14: Simulink® block scheme of a 2/2 NC proportional electrovalve

The pneumatic plant there are 4 valves so I put the model of a single valve in a subsystem and I conneted the 4 resulting subsystems as shown in figure 5.15 in order to make the pneumatic circuit.

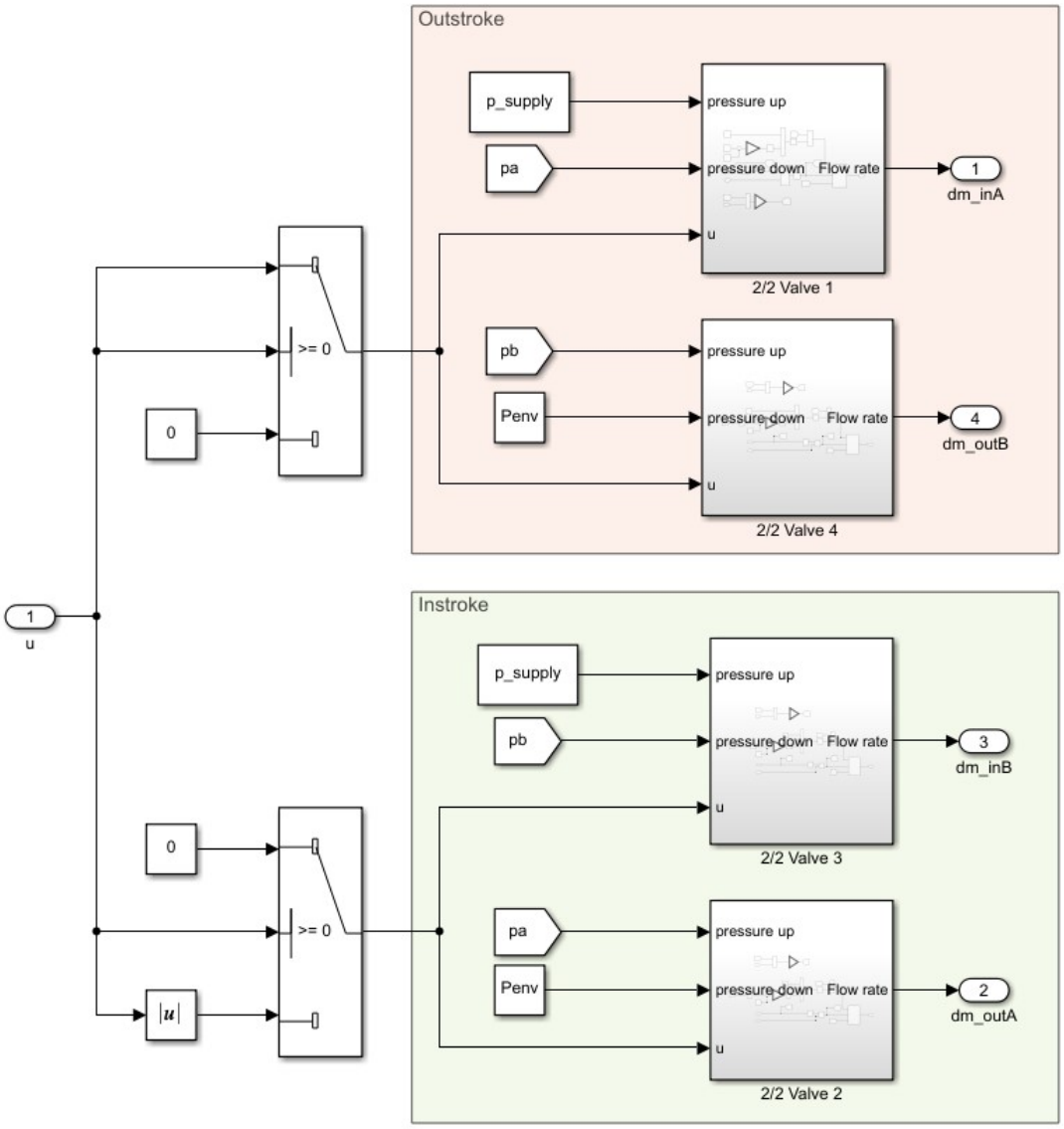


Figure 5.15: Simulink® block scheme of the pneumatic circuit

The implemented logic is very simple: if u is greater than zero the pneumatic actuator has to perform and outstroke and so the chamber A must be filled (Valve 1 open) and the chamber B must be discharged (Valve 4 open), the other valves must be closed; if u si lower than zero the situation is opposite, i.e. the pneumatic actuator has to perform an instroke and so chamber A must be discharged (Valve 2 open) and chamber B must be filled (Valve 3 open). So, according to this logic, the upstream pressure of the valve 1 and 3 is the supply pressure (p_{supply}) and the downstream pressure is the pressure in chambers A and B respectively; the upstream pressure of the valves 2 and 4 is the pressure in chambers A and B respectively and the downstream pressure is the pressure of the environment (p_{env}).

All the block scheme depicted in figure 5.15 constitutes the subsystem “Valve” shown in figure 5.7.

5.3 Model’s test and validation

In order to make sure that the model is built correctly and it works properly it needs to be validated. The validation consists on testing each subsystem and then the overall system. To test each subsystem I put a known input and I verified that the response is the expected one and the same for the complete model.
In the following I will explain in detail all of those tests.

5.3.1 Test of the mechanical plant

To test the mechanical plant I put as input a constant force ($0.1N$) and I made a plot of the linear position of the cart and the angle of the pendulum. The resulting behavior is shown in figure 5.16

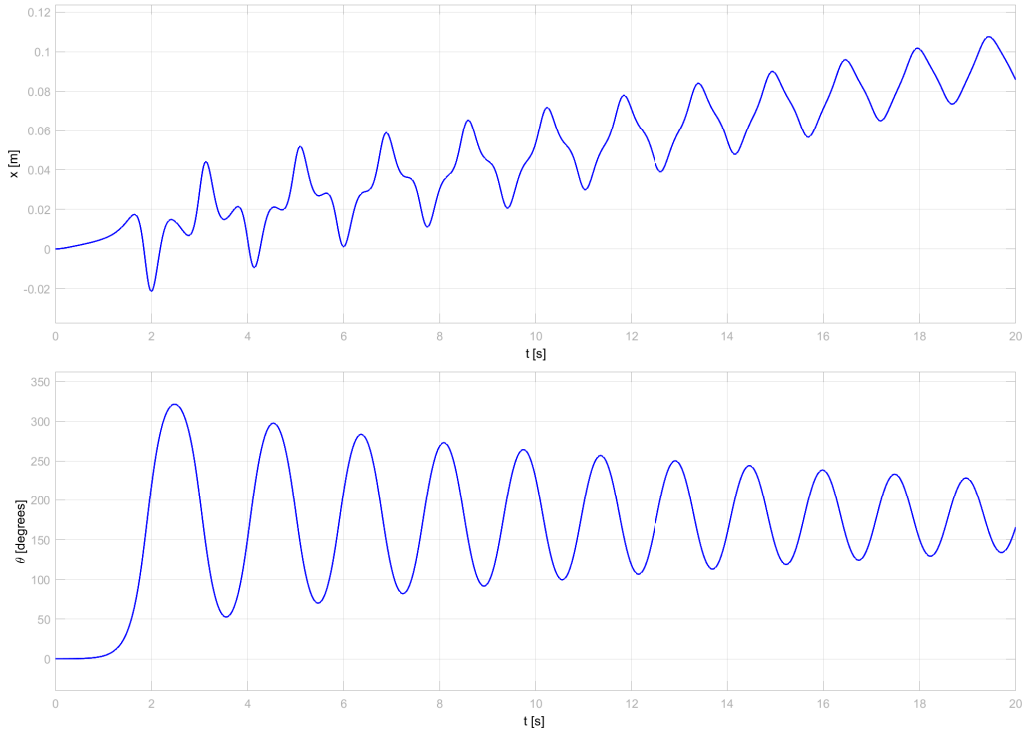


Figure 5.16: Plot of the response of the mechanical plant with a constant force of $0.1N$ as input

As expected the pendulum starts to oscillate around the stable equilibrium point that is 180° and the cart has some oscillations due to the oscillation of the pendulum but its

average position is increasing.

5.3.2 Test of the 2/2 NC proportional electrovalve

To test the valve I made three tests. In the first one I put as input a constant unit value that corresponds to the valve totally open. I obtain a constant flow rate as output as shown in figure 5.17. The upstream pressure and the downstream pressure are fixed and they are: $p_{up} = p_{supply} = 7bar; \quad p_{down} = 101325Pa.$

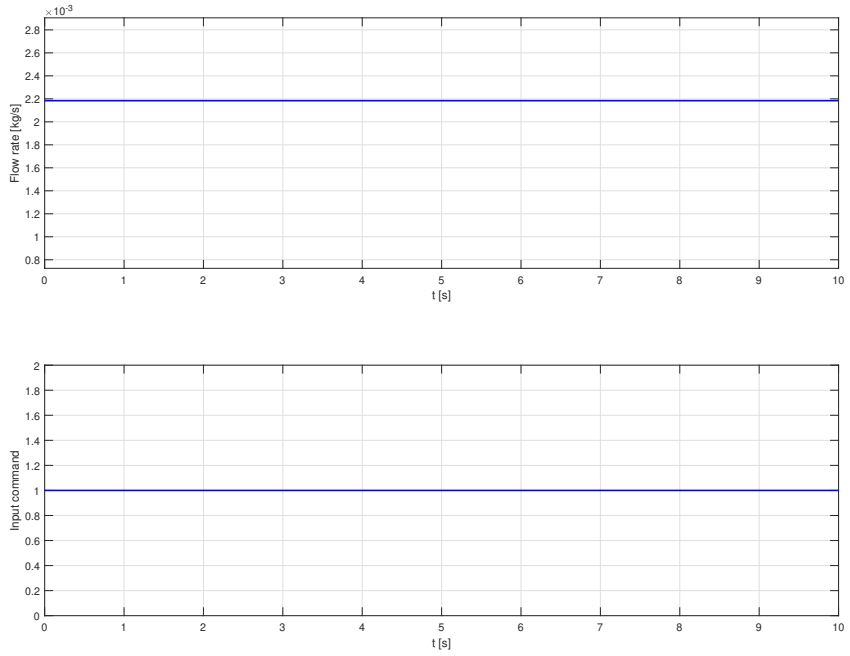


Figure 5.17: Plot of the flow rate behavior of the valve with a constant unit input

To verify if the obtained value is correct the flow rate equation can be used. In this case $r = \frac{p_{up}}{p_{down}} = 0.143$ so the flow rate is the critical one.

$$\dot{m} = C \cdot u \cdot \rho_{ANR} \cdot p_{UP} = 2.4 \cdot 10^{-9} \cdot 1 \cdot 1.3 \cdot 7 \cdot 10^5 = 0.0022kg/s \tag{5.19}$$

So according with the equation and the graph above the behavior is correct.
In the second test I put as input a ramp from 0 to 1 with slope 0.1 in order to reach the value 1 after 10 seconds. The behavior of the flow rate is shown in figure 5.18.

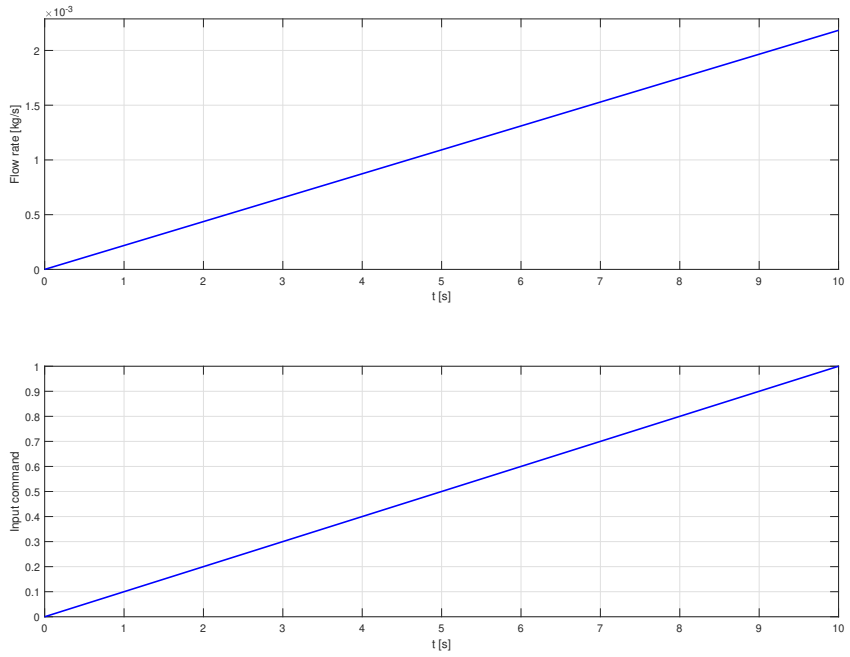


Figure 5.18: Plot of the flow rate behavior of the valve with a ramp input with 0.1 slope

As expected the flow rate starts from 0 and it reaches $0.0022 kg/s$ that is the value computed in (5.19) for the unit input.
The final test consists on computing the flow rate versus a varying of the pressure ratio. The test is made for 3 different upstream pressure and for a downstream pressure that varies from the environment pressure to the upstream pressure. The results is shown in figure 5.19.

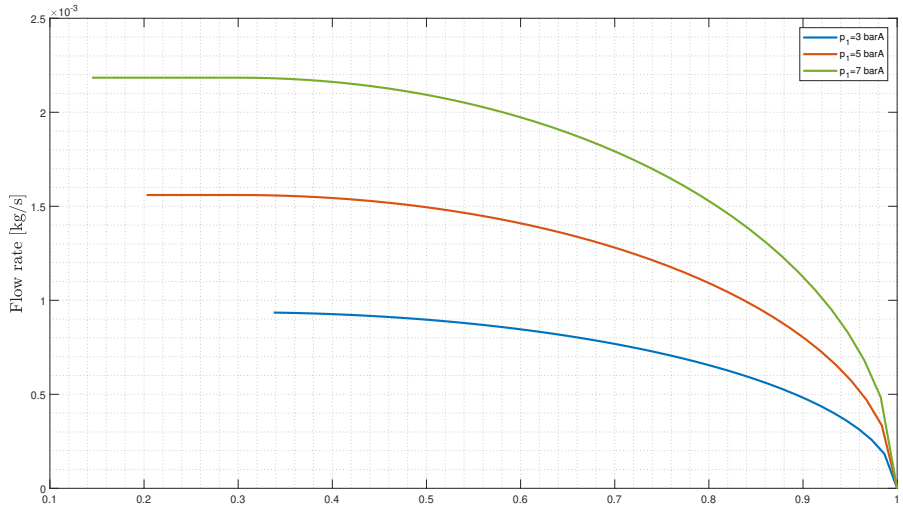


Figure 5.19: Plot of the flow rate behavior of the valve with different values of r

The graphs shows three characteristic curves of the flow rate in a valve. So taking into account the three tests the model can be considered valid.

5.3.3 Test of the pneumatic model

To test the pneumatic model I used the block scheme shown in figure 5.20.

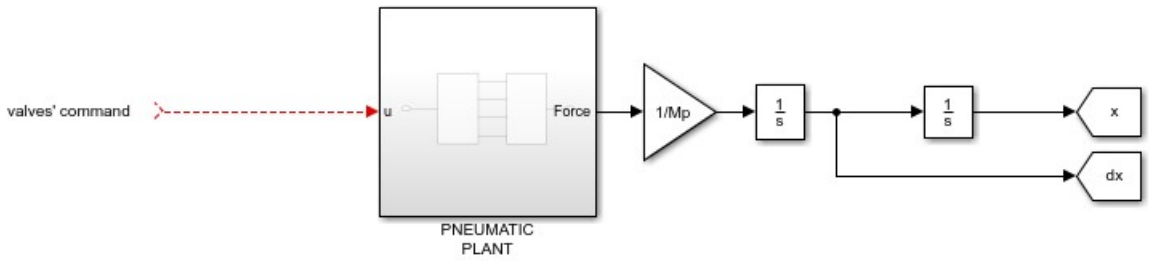


Figure 5.20: Block scheme for the test of the pneumatic model

The first test was done with the rod of the piston blocked ($x = 0$ and $\dot{x} = 0$) in order to keep constant the volume of the chambers of the actuator and to verify that the pressure model is working in a proper way. So, giving to the valves a constant command equal to 1 the resulting pressure behavior is shown in figure 5.21.

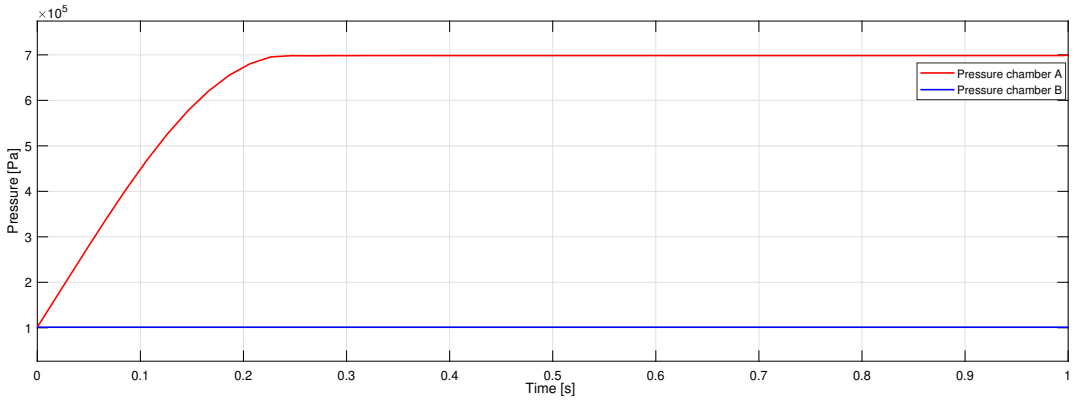


Figure 5.21: Test of the pressure model

The test can be considered valid because the behavior shown in figure 5.21 is the typical one of the filling a constant volume.
Another test is done with the rod of the piston free to move and giving two steps command to the valves (1 and -1). In this way the piston should move from its initial position (x=0) to its end stroke (x=0.25) and then, when the input signal i s-1 it should go to the other end stroke (x=-0.25). This behavior is shown in figure 5.22 where the resulting behavior is the expected one, so the model is valid.

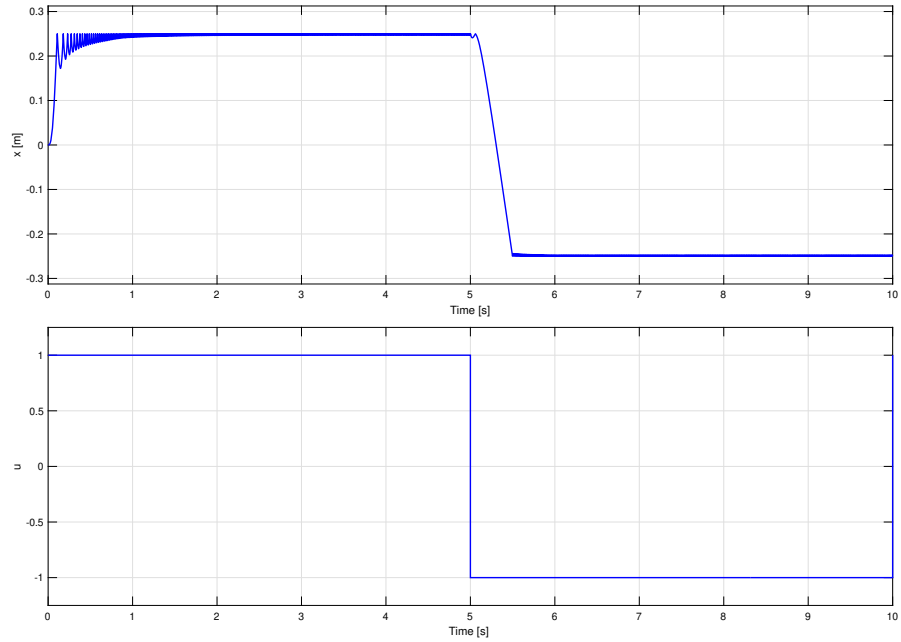


Figure 5.22: Test of the pneumatic model

In figure 5.22 there are some oscillations when the piston reaches the end stroke due to the fact that the model is frictionless.

5.3.4 Test of the complete model

The nonlinear model is tested setting as input a constant signal to the valves ($u = 0.5$). In this way the piston moves the cart to the endstroke ($x = 0.25$) and the pendulum starts oscillating and tends to stabilize in its stable position that is $\theta = 180°$. This behavior is shown in figure 5.23 that is the expected one.

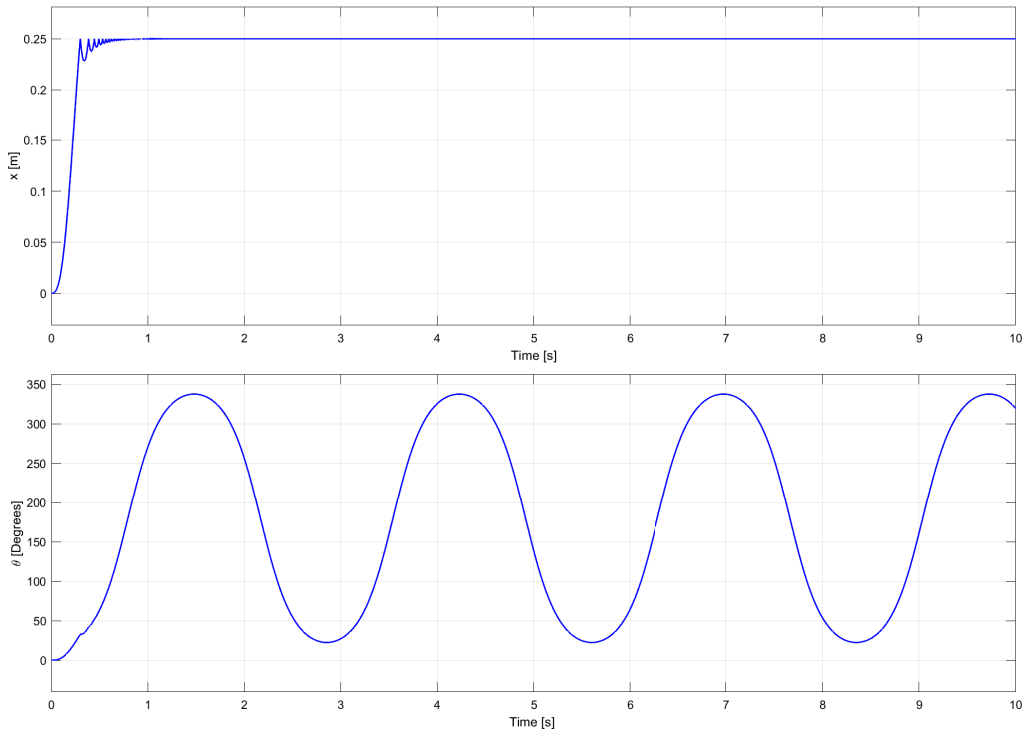


Figure 5.23: Test of the complete model

I did another test where the nonlinear model is compared against the linear model using the linear fuzzy controller. The two responses, setting x_{set}as a step of 0.05m with step time 1s, are shown in figure 5.24.

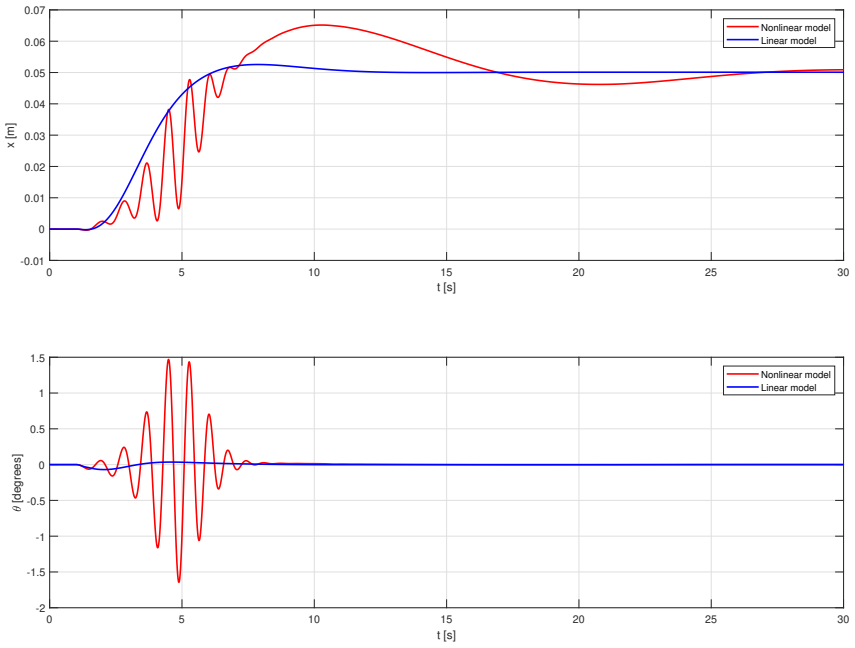


Figure 5.24: Step response with linear fuzzy controller of the two models

The two behaviors are more or less the same, but in addition there are the effects of the nonlinearities in the nonlinear model. So considering the test of each susbsysem the model was valid and this final two tests confirms that the linear approximation was not so far from the real behavior of the system.

Chapter 6

Design of the nonlinear fuzzy controller

As explained in [2] and reported in the previous chapter, the next step of the design procedure is to make the fuzzy controller nonlinear.
The control structure remains the same, I modified only the structure of the fuzzy controller.
The nonlinearization of the controller consists, first of all, in replacing the methods for the logic functions with the nonlinear functions. In particular the minimum function is used for the logic AND and the maximum function is used for the logic OR (the OR will not be used in the rules).
The minimum for the AND consists in taking into account the minimum truth value given by a certain rule for the given inputs. For example, considering a single rule, if the truth value of input 1 is 0.5 and the truth value of input 2 is 0.2 the truth value given by the rule is 0.2. Thinking about the Boolean AND it is True only when the two inputs are True. So extending this concept to the fuzzy logic we have to take into account only the values where the two inputs are true. So this value corresponds to the minimum of the two inputs. In a graphic view figure 6.1 can be observed.

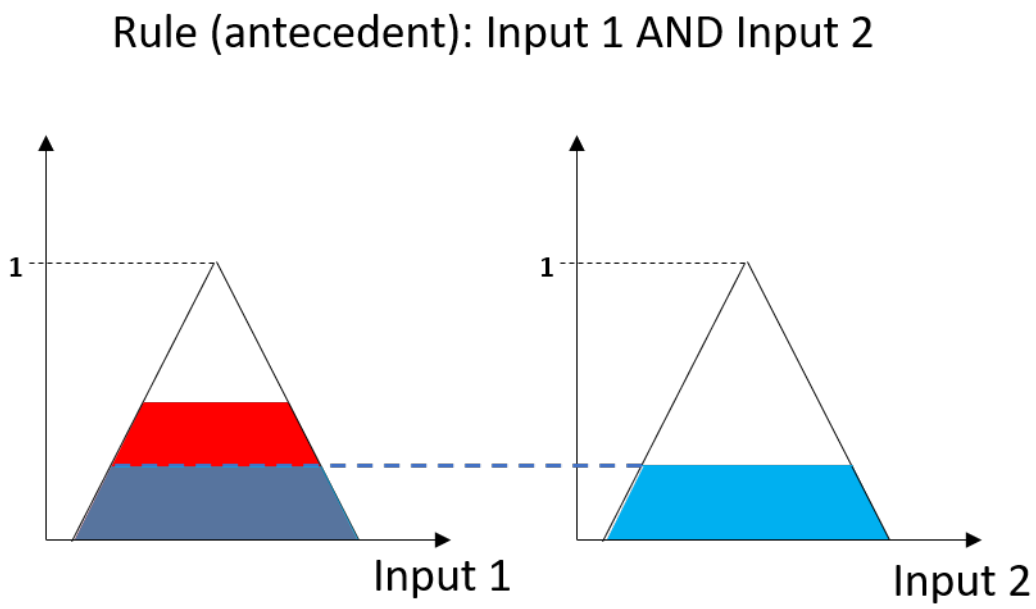


Figure 6.1: Graphic explanation of the minimum function for the logic AND

The maximum method for the logic OR works essentially in the same way of the logic AND. The defuzzification method remains the same of the linear case (COG).
The methods can be changed simply in the MatLab® tool for the fuzzy controller.
Continuing in the nonlinearization process of the fuzzy controller, I changed the set-membership functions of the output variable. In the linear case they were singletons, now they are modified into triangular functions. The centers of the triangles have the same values of the singletons. In figure 6.2 there is a graphical representation of the set-membership functions of the output variable.

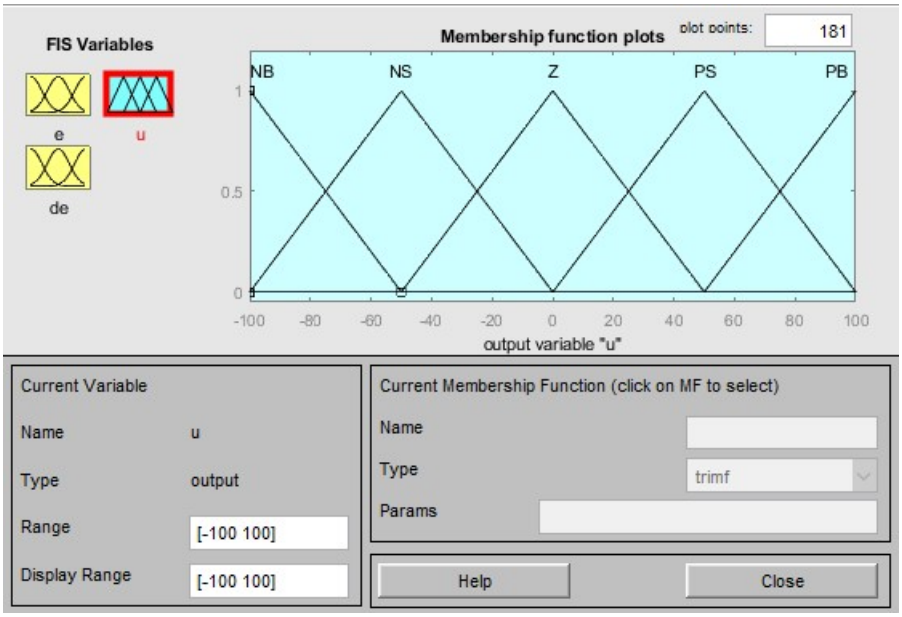


Figure 6.2: Set-membership functions of the output variable

In this way the control action should be more precise due to the fact that thanks to the triangular functions the output variable can assume also the values between the singletons, and in this way the controller can apply a "more continuous" control action. The rules are the last change. The new rule table is shown in the following table where in red are the modified linguistic values.

		$\dot{e}(t)$		
		N	Z	P
$e(t)$	N	NB	NS	NS
	Z	NS	Z	PS
	P	PS	PS	PB

Table 6.1: Rule table of the nonlinear fuzzy controller

I changed only two rules:

1. IF e is N AND $\dot{e}$ is P => u is NS

2. IF e is P AND $\dot{e}$ is N => u is PS

In this way the valves does not close during regulation but only when the pendulum or the cart are stabilized. So the control action should be faster and more ready to compensate a disturbance. On the other hand the overshoot should be higher.
Concerning the gains, they remain the same of the linear case. Their numerical values

are:

$$GE_a = 286.479 \qquad GE_l = 400$$
$$GU_a = 0.0073 \qquad GU_l = -3 \cdot 10^{-5}$$
$$GCE_a = 14.12 \qquad GCE_l = 1.017 \cdot 10^{3}$$
$$GIE_a = 865 \qquad GIE_l = 0.1556$$

6.1 Simulation results and disturbance analysis

First of all I made a comparison with the linear controller with the same input (x_{set}). The x_{set} was a step of $0.05m$ with step time $1s$. The two responses are shown in figure 6.3.

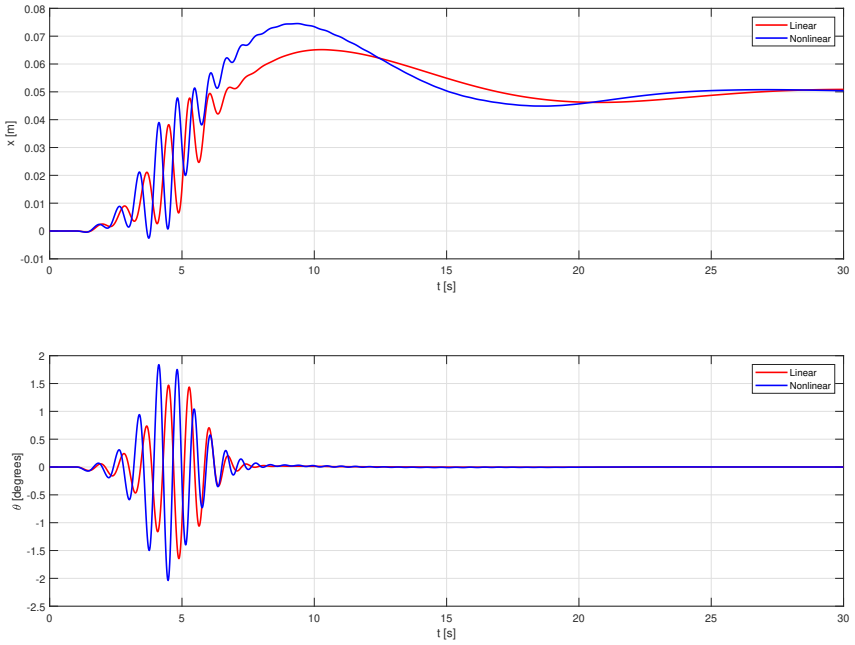


Figure 6.3: Step response with linear and nonlinear controller

The achieved performances are the following (in brackets the performances achieved with the linear controller):

- Rising time: $5.94s$ $(6.65s)$
- Settling time $\pm 10\%$: $13.85s$ $(15s)$

- Max overshoot: 49% (30%)

As expected the response is a bit faster but the overshoot increases.
Once the controller is designed its disturbance response can be analyzed.
A very common disturbance that can occur is a force applied on the cart. It can be impulsive or in the worst case it can be a step (a constant disturbance force applied on the cart). The response in the two cases of disturbance are shown in the figures 6.4 and 6.5 where the impulsive disturbance is of $50N$ at $1s$ ($x_{set} = 0m$) and the step disturbance is of $10N$ at $25s$ ($x_{set} = 0.05m$).

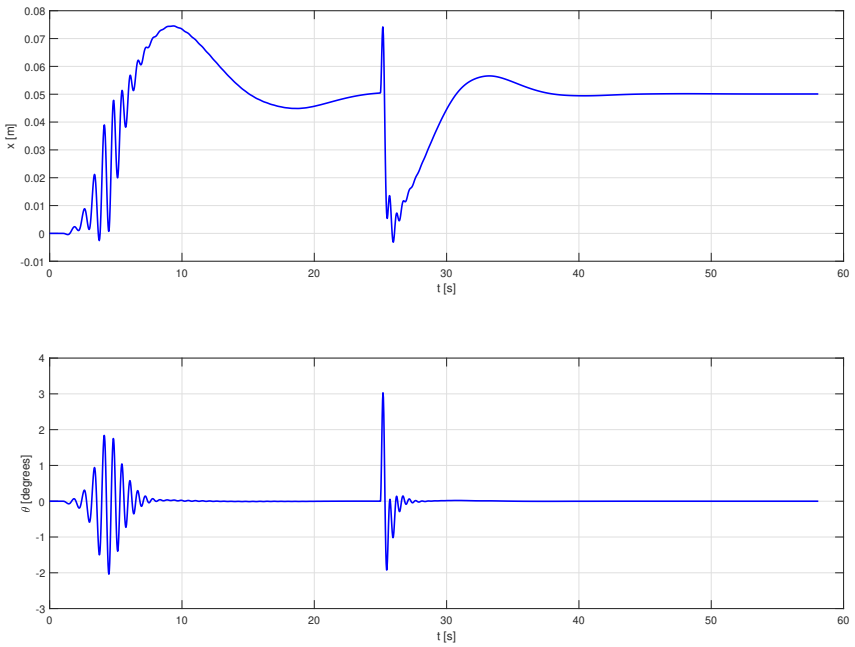


Figure 6.4: Response with a step disturbance

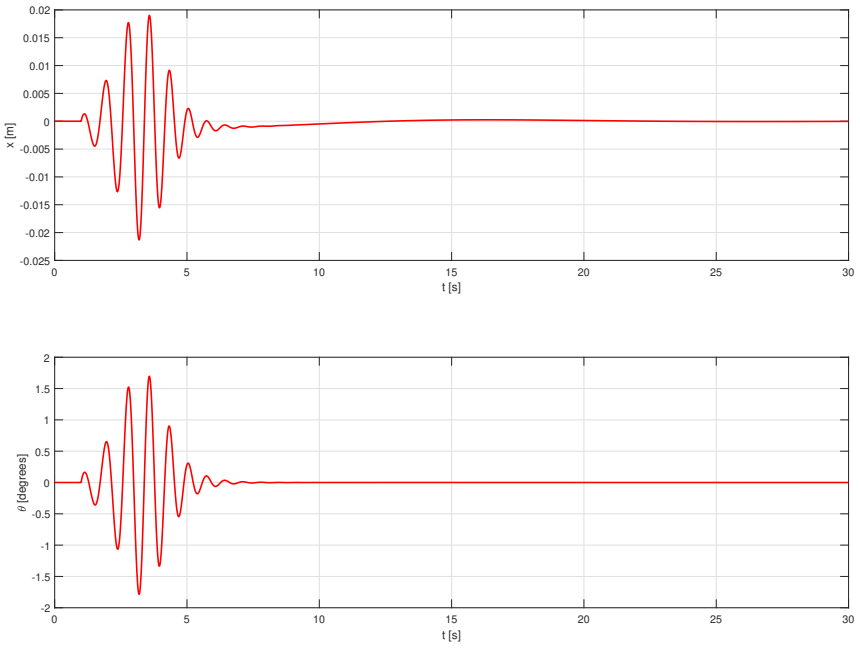


Figure 6.5: Response with an impulsive disturbance

A final simulation is done with non equal initial pressures in the two chambers of the actuator. In fact till now the initial conditions of the chambers of the actuator are with the atmospheric pressure. For this test I put in the chamber B an initial pressure greater than the atmospheric pressure of $0.5bar$ and in the chamber A the atmospheric pressure. The result is shown in figure 6.6.

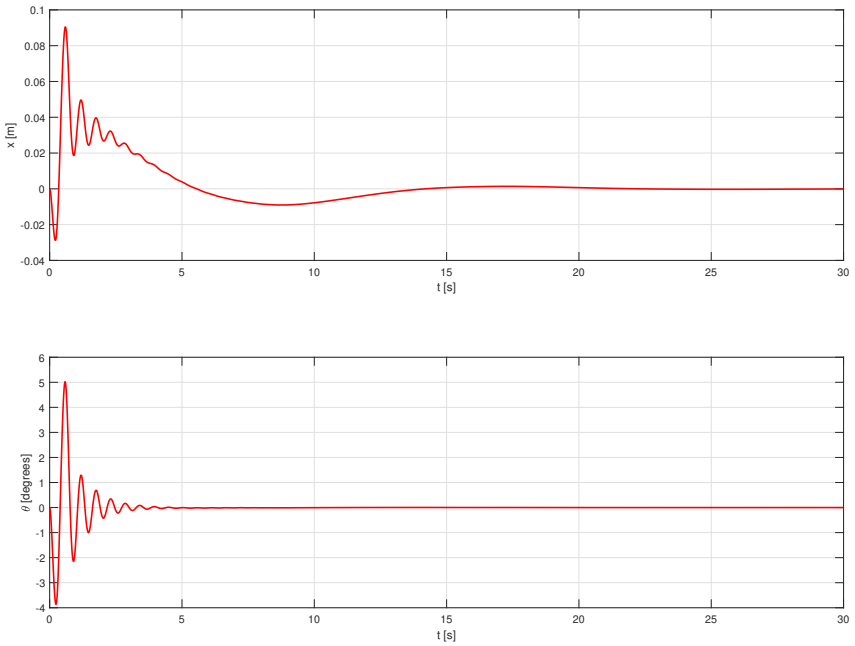


Figure 6.6: Behavior of the system with a non equal initial pressure in the two chambers of the actuator

Chapter 7

Arduino program

In this section I will present the prototype of an arduino code but it was not tested in real life. The code is complete in terms of functionality but an experimental test needs to be done.
The full code of the program is available in the appendix A. Essentially it is divided into 4 parts:

1. System's data storage
2. First main setup at starting
3. Setup of the timer interrupt
4. Instructions for the timer interrupt

In the following I will provide an explanation of each part.

7.1 System's data storage

In this part all the needed variables to store the data of the system are created and initialized.

Listing 7.1: Data part

```
#define OUTPUT_COMPARE 0x04E2
                        //0x0138 set to 1.25ms
                       //0x0271 set to 2.5ms
                       //0x04E2 set to 5ms
                       //0x09C4 set to 10ms
                       //0x0EA6 set to 15ms
                       //0x186A set to 25ms
                       //0x30D4 set to 50ms

Fuzzy *fc = new Fuzzy(); // Fuzzy variable
```

```
11
// PIN assignment
13 const int x_pin = A8, th_pin = A2; // Analog PIN for the sensors
const int v1_pin = 4, v2_pin=13; // Analog PIN for the valves
15
// Input variables
17 float x=0;
float th=0;
19 // Errors variables
float ex=0, eth=0, dex=0, deth=0, ex1=0, eth1=0, iex=0, ieth=0;
21
float u=0; // To store the input command for the plant
23 float x_set=0, th_set=0; // To set the x and theta
const float dt=0.005; // Sampling time fixed at 5 ms
25
// Fuzzy gains
27 float Kpa=2.09, Kia=6.31, Kda=0.103;
float Kpl=-0.000802, Kil=-4.68e-6, Kdl=-0.0306;
29 float x_max=0.25;
float GEl=100.0/x_max;
31 float GUl=Kpl/GEl*15.0;
float GCEl=Kdl/GUl;
33 float GIEl=Kil/GUl;
float th_max=3.14*20.0/180.0;
35 float GEa=100.0/th_max;
float GUa=Kpa/GEa;
37 float GCEa=Kda/GUa;
float GIEa=Kia/GUa;
```

Referring to the listing 7.1 the first line is a constant variable called “OUTPUT_COMPARE”. This variable represents the compare value for the timer interrupt.
In line 10 an object called “fc” from the class “Fuzzy” is created. The class “Fuzzy” is in the library “fuzzy.h”. The builder of the class does not want any input.
In lines 13-14 the pins variables are created for the inputs and outputs of the controller. The variables are of type “int” and they are declared as constant, because in this way they can not be modified. In particular the pins for the input are the analog pins, while the pins for the output are digital pin and they are written using the PWM technique at high frequency in order to approximate a continuous signal.
The last part of the code in listing 7.1 is devoted to store the gains values of the controller. They are computed in the same way of the simulation. In this way their computation is slower, but since this code is run just once it does not matter. On the other hand in this way they can be easily modified.

7.2 First setup at starting

The instructions to run at the starting of the board are written inside a function called “setup”. This function is run only once, so the instructions inside this function are to

setup the board (listing 7.2).

Listing 7.2: Setup

```
void setup() {

  Serial.begin(9600); // To set the speed of the serial monitor
  timerInterruptSetup(); // To set the timer interrupt
  FuzzySetup(); // To set the fuzzy controller
  TCCR0B = TCCR0B & 0b11111000 | 0x02; // Timer 0 to set the frequency to 8kHz

  // Zero initial value to the output ports for safety
  analogWrite(v1_pin, 0);
  analogWrite(v2_pin, 0);
}
```

Referring to the listing 7.2 in line 4 a void function called " timerInterruptSetup" is run. In this function there are all the instructions to setup the timer interrupt and they will be explained in the next section.
In line 5 another void function called "FuzzySetup" is run. In this function there are all the instructions to setup the fuzzy controller in OOP fashion. In fact the class "Fuzzy" provides all the methods to set in an easy way the structure of the controller (rules, fuzzy sets...). These instructions is not reported in this section, they are shown in the full code in the appendix A. Obviously the controller settings are the same of the nonlinear case in the simulations.
In line 6 there is a particular instruction to set the frequency of the PWM in ports 4 and 13. The PWM of these ports uses the clock of timer 0 (in the board four timers are available). So choosing a correct prescaler the frequency of the resulting PWM can be changed. The prescaler is chosen knowing that the base working frequency of timer 0 is $62500Hz$. Considering the fact that the valve's driver works with a PWM at $500Hz$ as output and with a continuous signal as input; a PWM with a frequency greater than $500Hz$ should be given by the Arduino. So I choose a value of the prescaler equal to 8:

$$f_{PWM} = \frac{62500}{8} = 7812.5Hz$$

This value is only indicative, it should be tested and eventually modified. I ensure that the instruction works properly in fact the frequency of the resulting PWM was measured with an oscilloscope.

7.3 Setup of the timer interrupt

As shown in the listing 7.2 in the main setup a function called "timerInterruptSetup" is run. It is a void function and its goal is to run a set of instructions to setup the timer/-counter 1 in order to rise an interrupt after a given time. The set of instructions of the function are shown in the listing 7.3.

Listing 7.3: Timer interrupt setup

```
// Timer interrupt setup
void timerInterruptSetup(){
    cli(); //disable the global interrupt
    //Timer/Counter 1
    TCCR1A = 0x00;
    TCCR1B = (_BV(WGM12)) | (_BV(CS11)) | (_BV(CS10)); //CTC mode, clk/64
    OCR1A = OUTPUT_COMPARE; //set to ...
    TCNT1 = 0x00; //initialise the counter
    TIMSK1 = _BV(OCIE1A); //Output Compare A Match Interrupt Enable
    sei(); //enable global interrupt
}
```

Essentially the main idea idea is to use a counter that is incremented by 1 every time the clock of timer 1 goes from 0 to 1. When the counter's value reaches the compare's value an interrupt is raised and a series of instructions are executed (they will be explained n the next section).
Then the counter is reset. In this way choosing properly the value of the compare, the sampling time of the controller is consequently chosen. In this case the sampling time is $5ms$. Also in this case this value should be tested and eventually modified.

7.4 Timer interrupt

When the interrupt from the counter of the timer 1 is raised the instructions inside a particular function called "ISR" are executed.
The first part of this function is devoted to print in the serial monitor the system's value x and θ. In this way is possible to collect them and to build a graph to make a comparison with the simulation. Then the "old" values of x and θ (corresponding to the prevoius cycle) are stored and also the actual values are read and stored (listing 7.4).

Listing 7.4: Data acquisition and storing

```
ISR(TIMER1_COMPA_vect){
  // Data acquisition
  Serial.print("u:␣");
  Serial.print(u,3);
  Serial.print("␣x:␣");
  Serial.print(x,3);
  Serial.print("␣th:␣");
  Serial.print(th,3);

  // Storing the error's values of the previous cycle
  ex1 = ex;
  eth1 = eth;

  // Reading the actual values
  x = (float) -1.0*(analogRead(x_pin)*0.5/1023.0-0.25);
```

```
  th = (float) -1.0*(analogRead(th_pin)*1.57/1023.0-0.785)-0.291;
17 // To have a precision of the third decimal
  th=th*1000;
19 th=th/1000.0;
```

Once the old and new values of the system's variables are updated the error's values and their integrator and derivative have to be computed. So, according to the control structure, the first computation is devoted to the position of the cart x (listing 7.5). Then the computed values are fuzzified and, according to the fuzzy rules, the value of θ_{set} is computed and defuzzified.

Listing 7.5: Error's computation for the position of the cart

```
1 // Computation of the x error's value
  ex = (float) x_set - x;
3
  // Derivative of the error and saturation at -100 100
5 dex = (float) ((ex - ex1)/dt)*GCEl;
  if (dex>100) dex = 100;
7 if (dex<-100) dex = -100;

9 // Integral of the error through trapezoidal approximation
  iex = (float) ((ex1+ex)*dt)/2;
11 // Application of the fuzzy rules and computation of the theta_set
  fc->setInput(1,ex*GEl);
13 fc->setInput(2,dex);
  fc->fuzzify();
15 th_set = (float) ((iex*GIEl)+(fc->defuzzify(1)))*GUl;
```

The derivative is computed using the linear equation and the integrator using the equation for the computation of the area of a trapezium. The use of these equations for the derivative and the integral is not a big approximation due to the fact that the sampling time is little ($5ms$) and so the variations from the previous value and the actual one is little and so can be considered linear.
Once the θ_{set} is computed on the same way the valves' command u can be computed, but this time the value of the angle should be considered (listing 7.6).

Listing 7.6: Error's computation for the angle of the pendulum

```
1 // Computation of the theta error's value
  eth = (float) th_set - th;
3 // Derivative of the error and saturation at -100 100
  deth = (float) ((eth-eth1)/dt)*GCEa;
5 if (deth>100) deth = 100;
  if (deth<-100) deth = -100;
7
  // Integral of the error through trapezoidal approximation
9 ieth = (float) ((eth1+eth)*dt)/2;
```

```
11  // Application of the fuzzy rules and computation of the command u
    fc->setInput(1,eth*GEa);
13  fc->setInput(2, deth);
    fc->fuzzify();
15  u = (float) ((ieth*GIEa)+(fc->defuzzify(1)))*GUa;
```

Finally the computed value of u is sent to the output ports considering the fact that it can be negative or positive. For this reason there are three if-statement (see the appendix A).

Chapter 8

Conclusions and further developments

The main idea of this work was to design and realize a fuzzy controller to stabilize a pneumatically actuated inverse pendulum. This type of system is widely used and studied in academic and industrial applications. The fuzzy logic introduce a newer control method appllied in this classical system. During my litherature review I did not find any other project like the one of my work.
The system needs a double control, the position of the cart and the angle of the pendulum. That problem was resolved by using two nested feedback control loops.
All of the design is done using the model-based approach. Thanks to that method the system can be simulated in a more or less realistic dynamic model. The simulation resulted very helpful because building a very simple linear model, a first controller can be designed and then it can be fine-tuned in more realistic model. Those steps were done without using any real system. The advantages of this method is that the design of the controller, and also the tuning, was faster. In an industrial environment that method should result in a cheaper design phase.
Unfortunately the experimental tests were not done because of time constraints for the graduation, so a first proposed further development is to implement and test my program in the Arduino board to show if the simulation corresponds to the real behavior of the system. In this way, with the realization of the controller, the design is completed.
Concerning the simulations they show that a stable controller can be developed and also that it is very robust against disturbance. The response is a little bit slower than the PID, but the control action is very precise and stable. The model is built without considering the Coulombian's friction, so in the future some tests can be done in the system in order to evaluate this effect.
Thinking about the response of the controller, it can be modified in order to get a specific performance. For example a faster response, less overshoot or less oscillations in the transient phase. So with this goal another fuzzy controller, with different gains or maybe with different structure, can be developed.
Once the controller is realized a very interesting evolution could be the possibility to change the variable x_{set} of the cart. A first idea should be to design and realize a sort

of manual controller to set manually the final position of the cart. Another idea is to program a remote control. For example building a web service a user can set the position of the cart in remote and can visualize the graphs of the response and, in this environment, he can set also the gains of the controller and see the difference in the behavior of the system. This idea can be very useful in a didactic environment.

Appendix A

Sketch of the program for Arduino

```
#include <Fuzzy.h>
#define OUTPUT_COMPARE 0x04E2
                      //0x0138 set to 1.25ms
                     //0x0271 set to 2.5ms
                     //0x04E2 set to 5ms
                     //0x09C4 set to 10ms
                     //0x0EA6 set to 15ms
                     //0x186A set to 25ms
                     //0x30D4 set to 50ms

Fuzzy *fc = new Fuzzy(); // Fuzzy variable

// PIN assignment
const int x_pin = A8, th_pin = A2; // Analog PIN for the sensors
const int v1_pin = 4, v2_pin=13; // PWM PIN for the valves

// Input variables
float x=0;
float th=0;
// Errors variables
float ex=0, eth=0, dex=0, deth=0, ex1=0, eth1=0, iex=0, ieth=0;

float u=0; // To store the input command for the plant
float x_set=0, th_set=0; // To set the x and theta
const float dt=0.005; // Sampling time fixed at 5 ms

// Fuzzy gains
float Kpa=2.09, Kia=6.31, Kda=0.103;
float Kpl=-0.000802, Kil=-4.68e-6, Kdl=-0.0306;
float x_max=0.25;
float GEl=100.0/x_max;
float GUl=Kpl/GEl*15.0;
float GCEl=Kdl/GUl;
float GIEl=Kil/GUl;
float th_max=3.14*20.0/180.0;
```

```
float GEa=100.0/th_max;
float GUa=Kpa/GEa;
float GCEa=Kda/GUa;
float GIEa=Kia/GUa;

void setup() {

  Serial.begin(9600); // To set the speed of the serial monitor
  timerInterruptSetup(); // To set the timer interrupt
  FuzzySetup(); // To set the fuzzy controller
  TCCR0B = TCCR0B & 0b11111000 | 0x02; // Timer 0 to set the frequency to 8kHz

  // Zero initial value to the output ports for safety
  analogWrite(v1_pin, 0);
  analogWrite(v2_pin, 0);
}

void loop() {
}


// Timer interrupt
ISR(TIMER1_COMPA_vect){
  // Data acquisition
  Serial.print("u:␣");
  Serial.print(u,3);
  Serial.print("␣x:␣");
  Serial.print(x,3);
  Serial.print("␣th:␣");
  Serial.print(th,3);

  // Storing the error's values of the previous cycle
  ex1 = ex;
  eth1 = eth;

  // Reading the actual values
  x = (float) -1.0*(analogRead(x_pin)*0.5/1023.0-0.25);
  th = (float) -1.0*(analogRead(th_pin)*1.57/1023.0-0.785)-0.291;
  // To have a precision of the third decimal
  th=th*1000;
  th=th/1000.0;

  // Computation of the x error's value
  ex = (float) x_set - x;

  // Derivative of the error and saturation at -100 100
  dex = (float) ((ex - ex1)/dt)*GCEl;
  if (dex>100) dex = 100;
  if (dex<-100) dex = -100;

```

```
  // Integral of the error through trapezoidal approximation
87  iex = (float) ((ex1+ex)*dt)/2;
  // Application of the fuzzy rules and computation of the theta_set
89  fc->setInput(1,ex*GEl);
  fc->setInput(2,dex);
91  fc->fuzzify();
  th_set = (float) ((iex*GIEl)+(fc->defuzzify(1)))*GUl;
93
  // Computation of the theta error's value
95  eth = (float) th_set - th;

97  // Derivative of the error and saturation at -100 100
  deth = (float) ((eth-eth1)/dt)*GCEa;
99  if (deth>100) deth = 100;
  if (deth<-100) deth = -100;
101
  // Integral of the error through trapezoidal approximation
103 ieth = (float) ((eth1+eth)*dt)/2;

105 // Application of the fuzzy rules and computation of the command u
  fc->setInput(1,eth*GEa);
107 fc->setInput(2, deth);
  fc->fuzzify();
109 u = (float) ((ieth*GIEa)+(fc->defuzzify(1)))*GUa;
    // Writing the corresponding pins to move the actuator
111   if (u==0){
      analogWrite(v1_pin, 0);
113   analogWrite(v2_pin, 0);
    }
115   if (u>0){
    analogWrite(v1_pin, u*255);
117 analogWrite(v2_pin, 0);
    }
119   if(u<0){
      analogWrite(v2_pin, abs(u)*255);
121   analogWrite(v1_pin, 0);
    }
123 }

125 // Timer interrupt setup
  void timerInterruptSetup(){
127   cli(); //disable the global interrupt
    //Timer/Counter 1
129   TCCR1A = 0x00;
    TCCR1B = (_BV(WGM12)) | (_BV(CS11)) | (_BV(CS10)); //CTC mode, clk/64
131   OCR1A = OUTPUT_COMPARE; //set to ...
    TCNT1 = 0x00; //initialise the counter
133   TIMSK1 = _BV(OCIE1A); //Output Compare A Match Interrupt Enable
    sei(); //enable global interrupt
135 }
```

```
137 // Fuzzy controller setup
    void FuzzySetup(){
139   // Fuzzy sets
      FuzzySet *N = new FuzzySet(-100, -100, -100, 0);
141   FuzzySet *Z = new FuzzySet(-100, 0, 0, 100);
      FuzzySet *P = new FuzzySet(0, 100, 100, 100);
143   FuzzySet *NB = new FuzzySet(-100, -100, -100, -50);
      FuzzySet *NS = new FuzzySet(-100, -50, -50, 0);
145   FuzzySet *ZE = new FuzzySet(-50, 0, 0, 50);
      FuzzySet *PS = new FuzzySet(0, 50, 50, 100);
147   FuzzySet *PB = new FuzzySet(50, 100, 100, 100);

149   // Input e
      FuzzyInput *e = new FuzzyInput(1);
151   e->addFuzzySet(N);
      e->addFuzzySet(Z);
153   e->addFuzzySet(P);
      fc->addFuzzyInput(e);
155
      //Input de
157   FuzzyInput *de = new FuzzyInput(2);
      de->addFuzzySet(N);
159   de->addFuzzySet(Z);
      de->addFuzzySet(P);
161   fc->addFuzzyInput(de);

163   // Output
      FuzzyOutput *u = new FuzzyOutput(1);
165   u->addFuzzySet(NB);
      u->addFuzzySet(NS);
167   u->addFuzzySet(ZE);
      u->addFuzzySet(PS);
169   u->addFuzzySet(PB);
      fc->addFuzzyOutput(u);
171
      // Rules
173   // (1) IF e is N AND de is N => u is NB
      FuzzyRuleAntecedent *eNdeN = new FuzzyRuleAntecedent();
175   eNdeN->joinWithAND(N, N);
      FuzzyRuleConsequent *uNB = new FuzzyRuleConsequent();
177   uNB->addOutput(NB);
      FuzzyRule *fuzzyRule1 = new FuzzyRule(1, eNdeN,uNB);
179   fc->addFuzzyRule(fuzzyRule1);

181   // (2) IF e is N AND de is Z => u is NS
      FuzzyRuleAntecedent *eNdeZ = new FuzzyRuleAntecedent();
183   eNdeZ->joinWithAND(N, Z);
      FuzzyRuleConsequent *uNS1 = new FuzzyRuleConsequent();
185   uNS1->addOutput(NS);
```

```
  FuzzyRule *fuzzyRule2 = new FuzzyRule(2, eNdeZ,uNS1);
187 fc->addFuzzyRule(fuzzyRule2);

189 // (3) IF e is N AND de is P => u is NS
  FuzzyRuleAntecedent *eNdeP = new FuzzyRuleAntecedent();
191 eNdeP->joinWithAND(N, P);
  FuzzyRuleConsequent *uNS2 = new FuzzyRuleConsequent();
193 uNS2->addOutput(NS);
  FuzzyRule *fuzzyRule3 = new FuzzyRule(3, eNdeP,uNS2);
195 fc->addFuzzyRule(fuzzyRule3);

197 // (4) IF e is Z AND de is N => u is NS
  FuzzyRuleAntecedent *eZdeN = new FuzzyRuleAntecedent();
199 eZdeN->joinWithAND(Z, N);
  FuzzyRuleConsequent *uNS3 = new FuzzyRuleConsequent();
201 uNS3->addOutput(NS);
  FuzzyRule *fuzzyRule4 = new FuzzyRule(4, eZdeN,uNS3);
203 fc->addFuzzyRule(fuzzyRule4);

205 // (5) IF e is Z AND de is Z => u is Z
  FuzzyRuleAntecedent *eZdeZ = new FuzzyRuleAntecedent();
207 eZdeZ->joinWithAND(Z, Z);
  FuzzyRuleConsequent *uZ = new FuzzyRuleConsequent();
209 uZ->addOutput(ZE);
  FuzzyRule *fuzzyRule5 = new FuzzyRule(5, eZdeZ,uZ);
211 fc->addFuzzyRule(fuzzyRule5);

213 // (6) IF e is Z AND de is P => u is PS
  FuzzyRuleAntecedent *eZdeP = new FuzzyRuleAntecedent();
215 eZdeP->joinWithAND(Z, P);
  FuzzyRuleConsequent *uPS1 = new FuzzyRuleConsequent();
217 uPS1->addOutput(PS);
  FuzzyRule *fuzzyRule6 = new FuzzyRule(6, eZdeP,uPS1);
219 fc->addFuzzyRule(fuzzyRule6);

221 // (7) IF e is P AND de is N => u is PS
  FuzzyRuleAntecedent *ePdeN = new FuzzyRuleAntecedent();
223 ePdeN->joinWithAND(P, N);
  FuzzyRuleConsequent *uPS2 = new FuzzyRuleConsequent();
225 uPS2->addOutput(PS);
  FuzzyRule *fuzzyRule7 = new FuzzyRule(7, ePdeN,uPS2);
227 fc->addFuzzyRule(fuzzyRule7);

229 // (8) IF e is P AND de is Z => u is PS
  FuzzyRuleAntecedent *ePdeZ = new FuzzyRuleAntecedent();
231 ePdeZ->joinWithAND(P, Z);
  FuzzyRuleConsequent *uPS3 = new FuzzyRuleConsequent();
233 uPS3->addOutput(PS);
  FuzzyRule *fuzzyRule8 = new FuzzyRule(8, ePdeZ,uPS3);
235 fc->addFuzzyRule(fuzzyRule8);
```

```
237   // (9) IF e is P AND de is P => u is PB
      FuzzyRuleAntecedent *ePdeP = new FuzzyRuleAntecedent();
239   ePdeP->joinWithAND(P, P);
      FuzzyRuleConsequent *uPB = new FuzzyRuleConsequent();
241   uPB->addOutput(PB);
      FuzzyRule *fuzzyRule9 = new FuzzyRule(9, ePdeP,uPB);
243   fc->addFuzzyRule(fuzzyRule9);
}
```

Bibliography

[1] Valerio Einaudi. *Studio Teorico Numerico e Sperimentale della Stabilità di un Pendolo Inverso su Base Mobile*. 1995.

[2] J. Jantzen. *Foundations of Fuzzy Control: A Practical Approach*. Wiley, 2013.

[3] Zeljko Situm Josko Petric. *Inverted Pendulum Driven by Pneumatics*. 2003.

[4] C. Krupke and J. Wang. *Modelling and robust control of an inverted pendulum driven by a pneumatic cylinder*. IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Busan, 2015.

[5] Thi Thanh Hoang Le, Anh Khoa Vo, Thien Van Nguyen, Van Khanh Doan, Duc Ha Vu, Manh Son Tran, and Van Dong Hai Nguyen. *Fuzzy controller of rotary inverted pendulum*. 2018.

[6] Pontin Marco. *Modellazione, realizzazione e controllo mediante PLC di un sistema a pendolo inverso ad attuazione pnematica*. 2018.

[7] K. Michels, F. Klawonn, R. Kruse, and A. Nürnberger. *Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers*. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, 2007.

[8] Lal Bahadur Prasad, Barjeev Tyagi, and Hari Om Gupta. Optimal control of nonlinear inverted pendulum system using pid controller and lqr: Performance analysis without and with disturbance input. *International Journal of Automation and Computing*, 2014.

[9] Tomomichi Sugihara, Yoshihiko Nakamura, and Hirochika Inoue. *Realtime Humanoid Motion Generation through ZMP Manipulation Based on Inverted Pendulum Control*. 2002.

[10] Marco Triverio. *Progetto E Implementazione Del Sistema Di Controllo Per Un Pendolo Inverso*. 2008.

[11] Junfeng Wu, Wanying Zhang, and Shengda Wang. *A Two-Wheeled Self-Balancing Robot with the Fuzzy PD Control Method*. 2012.