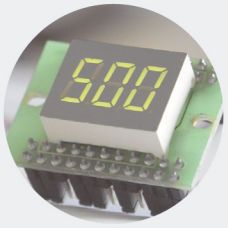
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES





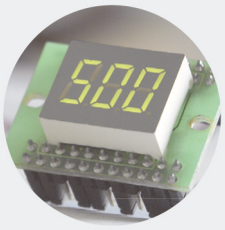

Control of a Heat Diffusion System Using Fuzzy Logic

RUI PEDRO SILVA MENDES
Julho de 2025

POLITÉCNICO
DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES





Control of a Heat Diffusion System Using Fuzzy Logic

RUI PEDRO SILVA MENDES
Julho de 2025

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Control of a Heat Diffusion System Using Fuzzy Logic

Rui Pedro Silva Mendes

Master's Degree in Electrical Engineering

Specialization Area of Autonomous Systems

Department of Electrical Engineering

Supervisor: Prof. Isabel Maria de Sousa de Jesus, PhD

July 3, 2025

"Apart from you and your supervisor, nobody will ever read your thesis."

Anonymous

THIS PAGE
INTENTIONALLY
LEFT BLANK

Abstract

Heat diffusion systems are typically controlled by **P**roportional **I**ntegral **D**erivative (PID) controllers but other approaches, such as **F**uzzy **L**ogic **C**ontroller (FLC), can offer better results. This technique transforms qualitative and imprecise information (expressed linguistically), into quantitative data which allows a more precise and adaptive control.

In this thesis, it is proposed a Fuzzy Logic based control strategy for a heat diffusion system, comparing its performance with traditional PID control. The methodology includes system modeling, design the fuzzy inference system, and simulation using MATLAB with additional tests using Anti-Windup compensation and Smith Predictor to improve transient response and time-delay handling. The final validation introduces simulated environmental noise representing real-world disturbances, (*e.g.* sudden temperature changes from opening the door) to evaluate controller robustness. Results demonstrate that Fuzzy Logic improves response accuracy, disturbance rejection, and adaptability to varying operating conditions, highlighting its potential for thermal systems with complex dynamics.

Keywords: Heat diffusion, Fuzzy Logic, PID controllers, Anti-Windup, Smith Predictor, disturbance.

THIS PAGE
INTENTIONALLY
LEFT BLANK

Acknowledgments

This report represents another important step in my academic and personal development. I would like to thank **I**nstituto **S**uperior de **E**ngenharia do **P**orto (ISEP) and all the teachers at **D**epartamento de **E**ngenharia **E**letrotécnica (DEE) for the way they prepared me for professional life throughout my Master's course.

To my supervisor, Professor Isabel Jesus, for her support, availability, understanding, patience and the trust she has always placed in me. Thank you for guiding me, supporting me and showing me the best way to prepare and implement the whole project.

To my family, especially my parents, who have always believed in me and supported me at every moment of my academic and personal life. A special thank you to them!

To all my friends for their availability and attention during the many hours I spent writing this report.

My sincere thanks to all of you.

THIS PAGE
INTENTIONALLY
LEFT BLANK

Contents

THIS PAGE
INTENTIONALLY
LEFT BLANK

List of Tables

THIS PAGE
INTENTIONALLY
LEFT BLANK

Lista de Siglas e Acrónimos

CPU	**C**ontrol **P**rocessing **U**nit
DEE	**D**epartamento de **E**ngenharia **E**letrotécnica
FIS	**F**uzzy **I**nference **S**ystem
FLC	**F**uzzy **L**ogic **C**ontroller
IAE	**I**ntegral **A**bsolute **E**rror
ISE	**I**ntegral **S**quared **E**rror
ISEP	**I**nstituto **S**uperior de **E**ngenharia do **P**orto
ITAE	**I**ntegral **T**ime **A**bsolute **E**rror
ITSE	**I**ntegral **T**ime **S**quared **E**rror
LTI	**L**inear **T**ime **I**nvariant
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
PID	**P**roportional **I**ntegral **D**erivative
SISO	**S**ingle **I**nput **S**ingle **O**utput
SSE	**S**um of **S**quared **E**rrors

Chapter 1

Introduction

1.1 Contextualization

Automatic control systems play a key role in the modern world, from complex industrial applications to building air conditioning or thermal process control. However, controlling thermal systems faces significant challenges [1].

A heating or cooling system does not operate in a static environment and external disturbances can happen like opening doors and windows which results in a change of temperature. In addition, heat diffusion involves temperature propagation delays and non-linear behaviors, which make precise control difficult with conventional methods.

Although PID controllers are widely used in industries, they have some drawbacks in their dynamic scenarios [2]. Since that they rely on linear mathematical models and fixed parameters, it makes them less effective when facing variations from external noise. This is where alternative approaches, such as Fuzzy Logic, gain relevance. Inspired by the way human reasoning deals with uncertainty, Fuzzy Logic allows qualitative rules (*e.g.,* "If the temperature is low, heat it more") to be translated into quantitative control actions, adapting better to complex and unpredictable systems [3].

1.2 Motivation

In the study of control systems, PID controllers often appear as the reference solution due to their simplicity and effectiveness in linear systems, but like any other, they have limitations that can affect their implementation. Since the possibility of exploring new control methods arose, even if they were initially unknown, Fuzzy Logic proved to be a promising approach once it represents human reasoning through linguistic rules, thus enabling more effective control in dynamic systems. Having this into consideration, this dissertation aims to investigate the implementation of Fuzzy Logic in the control of thermal diffusion systems, comparing its performance with the classical PID method.

1.3 Objectives

In general, the present project aims not only to deepen the knowledge related to PID controllers but also the Fuzzy Logic-based controllers and their implementation in MATLAB. To this end, several approaches will be implemented in order to compare the pros and cons of each one. Thus, the following tasks were performed:

- Theorical and pratical study on PID and Fuzzy controllers;
- Study of MATLAB software;
- Development of different control approaches;
- Comparison between all control implementations;
- Introduction of noise to simulate a real world scenario to test controllers robustness.

1.4 Thesis Structure

This report is divided into five chapters.

The first chapter provides an introductory approach to the topic, as well as the objectives outlined and the motivation.

The second chapter begins with an introduction to the state of the art, summarizing in more detail the concept of a heat diffusion system, followed by a theoretical explanation of both a classical controller and a Fuzzy controller, and a description of a First Order delayed system.

The third chapter explains the tools and systems required to be used such as MATLAB toolboxes and the types of controllers that can be applied are analyzed.

The fourth chapter implements and explains different approaches, comparing the performances of both the Fuzzy controller and the PID controller based on their responses and error rates.

The fifth and final chapter, concludes the work and drawn the future perspectives, highlighting what was and was not possible to achieve in the face of all the adversities encountered, as well as the experience gained from this final course project.

Chapter 2

State of Art

2.1 Introduction

Controlling systems is essential to optimize several industrial processes. For that, the PID are widely used due to their simplicity and effectiveness, but they can have limitations in nonlinear systems. To overcome these difficulties, Fuzzy controllers offer a more adaptive approach, better handling uncertainties, dynamic variations and a more friendly-user implementation.

In this work, it was developed a study to analyze the improvements introduced by Fuzzy controllers, when applied to systems with complex behaviors and time delay in their responses. The process considered for this analysis is a thermal system.

Thermal diffusion systems model the propagation of heat or cold and are described by differential equations. Studying these techniques allow us to understand their applications and choose the best approach for each scenario.

2.2 Heat diffusion system

A heat diffusion system [4] is a concept in thermodynamics that describes how thermal energy spreads within a material or between different materials. It refers to a physical system where heat transfer occurs due to a temperature difference, always moving from regions of higher temperature to lower temperature until thermal equilibrium is achieved. While thermodynamics [5] focuses on the overall energy exchange, engineering applications require analyzing the rate at which heat transfer occurs, as this influence the system's efficiency and performance.

Heat transfer in such systems happens through three fundamental mechanisms: conduction, convection, and radiation. These modes govern how energy propagates in different media, as illustrated in Figure 2.1.

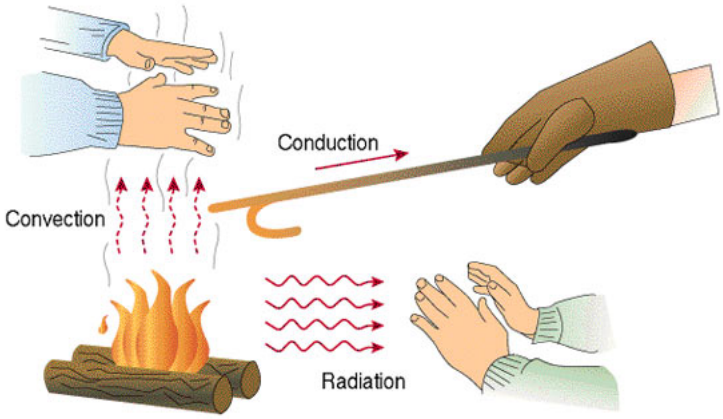


Figure 2.1: Heat Transfer Modes [6]

Applications of heat diffusion systems are critical in various industries. In electronics [7], efficient heat dissipation is essential for preventing overheating in devices like Control Processing Unit (CPU). In building design, heat diffusion aids in insulation and optimizing heating and cooling systems. Automotive engineering [8] relies on heat management in engine cooling to prevent failure.

In manufacturing, heat diffusion controls processes like metal casting and plastic molding. In renewable energy [9], solar thermal systems use it to distribute heat for water heating and power generation. It also plays a key role in data center climate control and in geothermal systems for harnessing Earth's natural heat.

These applications demonstrate the importance of heat diffusion in enhancing efficiency across different sectors.

2.2.1 Heat transfer modes - Conduction

Conduction [10] is the transfer of heat through a stationary medium, which may be a solid, liquid, or gas. It occurs due to interactions between more energetic and less energetic molecules or atoms. In gases, conduction occurs as high-energy molecules collide with lower-energy ones, transferring kinetic energy and creating a heat flux toward cooler regions. In liquids, the process is similar, but intermolecular forces play a greater role due to the closer spacing of molecules. In solids, conduction happens through lattice vibrations in non-metals, where atomic motion propagates thermal energy, and through free electron transport in metals, where highly mobile electrons make metals excellent heat conductors (Figure 2.2).

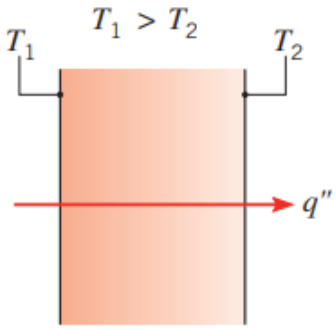


Figure 2.2: Heat transfer mode - Conduction [11]

The fundamental equation governing heat conduction is Fourier's Law, which states:

$$q'' = -kA\frac{dT}{dx} \tag{2.1}$$

where in equation 2.1:

- q'' is the rate of heat transfer (W)
- k is the thermal conductivity of the material ($\frac{W}{m \cdot k}$)
- A is the cross-sectional area perpendicular to heat flow (m^2)
- $\frac{dT}{dx}$ is the temperature gradient along the direction of heat transfer (K/m)

The negative sign indicates that heat flows from higher to lower temperature regions. For steady-state conduction through a uniform plane wall of thickness L, the equation simplifies to:

$$q'' = kA\frac{T1 - T2}{L} = kA\frac{\Delta T}{L} \tag{2.2}$$

where *T1* and *T2* are the temperatures of the two surfaces.

2.2.2 Heat transfer modes - Convection

Convection [10] is the transfer of heat through a solid material without the movement of the material itself. Unlike conduction, where heat moves through direct contact of particles, convection relies on the bulk motion of the fluid. This process is particularly important in natural phenomena like ocean currents, weather systems, and heat dissipation in engineering applications, such as cooling systems for electronics. It can occur in two primary forms: forced convection, where an external force drives the fluid motion (e.g., using a fan or pump), and natural convection, where fluid motion is caused by buoyancy forces

due to temperature-driven density changes. Thus, in Figure 2.3 it is possible to understand the importance of convection in thermal management and energy transfer systems.

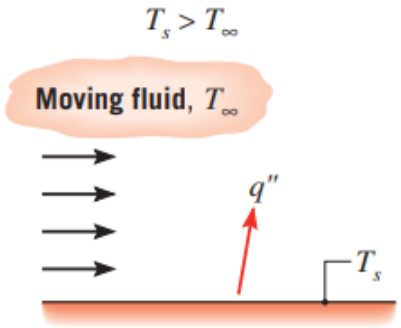


Figure 2.3: Heat transfer mode - Convection [11]

Newton's Law of Cooling describes convection heat transfer as:

$$q'' = hA(T_s - T_\infty) \tag{2.3}$$

where in equation 2.3:

- q'' is the rate of heat transfer (W)
- h is the convection heat transfer coefficient ($\frac{W}{m^2 \cdot K}$)
- A is the surface area through each heat is transferred (m^2)
- T_s is the surface temperature (K)
- T_∞ is the temperature of the surrounding fluid (K)

2.2.3 Heat transfer modes - Radiation

Radiation [10] is the transfer of heat in the form of electromagnetic waves. Differently from conduction and convection, radiation does not require a medium and can occur in a vacuum, such as the heat from the sun reaching the Earth. As shown in Figure 2.4, all objects emit radiation based on their temperature, with hotter bodies emitting more radiation at shorter wavelengths. The amount of radiation emitted depends on the object's surface properties and temperature, and it follows laws like the Stefan-Boltzmann law. Radiation plays a significant role in numerous applications, including solar energy systems, infrared heating, and heat dissipation from hot objects. It is also vital in fields like astrophysics and space exploration, where the vacuum of space requires heat transfer to occur solely through radiation.

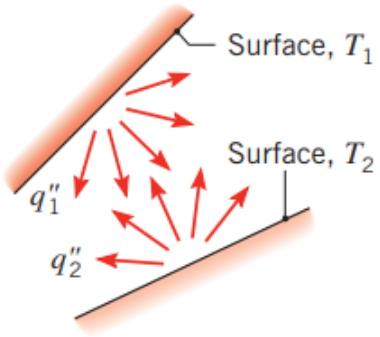


Figure 2.4: Heat transfer mode - Radiation [11]

The rate of thermal radiation emitted by a perfect blackbody is given by the Stefan-Boltzmann Law:

$$q'' = \varepsilon \sigma A T^4 \tag{2.4}$$

where in equation 2.4:

- q'' is the heat transfer rate due to radiation (W)
- σ is the Stefan-Boltzmann constant ($5.67 \times 10^{-8} W/m^2K^4$)
- A is the emitting surface area (m^2)
- T is the absolute temperature of the surface (K)

Emissivity ($0 \leq \varepsilon \leq 1$) represents the efficiency of the surface emitting thermal radiation. When two surfaces exchange radiation, the net heat transfer between them is:

$$q'' = \varepsilon \sigma A (T_1^4 - T_2^4) \tag{2.5}$$

therefore, T_1 and T_2 are the absolute temperature of the two surfaces exchanging radiation.

2.2.4 The First Law of Thermodynamics

The Laws of Thermodynamics provide fundamental principles governing energy interactions in physical systems. The First Law of Thermodynamics [12, 13] states that *"energy can neither be created nor destroyed, it can only change forms"* and this is often called the principle of energy conservation. Mathematically, for any system undergoing a process, the change in total energy (increase or decrease) is equal to the difference between the total energy entering in the system and the energy leaving the system. Considering that energy can be transferred into or out of a system through heat, work, and mass flow, and

recognizing that the total energy of a simple compressible system comprises internal, kinetic, and potential energies, the energy balance for any system undergoing a process can be written as:

$$\Delta E_{system} = E_{in} - E_{out} \qquad \text{(J)} \tag{2.6}$$

Energy is a property, meaning its value remains constant unless the system's state changes. Consequently, if the system's state remains unchanged throughout the process, there is no energy change ($\Delta E_{system} = 0$). In this case, the energy balance reduces to the following equation :

$$\dot{E}_{in} = \dot{E}_{out} \tag{2.7}$$

In steady operation, the rate of energy transfer to a system is equal to the rate of energy transfer from the system as shown in Figure 2.5

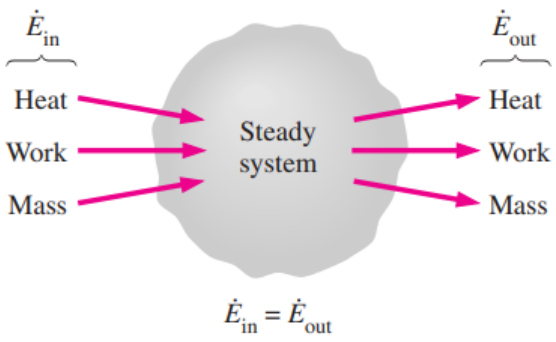


Figure 2.5: Energy balance [12]

When the effects of electricity, magnetism, motion, gravity, and surface tension are negligible, the total energy change of a system during a process reduces to the change in its internal energy ($\Delta E_{system} = \Delta U_{system}$).

In heat transfer analysis, the primary focus is on energy forms that transfer due to a temperature difference - heat or thermal energy. In such cases, it is useful to express a heat balance, considering the conversion of nuclear, chemical, and electrical energies into thermal energy as heat generation. The energy balance in this scenario can be written as:

$$\underbrace{Q_{\text{in}} - Q_{\text{out}}}_{\text{net heat transfer}} + \underbrace{E_{\text{gen}}}_{\text{heat generation}} = \underbrace{\Delta E_{\text{thermal, system}}}_{\text{thermal energy of the system}} \qquad \text{(J)} \tag{2.8}$$

2.3 General heat equation

A primary goal in conduction analysis is to determine the temperature distribution within a medium, based on the conditions applied to its boundaries. Moreover, we aim to understand how temperature varies with position in the material. Once the temperature distribution is known, the conduction heat flux at any point in the medium or on its surface can be calculated using Fourier's law. Additionally, other key quantities of interest can also be determined. For solids, the temperature distribution can be used to assess structural integrity by analyzing thermal stresses, expansions, and deflections [11, 12]. That being said, the heat diffusion equation in Cartesian coordinates is given by:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+q=\rho c_p\frac{\partial T}{\partial t} \tag{2.9}$$

Furthermore, if the thermal conductivity is constant the heat equation is then described as:

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}+\frac{q}{k}=\frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.10}$$

where $\alpha=\frac{k}{\rho c_p}$ is the thermal diffusivity. Equation 2.10 is known as the Fourier-Biot equation [14], and it reduces to these forms under specified conditions:

1. Steady-state (**Poisson equation**):

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}+\frac{\dot{q}}{k}=0 \tag{2.11}$$

2. Transient, no heat generation (**Diffusion equation**):

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}+\frac{q}{k}=\frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.12}$$

3. Steady-state, no heat generation (**Laplace equation**):

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}=0 \tag{2.13}$$

2.4 Classic Controllers

Classical control theory forms the foundation of control system design, offering well-established methods for regulating dynamic systems. These techniques, developed primarily in the early to mid-20th century, focus on Single Input Single Output (SISO) systems and are characterized by their frequency-domain and time-domain analysis tools. Among the most widely used classical controllers is the PID controller, renowned for its simplicity, robustness, and adaptability to diverse industrial applications such as temperature regulation, motor speed control, and process automation. The PID controller's three term structure, proportional (P) for immediate error response, integral (I) for eliminating steady-state error, and derivative (D) for damping oscillations enables effective control across many linear systems.

Despite their strengths, classical controllers exhibit limitations when applied to complex modern systems. Their reliance on Linear Time Invariant (LTI) models makes them less effective for highly nonlinear, time-varying, or Multiple Input Multiple Output (MIMO) systems scenarios where modern state-space or adaptive control methods excel. Additionally, classical designs often require manual tuning and may struggle with unmodeled dynamics or parametric uncertainties. Nevertheless, their intuitive design, computational efficiency, and proven reliability ensure their continued dominance in industries where system dynamics are well-understood and performance requirements align with SISO paradigms [1, 15].

2.4.1 The Feedback Principle

The concept of feedback is both simple and remarkably powerful, serving as a cornerstone of technological advancement. The principles of feedback have fueled significant progress in control systems, leading to numerous innovations and patents. At its core, feedback involves adjusting the manipulated variable in response to changes in the process variable. If the process variable drops below the desired setpoint, the manipulated variable is increased, whereas if the process variable rises above the setpoint, the manipulated variable is decreased.

This process is referred to as negative feedback, as it ensures that the manipulated variable acts in opposition to the process variable, thereby stabilizing the system. Feedback control systems are commonly depicted using block diagrams, where the controller and the process are linked in a closed-loop configuration (Figure 2.6). The feedback signal is combined with the input signal at a specific junction, often represented by a summation symbol, with a sign-inversion block highlighting the negative feedback mechanism [2, 16].

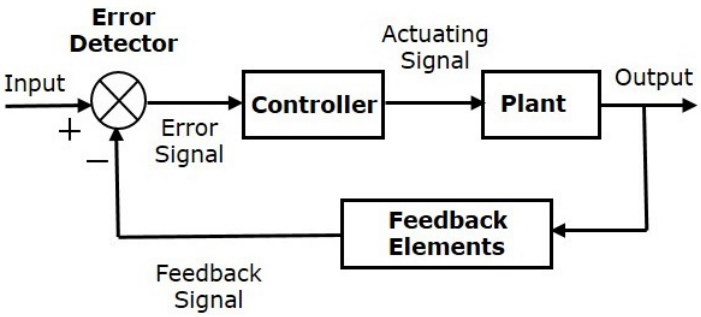


Figure 2.6: Block diagram of a system with feedback [17]

2.4.1.1 On-Off Control

The feedback control can be implemented in various ways, with one of the simplest one being the On-Off control, which can be mathematically represented as:

$$u = \begin{cases} u_{max}, & \text{if } e > 0 \\ u_{min}, & \text{if } e < 0 \end{cases}$$

In this context, $e = y_{sp} - y$ defines the control error, which is the difference between the setpoint, y_{sp} , and the actual process output, y. This form of control applies the maximum corrective action at all times. Specifically, when the error is positive, the manipulated variable is set to its highest value, and when the error is negative, it is set to its lowest value (Figure 2.7).

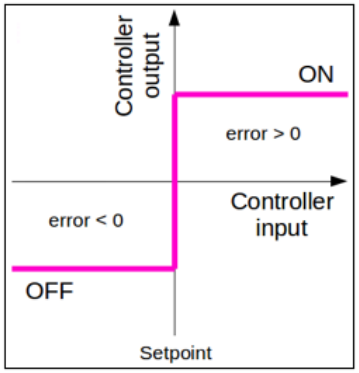


Figure 2.7: Setup of an On-Off controller called "hunting" [18]

While this control method is straightforward and eliminates the need for parameter tuning, it frequently leads to oscillations around the setpoint because of constant switching between states [2]. To mitigate this issue and minimize excessive switching, enhancements such as dead zones and hysteresis are commonly introduced (Figure 2.8).

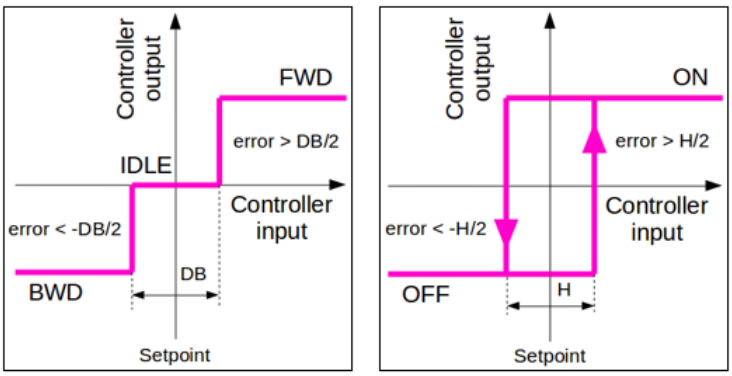


Figure 2.8: On-Off control with dead zone and hysteresis implementation [19]

2.4.2 PID Control

PID controllers are one of the most widely used control algorithms in industrial automation. Their reliability, ease of implementation, and versatility across diverse applications, have established them as a cornerstone of control engineering. PID controllers operate by continuously modifying control inputs to minimize the error between the desired setpoint and the measured process variable.

These controllers are extensively used in industries such as process control, including applications like temperature regulation and robotic motion control. They are typically implemented in standalone controllers and embedded systems [1, 2, 20]. The PID algorithm can be expressed in the time domain as:

$$u_{sp}(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{2.14}$$

and in the Laplace domain, by this equation:

$$U(s) = \left(K_p + \frac{K_i}{s} + K_d s\right)E(s) \tag{2.15}$$

here, u is the control variable and e is the steady-state error. The control variable is calculated as the sum of three components: the proportional (P) term, which is directly proportional to the error; the integral (I) term, which incorporates the accumulated error over time; and the derivative (D) term, which reflects the rate of change of the error [1, 2]. A detailed explanation of each one of these terms is provided below.

2.4.2.1 Proportional term

As the foundational element of a PID controller, the proportional (P) term produces a control signal that is directly proportional to the instantaneous error, defined as the difference between the desired setpoint $u_{sp}(t)$) and the measured process variable ($y(t)$). This relationship is mathematically expressed as:

$$f(t) = Ke(t), \qquad \text{where: } e(t) = u_{sp}(t) - y(t) \tag{2.16}$$

In the Laplace domain:

$$F(s) = K_p E(s) \tag{2.17}$$

here, K represents the proportional gain, and $e(t)$ is the error at time t. The proportional term offers an immediate response to the error. A higher proportional gain results in a stronger reaction to deviations from the setpoint, which can reduce steady-state error and enhance the system's responsiveness. However, if the gain is set too high, it may lead to oscillations or even cause instability in the system over time [2, 15, 21].

In industrial controllers, the proportional term is commonly replaced by the term "proportional band". This term refers to the percentage change in error required to move the actuator to its full scale. It is related to the gain through the following equation:

$$\text{proportional band } (\%) = \frac{100}{K_p} \tag{2.18}$$

2.4.2.2 Integral

The integral (I) term overcomes the limitation of the proportional term system accounting for the accumulated error over time. It integrates the error throughout the duration and generates a control signal that is proportional to the total accumulated error. This term can be written as:

$$f(t) = \frac{K}{T_i} \int_0^t e(\tau) d\tau \tag{2.19}$$

In the Laplace domain:

$$F(s) = \frac{K_i}{s} E(s) \tag{2.20}$$

here, T_i is the integral time constant. The integral term plays a crucial role in eliminating steady-state errors, which arise when the proportional term alone is insufficient to drive the process variable to the setpoint. By continuously integrating the error over time, the integral term ensures that even small, persistent errors are gradually corrected. However, if not tuned appropriately, the integral term can lead to a slower system response. Moreover, if it accumulates too much error, it can cause the system to overshoot or oscillate [2, 15, 22].

2.4.2.3 Derivative

The derivative (D) term enables the prediction of future error by considering the current rate of change of the error. It produces a control signal proportional to the derivative of

the error, which helps to dampen the system's response and minimize overshoot. This term can be expressed as:

$$f(t) = KT_d \frac{de(t)}{dt} \tag{2.21}$$

In the Laplace domain:

$$F(s) = K_d s E(s) \tag{2.22}$$

In this context, T_d denotes the derivative time constant. This component is particularly beneficial in systems where the process variable undergoes rapid changes. By anticipating future errors, the derivative term introduces a damping effect that minimizes oscillations and improves system stability. However, because the derivative term relies on the rate of change of the error, it is highly susceptible to high-frequency noise in the measurement signal. To mitigate this issue, the derivative term is typically filtered or constrained in practical applications [2, 15, 22].

By integrating all three components of a PID controller — proportional (P), integral (I), and derivative (D) — effective control can be achieved across a diverse range of processes. Proper tuning of these components is essential for optimal performance. Understanding their distinct roles — the P term provides an immediate response, the I term eliminates steady-state errors, and the D term enhances damping and stability — is critical for designing robust and efficient control systems [2, 15, 21, 22].

2.4.2.4 The effects of PID gains

The effects of PID gains directly influence a system's response, impacting its stability, speed and accuracy in reaching the desired setpoint. To complement what was previously discussed in the previous subsection 2.4.2, the table 2.1 shows how the change of PID parameters affects the system response.

Table 2.1: Effects of PID Gains on System Response

Closed-Loop Response	**Rise Time (t_r)**	**Overshoot (%)**	**Settling Time (t_s)**	**Steady-State Error (e_{ss})**	**Stability**
Increasing K_p	Decrease	Increase	Small Increase	Decrease	Degrade
Increasing K_i	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing K_d	Small Decrease	Decrease	Decrease	Minor Change	Improve

2.4.3 Open-Loop vs Closed-Loop

Control systems are generally classified as Open-Loop or Closed-Loop, depending on whether feedback is used or not. The main difference is that Closed-Loop systems monitor their output to make adjustments, while Open-Loop systems do not [1].

Open-Loop control systems are those in which the output does not influence the control action. In other words, the system does not measure or feedback its output for comparison with the input. A common example is a washing machine, where soaking, washing, and rinsing occur based on a set time, without assessing the cleanliness of the clothes.

An open-loop control $G_{(s)}$ system typically consists of two main components: the controller $C_{(s)}$ and the process, as illustrated in Figure 2.9. The system operates by receiving an input signal $X_{(s)}$, which is processed by the controller to generate an actuating signal. This signal then directs the process, influencing the output variable $Y_{(s)}$ to follow a predefined standard [16].

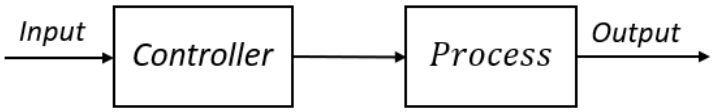


Figure 2.9: Open-Loop control system

Since the output is not compared to the reference input, each input results in a fixed operating condition, making system accuracy reliant on calibration. If disturbances occur, the system cannot adjust, leading to potential errors. Open-loop control is practical only when the relationship between input and output is well-defined and remains unaffected by internal or external disturbances.

Mathematically, the general open-loop transfer function can be described as:

$$Y_{(s)} = [C_{(s)} \cdot G_{(s)}] \cdot X_{(s)} \tag{2.23}$$

where in equation 2.23:

- $X_{(s)}$ is the input of the system
- $C_{(s)}$ is the controller applied
- $G_{(s)}$ is the transfer function process
- $Y_{(s)}$ is the output of the system

Closed-Loop control systems are those in which the output influences the control action through a feedback mechanism [1]. In other words, the system continuously measures its output and compares it with the desired input, adjusting its operation accordingly to minimize errors. A common example is an air conditioning system with a thermostat, where the actual room temperature is monitored and the system adjusts cooling or heating to maintain the desired setpoint.

A closed-loop control system $H_{(s)}$ generally includes three key components: the controller, the process, and the feedback loop, as shown in Figure 2.10. The system begins by receiving an input signal, which the controller processes to produce an actuating signal. This actuating signal is then applied in the process, affecting the output variable. The output is then monitored and compared to the input via the feedback loop, expressed as "Sensor Feedback", enabling the system to make adjustments and maintain the output according to the desired standard [16].

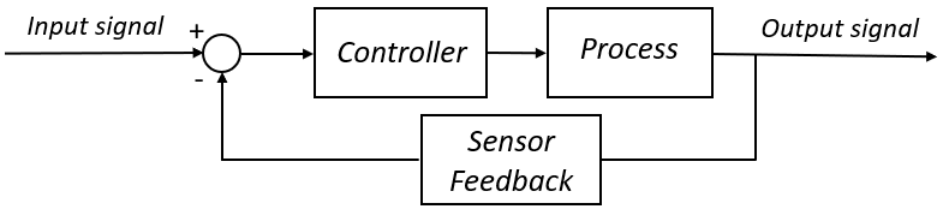


Figure 2.10: Closed-Loop control system

Since that closed-loop control systems are widely used for precise control, the feedback mechanism allows them to correct for disturbances and variations in system parameters, ensuring stable and accurate performance. For example, in industrial applications, closed-loop systems maintain desired conditions like temperature or pressure despite external changes [23].

However, closed-loop systems can be more complex and may suffer from instability or oscillations if not properly tuned. Despite these challenges, they are preferred when accuracy and disturbance rejection are critical, as the benefits of feedback often outweigh the added complexity and cost.

The open-loop block diagrams are a valuable tool in control systems, helping to visualize the signal flow through various components.

Mathematically, the general closed-loop transfer function can be described as:

$$Y_{(s)} = \left[\frac{G_{(s)}}{1 + G_{(s)} \cdot H_{(s)}}\right] . X_{(s)} \tag{2.24}$$

where in equation 2.24:

- $X_{(s)}$ is the input of the system

- $C_{(s)}$ is the controller applied
- $G_{(s)}$ is the transfer function process
- $H_{(s)}$ is the feedback
- $Y_{(s)}$ is the output of the system

2.4.4 Ziegler-Nichols tuning method

The Ziegler-Nichols method is a commonly used approach for tuning Proportional-Integral-Derivative (PID) controller parameters. It provides a structured way to determine the proportional gain (K_p), integral time (T_i), and derivative time (T_d) using experimental data, primarily by analyzing the system's response to a step input [24]. This method includes two main tuning methods, one for open-loop (continuous) processes and another for closed-loop processes.

The **Open-Loop method** is used when the system has no feedback, so the system does not oscillate easily. A step input is then applied to the system, and its output response is observed as shown in Figure 2.11.

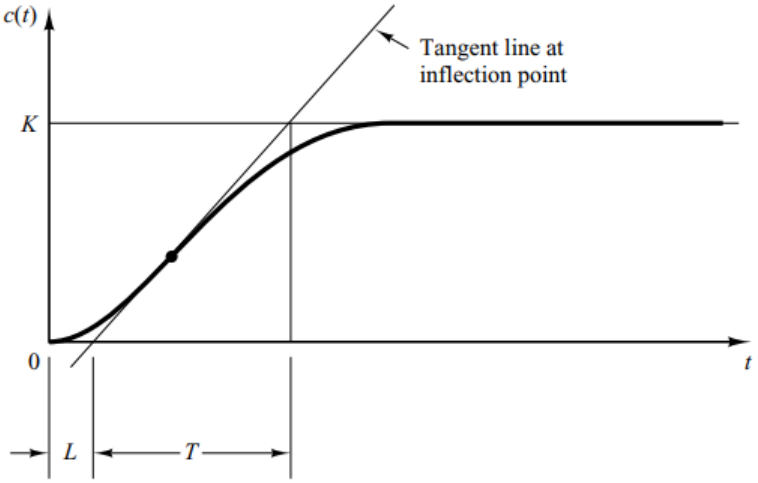


Figure 2.11: Open-Loop step response [1]

Next, the system's reaction to the step input is analyzed by measuring key parameters. The delay time (L) is recorded, which represents the time it takes for the system to begin responding after the input is applied. Additionally, the time constant (T) is measured, which indicates how long the system takes to reach approximately 63% of its final value. These values are essential for determining the appropriate PID controller settings [2].

Based on the delayed time (L) and the time constant (T), it is possible to use those values to calculate the PID parameters as shown in table 2.2

Table 2.2: Open-Loop PID parameters

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

The **Closed-Loop method** is typically used when the system is already operating in a closed loop configuration. First, the integral (K_i) and derivative (K_d) gains are set to zero, leaving only the proportional gain (K_p) active as seen in Figure 2.12. The proportional gain is then gradually increased until the system exhibits sustained oscillations with a constant amplitude. At this point, the proportional gain is referred to as the ultimate gain (K_u), and the time it takes to complete one full oscillation cycle is known as the ultimate period (P_u). These values are crucial for determining the appropriate PID controller settings [1].

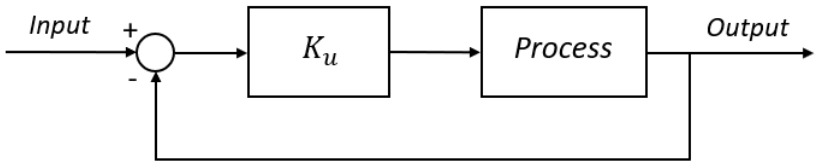


Figure 2.12: Closed-loop system with a proportional controller

Based on the ultimate gain (K_u) and the ultimate period (P_u), it is possible to use those values to calculate the PID parameters that can be seen in table 2.3.

Table 2.3: Closed-Loop PID parameters

Type of Controller	K_p	T_i	T_d
P	$0.5K_u$	∞	0
PI	$0.45K_u$	$\frac{1}{1.2}P_u$	0
PID	$0.6K_u$	$0.5P_u$	$0.125P_u$

2.4.5 First Order System

First order systems (Figure 2.13) are, by definition, systems whose input/output relationship is given by a first order differential equation:

$$\tau\frac{dy}{dt} + y = K_p x_{(t)} \tag{2.25}$$

where:

- $x_{(t)}$ is the input signal
- K_p is the static gain which scales the input's effect on the steady state output
- τ is the time constant that determines the system's response speed
- y is the system output

That being said, first order systems contain a single energy storage element and for that purpose, the order of the input/output differential equation will be the same as the number of independent energy storage elements in the system [23].

When applying Laplace transformation we found equation 2.26:

$$\frac{Y_{(s)}}{X_{(s)}} = \frac{K_p}{\tau_s + 1} \tag{2.26}$$

The result of it, takes place in the following transfer function of a first-order system as shown in Figure 2.13.

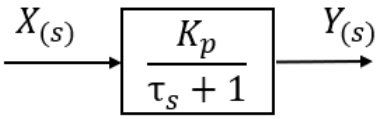


Figure 2.13: Simplified First-Order block diagram

2.4.5.1 First Order System - Step response

The step response of a system refers to its output when subjected to a sudden change in input, typically represented by a step function which can be seen in Figure 2.14. In the context of a first order system, this input is a step from zero to a constant value, and the system's response is the change in its output over time as shown in equation 2.27.

$$X_{(t)} = \left\{ \begin{array}{l} 0 \text{ for } t < 0 \\ 1 \text{ for } t > 0 \end{array} \right\} \tag{2.27}$$

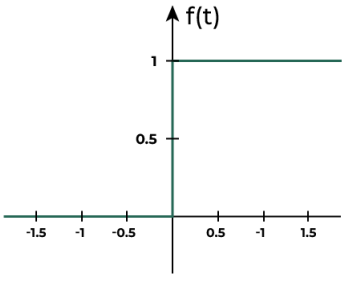


Figure 2.14: Step function response [25]

The step response provides valuable information about the system's dynamic behavior, including its speed of response, stability, and how it reaches a steady state after the input change. Key parameters derived from the step response include the time constant, which characterizes the rate at which the system responds, and the settling time, which indicates how long it takes for the output to stabilize within a certain percentage of its final value [16, 23].

When applying Laplace transformation to a step function, we can define the input step signal as:

$$U_{(s)} = \frac{1}{s} \tag{2.28}$$

Multiplying the previous transfer function (2.26) by the input in the Laplace domain we get:

$$Y_{(s)} = G_{(s)} \cdot U_{(s)} = \frac{K}{\tau_s + 1} \cdot \frac{1}{s} \tag{2.29}$$

Using partial fraction decomposition and inverse Laplace transform, the time-domain solution is:

$$y(t) = K\left(1 - e^{-\frac{t}{\tau}}\right), \quad t \geq 0 \tag{2.30}$$

Some key characteristics of the step response include the initial condition, where the output at time zero is assumed to be zero ($y(0) = 0$), assuming the system starts from rest. The final or steady-state value of the response is $y(\infty) = K$, where K is the output system settles to after the transient effects have dissipated.

The settling time (T_s) refers to the time it takes for the output to fall within $\pm 98\%$ of its final value, which is approximately four times the time constant (4τ). These parameters collectively characterize the speed and stability of the system's response to the step input [26].

To sum up, a first order step response curve has an exponential rise and looks like Figure 2.15.

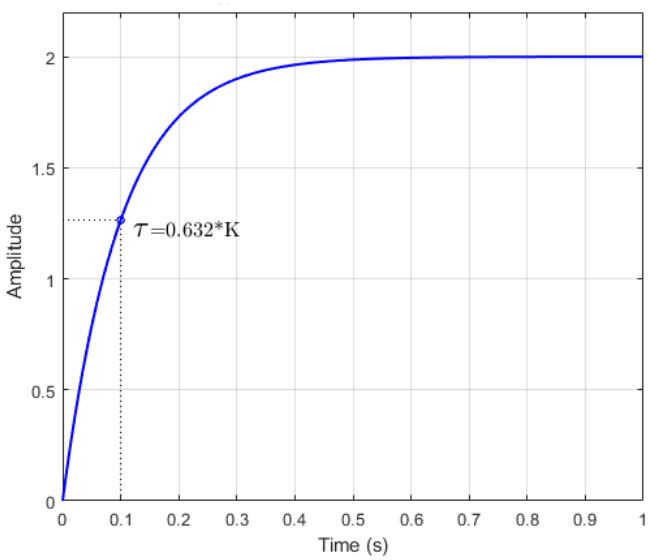


Figure 2.15: First Order system step response

2.4.5.2 First Order System with delay

In practical applications, many dynamic systems experience a delay before reacting to a change in input. This delay, known as dead time (t_d), refers to the time interval between the applied action and the first observable effect. Such delays occurs in various systems, like the thermal systems [4] where heat diffusion takes time to reach sensors or chemical processes [27] where reaction times introduce inherent delay.

The transfer function of a first order delayed system is the same as equation 2.26, but now with a shift in time, where we apply an exponential to perform that delay in the Laplace domain. The exponential term causes the output to be delayed by t_d seconds, meaning that the system remains unresponsive until the delay period has occur. That being said, the transfer function is modified as:

$$G(s) = \frac{K}{\tau s + 1} e^{-t_d s} \tag{2.31}$$

Furthermore, if a unit step (equation 2.28) is applied to the delayed first order system, the output in the time domain is given by:

$$y(t) = \begin{cases} 0, & t < t_d \\ K\left(1 - e^{-\frac{t-t_d}{\tau}}\right), & t \geq t_d \end{cases} \tag{2.32}$$

Before t_d, the output remains zero because the system has not yet started responding. Once $t \geq t_d$, the system exhibits the standard first-order exponential response with the

defined time shift. This delay does not alter the final steady-state value, which remains K, but it extends the time required for the system to reach that value due to the initial response lag [23].

To sum up, the effect of the delay on the system's response is shown in Figure 2.16.

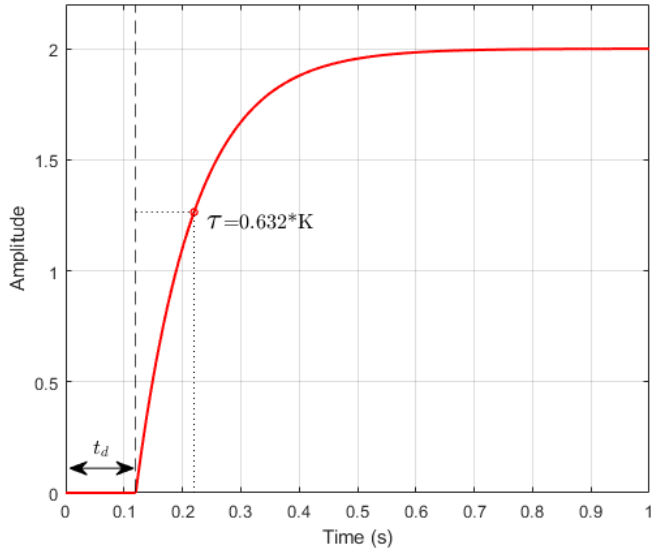


Figure 2.16: First Order delayed system step response

2.5 Fuzzy Logic

Fuzzy logic is an extension of traditional binary logic (true or false) that accommodates partial truths. Unlike classical logic, which operates in absolutes, fuzzy logic allows for degrees of truth, represented by values between 0 and 1. This capability makes it particularly effective for handling uncertainty, imprecision, and ambiguity in real-world systems.

The core of fuzzy logic lies in its use of linguistic variables and membership functions. Linguistic variables are terms like "Cold," "Warm," or "Hot" that describe system states in natural language. Membership functions map input values to degrees of membership in these linguistic categories. For example, a temperature of 25°C might be 0.6 "Warm" and 0.4 "Hot." This representation allows fuzzy logic to model human-like reasoning and decision-making [2].

Fuzzy logic operations, such as AND, OR, and NOT, are applied to these linguistic variables to perform reasoning. These operations enable fuzzy logic to handle imprecise or ambiguous inputs, making it suitable for complex, nonlinear, or uncertain systems where traditional methods may struggle.

2.5.1 Fuzzy Logic applications

Fuzzy Logic has been effectively applied in real-world systems where precision is challenging to achieve. In automotive engineering, it enhances anti-lock braking systems (ABS) and adaptive cruise control by enabling smooth, human-like decision-making [28]. Consumer electronics, such as washing machines and air conditioners, utilize Fuzzy Logic to dynamically adjust operations based on sensor inputs [29]. Healthcare benefits from its diagnostic capabilities, particularly in cancer detection and ECG analysis, where data ambiguity is common [3]. In finance, Fuzzy Logic supports risk assessment, credit scoring, and algorithmic trading by handling uncertain market variables [30]. These applications demonstrate its versatility in managing complex, non-linear systems across diverse fields.

2.6 Membership functions

Membership functions are fundamental components of fuzzy logic systems, serving as mathematical tools that map crisp input values (*e.g.,* temperature, pressure, or speed) to degrees of membership within fuzzy sets. These functions quantify how much an input value belongs to a specific linguistic category, such as "Cold", "Moderate", or "Hot" in the case of temperature (Figure 2.17). By assigning a membership degree between 0 and 1, they bridge the gap between precise numerical data and human-like, qualitative reasoning.

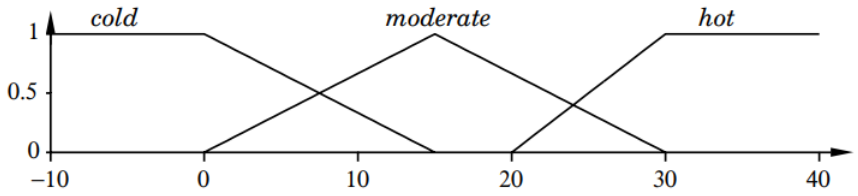


Figure 2.17: Example of Fuzzy membership function [2]

While membership functions can take on various shapes depending on the application and the nature of the system, three types are most commonly used due to their simplicity, computational efficiency, and intuitive interpretation: Triangular, Trapezoidal, and Gaussian (Figure 2.18). Each of these shapes has unique characteristics that make it suitable for different scenarios, and understanding their properties is the key to designing effective fuzzy logic systems [31].

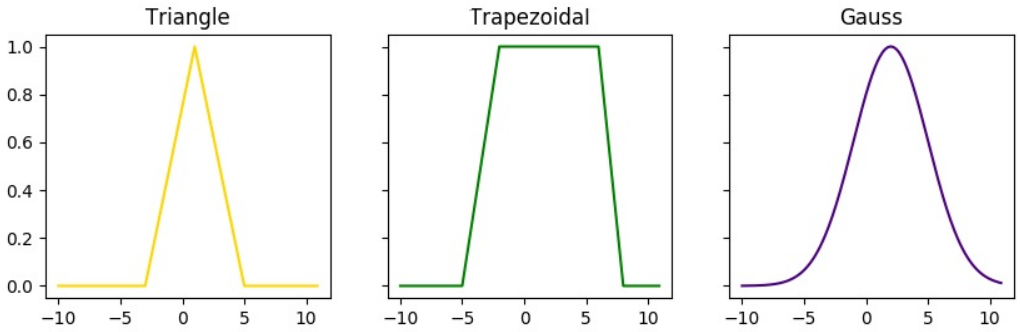


Figure 2.18: Different types of membership functions [32]

2.6.1 Triangular membership function

A triangular membership function [33] is defined by three parameters: the start point a, the peak point b, and the end point c. The membership function (2.33) quantifies the degree to which element x belongs to a fuzzy set A, and is defined as:

$$\mu_A(x) = \begin{cases} 0 & \text{if } x \leq a, \\ \frac{x-a}{b-a} & \text{if } a < x \leq b, \\ \frac{c-x}{c-b} & \text{if } b < x \leq c, \\ 0 & \text{if } x > c. \end{cases} \tag{2.33}$$

For example, for a generic linguistic variable "Warm", the parameters might be set as $a = 20$, $b = 25$ and $c = 30$. In this case:

- At $x = 25$, $\mu_{Warm}(x) = 1$ (fully "Warm")
- At $x = 22$, $\mu_{Warm}(x) = 0.4$ (partially "Warm")

2.6.2 Trapezoidal membership function

A trapezoidal membership function [33] is defined by four parameters: the start point a, the start of the plateau b, the end of the plateau c, and the end point d. The membership function (2.34) quantifies the degree to which element x belongs to a fuzzy set B, and is defined as:

$$\mu_B(x) = \begin{cases} 0 & \text{if } x \leq a, \\ \frac{x-a}{b-a} & \text{if } a < x \leq b, \\ 1 & \text{if } b < x \leq c, \\ \frac{d-x}{d-c} & \text{if } c < x \leq d, \\ 0 & \text{if } x > d. \end{cases} \tag{2.34}$$

For example, for a generic linguistic variable "Comfortable", the parameters might be set as $a = 18$, $b = 22$, $c = 26$ and $d = 30$. In this case:

- At $x = 24$, $\mu_{Comfortable}(x) = 1$ (fully "Comfortable")
- At $x = 20$, $\mu_{Comfortable}(x) = 0.5$ (partially "Comfortable")

2.6.3 Gaussian membership function

A Gaussian membership function [33] is defined by a center c and a width spread σ. The membership function (2.35) quantifies the degree to which element x belongs to fuzzy a set C, and is defined as:

$$\mu_C(x) = e^{-\frac{(x-c)^2}{2\sigma^2}} \tag{2.35}$$

Suppose that we are defining the linguistic variable "Hot", the parameters might be set as $c = 30$ and $\sigma = 5$. In this case:

- At $x = 30$, $\mu_{Hot}(x) = 1$ (fully "Hot")
- At $x = 25$, $\mu_{Hot}(x) \approx 0.6$ (partially "Hot")

2.7 Fuzzy Logic Controllers

A Fuzzy Logic Controller (FLC) is a type of intelligent control system that uses fuzzy logic to map inputs to outputs. It mimics human decision-making by using linguistic rules (*e.g.,* "If the temperature is high, then cool the system down") and fuzzy sets to handle imprecise or uncertain information. Unlike conventional controllers (*e.g.,* PID) that rely on precise mathematical models, fuzzy controllers can handle ambiguity and vagueness in the input data [34].

The block diagram of the fuzzy controller is illustrated in Figure 2.19, depicting the fuzzy controller integrated into a closed-loop control system. The outputs of the plant are represented by $y(t)$, its inputs by $u(t)$, and the reference input to the fuzzy controller by $r(t)$.

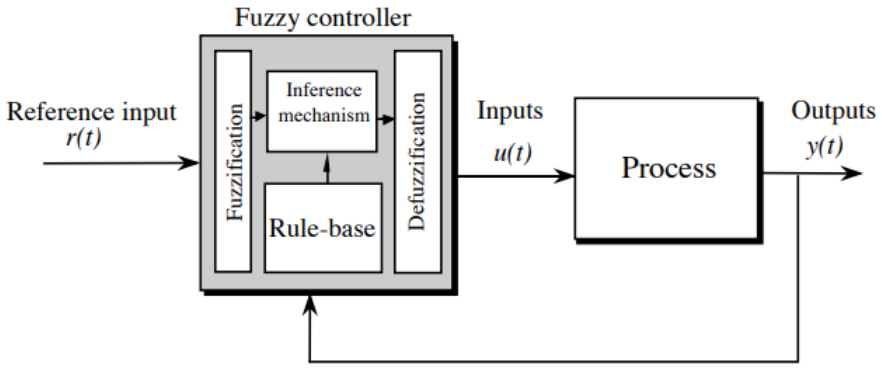


Figure 2.19: Fuzzy controller architecture [28]

Having into consideration the figure 2.19, a fuzzy controller consists of four main components:

1. **Fuzzification**
2. **Rule Base (Knowledge Base)**
3. **Inference Mechanism**
4. **Defuzzification**

2.7.1 Fuzzification

Fuzzification bridges the gap between precise numerical inputs and the linguistic reasoning of fuzzy logic. Unlike classical binary sets, fuzzy sets allow partial membership (*e.g.*, a temperature of 25°C can be "warm" and "slightly hot" simultaneously), enabling systems to model human-like ambiguity.

A fuzzy set A over a universe X is defined by its membership function $\mu_A(x)$, where $\mu_A(x) \in [0, 1]$ quantifies the degree to which x belongs to A [35]. The support of A is the set of points where $\mu_A(x) > 0$:

$$S(A) = cl\{x \in X \mid \mu_A(x) > 0\} \tag{2.36}$$

For a crisp input x_i, fuzzification maps x_i to a fuzzy set A by combining scaled linguistic fuzzy sets $Q_{(x_i)}$ is [36]:

$$A = \bigcup_{i=1}^{n} \mu_i \cdot Q_{(x_i)}, \tag{2.37}$$

here, $\bigcup$ denotes the fuzzy union (point wise maximum membership), μ_i is the membership grade of x_i in $Q_{(x_i)}$, and $Q_{(x_i)}$ represents linguistic terms (*e.g.*, "Warm").

For a better understanding of the fuzzification process, a Python script was made to complement the explanation. Therefore, for the crisp input x_i of 22°C, we have the following parameters shown in table 2.4:

Table 2.4: Membership function parameters and results for $x_i = 22$°C.

Term	**Type**	**Parameters**	$\mu(x_i)$
Cold	Triangular	$a = 4, b = 8, c = 12$	0.00
Moderate	Trapezoidal	$a = 10, b = 15, c = 20, d = 25$	0.60
Warm	Triangular	$a = 20, b = 25, c = 30$	0.40
Hot	Triangular	$a = 25, b = 30, c = 35$	0.00

For $x_i = 22$°C, the membership degrees (Table 2.4) indicate that the temperature is primarily "Moderate" (60%) with a slight contribution from "Warm" (40%), aligning with human intuition. This indicates that 22°C is strongly associated with moderate temperatures while still having a minor warm characteristic. The output is derived from equation (2.37) and visualized in Figure2.20.

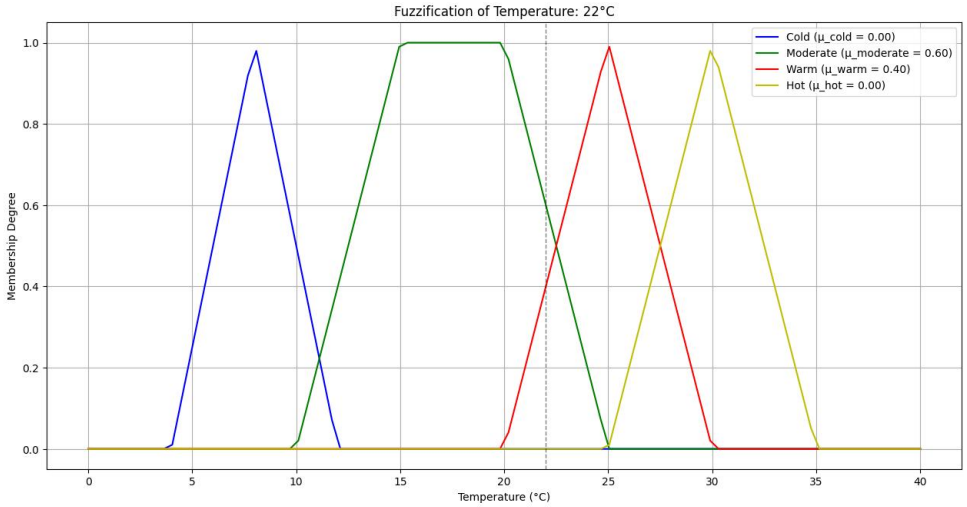


Figure 2.20: Membership functions for temperature. The vertical line marks $x_i = 22$°C, showing its membership degrees (see Table 2.4).

2.7.2 Rule Base

The Rule Base is the cornerstone of a fuzzy control system, serving as the repository of expert knowledge that guides decision-making. Unlike conventional controllers that rely on precise mathematical models, a fuzzy controller operates through a set of linguistic if-then rules, mimicking human reasoning. That being said, these rules define how the system

should respond to various combinations of input conditions, translating vague or imprecise data into actionable control strategies [37]:

$$IF \quad \text{(condition)} \quad THEN \quad (action)$$

The conditions and actions in these rules are expressed using linguistic variables (*e.g.,* "Temperature is High" or "Pressure is Low"), where each term is mathematically defined by membership functions. For example, a temperature of 35°C might belong to the fuzzy set "Hot Temperature" with a membership degree ($\mu_{Hot}(x)$) of 0.8. Rules combine these variables using logical operators:

- The **AND** operator typically uses the minimum of membership values to determine the rule's firing strength (e.g., "IF Temperature is High **AND** Humidity is Low").

$$w = min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), ...)$$

- The **OR** operator employs the maximum of memberships (*e.g.,* IF Temperature is High **OR** Pressure is Low).

$$w = max(\mu_{A_1}(x_1), \mu_{A_2}(x_2), ...)$$

- The **NOT** operator inverts a membership degree (*e.g.,* IF Temperature is **NOT** Low).

$$\mu_{\neg A}(x_i) = 1 - \mu_A(x_i)$$

In essence, the rule base acts as the "brain" of the fuzzy controller, encoding heuristic knowledge, such as an operator's experience or empirical observations, into a structured framework. Each rule evaluates fuzzy inputs *e.g.,* "Temperature is High" or "Pressure is Low" and prescribes corresponding fuzzy outputs *e.g.,* "Increase Cooling" or "Reduce Valve Opening". By aggregating the results of multiple rules, the system can handle complexity, nonlinearity, and uncertainty effectively [28].

2.7.3 Inference Mechanism

The inference mechanism acts as the fuzzy controller's core decision-making process. It transforms fuzzified inputs into fuzzy outputs by systematically applying IF-THEN rules from the Fuzzy Rule Base *e.g.,* "IF temperature is high AND pressure is low, THEN valve opening is medium". Fuzzy Operators (AND/OR), combine input conditions using methods like *minimum* or *maximum*, while the Implication Method adjusts each rule's

output based on antecedent satisfaction.The system then aggregates all rule outputs before defuzzification, enabling effective control under uncertainty [38].

That being said, the inference mechanism evaluates fuzzy rules to map inputs to outputs through a three-step process: **rule evaluation, implication and aggregation** [28, 34]:

1. **Rule Evaluation (Firing Strength Calculation):**

   Each fuzzy rule follows the structure:

   $$IF(x_1 \ is \ A_1) \wedge (x_2 \ is \ A_2) \wedge ..., THEN \ y \ is \ B,$$

   where A_i are the fuzzy sets for the input x_i, B is the fuzzy set for output y and $\wedge$ is the logical operator AND each is typically used for the **minimum** (Mamdani) or **product** (Larsen). Moreover, the **firing strength** (α_r) of rule r quantifies how well the inputs match the antecedents:

   - For **AND** conditions:

     $$\alpha_r = min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), ...) \quad (Mamdani)$$

     or

     $$\alpha_r = \prod_i \mu_{A_i}(x_i) \quad (Larsen)$$

   - For **OR** conditions:

     $$\alpha_r = max(\mu_{A_1}(x_1), \mu_{A_2}(x_2), ...)$$

   For example, for a rule with $\mu_{comfortable}(x1) = 0.4$ and $\mu_{negative\_small}(x1) = 0.6$, the firing strength is $min(0.4, 0.6) = 0.4$

2. **Implication (Consequent Modification):**

   The firing strength α_r modifies the consequent fuzzy set B:

   - **Mamdani Implication:** Clips B at α_r:

     $$\mu_{B_r}(y) = min(\alpha_r, \mu_B(y))$$

     This truncates the output membership function (*e.g.,* a triangular set becomes trapezoidal)

   - **Larsen Implication:** Scales B by α_r:

     $$\mu_{B_r}(y) = \alpha_r \cdot \mu_B(y)$$

     For example, if $\mu_{medium}(y) = 0.8$ and $\alpha_r = 0.4$, Mamdani yields $min(0.4, 0.8) = 0.4$

3. **Aggregation of Rule Outputs:**

   The system combines all modified consequents ($\mu_{B_r}(y)$) into a single fuzzy set using the maximum operator or alternatives like probabilistic sum each can be described as:

$$\mu_{agg}(y) = max(\mu_{B1}(y), \mu_{B2}(y), ...)$$

   For example, if Rule 1 produces $\mu_{B1}(y)$ clipped at 0.4 and Rule 2 at 0.6, the aggregated output is $max(0.4, 0.6) = 0.6$. After aggregation, the system produces a single fuzzy set $\mu_{agg}(y)$ that represents the possible output values. However, since real-world systems require crisp control signals, this fuzzy output must be transformed into a definitive numerical value.

   Thus, Defuzzification, ensures that the fuzzy inference system delivers a precise and actionable control decision.

2.7.4 Defuzzification

Defuzzification is the process of converting fuzzy outputs into precise numerical values, that can be interpreted and acted upon by a system. Since fuzzy controllers operate using fuzzy logic where outputs are represented as degrees of membership in linguistic categories, defuzzification is essential for generating concrete control signals, such as motor speed in RPM or temperature in °C.

This transformation ensures that the output is both actionable and stable, allowing for smooth system behavior. The process involves various defuzzification methods, each employing a different strategy to compute the final crisp value [28, 34, 38]:

1. **Centroid (Center of Gravity - COG):** Computes the weighted average of all possible output values, where the weights are their membership degree and the goal is to find the "balance point" of the fuzzy set.

$$COG = \frac{\int \mu(x) \cdot x dx}{\int \mu(x) dx}$$

2. **Mean of Maximum (MOM):** Averages all output values (x_i) where the membership $\mu(x)$ is maximized and ignores the shape of the fuzzy set. Only looks at the highest confidence values.

$$MOM = \frac{\sum_{i=1}^{N} x_i}{N}, \quad \text{where } \mu(x_i) = max(\mu(x))$$

3. **Weighted Average (for singleton outputs):** Used when each rule outputs a crisp value (singleton) and the combination of the outputs comes from weighing their rule firing strenghts (μ_i).

$$WA = \frac{\sum_{i=1}^{n} \mu_i \cdot x_i}{\sum_{i=1}^{n} \mu_i}$$

4. **Bisector Method:** Finds the vertical line ($x = B$) that splits the area under $\mu(x)$ into two equal halves, each is similar to the COG but just equalizes the area and does not weight by x.

$$\int_a^B \mu(x)dx = \int_B^b \mu(x)dx, \quad \text{where [a,b] is the support of the fuzzy set}$$

5. **Smallest/Largest of Maximum (SOM/LOM):** The SOM picks the leftmost x where $\mu(x)$ is maximized (conservative choice), and LOM picks the rightmost x where $\mu(x)$ is maximized (aggressive choice).

$$SOM = min\ arg\ max(\mu(x))$$

$$LOM = max\ arg\ max(\mu(x))$$

Based on the COG defuzzification method, the following Figure 2.21 shows the result of what previously explain. Since that COG is one of the most used methods, a graphical explanation helps to understand its concept.

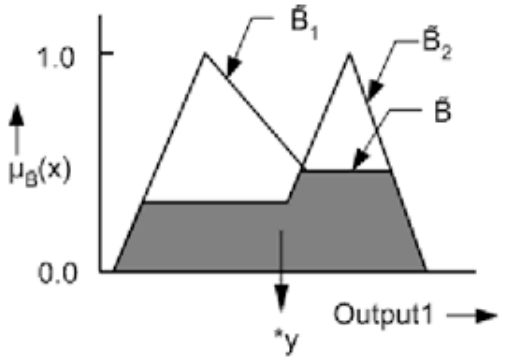


Figure 2.21: COG defuzzyfication method [39]

2.7.4.1 Comparison of Mamdani and Sugeno Fuzzy Rule Structures

Fuzzy rule-based systems employ two primary rule paradigms, Mamdani and Sugeno, each with distinct mathematical structures and applications in SISO and MIMO configurations. The choice between these types depends on the system's complexity, interpretability

requirements, and computational constraints. Below, we analyze their rule structures, operational characteristics, and practical implications. The Mamdani method [40] employs linguistic variables in both antecedents and consequents, mirroring human reasoning. A generic Mamdani rule is expressed as:

$$\text{IF } \bigcap_{i=1}^{n}(x_i \text{ is } A_i) \text{ THEN } y \text{ is } B \tag{2.38}$$

where:

- x_i are crisp input variables (*e.g.,* temperature, humidity),
- A_i are fuzzy sets for the antecedent (*e.g.,* "High", "Low"),
- B is a fuzzy set for the consequent (*e.g.,* "Fast" fan speed),

closely emulates human reasoning through its linguistic framework. For instance, the rule "IF temperature is High AND humidity is High, THEN cooling intensity is Very Strong" first computes a firing strength:

$$w_k = min(\mu_{High}(x_{temp}), \mu_{High}(x_{hum}))$$

that scales the output fuzzy set B. The aggregated output:

$$\mu_{agg}(y) = max(w_1 \cdot \mu_{C_2}(y), w_2 \cdot \mu_{C_1}(y), ..., w_n \cdot \mu_{C_n}(y))$$

requires defuzzification and typically uses the centroid method:

$$y_{crisp} = \frac{\int y \cdot \mu_{\text{agg}}(y)dy}{\int \mu_{\text{agg}}(y)dy} \tag{2.39}$$

This process preserves interpretability but introduces computational overhead due to the integral calculations, making Mamdani systems less ideal for real-time applications. Based on the previous explanation, Figure 2.22 shows the output of two rules applied with Mamdani's method.

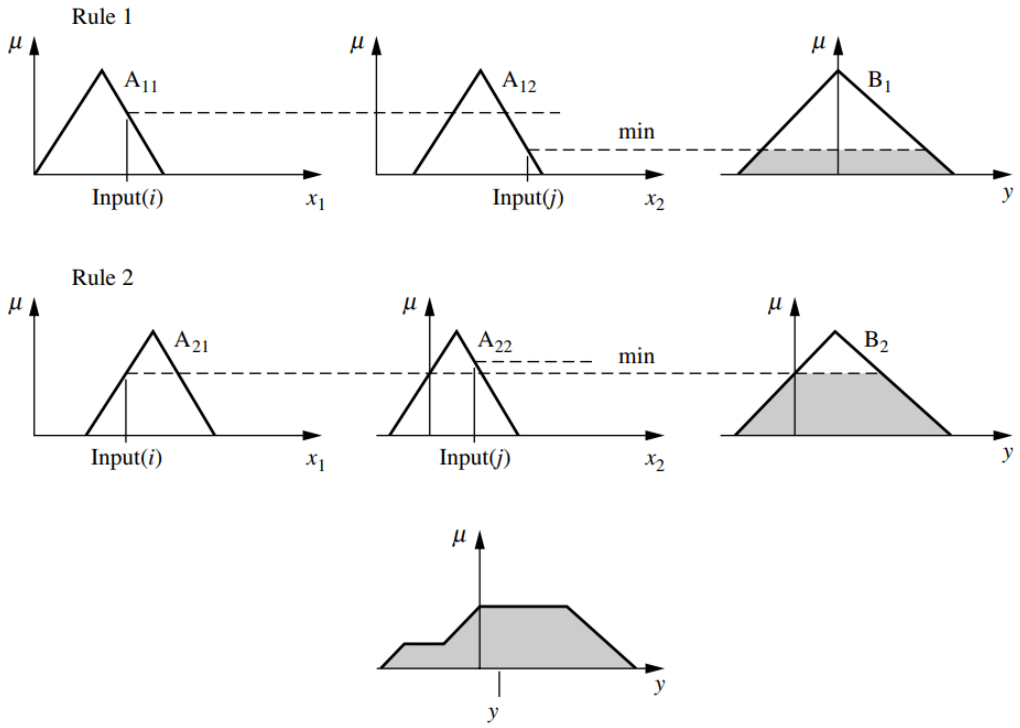


Figure 2.22: Example of Mamdani's method [38]

In contrast, the Sugeno method [41], replaces fuzzy consequents with crisp functions. Its rule structure is formalized as:

$$\text{IF} \bigcap_{i=1}^{n} (x_i \text{ is } A_i) \text{ THEN } y = f(\cdot) \tag{2.40}$$

where $f(\cdot)$ is typically a linear function (*e.g.*, $y_k = p_k x_1 + q_k x_2 + r_k$) or a constant (*e.g.*, $y_k = c_k$). For the same inputs, the system calculates the firing strength identically to Mamdani, but the output is computed as a weighted average:

$$y_{crisp} = \frac{\sum_{k=1}^{n} w_k \cdot y_k}{\sum_{k=1}^{n} w_k} \tag{2.41}$$

where:

- w_k is the rule activation strength
- y_k is the consequent function output

For a better understanding, Figure 2.23 shows the output of two rules applied with Sugeno's method.

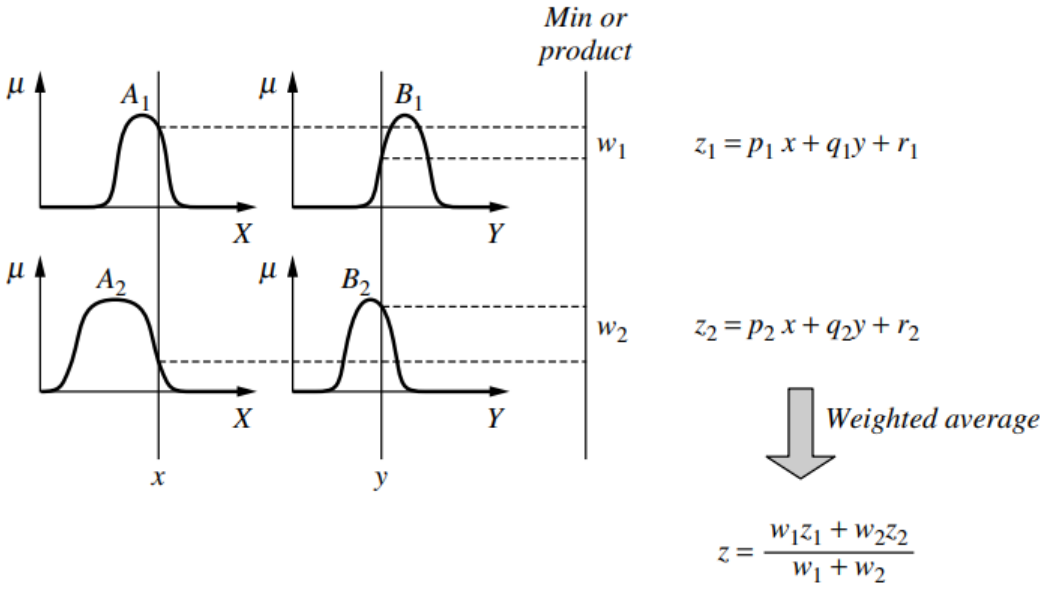


Figure 2.23: Example of Sugeno's method [38]

While both methods share the foundational principles of fuzzy logic, their operational and practical characteristics diverge significantly. To summarize these distinctions and guide method selection, Table 2.5 provides a consolidated comparison across critical dimensions, including computational efficiency, interpretability, and typical application domains.

Table 2.5: Comparative analysis between Mamdani and Sugeno

Criterion	**Mamdani Method**	**Sugeno Method**
Rule Consequent	Fuzzy set (linguistic)	Crisp function (mathematical)
Defuzzification	Required (high cost)	Not needed (low cost)
Interpretability	High (human-readable)	Moderate (requires expertise)
Computational Speed	Slow (integral calculations)	Fast (weighted average)
Optimization	Challenging	Easy (gradient-compatible)
Typical Applications	• HVAC systems • Medical diagnosis • Consumer appliances	• Robotics • Automotive control • Real-time signal processing

Chapter 3

Heat Diffusion System Modeling

3.1 Introduction

This chapter presents the theoretical foundation and the tools used in the modeling of the heat diffusion system. Firstly, it begins with the justification of the first-order delayed system parameters. Secondly, the MATLAB Fuzzy Logic Toolbox was explained as well as the self-tuning mechanisms that a PID controller also provides. Finally, Fuzzy-PID approaches, anti-windup techniques to mitigate integral saturation, and the use of the Smith Predictor as a delay compensation method are also explored.

3.2 System Specifications

The system under study is a heat diffusion whose dynamics were previously discussed in subsection 2.4.5. For simulation purpose and to have a little reminder, the system is represented by the following Laplace-domain transfer function:

$$G(s) = \frac{K}{\tau s + 1} e^{-Ts} \tag{3.1}$$

This transfer function corresponds to a first-order system with a time delay, commonly used to model thermal systems where the output (temperature) responds gradually to changes in input (*e.g.,* heating power) and exhibits a delay due to physical properties like thermal inertia. The system is designed to operate within a temperature range of $15-30°C$, which simulates a practical thermal control scenario such as residential or office heating. This range reflects common indoor temperature requirements and allows for meaningful evaluation of the control system's responsiveness and stability.

The model parameters, gain (K), time constant (τ), and delay (T) were adopted from the article [42] and are given as:

$$K = 0.52, \quad \tau = 162, \quad T = 28$$

3.3 Fuzzy Logic Toolbox

The application of fuzzy logic can be carried out in MATLAB software, using the Fuzzy Logic Toolbox [43] which simplifies designing intelligent control systems by replacing complex math with intuitive, rule-based approaches. In this section, we will explore the Fuzzy Logic Designer (Figure 3.1), a central component of the toolbox. This app can be found in MATLAB Apps area, more specifically in the *Control System Design and Analysis*.

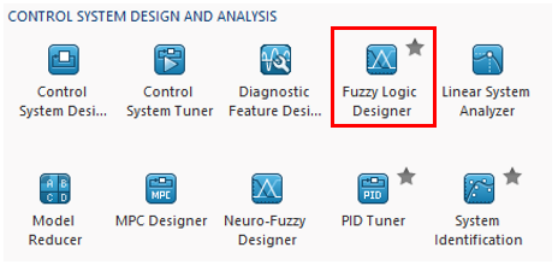


Figure 3.1: Fuzzy Logic Designer app

After initializing the app, the menu presented in Figure 3.2 gives users permission to select one of the four already existing templates: Mamdani Type 1 or 2 and Sugeno Type 1 or 2. If none of them is suitable, it is also possible to create a customized model.

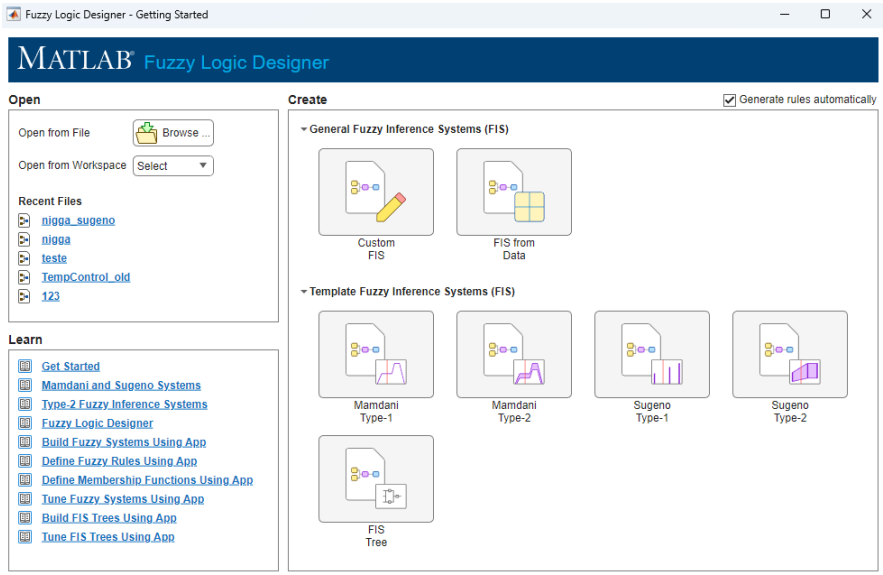


Figure 3.2: Initial menu

For explanation only, it was chosen the Mamdani Type 1 as an example and it provides a generic pre-created model which can be seen in Figure 3.3. This model has two inputs and one output with three membership functions each one, and nine rules for the Fuzzy Inference System (FIS).

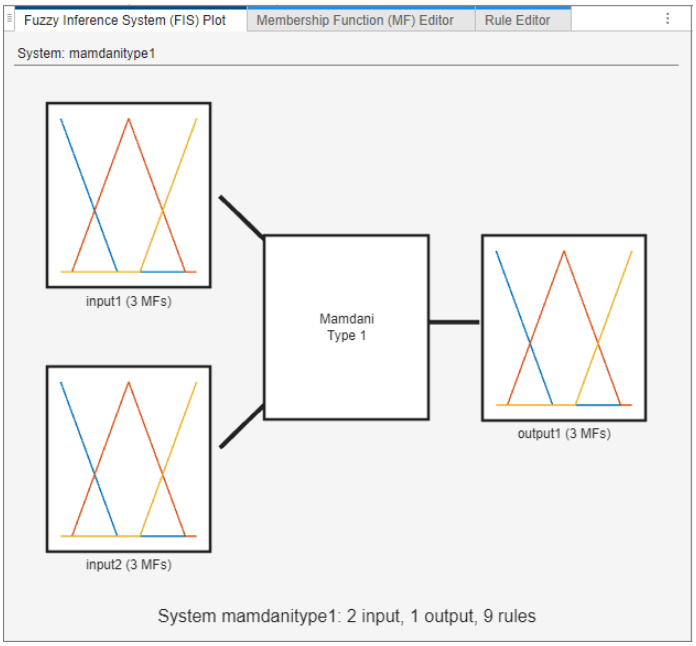


Figure 3.3: Mamdani Type 1 controller general view

If needed, it is possible to adjust/configure the FIS. In Figure 3.4 we can see the menu that comes along with the controller which not only has the different operating modes (AND, OR, implication and aggregation) but also the pretended deffuzification method.

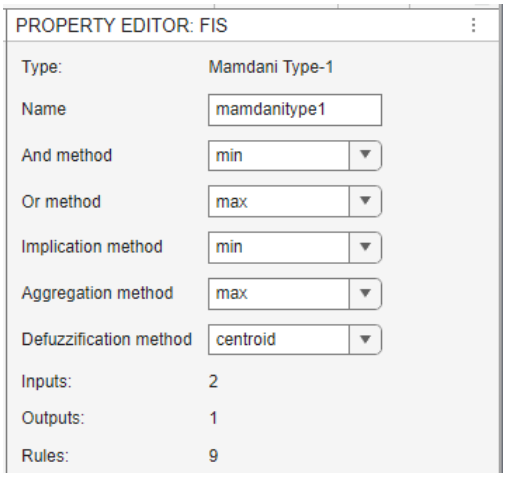


Figure 3.4: Mamdani Type 1 controller property editor

The *Membership Function (MF) Editor* window (Figure 3.5), is used to visualize the membership functions assigned to each system's inputs and outputs.

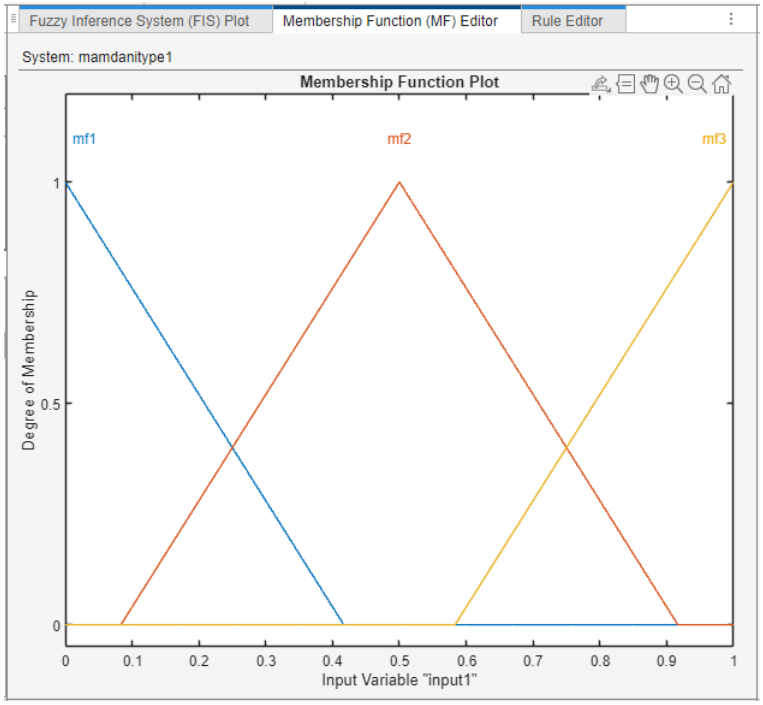


Figure 3.5: Membership functions for the inputs

The distribution values and shapes can also be changed according to what fits bets in the system, and can be configured in the *Membership Functions property editor* which can be seen in Figure 3.6.

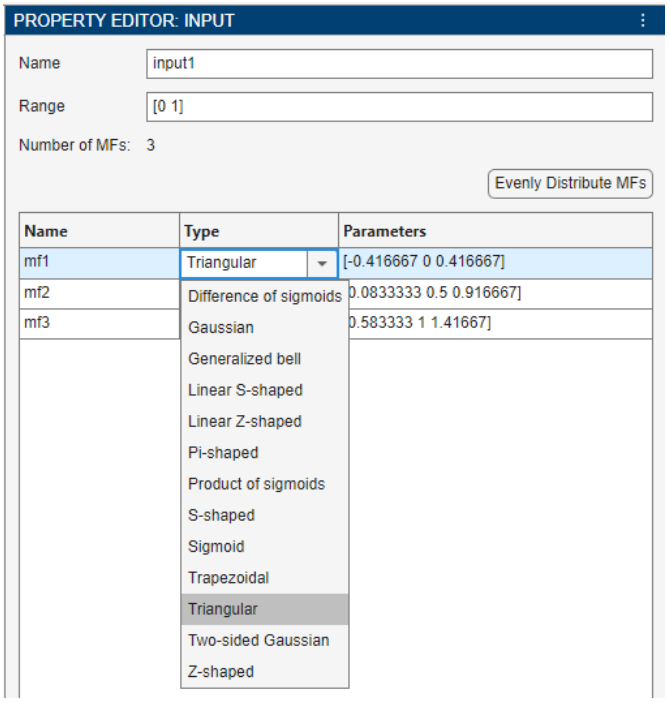
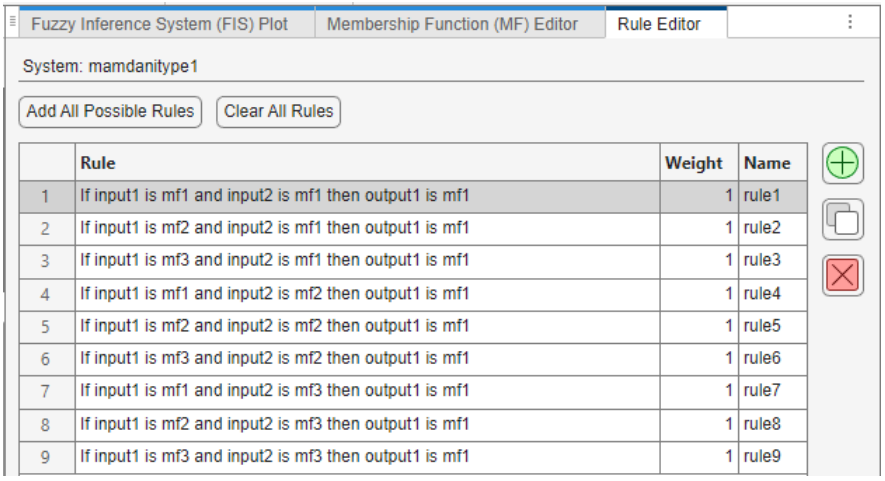


Figure 3.6: Membership functions property editor

The last step to build the FIS is to make the rules base in the *Rule Editor* window. In Figure 3.7 are shown all the possible rules that interact with the membership functions and also the respective weight of each one.



	Rule	Weight	Name
1	If input1 is mf1 and input2 is mf1 then output1 is mf1	1	rule1
2	If input1 is mf2 and input2 is mf1 then output1 is mf1	1	rule2
3	If input1 is mf3 and input2 is mf1 then output1 is mf1	1	rule3
4	If input1 is mf1 and input2 is mf2 then output1 is mf1	1	rule4
5	If input1 is mf2 and input2 is mf2 then output1 is mf1	1	rule5
6	If input1 is mf3 and input2 is mf2 then output1 is mf1	1	rule6
7	If input1 is mf1 and input2 is mf3 then output1 is mf1	1	rule7
8	If input1 is mf2 and input2 is mf3 then output1 is mf1	1	rule8
9	If input1 is mf3 and input2 is mf3 then output1 is mf1	1	rule9

Figure 3.7: Rules Base

Like the previous ones, it is also possible to configure the rules (Figure 3.8) and adjust them as needed, by selecting the rules connector (AND or OR) and change the expected membership function for each input. Finally, the outputs are also assigned to the respective membership function and it is ready to use.

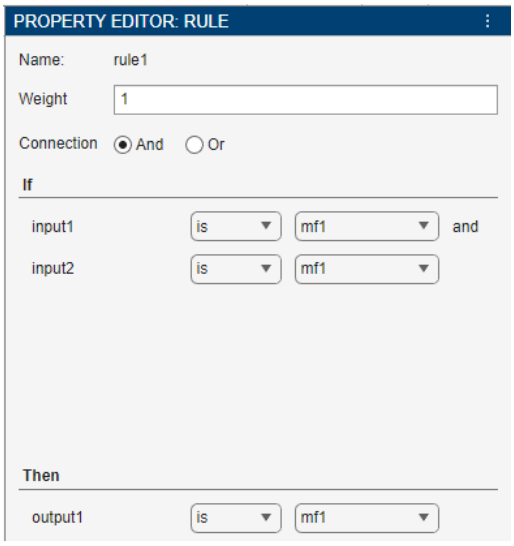


Figure 3.8: Rule Base property editor

After configure the FIS with the previous steps, the user is also able to simulate the inference of each rule and get a control surface representation. For that, just go to the *Simulation* menu that is shown in Figure 3.9.

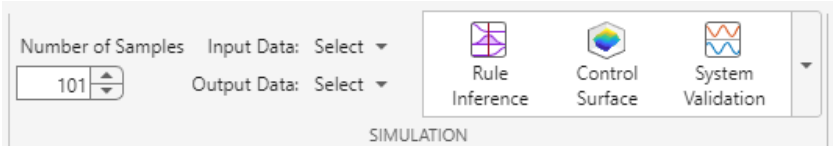


Figure 3.9: Simulation menu

When choosing the *Rule Inference* in the simulation menu, the window in the Figure 3.10 is displayed. All rules are defined based on the input values (Input 1 and Input 2) and their corresponding output. By combining the results of these outputs, the final output set is generated.

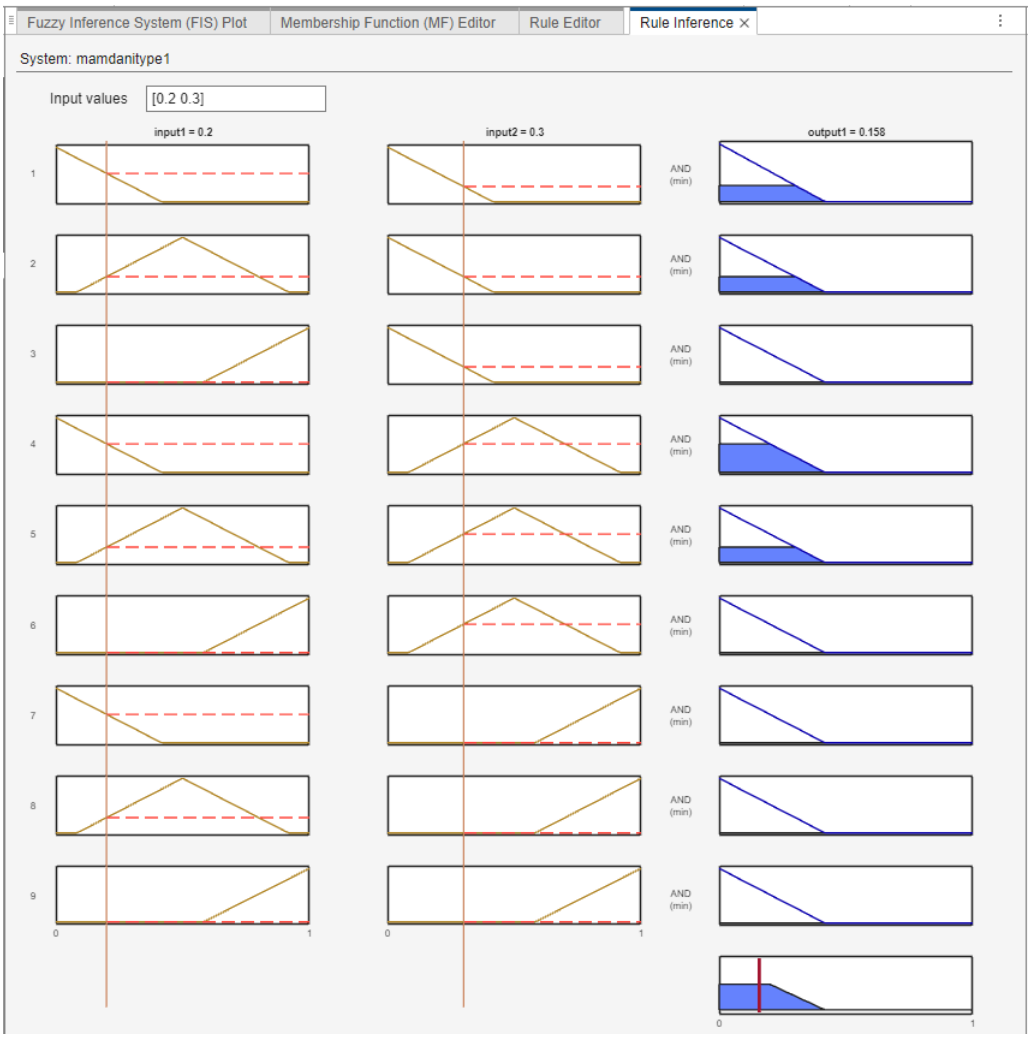


Figure 3.10: Rule Inference process

The *Control Surface* option in the simulation menu, gives a three dimensional perspective of the controlled variable result that can be seen in Figure 3.11.

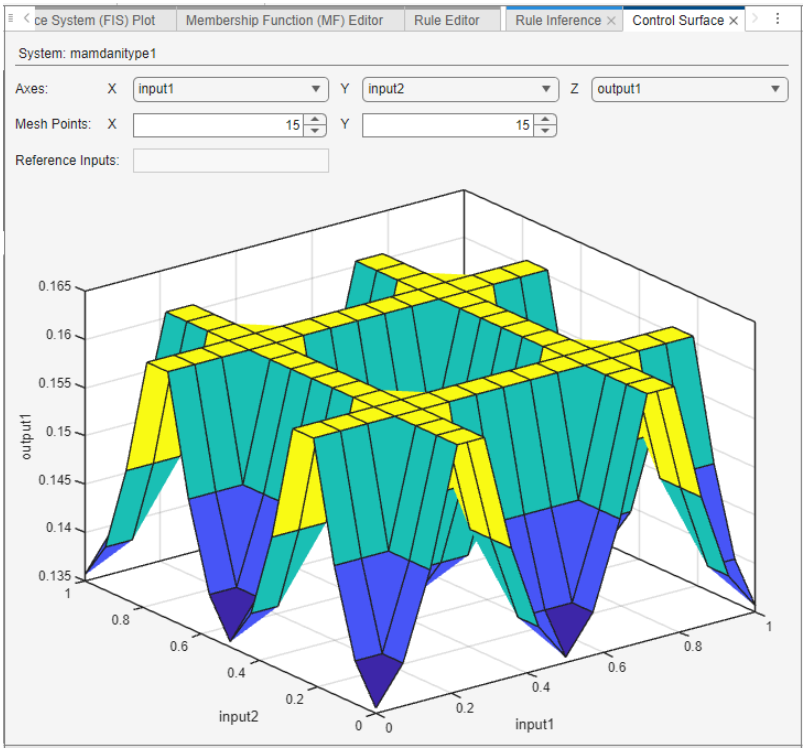


Figure 3.11: Control Surface

To sum up, the Fuzzy Logic Toolbox offers a user-friendly and intuitive environment for designing and simulating fuzzy controllers, making it easier to develop, test, and refine control strategies.

3.4 Tune PID Controller with MATLAB

To obtain the optimal tuning parameters for the controller, the PID Tuner application from MATLAB was used [44]. The app provides an interactive interface where the user can balance between fast response and robustness using a sliding knob. The resulting controller ensures closed-loop stability and improves the transient response. Unlike classical tuning rules like Ziegler–Nichols, MATLAB's PID Tuner uses a model-based optimization approach and it relies on [45]:

- Linearization of the plant around an operating point;
- Frequency-domain optimization which ensures robust stability;
- Internal optimization algorithms to help minimizing the error over time.

That being said, to use the PID Tuner, it is simple as opening the PID block and then click on the "Tune" option as shown in the red rectangle from Figure 3.12. The green rectangle is to chose the desired tuning method: Frequency response or transfer function.

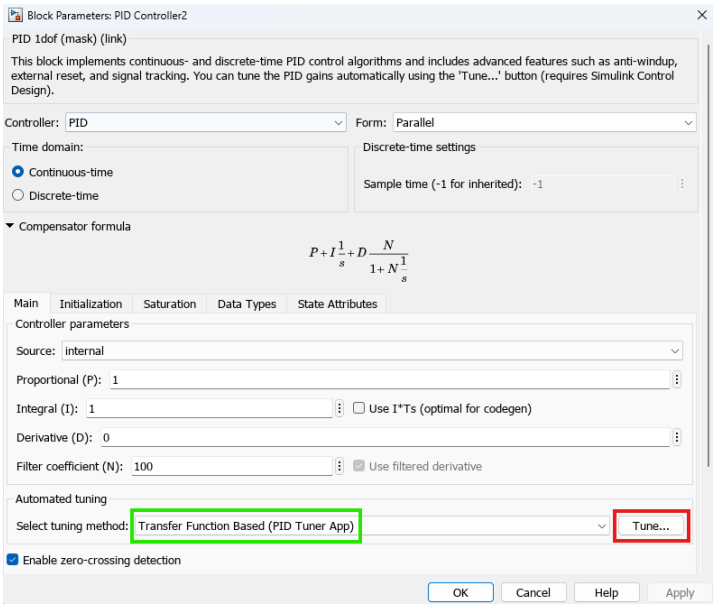


Figure 3.12: PID Tuner option in the PID Controller

When the app is opened, the layout from Figure 3.13 will be presented and it is possible to make all the desired adjustments to the controller and see its response in real time by sliding the knobs related to the "Response Time" and "Transient Behavior".

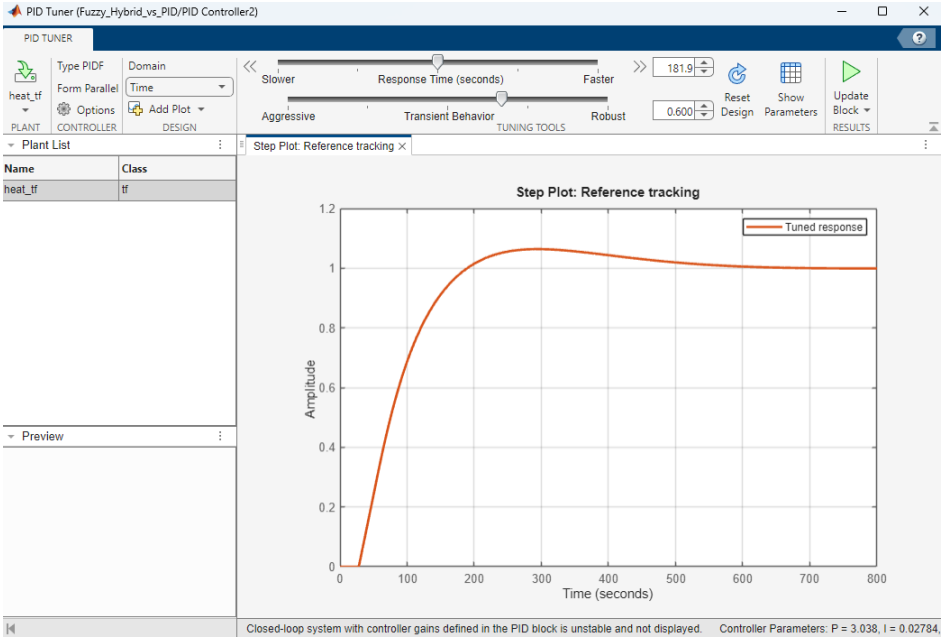


Figure 3.13: PID Tuner layout

Once tuned, the gains can be directly applied to the Simulink model for further analysis and simulation. It is also possible to see the individual contributions of each one, just by clicking on the "Show Parameters". The PID Tuned parameters in this case are:

$$K_p = 1.9489, \quad K_i = 0.012318, \quad K_d = 0.28145$$

3.5 Fuzzy-PID Controllers

Control systems in dynamic environments often face challenges such as non-linearities, parameter uncertainties, and time-varying conditions. As mentioned previously, while classical PID controllers offer simplicity and reliability, their fixed-gain structure limits adaptability in such scenarios. To address this, hybrid approaches combining PID control with Fuzzy Logic, also referred as Fuzzy-PID controllers (Figure 3.14), have emerged [28].

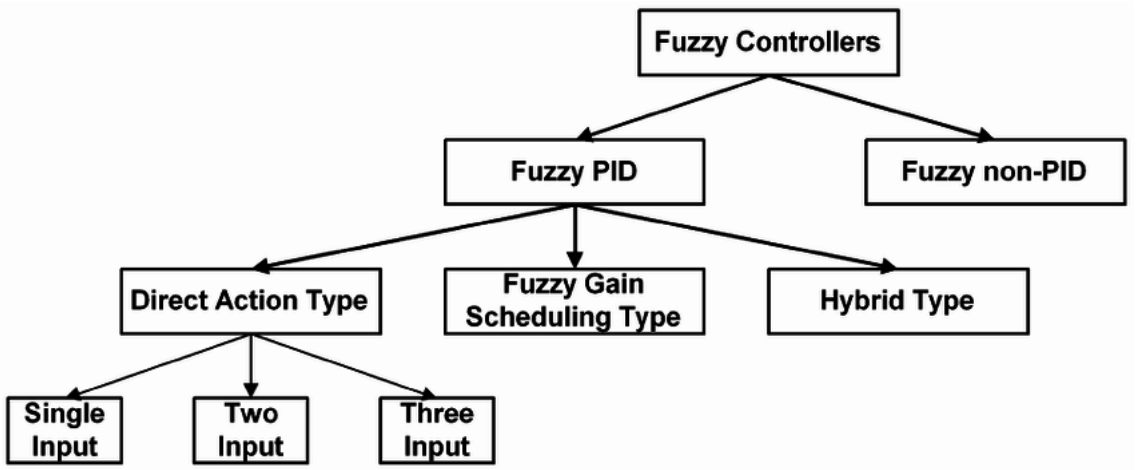


Figure 3.14: Fuzzy Controllers Classification [46]

On the one hand, the Fuzzy non-PID controllers, as the name implies, are controllers that don't follow the classical PID structure and can be designed on empirical or heuristic knowledge of the system. They use fuzzy rules to directly modulate the relationship between the system's inputs and outputs which is particularly useful when a precise mathematical model is not available or when the controlled system has highly complex or unknown dynamics.

On the other hand, the Fuzzy-PID controllers combine the traditional structure of a PID controller with fuzzy logic, allowing a more robust response in complex or uncertain systems. That being said, there are three different categories [47]:

- **Direct Action Type**

  It is the simplest fuzzy controller, where the output is calculated directly from the inputs using fuzzy rules. The controller can have:

1. **One Input:** It only uses one input variable (Figure 3.15) which is normally the error(e) to calculate the output (u). This is the simplest form with low complexity.

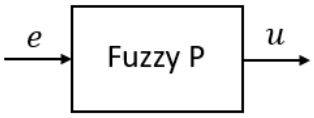
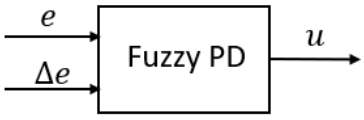
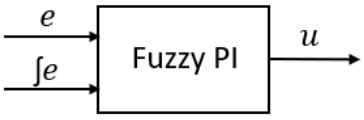


Figure 3.15: One input Fuzzy controller - Fuzzy P

2. **Two Input:** Uses two variables (Figure 3.16), typically the error (e) and the derivative of the error (Δe) for a Fuzzy PD or the error (e) and the integral of the error ($\int e$) for a Fuzzy PI. This format allows faster reactions to system variations because it just not focus on the error itself but also in the way it is changing or incrementing over time.



(a) Fuzzy PD



(b) Fuzzy PI

Figure 3.16: Two input Fuzzy controller

3. **Three Input:** This one is basically complementing the "Two Input" with another variable, which is the integral of the error ($\int e$). This approach aims to fully capture the dynamic behavior of the system and makes it possible to simulate the PID controller with more precision which can be seen in Figure 3.17.

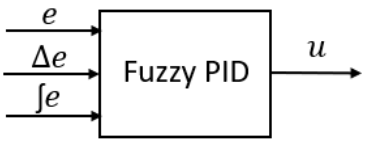


Figure 3.17: Three input Fuzzy controller - Fuzzy PID

For that, two different Fuzzy Logic algorithms exists:

- **Fuzzy Gain Scheduling Type**

  In this case, fuzzy logic does not directly generate the control action, but is used to dynamically adjust the traditional PID controller (Kp, Ki, Kd) gains [48]. Thus,

the PID controller is still present, but its parameters are no longer fixed and become adaptive. This type is useful when the system presents variable behavior over time, allowing the controller to adapt continuously, improving performance and stability. The Figure 3.18 shows a simple block diagram implementation of the controller.

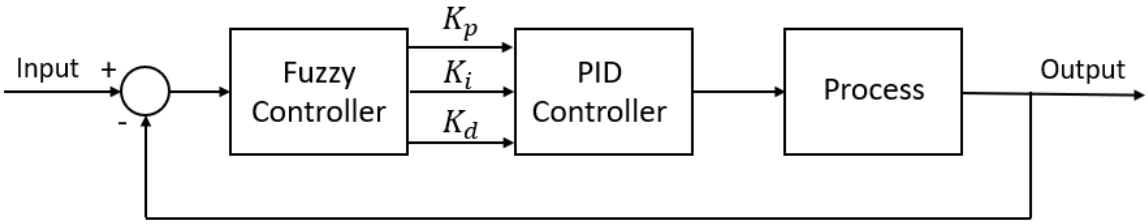


Figure 3.18: Fuzzy Gain Scheduling Type block diagram

The most commons approach is to combine the two inputs, Fuzzy PD or Fuzzy PI, and adjust the K_p, K_i and K_d gains with the fuzzy rules.

- **Hybrid Type**

  In the Hybrid Type [49], the fuzzy controller is combined with other computational intelligence techniques, forming more robust and adaptive hybrid solutions. A typical example is the Neuro-Fuzzy model (ANFIS), where neural networks are used to automatically optimize fuzzy rules and membership functions, which is common in applications such as medical diagnostics or adaptive robotics. Another effective combination is between Fuzzy Logic and Genetic Algorithms (GAs), where GAs automatically adjust the fuzzy rule base and system parameters, which has been successfully applied in traffic control systems, such as smart traffic lights. In the Hybrid Fuzzy-PID, a fuzzy module continuously supervises a conventional PID controller, readjusting its parameters as necessary, which is a common architecture in industrial automation systems where precision and fast response are required. These hybrid approaches offer high flexibility and generalization, and are especially suitable for complex and dynamic systems.

3.6 Anti-Windup Mechanism

Anti-Windup [50] is a technique used in control systems to mitigate the negative effects of actuator saturation, which occurs when the control signal calculated by the controller exceeds the physical limits of the actuator. When a controller (*e.g.,* PID controller) enters saturation mode, the integral term (I) can accumulate excessive error (a phenomenon called "windup"), leading to large overshoots, oscillations, or even instability. Anti-Windup has the block diagram of Figure 3.19 and aims to correct this problem, ensuring smoother and

more stable performance by limiting the accumulation of the integral action outside the valid limits of the control signal.

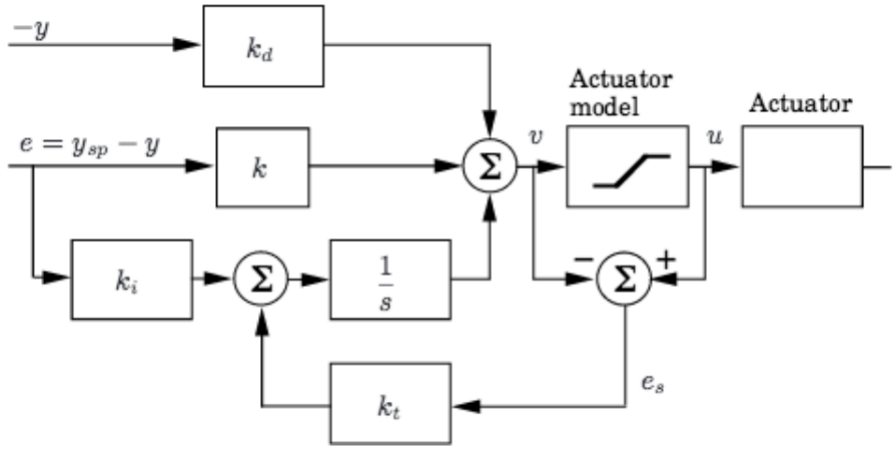


Figure 3.19: Anti-Windup block diagram [50]

That being said, these are the most common approaches to take [51]:

1. **Clamping or Conditional Integration**: The idea is to turn off the integrator when the actuator is saturated, and the error is increasing in that situation. Hence, the integrator only accumulates when the control signal is not saturated or the error helps to "de-saturate" the actuator.

$$\frac{dI(t)}{dt} = \begin{cases} e(t), & \text{if } u(t) = u_{\text{sat}}(t) \text{ and } e(t) \cdot (u_{\text{unsat}}(t) - u_{\text{sat}}(t)) \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

   where $e(t)$ is the error, $u_{unsat}(t)$ is the control singal without saturation and $u_{sat}(t)$ is the saturated signal applied to the system. This one is easier to implement, but may cause some discontinuities.

2. **Back Calculation**: Instead of turning off the integrator, it keeps continuously correcting the difference between the saturated and non-saturated signal. The anti-windup gain K_{aw} forces the integrator to fall back once the actuator is fully saturated, helping to make a softer correction:

$$\frac{dI(t)}{dt} = e(t) + K_{\text{aw}} \cdot (u_{\text{sat}}(t) - u(t)) \tag{3.3}$$

   where $u(t)$ is the controller output without saturation, $u_{sat}(t)$ is the output after the saturation is applied and K_{aw} is the adjustable anti-windup gain. This method is used in a lot of industrial applications and its response is softer than the previous method.

3.7 Smith Predictor

The Smith Predictor [50] is a predictive controller designed to improve the performance of systems that have significant time delay (*e.g.*, control of temperature) in which the response to the control is not immediate and this delay can cause instability or slow responses if not properly compensated for. That being said, the Smith Predictor helps solving this problem by using an internal model of the system to "predict" future behavior compensating for the delay.

To address this, the Smith Predictor introduces an internal model of the plant to anticipate the system's future behavior and separates the plant into two components: the transfer function model $G_{(s)}$, and the delay term e^{-T_s}, as illustrated in Figure 3.20

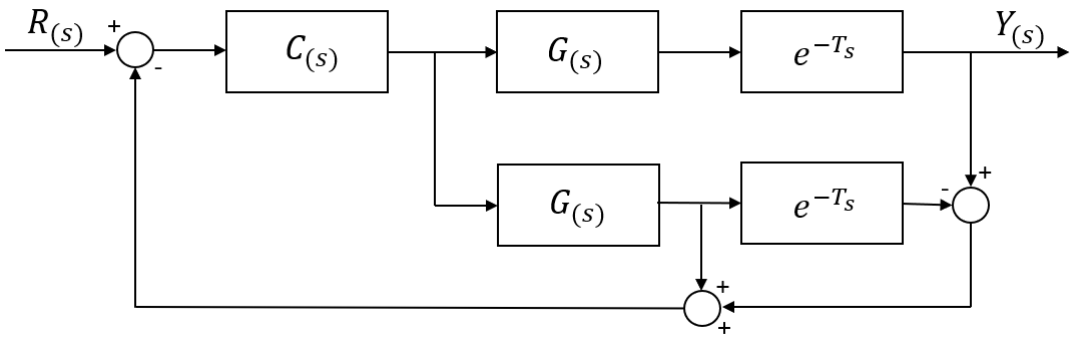


Figure 3.20: Smith Predictor block diagram

By applying the Smith Predictor to the system, it helps improving stability with long delays, has faster response based on the prediction and can be combined with PID controllers for better performance.

3.8 Performance Metrics

Evaluating the error provides insight into how well the system is tuned. That being said, a well-tuned system will exhibit a low error, while poor tuning results in a higher error. The main method of evaluating the system performance is to track the Sum of Squared Errors (SSE) and is given by the following equation [52]:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.4}$$

where:

- y_i is the reference signal (desired setpoint) at sample i,
- $\hat{y}_i$ is the actual system output at sample i,

- n is the total number of samples.

Other applicable indices include those based on the integral error criterion, which aim to minimize the integral error within a closed-loop system and are presented by the following four indexes [53]:

1. **Integral Squared Error (ISE):** It integrates the squared value of the error over time. This criterion penalizes large errors more heavily due to the squaring operation, which can lead to faster system responses but also cause overshoot:

$$I_{ISE} = \int_0^\infty e^2(t)dt \tag{3.5}$$

2. **Integral Time Squared Error (ITSE):** Makes the same as ISE, but includes a time factor multiplier, which means that errors that occur later in time are penalized more. This makes the system to settle quickly and reduces last-longing deviations:

$$I_{ITSE} = \int_0^\infty te^2(t)dt \tag{3.6}$$

3. **Integral Absolute Error (IAE):** It integrates the absolute value of the error over time. Thus, it gives equal weight to all errors regardless of magnitude, resulting in smoother and generally slower responses making it less sensitive to large spikes in error:

$$I_{IAE} = \int_0^\infty |e(t)|dt \tag{3.7}$$

4. **Integral Time Absolute Error (ITAE):** Makes the same as IAE, but includes a time factor multiplier. This criterion is often used in controller design as it tends to minimize settling time and overshoot, which leads to improved transient performance:

$$I_{ITAE} = \int_0^\infty t|e(t)|dt \tag{3.8}$$

Chapter 4

Simulation and Results

4.1 Introduction

This chapter presents the practical results obtained with the implementation and simulation of the studied controllers. By using MATLAB/Simulink, it was possible to compare different control strategies, starting with a more basic analysis and then apply a more robust version of the Fuzzy Controller. To evaluate each implementation, we not only rely on the performance metrics but also on the visual response of the system. Furthermore, a practical case was purposed in order to see the controller response when an entropy is added to the signal, in order to validate its implementation.

4.2 Heat Diffusion System - Step Response Analysis

Based on what was explained previously in the 3.2 subsection, a step response was then performed. Thus, the thermal system presents the behaviour shown in Figure 4.1.

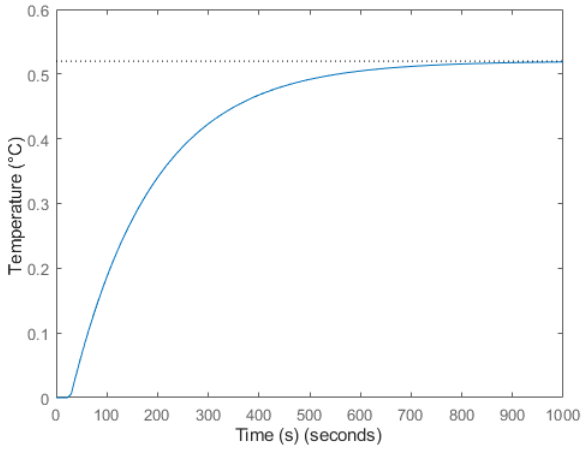


Figure 4.1: Heat system step response

By analysing the step response of this system, it is possible to observe the main dynamic characteristics of a thermal system:

- **Presence of delay:** The system starts its response after some time ($T = 28$ s). This behaviour represents a challenge for control, since any control action has a delayed effect on the output.
- **Slow response:** The system takes several minutes to reach steady state because it has a high time constant ($\tau = 162$ s). This behaviour confirms the thermal systems characteristics, and slow nature of the process.
- **Static gain:** The output stabilizes at a value corresponding to the gain ($K = 0.52$) and the final output value in steady state.

All these factors demonstrate that the dynamics of the system are slow and present a significant delay, thus being typical characteristics of thermal systems and the implementation of the controller needs to have this in consideration.

4.3 Comparison between Fuzzy Gain Scheduling Controller and classical PID Controllers

This section presents a comparative analysis between classical controllers and their corresponding Fuzzy implementation. The objective is to evaluate the performance and advantages that each one offers when applied to systems with time delay. That being said, successive comparisons were made in order to adjust the controllers parameters, based on the errors and its graphical response . For testing and comparison purpose, a temperature of 18°C was used to simulate a comfortable room temperature. For the Fuzzy controller, it was used the same input **"Error"** membership functions for all the P, PI, PD and PID versions which can be seen in Table 4.1. All simulations were carried out over a total duration of 3000 seconds.

Table 4.1: **Error** input values

Name	**Type**	**Parameters**
NegativeLarge	Trapezoidal	[-10 -10 -6 -2]
NegativeSmall	Triangular	[-4 -2 0]
Zero	Triangular	[-0.1 0 0.1]
PositiveSmall	Triangular	[0 1.5 4]
PositiveLarge	Trapezoidal	[3 7 35 35]

The output membership functions for each version (P, PI, PD and PID) were not the same, since it was necessary to do various tests in order to fit the best ranges.

4.3.1 Fuzzy-P vs P Controllers

The first comparison made was only with the proportional gain (P) of each controller as can be seen in the block diagram in Figure 4.2.

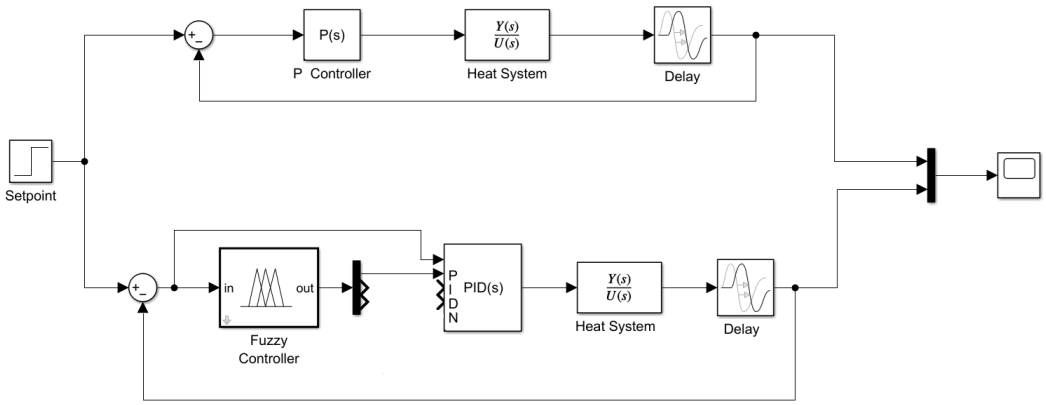


Figure 4.2: Fuzzy-P vs P block diagram

A proportional action (P) provides a control signal that is proportional to the error, which is the difference between the defined setpoint and the output. In systems with delay, a P controller alone may not be sufficient to eliminate the error in steady state, especially if the gain is not very high. On the other hand, if the gain is very high it may still not be able to reach the desired target value and cause overshoot or even instability when the delay is significant. Even though the K_p gain for the P controller were already given by the PID Tuner described in subsection 3.4, several values of K_p were tested to see how the system responds as it increases value. For the Fuzzy-P output membership function K_p, the initial values used were those shown in Table 4.2.

Table 4.2: $\mathbf{K_p}$: Fuzzy-P output values

Name	**Type**	**Parameters**
Low	Trapezoidal	[0.5 0.5 0.8 1.2]
Medium	Triangular	[1 1.5 2]
High	Trapezoidal	[1.8 2.2 2.5 2.5]

The Fuzzy-P rules applied to the controller were the following shown in Figure 4.3.

	Rule	Weight	Name
1	If Error is NegativeLarge then Kp is Low	1	rule1
2	If Error is NegativeSmall then Kp is Low	1	rule2
3	If Error is Zero then Kp is Low	1	rule3
4	If Error is PositiveSmall then Kp is Medium	1	rule4
5	If Error is PositiveLarge then Kp is High	1	rule5

Figure 4.3: Fuzzy-P rule base

Thus, for the response in Figure 4.4, $K_p = 5$ was applied to both controllers, since that the initial values were too small, and the gain was amplified in order to see what it would do to the controllers responses. When analysing the system response and based on the previous explanation, it can be seen that both controllers present a similar response, but a high value of K_p does not solve the problem because, it does not even reach the desired setpoint (18° C) and also presents overshoot, which is not desirable. To help complement this, the performance indices of each controller were taken into account. Thus, Figure 4.5 shows the errors of controller P.

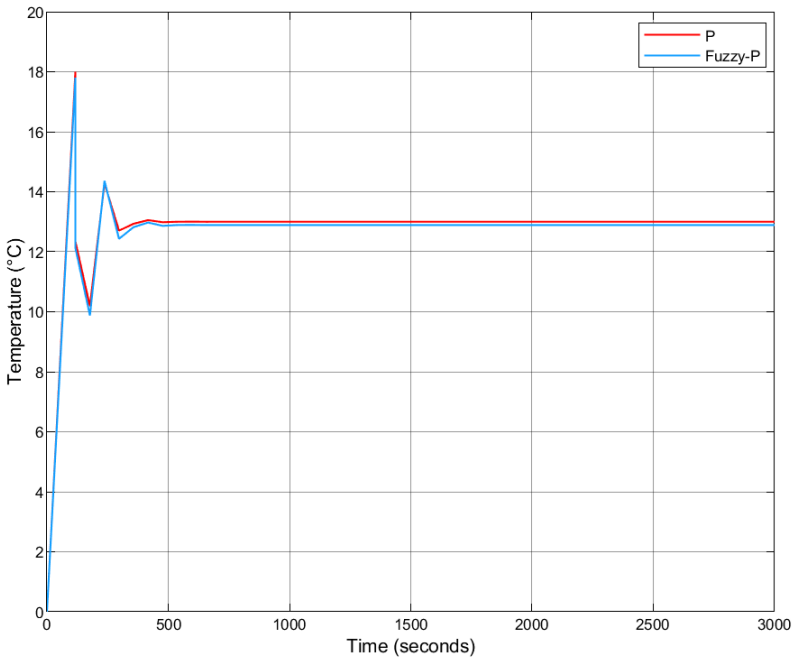
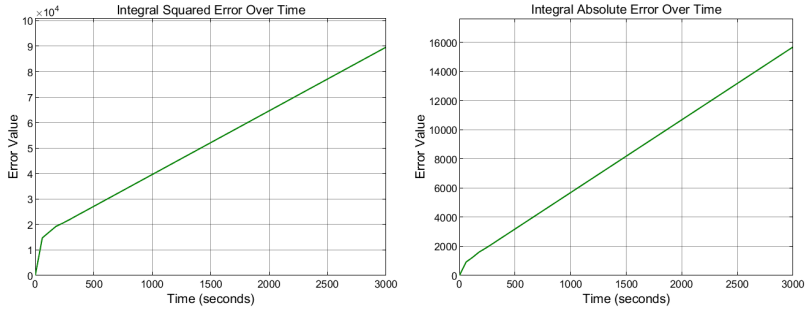


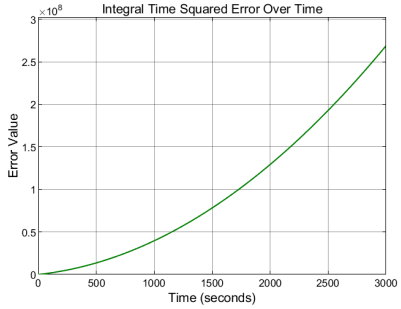
Figure 4.4: Fuzzy-P vs P step responses

By analysing the Fuzzy-P and P controllers, for the ISE or IAE performance metric we can conclude that it shows a constant increase of error over time. This behaviour confirms that the system never eliminates the error and maintains a stationary error. Since that the response obtained with only proportional action was not the most desired, it was then necessary to carry out new experiments to find the best system response.

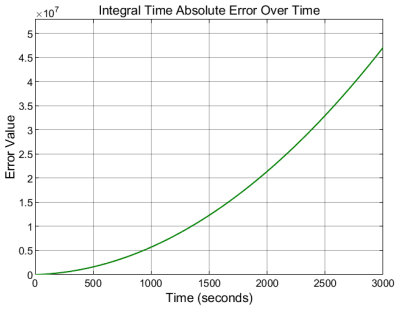


(a) ISE

(b) IAE



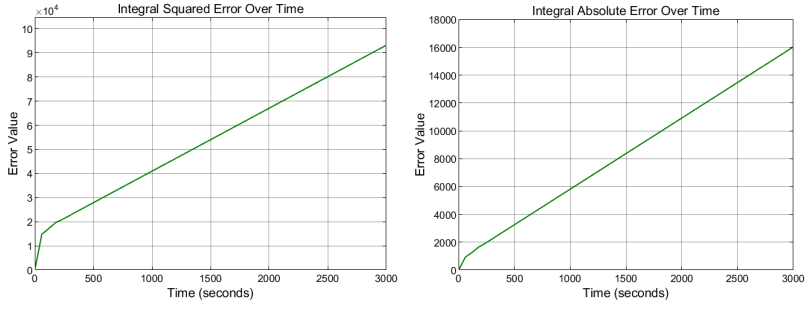
(c) ITSE



(d) ITAE

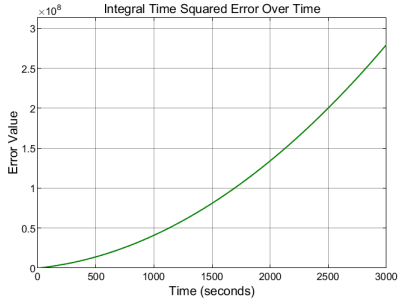
Figure 4.5: P controller performance metrics

The Fuzzy-P controller, it presents the following performance indices in Figure 4.6.

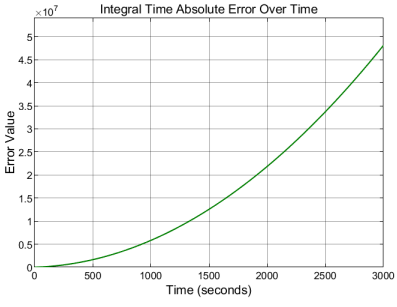


(a) ISE

(b) IAE



(c) ITSE



(d) ITAE

Figure 4.6: Fuzzy-P controller performance metrics

4.3.2 Fuzzy-PD vs PD Controllers

The second comparison made was with the addition of the derivative component (D) in each controller as can be seen in Figure 4.7. The value **"1"** in Figure 4.7 represents the derivative filter coefficient (N) of D component, which can vary. A higher N value results in a faster derivative response but makes the system more sensitive to noise, while a lower N value yields a smoother but slower response. The value of $\boldsymbol{N = 1}$ was used because it had no significant impact on the system's response when compared to higher values (*e.g.*, $N = 100$), and it also resulted in a faster and lighter simulation, avoiding unnecessary computation load.

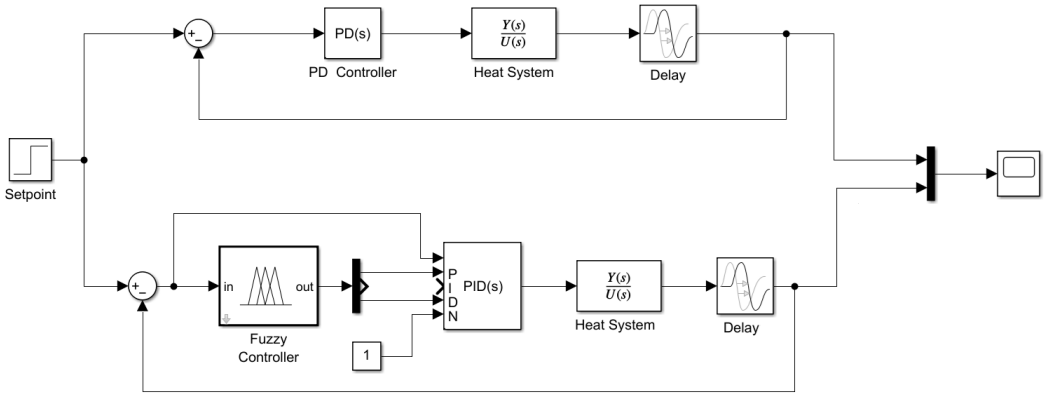


Figure 4.7: Fuzzy-PD vs PD block diagram

A derivative action (D) reacts to the rate change of the error and helps to predict the behaviour of the system, because it can dampen oscillations and improve the stability. In systems with delay it is quite useful the use of this component because if the error is changing very quickly, K_d increases to dampen the response. If the system is already stable, K_d decreases to minimize noise sensitivity. For the Fuzzy-PD output membership function K_d, the initial values used were those shown in Table 4.3.

Table 4.3: $\mathbf{K_d}$: Fuzzy-PD output values

Name	**Type**	**Parameters**
Low	Trapezoidal	[0.1 0.1 0.25 0.45]
Medium	Triangular	[0.35 0.5 0.7]
High	Trapezoidal	[0.6 0.85 1 1.5]

The Fuzzy-PD rules applied to the controller were the following shown in Figure 4.8.

	Rule	Weight	Name
1	If Error is NegativeLarge then Kp is Low, Kd is High	1	rule1
2	If Error is NegativeSmall then Kp is Low, Kd is Medium	1	rule2
3	If Error is Zero then Kp is Low, Kd is Medium	1	rule3
4	If Error is PositiveSmall then Kp is Medium, Kd is Low	1	rule4
5	If Error is PositiveLarge then Kp is High, Kd is Low	1	rule5

Figure 4.8: Fuzzy-PD rule base

Having this into consideration, to reach the values of Table 4.3 several tests were made, and based on the step response of each controller they were adjusted to reach the best possible response which resulted in Figure 4.9. The definition of values $K_d = 0.28145$ for the PD controller were obtained through the PID Tuner previously explained in the subsection 3.4 and it should be noted that the value of K_p remained the same for this part ($K_p = 5$) applied on both controllers.

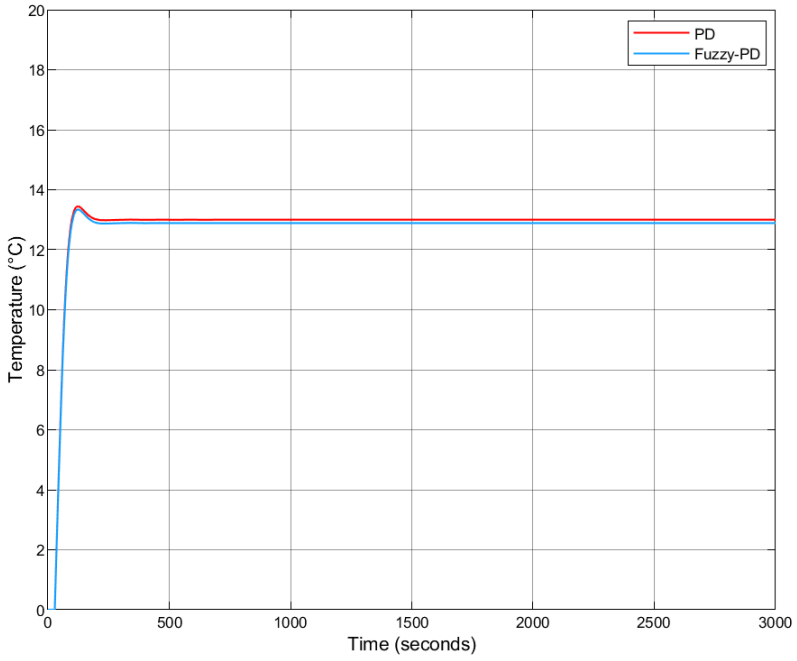
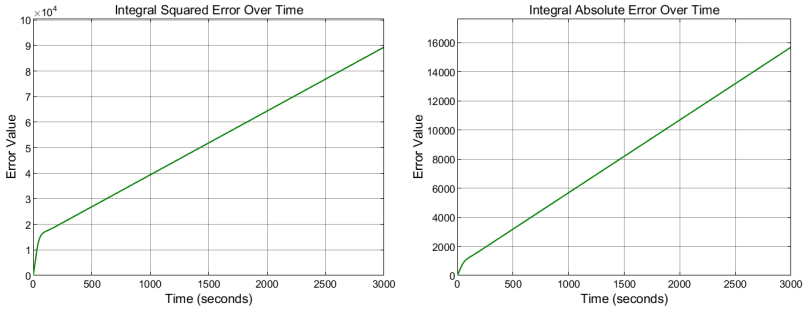


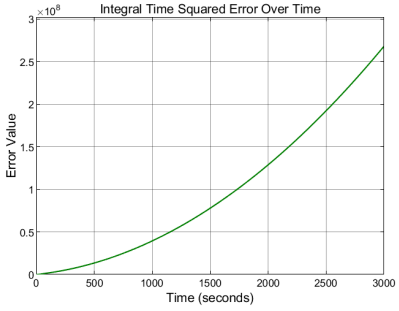
Figure 4.9: Fuzzy-PD vs PD step responses

Based on the step response of each controller (Figure 4.9), it is possible to notice that the derivative action helps to combat the overshoot problems that the proportional action applied, but even so it is still not possible to reach the target value, because we still have a lot of error. To complement the analysis, the performance indexes of each controller were also obtained. Thus, the PD controller presents the errors shown in Figure 4.10.

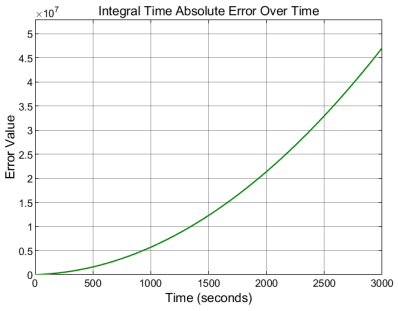


(a) ISE

(b) IAE



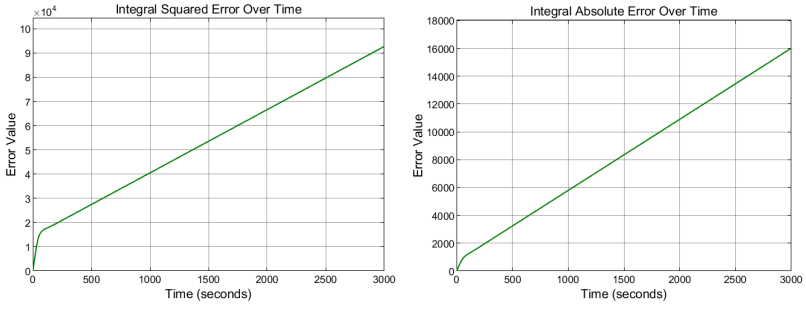
(c) ITSE



(d) ITAE

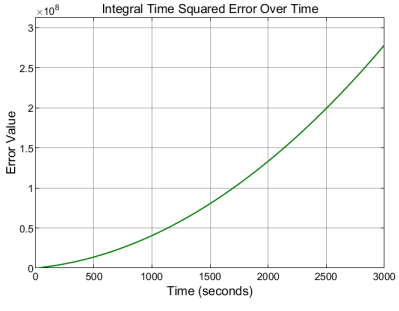
Figure 4.10: PD controller performance metrics

As for the Fuzzy-PD controller, they are the ones illustrated in Figure 4.11.

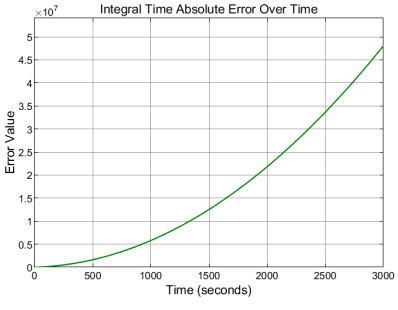


(a) ISE

(b) IAE



(c) ITSE



(d) ITAE

Figure 4.11: Fuzzy-PD controller performance metrics

By analysing the Fuzzy-PD and PD controllers ISE or IAE performance metrics for example, we can conclude that it still shows a constant increase of error over time and still not help to correct this error and has a response similar to the P term alone. Since the response obtained with proportional and derivative actions was still unable to help solve the problem, it was necessary to carry out new experiments with the introduction of new contributions.

4.3.3 Fuzzy-PI vs PI Controllers

The third experiment was to swap the derivative action (D) for the integrative action (I) in each controller, which can be see from the block diagram in Figure 4.12.

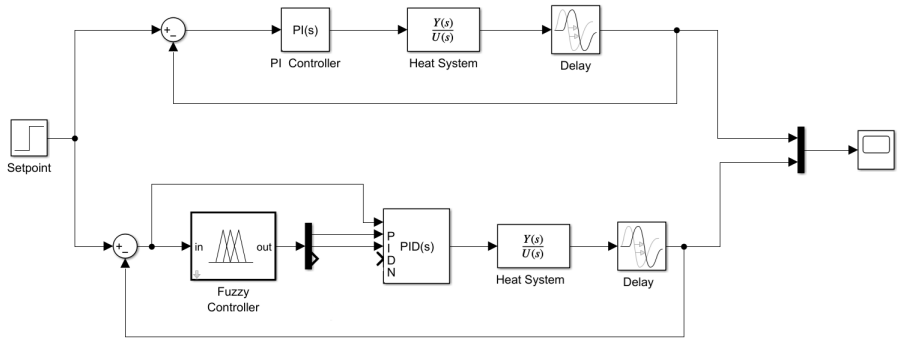


Figure 4.12: Fuzzy-PI vs PI block diagram

The integral term helps to eliminate the steady-state error by accumulating past errors. In a delayed system where the proportional term may not be enough to achieve the desired value, introducing the integral ensures that the error tends towards zero over time. Thus, by increasing K_i the system corrects the error more quickly but can cause overshoot or oscillations as the control continues to accumulate error. On the other hand, decreasing K_i slows down the error correction, so the system tends to be more stable but less responsive. For the Fuzzy-PI output membership function K_i, the initial values used were those shown in Table 4.4.

Table 4.4: $\mathbf{K_i}$: Fuzzy-PI output values

Name	Type	Parameters
Low	Triangular	[0.001 0.01 0.03]
Medium	Triangular	[0.025 0.05 0.075]
High	Triangular	[0.06 0.085 0.1]

The Fuzzy-PI rules applied to the controller were the following shown in Figure 4.13.

	Rule	Weight	Name
1	If Error is NegativeLarge then Kp is Low, Ki is High	1	rule1
2	If Error is NegativeSmall then Kp is Low, Ki is Medium	1	rule2
3	If Error is Zero then Kp is Low, Ki is Medium	1	rule3
4	If Error is PositiveSmall then Kp is Medium, Ki is Low	1	rule4
5	If Error is PositiveLarge then Kp is High, Ki is Low	1	rule5

Figure 4.13: Fuzzy-PI rule base

The values set for $K_i = 0.012318$ for the PI controller were again obtained using the PID Tuner. As for the Fuzzy-PI controller, the values were adjusted based on the system's response until the intervals in the Table 4.4 were reached, which are the ones with the best results. An adjustment that was also made was in the proportional gain membership function that went back to its original values that can be seen in the previous Table 4.2 and the classical controller also went back to its original $K_p = 1.9489$ previously obtained with the PID Tuner. After all the adjustments, the response of Figure 4.14 is obtained.

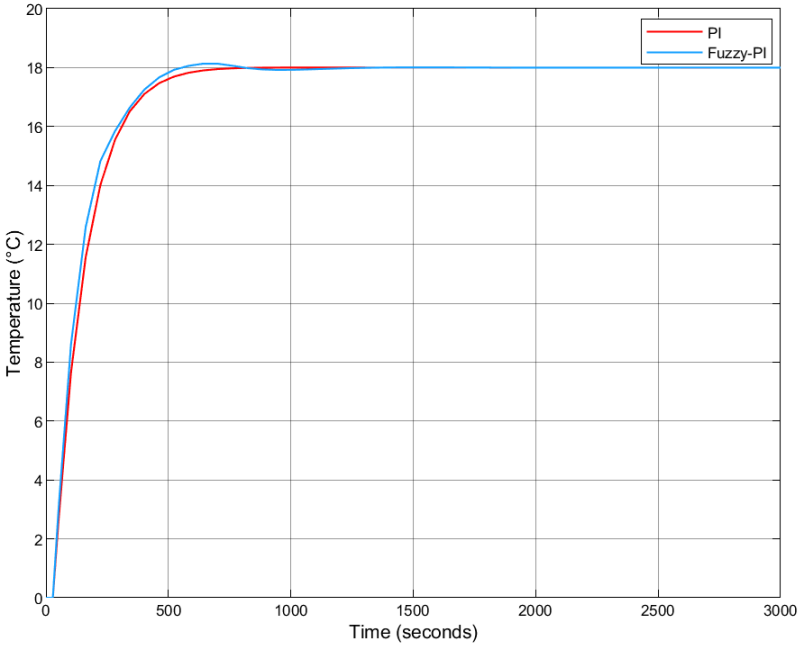


Figure 4.14: Fuzzy-PI vs PI step responses

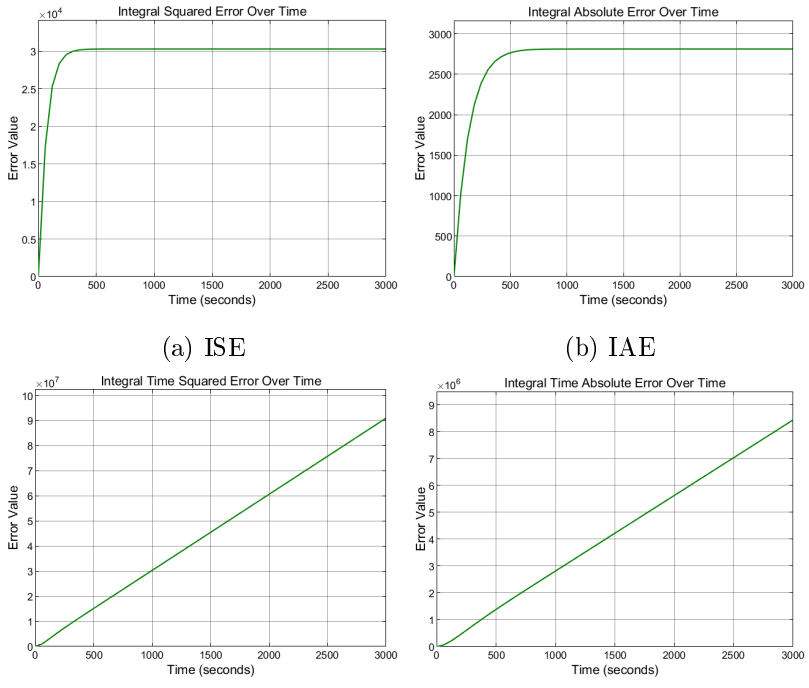
To perform a more detailed comparison between the Fuzzy-PI and PI controllers, temporal response metrics were analysed. For this purpose, the **"To Workspace"** block was used in Simulink to extract the simulation data into MATLAB workspace. With the arrays of output (***Temperature* (°*C*)**) and time (***Time* (*seconds*)**), the **stepinfo()** function was applied to obtain key performance indicators, such as rise time, settling time, over-

shoot and steady-state error. This analysis allows for a quantitative comparison between both controllers. Based on the values obtained, which can be seen in Table 4.5, the Fuzzy-PI controller presents a faster rise time and a lower settling time when compared to the classical PI controller. Although it exhibits a slightly higher overshoot, both values are below 1%, which is generally acceptable indicating a well-damped response. Additionally, both controllers achieve zero steady-state error.

Table 4.5: Time metrics comparison between Fuzzy-PI and PI controllers

Controller	**Rise Time (t_r)**	**Settling Time (t_s)**	**Overshoot (%)**	**Steady-State Error (e_{ss})**
Fuzzy-PI	263.6628	465.6549	0.7363	0
PI	281.7529	506.7102	0.0094	0

To help complement the analysis, the performance indexes were analysed again. Thus, the PI controller shows the following errors in Figure 4.15.

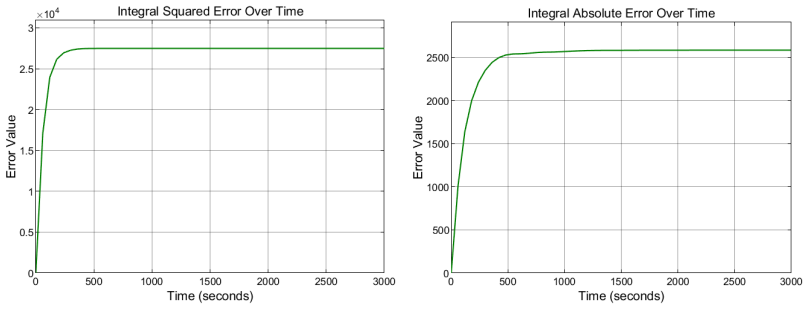


(a) ISE (b) IAE

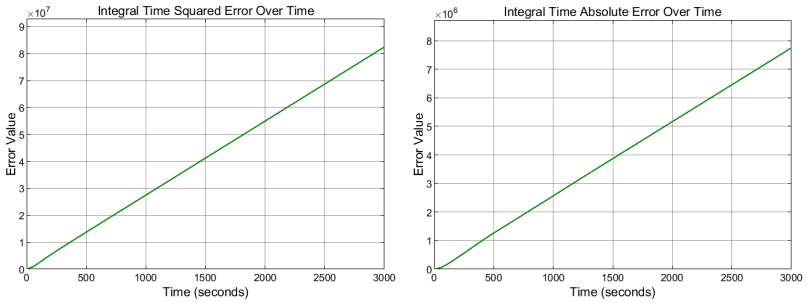
(c) ITSE (d) ITAE

Figure 4.15: PI controller performance metrics

The Fuzzy-PI controllers are illustrated in Figure 4.16.



(a) ISE (b) IAE



(c) ITSE (d) ITAE

Figure 4.16: Fuzzy-PI controller performance metrics

Analysing Figures 4.15 and 4.16, it can be concluded that the Fuzzy-PI controller has smaller errors compared to the PI controller, which can be seen in Table 4.6. When taking the ISE or IAE that are easier to compare, it is possible to see that the graph grows rapidly at the start, because the error is maximum just after the start of the simulation, as it is still a long way from reaching the desired setpoint. As it gets closer to the setpoint, the error tends towards zero and the growth of the ISE or IAE slows down which means that the system stabilize at a final value that represents the amount of total error accumulated over the course of the controller's responses. It can therefore be assumed that with proportional (P) and integrative (I) action it is possible to solve the issue.

Table 4.6: Performance metric comparison between Fuzzy-PI and PI controllers

Type	**ISE**	**IAE**	**ITSE**	**ITAE**
Fuzzy-PI	2.748×10^4	2.583×10^3	8.242×10^7	7.745×10^6
PI	3.032×10^4	2.811×10^3	9.093×10^7	8.431×10^6

4.3.4 Fuzzy-PID vs PID Controllers

The fourth and final experiment was carried out using all the actions, proportional (P), integrative (I) and derivative (D), resulting in the block diagram in Figure 4.17.

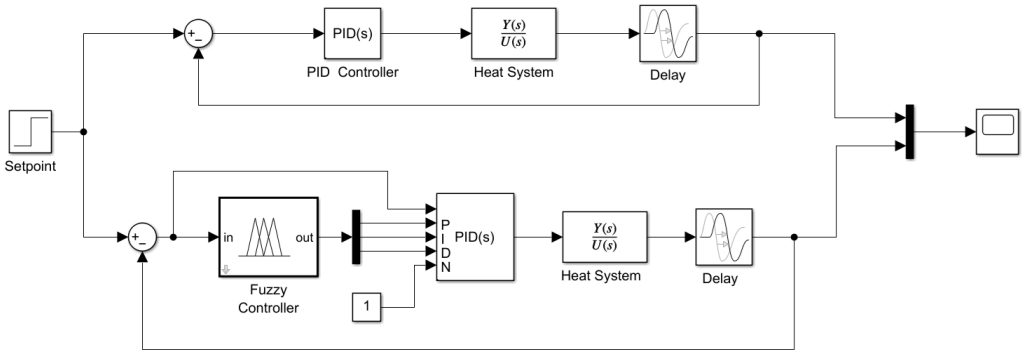


Figure 4.17: Fuzzy-PID vs PID block diagram

Since both the K_p, K_i and K_d values for the Fuzzy and classical controllers had already been defined and obtained from the previous experiments, all that was needed was to put everything together. Thus, the only change made was to complement the rule bases with the three components, which can be seen in Figure 4.18.

	Rule	Weight	Name
1	If Error is NegativeLarge then Kp is Low, Ki is High, Kd is High	1	rule1
2	If Error is NegativeSmall then Kp is Low, Ki is Medium, Kd is High	1	rule2
3	If Error is Zero then Kp is Low, Ki is Low, Kd is Medium	1	rule3
4	If Error is PositiveSmall then Kp is Medium, Ki is Low, Kd is Low	1	rule4
5	If Error is PositiveLarge then Kp is High, Ki is Low, Kd is Low	1	rule5

Figure 4.18: Fuzzy-PID rules base

That being said, the following response was obtained from Figure 4.19.

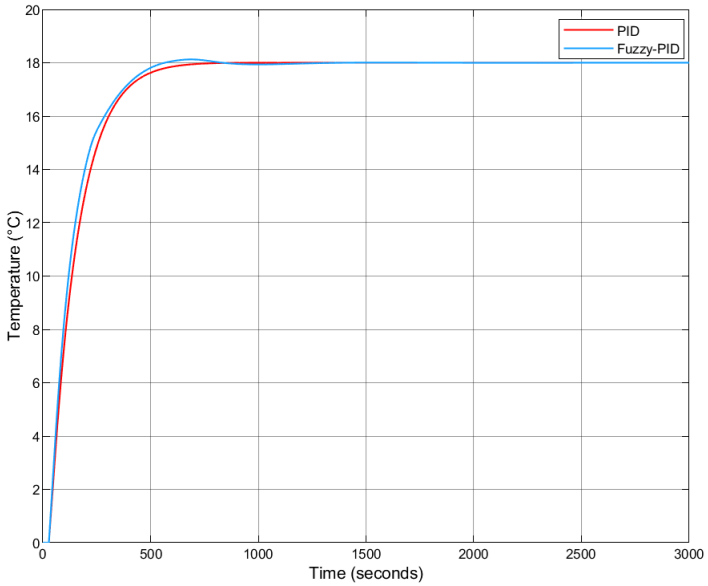


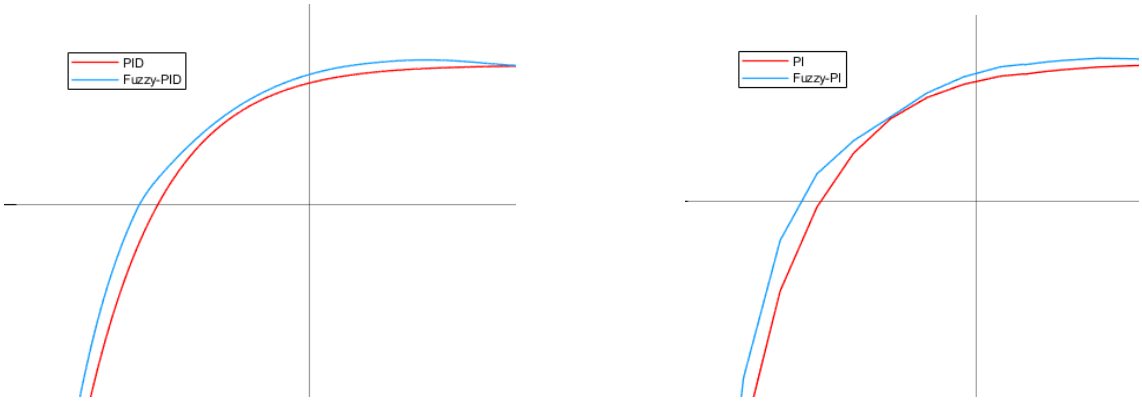
Figure 4.19: Fuzzy-PID vs PID step responses

To perform a more detailed comparison between the Fuzzy-PID and PID controllers, temporal response metrics were analysed. For this purpose, the **"To Workspace"** block was used in Simulink to extract the simulation data into MATLAB workspace. With the arrays of output (***Temperature* (°*C*)**) and time (***Time* (*seconds*)**), the **stepinfo()** function was applied to obtain key performance indicators, such as rise time, settling time, overshoot and steady-state error. This analysis allows for a quantitative comparison between both controllers. Based on the values obtained, which can be seen in Table 4.7, the Fuzzy-PID controller presents a faster rise time and a lower settling time when compared to the classical PID controller. Although it exhibits a slightly higher overshoot, both values are below 1% which is generally acceptable indicating a well-damped response. Additionally, both controllers achieve zero steady-state error.

Table 4.7: Time metrics comparison between Fuzzy-PID and PID controllers

Controller	**Rise Time (t_r)**	**Settling Time (t_s)**	**Overshoot (%)**	**Steady-State Error (e_{ss})**
Fuzzy-PID	259.3701	462.6557	0.6933	0
PID	274.5492	504.8539	0.0099	0

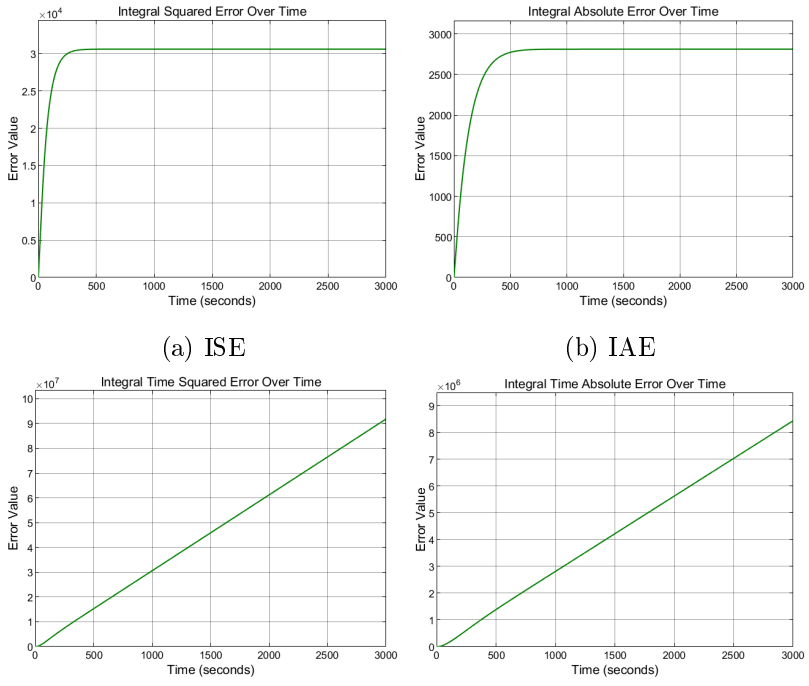
Another contribution that can be seen is the derivative action, because although it is a small change, it makes both controllers reach the stabilization value a little faster (around 2 seconds in each) and it can also be seen a smoother response, as can be seen in Figure 4.20 when compared to the Fuzzy-PI and PI implementations.



(a) Fuzzy-PID vs PID curve step responses (b) Fuzzy-PI vs PI curve step responses

Figure 4.20: Contribution of derivative action in the step responses

To complete the explanation, the performance indexes of each controller were analysed once again. Thus, the PID controller shows the following errors in Figure 4.21.

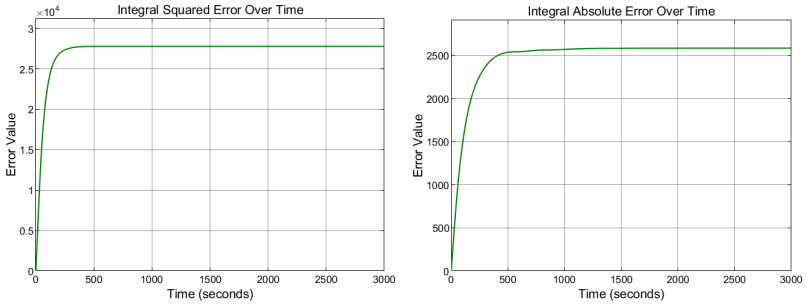


(a) ISE (b) IAE

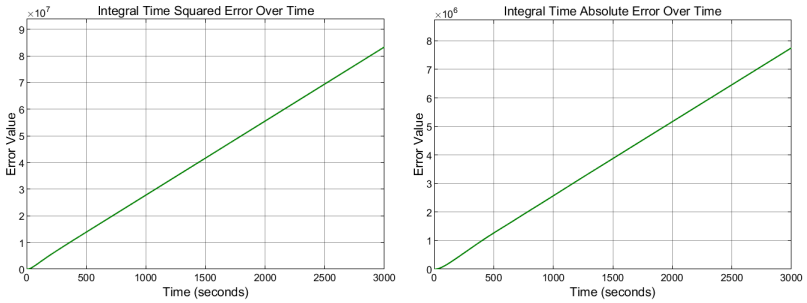
(c) ITSE (d) ITAE

Figure 4.21: PID controller performance metrics

As for the Fuzzy-PID controller, it shows the ones in Figure 4.22.



(a) ISE (b) IAE



(c) ITSE (d) ITAE

Figure 4.22: Fuzzy-PID controller performance metrics

Analysing Figures 4.21 and 4.22 it can be concluded that the Fuzzy-PID controller has smaller errors compared to the PID controller which can be seen in Table 4.8. When taking the ISE or IAE that are easier to compare, it is possible to see that the graph grows rapidly at the start, because the error is maximum just after the start of the simulation, as it is still a long way from reaching the desired setpoint. As it gets closer to the setpoint, the error tends towards zero and the growth of the ISE or IAE index slows down which means that the system stabilize at a final value, that represents the amount of total error accumulated over the course of the controllers' responses. It can therefore be assumed that with proportional (P), integrative (I) and derivative (D) actions it is possible to solve the issue.

Table 4.8: Performance metric comparison between Fuzzy-PID and PID controllers

Type	**ISE**	**IAE**	**ITSE**	**ITAE**
Fuzzy-PID	2.778×10^4	2.585×10^3	8.332×10^7	7.751×10^6
PID	3.060×10^4	2.812×10^3	9.178×10^7	8.432×10^6

More, compared the errors of PI and PID controllers, we can conclude that the total errors are very similar.

4.3.5 Impact of Anti-Windup in Fuzzy-PI and PI controllers

Another experiment carried out was the introduction of the Anti-Windup technique, previously explained in the subsection 3.6. Thus, in a first instance, it was necessary to build the subsystem to implement the Anti-Windup gain (K_{aw}) as can be seen in Figure 4.23.

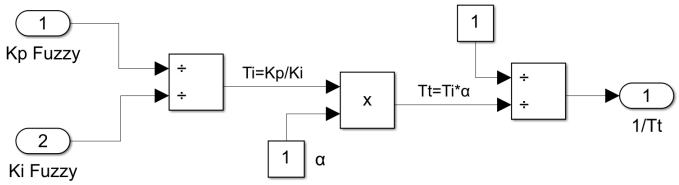


Figure 4.23: Ultimate gain block diagram

Thus and since the system in question already adjusts the parameters $K_p\ Fuzzy$ and $K_i\ Fuzzy$ dynamically, the values of Ti will also suffer from this change since $K_p\ Fuzzy$ and $K_i\ Fuzzy$ will be adjusted over time. As for the parameter α, it is an adjustment constant and is responsible for determining the aggressiveness of the integrator correction when the control signal saturates. For small α values (*e.g,* $Ti * 0.1$) the system recovery will be slow and possibly have overshoot and, for larger α values (*e.g.,* $Ti * 2$) the recovery will be faster but there may be a risk of the system to oscillate. Ideally, the values should be adjusted so that there is a balance between speed and stability. That being said, Figure 4.24 shows the block diagram of the implementation of the Anti-Windup technique.

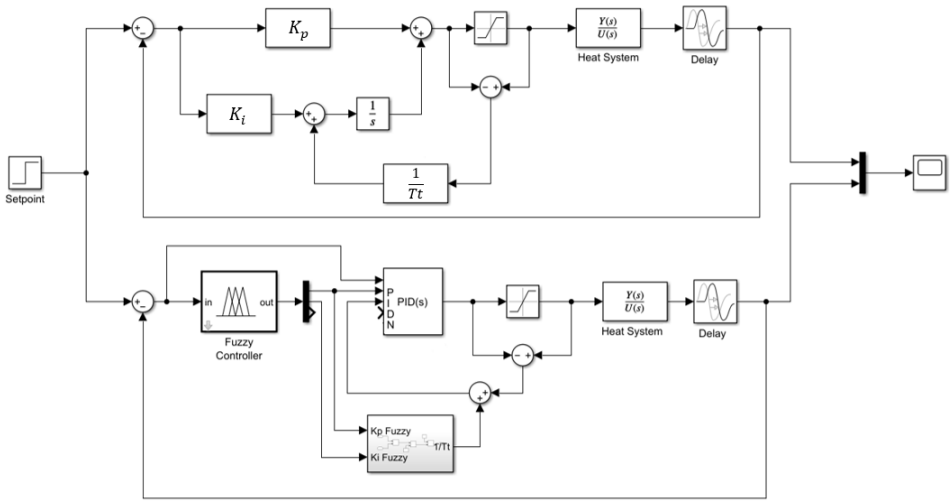
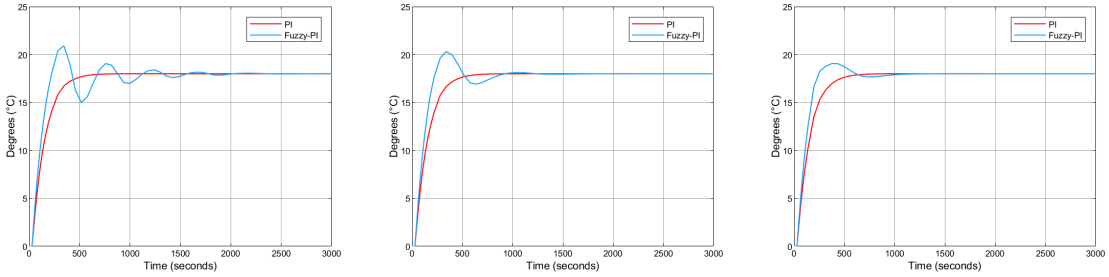


Figure 4.24: Fuzzy-PI vs PI with Anti-Windup block diagram

Tests were carried out with the Anti-Windup technique by Back Calculation with a saturation of 50%, because for values above this, the system wouldn't even react. The adjustment constant α was varied with the values 0.5, 1.0 and 1.5. Figure 4.25 shows the comparison between the Fuzzy-PI and PI controllers with Anti-Windup for each value of α.

Figure 4.25: Fuzzy-PI vs PI with Anti-Windup step responses



(a) $\alpha = 0.5$ and saturation of 50%

(b) $\alpha = 1.0$ and saturation of 50%

(c) $\alpha = 1.5$ and saturation of 50%

By analysing the graphs, it can be concluded that applying the Anti-Windup technique did not bring any clear benefit to the system's response, especially in Fuzzy-PI. On the contrary:

- with $\alpha = 0.5$ (Figure 4.25a), the system shows sharp oscillations in the transient regime and takes longer time to stabilize.
- with $\alpha = 1.0$ (Figure 4.25b), a considerable overshoot is still observed, although with less oscillation.
- with $\alpha = 1.5$ (Figure 4.25c), the response shows a small additional overshoot but no significant oscillations, but even so the version without Anti-Windup has a more stable response.

Although useful in other contexts, the Anti-Windup technique can worsen performance in first-order systems with delays, such as the thermal system under study. Furthermore, if both controllers have well-tuned gains, the integral term does not accumulate excessive error before reaching saturation, making Anti-Windup unnecessary or even counterproductive. That said, the implementation of this control structure will be discarded.

4.3.5.1 Anti-Windup in Fuzzy-PID and PID controllers

One more test that was carried out was the introduction of the derivative action, but it showed no significant improvement over the Fuzzy-PI and PI Anti-Windup responses (Figure 4.25). Thus, its use in Fuzzy-PID and PID controllers was also discarded for further analysis.

4.3.6 Fuzzy-PI with Smith Predictor vs PI with Smith Predictor

The last test applied to each of the controllers, was the introduction of the Smith Predictor explained earlier in the sub-section 3.7, where the aim is to assess how its integration impacts the performance of each systems. It was therefore first tested for the Fuzzy-PI and PI controllers, as can be seen from the block diagram in Figure 4.26.

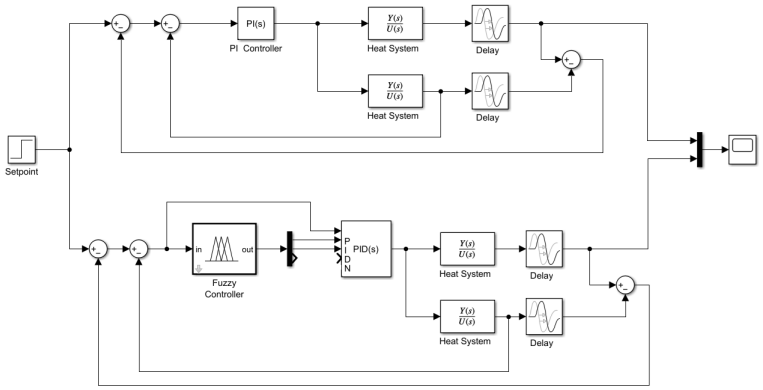


Figure 4.26: Fuzzy-PI vs PI with Smith Predictor block diagram

That being said, the responses in Figure 4.27 was obtained.

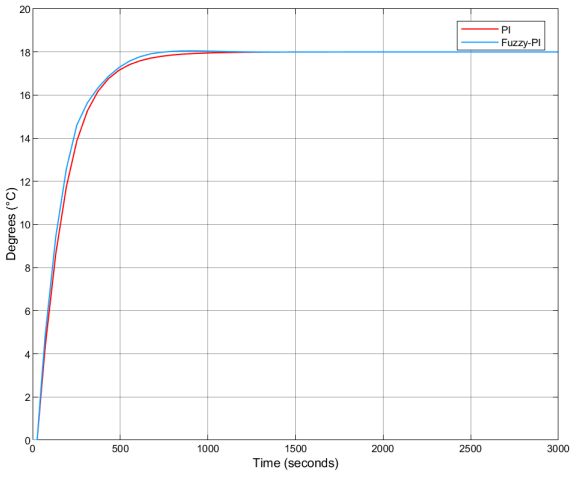


Figure 4.27: Fuzzy-PI vs PI with Smith Predictor step response

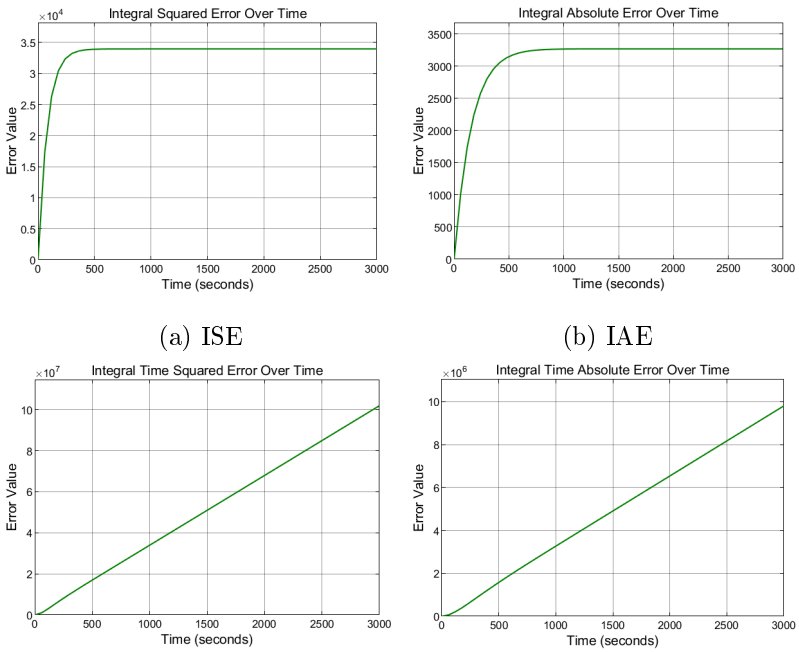
To perform a more detailed comparison between the Fuzzy-PI and PI controllers with Smith Predictor implementation, temporal response metrics were analysed. For this purpose, the **"To Workspace"** block was used in Simulink to extract the simulation data into MATLAB workspace. With the arrays of output (***Temperature* (°*C*)**) and time (***Time* (*seconds*)**), the **stepinfo()** function was applied to obtain key performance in-

dicators, such as rise time, settling time, overshoot and steady-state error. This analysis allows for a quantitative comparison between both controllers. Based on the values obtained which can be seen in Table 4.9, the Fuzzy-PI controller presents a faster rise time and a lower settling time when compared to the classical PI controller. Although it exhibits a slightly higher overshoot, both values are below 1% which is generally acceptable indicating a well-damped response. Additionally, both controllers achieve zero steady-state error.

Table 4.9: Time metrics comparison between Fuzzy-PI and PI controllers with Smith Predictor implementation

Controller	**Rise Time (t_r)**	**Settling Time (t_s)**	**Overshoot (%)**	**Steady-State Error (e_{ss})**
Fuzzy-PI	338.8536	540.2515	0.5379	0
PI	345.5608	613.4476	6.7723×10^{-4}	0

When compared with the versions without Smith Predictor, the Fuzzy-PI and PI controllers show increases in settling time of about 74 seconds and 110 seconds, respectively. However, the Fuzzy-PI with Smith Predictor still stabilizes approximately 76 seconds faster than the PI with Smith Predictor. To complement this analysis, performance indexes were also evaluated, and for the PI with Smith Predictor, the corresponding errors are shown in Figure 4.28.

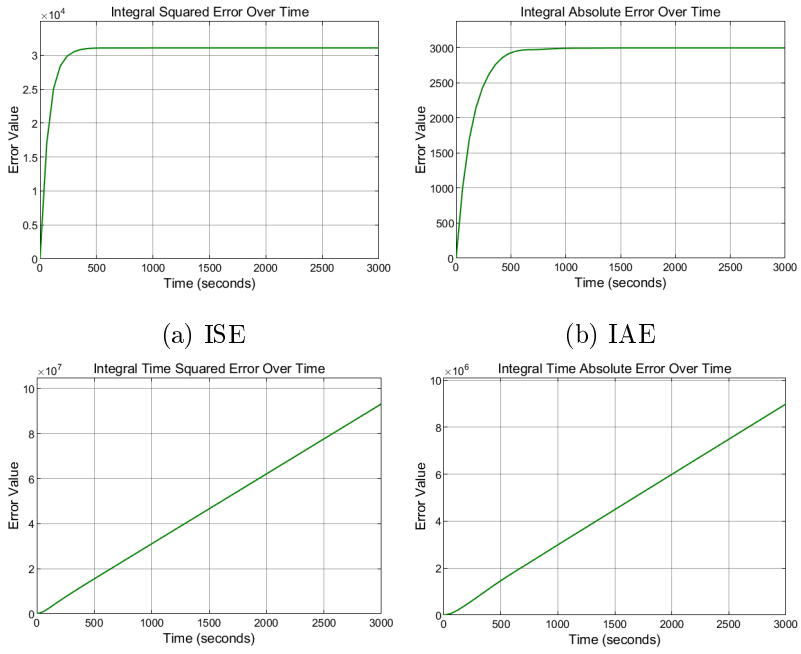


(a) ISE (b) IAE

(c) ITSE (d) ITAE

Figure 4.28: PI controller with Smith Predictor performance metrics

As for the Fuzzy-PI with Smith Predictor, it shows the errors in Figure 4.29.



(a) ISE (b) IAE

(c) ITSE (d) ITAE

Figure 4.29: Fuzzy-PI controller with Smith Predictor performance metrics

Analysing Figures 4.28 and 4.29, it can be concluded that the Fuzzy-PI controller with Smith Predictor yields smaller errors than the PI controller with Smith Predictor. Observing the ISE or IAE, the graph initially rises quickly due to the large initial error, which then decreases as the setpoint is approached. This causes the ISE/IAE growth to slow, reflecting system stabilization and total accumulated error. Compared to Fuzzy-PI and PI without the Smith Predictor, the error increases with its use (see Table 4.10), but the benefits in reducing oscillations and smoothing the response justify its implementation.

Table 4.10: Performance metrics comparison between Fuzzy-PI and PI controllers with and without Smith Predictor

Performance Metrics	**With Smith Predictor**		**Without Smith Predictor**	
	Fuzzy-PI	**PI**	**Fuzzy-PI**	**PI**
ISE	3.106×10^4	3.396×10^4	2.748×10^4	3.032×10^4
IAE	2.997×10^3	3.267×10^3	2.583×10^3	2.811×10^3
ITSE	9.316×10^7	1.019×10^8	8.242×10^7	9.093×10^7
ITAE	8.989×10^6	9.799×10^6	7.745×10^6	8.431×10^6

4.3.7 Fuzzy-PID with Smith Predictor vs PID with Smith Predictor

The second experiment carried out with the introduction of the Smith Predictor was with PID-based controllers, as can be seen from the block diagram in Figure 4.30.

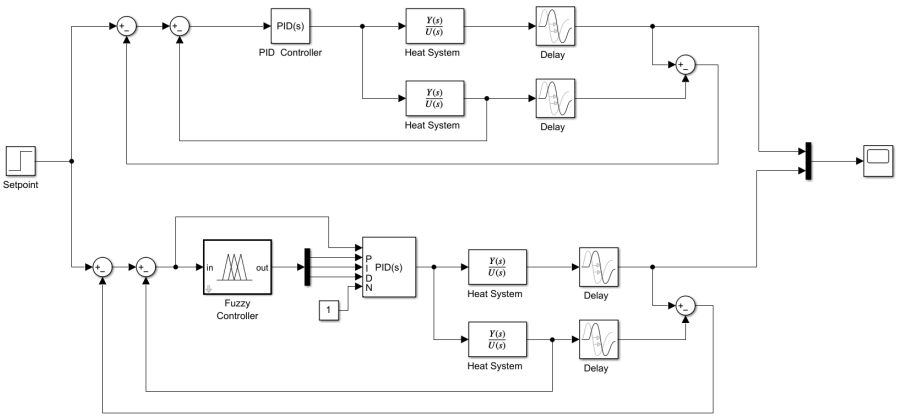


Figure 4.30: Fuzzy-PID vs PID with Smith Predictor block diagram

After applying the Smith Predictor, it is clear that it helps in smoothing the response obtained from the controllers and also correcting the slight overshoot that the Fuzzy-PID controller previously presented, which went from 18.12°C to 18.09°C (0.69% to 0.51%), that can be seen in Figure 4.31.

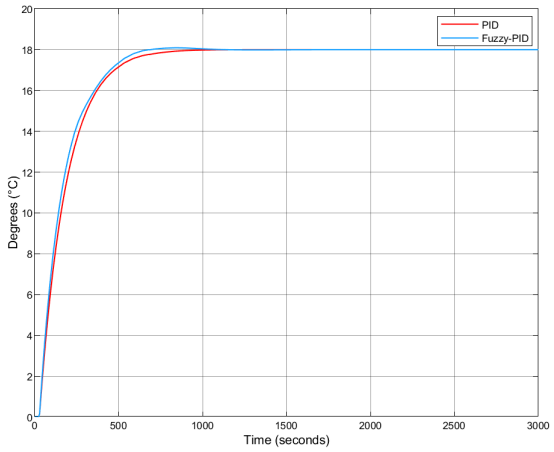


Figure 4.31: Fuzzy-PID vs PID step responses with Smith Predictor

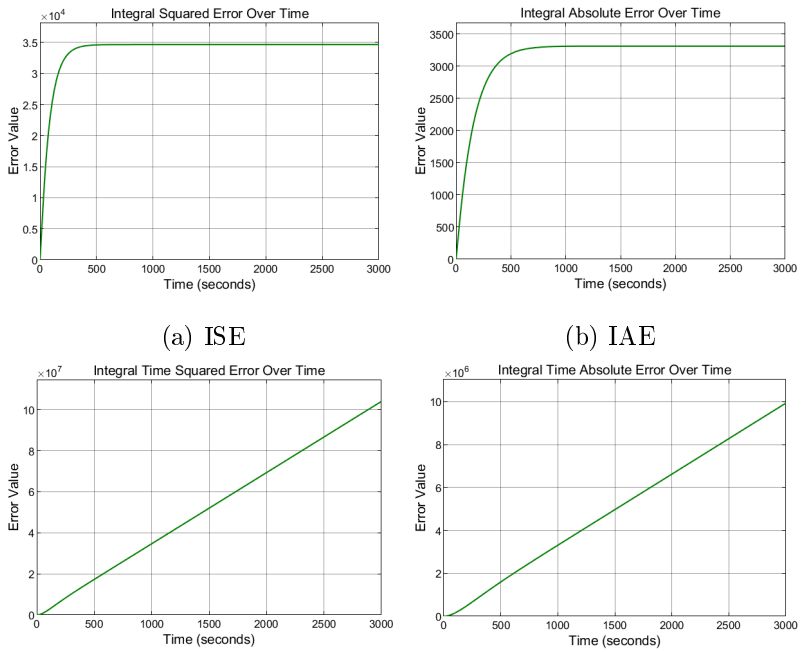
To perform a more detailed comparison between the Fuzzy-PID and PID controllers with Smith Predictor implementation, temporal response metrics were analysed. For this purpose, the **"To Workspace"** block was used in Simulink to extract the simulation data into MATLAB workspace. With the arrays of output (***Temperature* (°*C*)**) and time (***Time* (*seconds*)**), the **stepinfo()** function was applied to obtain key performance

indicators, such as rise time, settling time, overshoot and steady-state error. This analysis allows for a quantitative comparison between both controllers. Based on the the values obtained which can be seen in Table 4.11, the Fuzzy-PID controller presents a faster rise time and a lower settling time when compared to the classical PID controller. Although it exhibits a slightly higher overshoot, both values are below 1% which is generally acceptable indicating a well-damped response. Additionally, both controllers achieve zero steady-state error.

Table 4.11: Time metrics comparison between Fuzzy-PID and PID controllers with Smith Predictor implementation

Controller	**Rise Time (t_r)**	**Settling Time (t_s)**	**Overshoot (%)**	**Steady-State Error (e_{ss})**
Fuzzy-PID	329.2864	549.9743	0.5102	0
PID	341.9797	616.1027	0.0027	0

When compared with the versions without Smith Predictor, the Fuzzy-PID and PID controllers show increases in settling time of about 87 seconds and 111 seconds, respectively. However, the Fuzzy-PID with Smith Predictor still stabilizes approximately 66 seconds faster than the PID with Smith Predictor. To complement this analysis, performance indexes were also evaluated, and for the PID with Smith Predictor, the corresponding errors are shown in Figure 4.32.

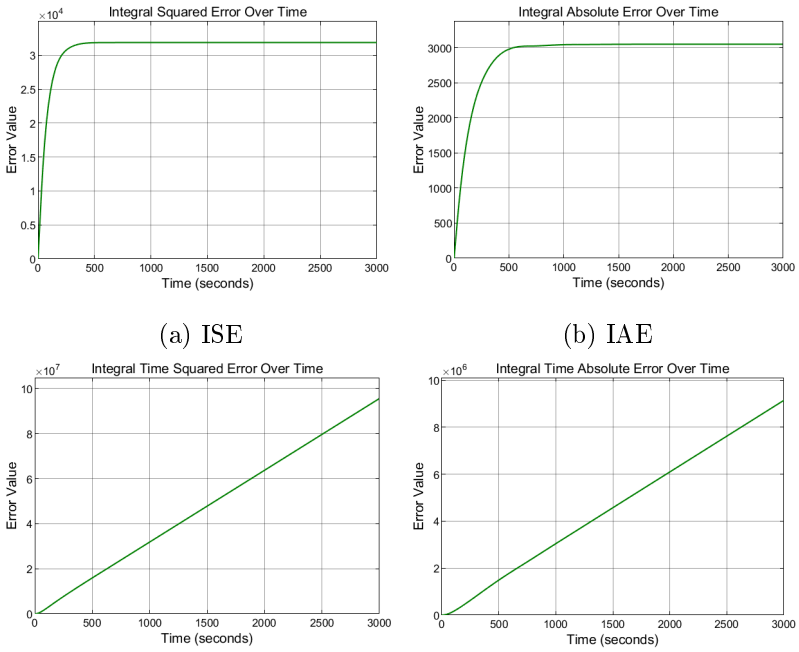


(a) ISE (b) IAE

(c) ITSE (d) ITAE

Figure 4.32: PID controller with Smith Predictor performance metrics

As for the Fuzzy-PID with Smith Predictor, it shows the errors in Figure 4.33.



(a) ISE (b) IAE

(c) ITSE (d) ITAE

Figure 4.33: Fuzzy-PID controller with Smith Predictor performance metrics

Analysing Figures 4.32 and 4.33, it can be concluded that the Fuzzy-PID controller with Smith Predictor yields smaller errors than the PID controller with Smith Predictor. Observing the ISE or IAE, the graph initially rises quickly due to the large initial error, which then decreases as the setpoint is approached. This causes the ISE/IAE growth to slow, reflecting system stabilization and total accumulated error. Compared to Fuzzy-PID and PID without the Smith Predictor, the error increases with its use (see Table 4.12), but the benefits in reducing oscillations and smoothing the response justify its implementation.

Table 4.12: Performance metrics comparison between Fuzzy-PID and PID with and without Smith Predictor

Performance Metrics	**With Smith Predictor**		**Without Smith Predictor**	
	Fuzzy-PID	**PID**	**Fuzzy-PID**	**PID**
ISE	3.186×10^4	3.466×10^4	2.748×10^4	3.060×10^4
IAE	3.050×10^3	3.310×10^3	2.585×10^3	2.812×10^3
ITSE	9.554×10^7	1.040×10^8	8.332×10^7	9.178×10^7
ITAE	9.146×10^6	9.927×10^6	7.751×10^6	8.432×10^6

4.3.8 Choice of Controller Type

After the tests carried out above, it can be concluded that there are two possible approaches to controlling systems with delays, Fuzzy-PI/PI or Fuzzy-PID/PID controllers. Although both can be valid, it was necessary to choose which one to use in the next section 4.4 where the chosen controller will be simulated in a 'real' environment. The results obtained show that the Fuzzy-PI and PI controllers, with and without Smith Predictor, have slightly lower errors when compared to the Fuzzy-PID and PID controllers in both approaches, with and without the Smith Predictor. On the other hand, when analysing the response of each controller and time metrics, the Fuzzy-PID and PID controllers, with and without Smith Predictor, have a smoother response and slightly faster response when compared to the Fuzzy-PI and PI implementations. Therefore, as the difference in error between both controllers is negligible and the visual response has some impact, the Fuzzy-PID and PID, with and without Smith Predictor approaches were chosen.

4.4 Practical Case Study

To help complement the comparison between Fuzzy-PID and PID controllers, a more realistic environment was simulated where it was proposed to introduce noise/entropy and see if the controllers can correct it. This noise corresponds, for example, that we have the indoor temperature set to 18°C and someone were to open a door/window and the outside temperatures were too hot or too cold (*e.g.,* −10°C or 35°C), and thus affecting the controllers' response. For this purpose, the "Noise" block was developed, which basically consists of applying a random value (either positive or negative) after the controllers have already stabilized at the desired target value, which can be seen in Figure 4.34.

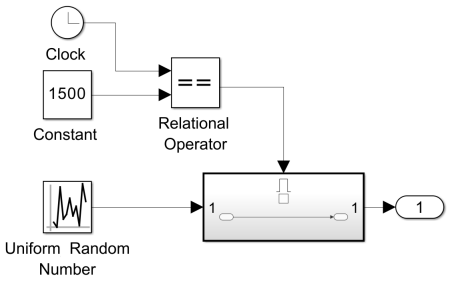


Figure 4.34: Noise model components

Once the noise model was obtained, it was necessary to test for both Fuzzy-PID vs PID with and without the Smith Predictor, in order to see which one gets the best response. The noise was applied after the system had already stabilise, and the behaviour of the controllers could be seen. In this way, the noise is introduced at 1500 seconds.

4.4.1 Noise implementation without Smith Predictor

Following the same block diagram structure from subsection 4.3.4, the only change is adding the noise block to the system that can be seen in Figure 4.35.

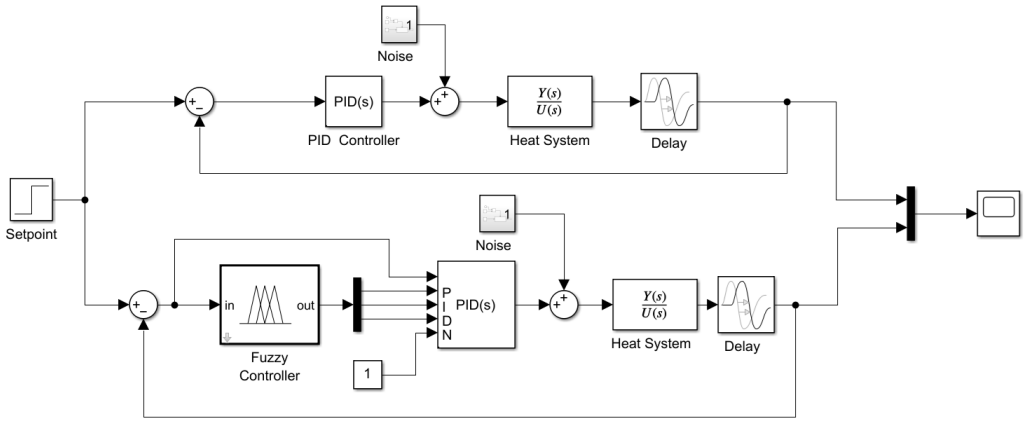


Figure 4.35: Fuzzy-PID vs PID noise implementation

Thus, the behaviour of the controllers was first tested when the outside temperature is hotter than the inside temperature. The result obtained was the following in Figure 4.36.

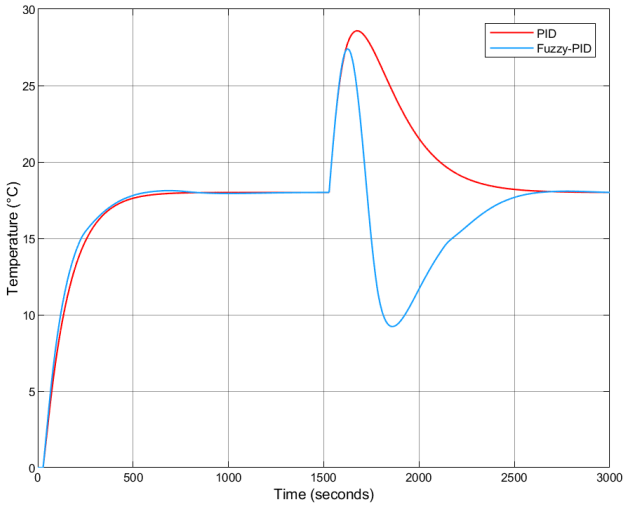


Figure 4.36: Fuzzy-PID vs PID step response for outside warmer temperatures

It was also tested when the outside temperatures are colder than the inside temperatures and the answer in Figure 4.37 was obtained.

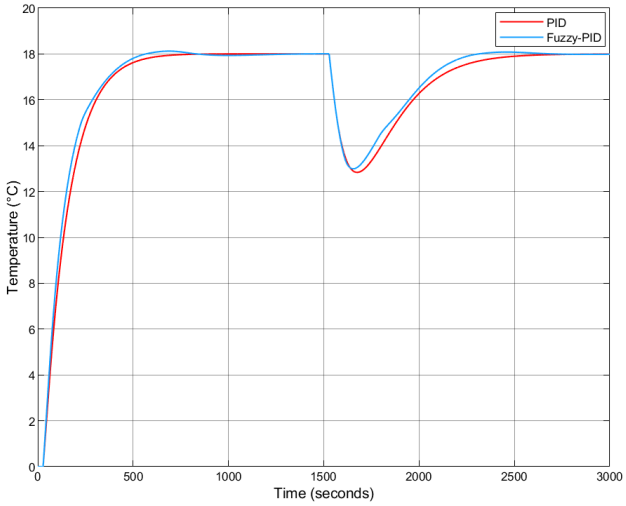
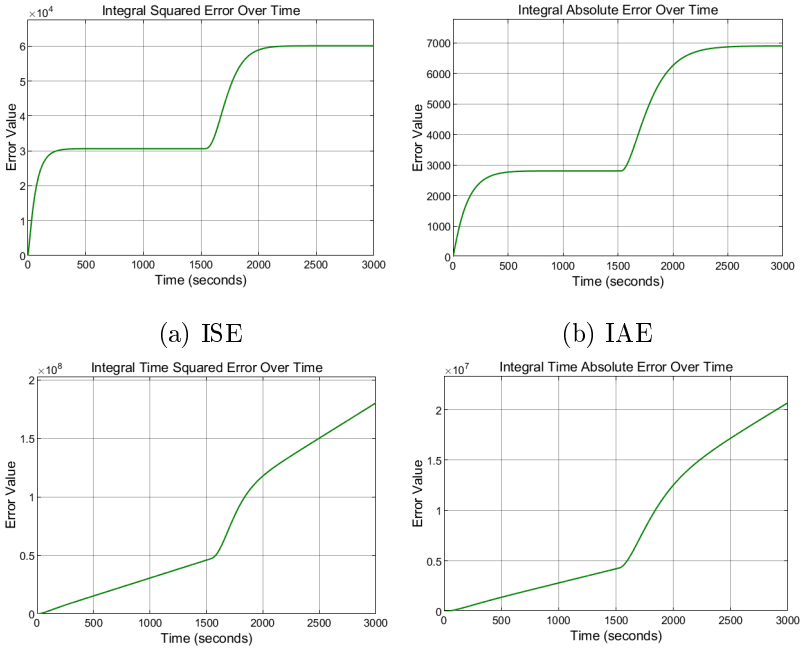


Figure 4.37: Fuzzy-PID vs PID step response for outside colder temperatures

To complement the application of noise, a comparison of the performance indexes of each controller was made again, but it was only taken the "Warmer Temperatures" since it has more variation compared to the "Colder Temperatures", making the visualization better. As expected, when noise is introduced, the errors undergo a change because this is when correction is necessary. Thus, the PID controller presents the error responses shown in Figure 4.38.

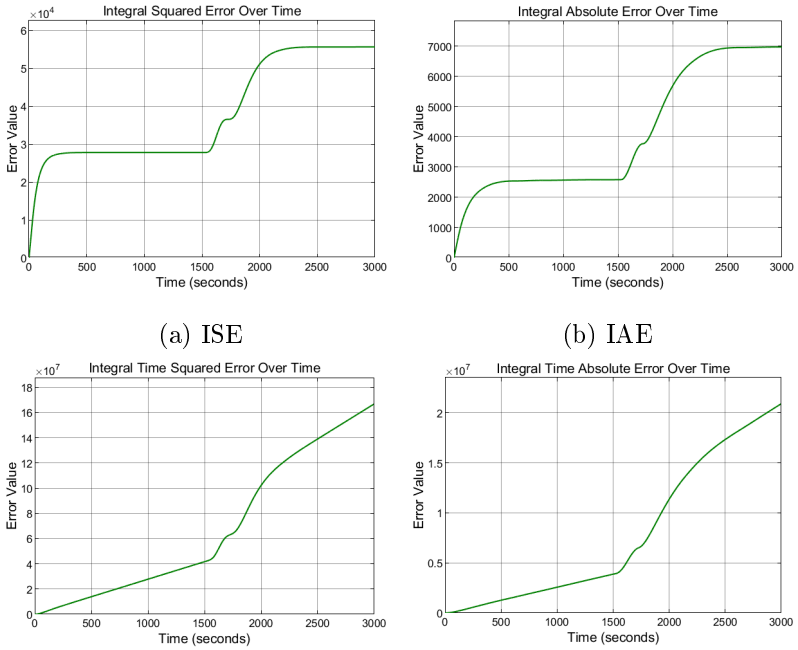


(a) ISE (b) IAE

(c) ITSE (d) ITAE

Figure 4.38: PID controller without Smith Predictor performance metrics with noise

As for the Fuzzy-PID controller, it has the following error response in Figure 4.39.



(a) ISE (b) IAE

(c) ITSE (d) ITAE

Figure 4.39: Fuzzy-PID controller without Smith Predictor performance metrics with noise

Analysing Figures 4.38 and 4.39, it can be concluded that the Fuzzy-PID controller has smaller errors compared to the PID controller, even though the IAE metric this time is a little bit higher than the PID which can be seen in Table 4.13. When taking the ISE or IAE that are easier to compare, it is possible to see that the graph grows rapidly at the start, as it is still a long way from reaching the desired setpoint. As it gets closer to the setpoint, the error tends towards zero and the growth of the ISE or IAE slows down which means that the system stabilize at a final value that represents the amount of total error accumulated over the course of the controllers' responses. When the noise is introduced, it is possible to see that all graphs change towards a new error value, which makes sense because the controller needs to correct again and adjust for the desired setpoint.

Table 4.13: Performance metric comparison between Fuzzy-PID and PID without Smith Predictor and noise implementation

Type	**ISE**	**IAE**	**ITSE**	**ITAE**
Fuzzy-PID	5.563×10^4	6.971×10^3	1.669×10^8	2.091×10^7
PID	6.008×10^4	6.091×10^3	1.802×10^8	2.070×10^7

With these results, we can conclude that the proposed algorithm reach the goals proposed in this work, even when the system is subjected to external interference to its oper-

ation mode.

4.4.2 Noise implementation with Smith Predictor

This implementation was also based on the subsection 4.3.7 with the introduction of the noise block as well. That being said, we get the following block diagram from Figure 4.40.

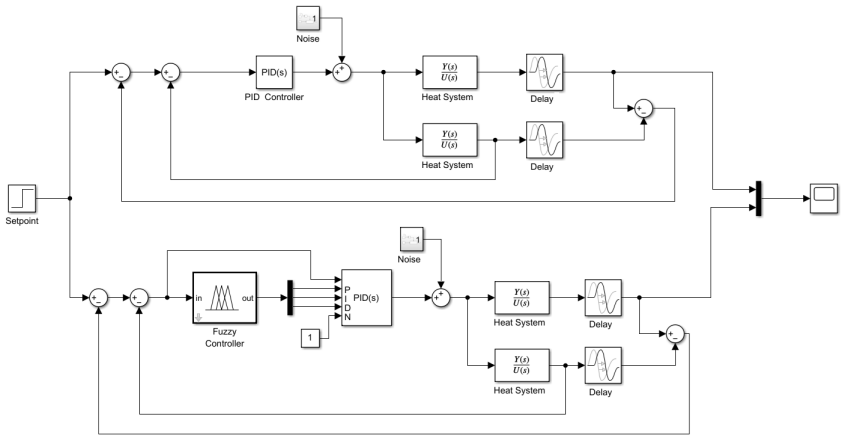


Figure 4.40: Fuzzy-PID vs PID block diagram with noise implementation

Like previously done, it was first started with hotter outside temperatures, and the result obtained was the following in Figure 4.41.

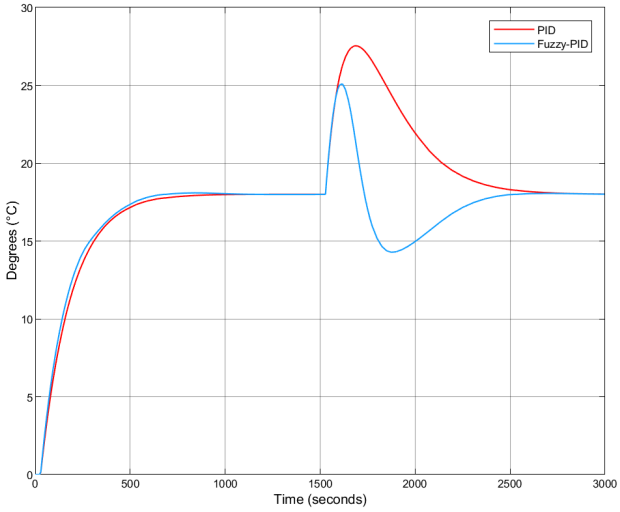


Figure 4.41: Fuzzy-PID vs PID step response for outside warmer temperatures with Smith Predictor implementation

It was then tested when the outside temperatures are colder than the inside tempera-

tures and the answer in Figure 4.42 was obtained.

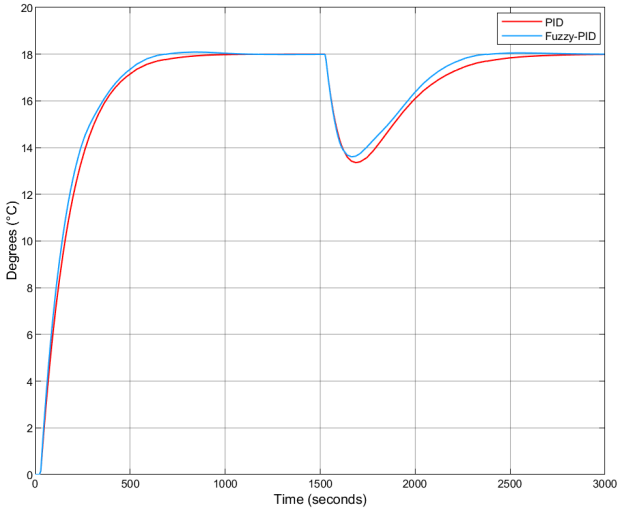
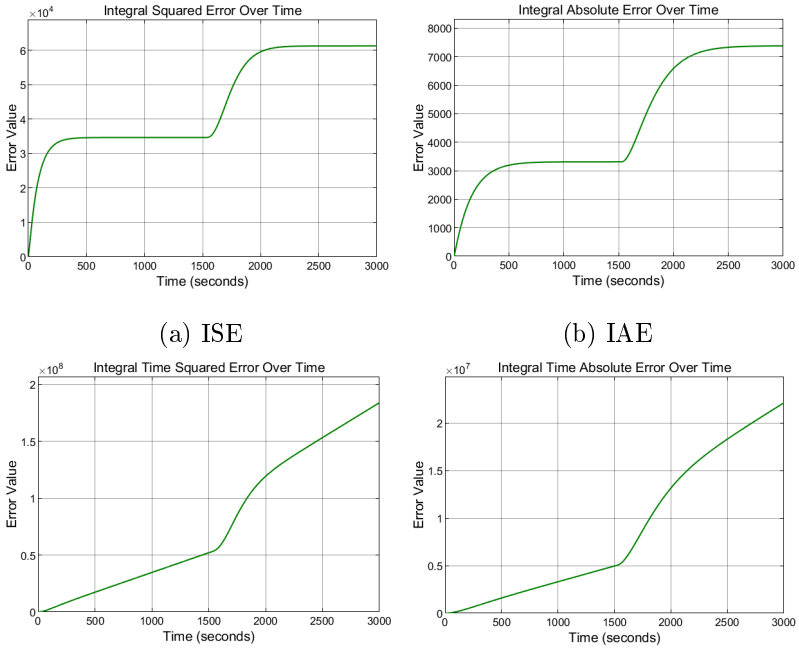


Figure 4.42: Fuzzy-PID vs PID step response for outside colder temperatures with Smith Predictor implementation

To complement the noise analysis, a new comparison of performance indexes was made using only the "Warmer Temperatures" case, as it shows more variant and clearer visualization. That being said, the PID controller responses are shown in Figure 4.43.

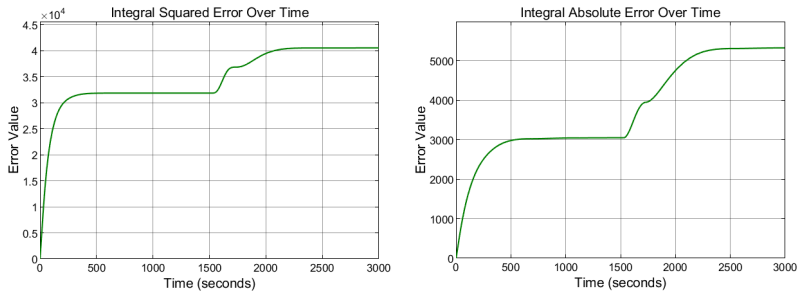


(a) ISE (b) IAE

(c) ITSE (d) ITAE

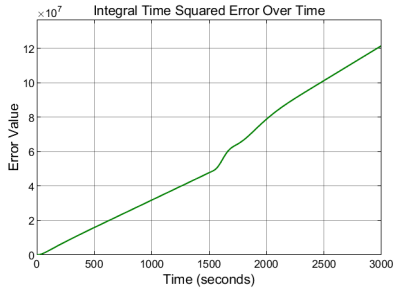
Figure 4.43: PID controller with Smith Predictor performance metrics when noise is applied

As for the Fuzzy-PID controller, it has the following response in Figure 4.44.

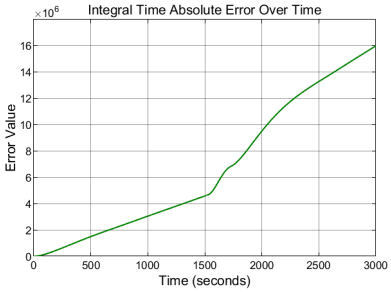


(a) ISE

(b) IAE



(c) ITSE



(d) ITAE

Figure 4.44: Fuzzy-PID controller with Smith Predictor performance metrics when noise is applied

Analysing Figures 4.43 and 4.44, it can be concluded that the Fuzzy-PID controller has smaller errors compared to the PID controller which can be seen in Table 4.14. When taking the ISE or IAE that are easier to compare, it is possible to see that the graph grows rapidly at the start, as it is still a long way from reaching the desired setpoint and as it gets closer to the setpoint, the error tends towards zero and the growth of the ISE or IAE slows down, which means that the system stabilize at a final value that represents the amount of total error accumulated over the course of the controllers' response. When the noise is introduced, it is possible to see that all graphs change towards a new error value which makes sense because the controller needs to correct again and adjust for the desired setpoint.

Table 4.14: Performance metric comparison between Fuzzy-PID and PID with Smith Predictor and noise implementation

Type	**ISE**	**IAE**	**ITSE**	**ITAE**
Fuzzy-PID	4.053×10^4	5.321×10^3	1.216×10^8	1.597×10^7
PID	6.127×10^4	7.377×10^3	1.838×10^8	2.213×10^7

4.4.3 Comparison between both implementations

After analysing the controllers' responses to the introduction of noise, it can be concluded that both are able to adjust and make the correction to reach the target value again, however the Fuzzy-PID controller presents a better response than the PID and ends up reaching the target value faster than the PID controller.

When it comes to the implementation of the Smith Predictor, what is concluded is that it not only promotes a smoother system response but it also helps the controller to behave fast and do not allow it to overshoot so much. By analysing and compare for example, Figure 4.36 with Figure 4.41 that are related to the "Warmer Temperatures Noise" it is noticeable that the values obtained and the graphical responses from the Smith Predictor implementation are better than the other one without the Smith Predictor.

It should also be noted that the errors obtained also demonstrate that the Fuzzy-PID approach continues to be more advantageous than implementing only a classic PID controller as it presents lower error values.

To sum up, the best implementation is the Fuzzy-PID with Smith Predictor since that it had the best responses.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

After a comparative analysis of different controller configurations, it was concluded that the Fuzzy-PID controller showed better overall performance than the classic PID controller. The Fuzzy-P, Fuzzy-PD, Fuzzy-PI and Fuzzy-PID variants were tested, the latter providing the best balance between dynamic response, stability and robustness.

The possibility of using only a Fuzzy-PI and PI controller approach was initially considered because, in systems with delays, has a good capacity for correcting the stationary error. However, the introduction of the derivative component helped reducing the rise time and settling time and was also crucial in smoothing the response obtained by reducing the overshoot. For these reasons, the Fuzzy-PI and PI strategy was discarded.

Complementary mechanisms such as Anti-Windup and Smith Predictor were also evaluated. It was observed that Anti-Windup, did not improve the system and even ended up degrading the response. As for the Smith Predictor, although it introduces a slight additional delay to the system, it proved to be beneficial especially in the presence of external disturbances, where its compensatory action smoothed out the response, reducing overshoot and consequently increasing the robustness of the controller's performance.

To sum up the combination of a Fuzzy-PID controller with Smith Predictor provides the most effective and robust solution for the system under study.

5.2 Future Work

When it comes to future work, it would be interesting to explore hybrid approaches to optimize the controller, such as PSO or genetic algorithms, to automatically adjust the parameters or even its rule base.

In addition, the use of neural networks, like Neuro-Fuzzy, or other learning methods could allow for greater adaptation to variations in the system. Finally, it would be im-

portant to validate the approach in a real environment, using embedded hardware to see practical performance of the controller.

Bibliography

[1] Katsuhiko. Ogata. *Modern control engineering*. Prentice Hall, 2022. ISBN: 0136156738.

[2] Karl J.. Åström and Tore. Hägglund. *PID controllers*. International Society for Measurement and Control, 1995, p. 343. ISBN: 1556175167.

[3] Lotfi A Zadeh. "Fuzzy logic=computing with words". In: *IEEE Transactions on Fuzzy Systems* 4.2 (1996), pp. 103–111.

[4] John H Lienhard. *A Heat Transfer Textbook*. 5th ed. Massachusetts: Phlogiston Press, 2020. URL: http://ahtt.mit.edu.

[5] Keith Sherwin. *Introduction to Thermodynamics*. 1st ed. Chapman & Hali, 1993. ISBN: 9780412476402.

[6] Green Mechanic. *Conduction, Convection and Radiation*. Accessed: 2025-03-15. 2017. URL: https://www.green-mechanic.com/2017/01/conduction-convection--radiation.html.

[7] C.P. Hong. *Thermal Management of Electronics*. CRC Press, 2006.

[8] M.D. Kearney and M.S. Allen. *Automotive Thermal Management: Theory and Practice*. Wiley, 2015.

[9] John A. Duffie and William A. Beckman. *Solar Engineering of Thermal Processes*. Wiley, 2013.

[10] Michael J Moran and Howard N Shapiro. *Fundamentals of Engineering Thermodynamics*. 5th ed. John Wiley & Sons, Inc, 2006. ISBN: 139780470030370.

[11] Frank P. Incropera and David P. DeWitt. *Fundamentals of Heat and Mass Transfer*. 4th ed. New York, NY: John Wiley & Sons, 1996. ISBN: 9780471304609.

[12] Yunus A Cengel. *Steady versus Transient Heat Transfer*. 2022.

[13] E. Fermi. *Thermodynamics*. Dover Books on Physics. Dover Publications, 2012. ISBN: 9780486134857. URL: https://books.google.pt/books?id=xCjDAgAAQBAJ.

[14] T. N. Narasimhan. "Fourier's heat conduction equation: History, influence, and connections". In: *Reviews of Geophysics* 37 (1 Feb. 1999), pp. 151–172. ISSN: 87551209. DOI: 10.1029/1998RG900006.

[15] William J.. Palm. *System dynamics*. McGraw-Hill Education, 2021, p. 904. ISBN: 9780078140051.

[16] Benjamin C. Kuo Farid Golnaraghi. *Automatic Control Systems*. Mc Graw Hill, 2022.

[17] *Feedback System Image*. Accessed: 2025-03-22. 2019. URL: https://cdn.automationforum.co/uploads/2019/01/feedback-1.png.

[18] *On-Off Control Output Image*. Accessed: 2025-03-22. 2019. URL: https://x-engineer.org/wp-content/uploads/2019/01/On-off-control-output.png.

[19] *On-Off Control with Hysteresis Output Image*. Accessed: 2025-03-22. 2019. URL: https://x-engineer.org/wp-content/uploads/2019/01/On-off-control-with-hysteresis-output.png.

[20] Aidan O'Dwyer. *Handbook of PI and PID Controller Tuning Rules*. A comprehensive guide to tuning rules for PI and PID controllers. London: Imperial College Press, 2009. ISBN: 978-1-84816-242-6.

[21] Liuping Wang. *PID Control System Design and Automatic Tuning using MATLAB/Simulink*. 1st. John Wiley & Sons, 2019. ISBN: 978-1119469445. DOI: 10.1002/9781119469414. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119469414.

[22] Graham C. Goodwin, Stefan F. Graebe, and Mario E. Salgado. *Control System Design*. 1st. Upper Saddle River, NJ: Prentice Hall, 2001. ISBN: 978-0139586538.

[23] N. S. Nise. *Control Systems Engineering*. 6th. Hoboken, NJ: Wiley, 2011.

[24] J.G. Ziegler and N.B. Nichols. "Optimum Settings for Automatic Controllers". In: (1942).

[25] GeeksforGeeks. *Stair Step Function*. Accessed: 2025-03-22. 2023. URL: https://media.geeksforgeeks.org/wp-content/uploads/20231110110413/Stair-Step-function-03.png.

[26] Gene F Franklin, J David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems Sixth Edition*. 2010.

[27] Thomas E Marlin. *Process Control Designing Processes and Control Systems for Dynamic Performance*. 2015.

[28] Kevin M.. Passino and Stephen. Yurkovich. *Fuzzy control*. Addison-Wesley, 1998, p. 475. ISBN: 020118074X.

[29] Chuen Chien Lee. "Fuzzy logic in control systems: Fuzzy logic controller—Part I". In: *IEEE Transactions on Systems, Man, and Cybernetics* 20.2 (1990), pp. 404–418.

[30] Shu-Heng Chen and Paul P. Wang. *Computational Intelligence in Economics and Finance*. Vol. 1. Berlin, Heidelberg: Springer, 2001. ISBN: 978-3-642-07530-9. DOI: 10.1007/978-3-662-04323-3.

[31] IIT Kharagpur. *Chapter 3: Fuzzy Membership Functions*. Accessed: 2025-03-23. n.d. URL: https://cse.iitkgp.ac.in/~dsamanta/courses/archive/sca/Archives/Chapter%203%20Fuzzy%20Membership%20Functions.pdf.

[32] *Image from User Repository*. Accessed: 2025-03-22. 2020. URL: https://user-images.githubusercontent.com/39605819/72969382-f8f7ec00-3d8a-11ea-9244-3c3b5f23b3ac.png.

[33] Arpit Jain and Abhinav Sharma. *Membership function formulation methods for Fuzzy Logic systems: A comprehensive review*.

[34] H.-J. Zimmermann. *Fuzzy Set Theory—and Its Applications*. Springer Netherlands, 2001. DOI: 10.1007/978-94-010-0646-0.

[35] Didier Dubois and Henri Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, 1980.

[36] J.-S.R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice Hall, 1997.

[37] Elmer P. Dadios, ed. *Fuzzy Logic - Controls, Concepts, Theories and Applications*. InTech, 2012. ISBN: 978-953-51-0396-7.

[38] Timothy J. Ross. *Fuzzy Logic with Engineering Applications*. 3rd. Chichester, UK: Wiley, 2010. ISBN: 978-0-470-74376-8.

[39] *Centroid defuzzification method*. 2011. URL: https://www.researchgate.net/figure/Centroid-defuzzification-method_fig8_282586601 (visited on Nov. 15, 2023).

[40] E. H. Mamdani and S. Assilian. "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller". In: *International Journal of Man-Machine Studies* 7.1 (1975), pp. 1–13. DOI: 10.1016/S0020-7373(75)80002-2.

[41] Tomohiro Takagi and Michio Sugeno. "Fuzzy Identification of Systems and Its Applications to Modeling and Control". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15.1 (1985), pp. 116–132. DOI: 10.1109/TSMC.1985.6313399.

[42] Isabel S Jesus, J A Tenreiro Machado, and Ramiro S Barbosa. *On the Fractional Order Control of Heat Systems*. Springer. DOI: 10.1007/978-1-4020-8678-632.

[43] The MathWorks, Inc. *Fuzzy Logic Toolbox User's Guide*. Available: https://www.mathworks.com/products/fuzzy-logic.html. MathWorks. Natick, Massachusetts, USA, 2023.

[44] MathWorks. *PID Tuner - Tune PID controllers*. Accessed: May 18, 2025. The MathWorks, Inc. 2025. URL: https://www.mathworks.com/help/control/ref/pidtuner-app.html.

[45] The MathWorks, Inc. *PID Controller Design for Fast Reference Tracking*. Acedido em: July 3, 2025. The MathWorks, Inc. 2023. URL: https://www.mathworks.com/help/control/getstart/pid-controller-design-for-fast-reference-tracking.html.

[46] Oscar Castillo and Patricia Melin. *A classification of fuzzy controllers*. Accessed: 2025-05-13. 2013. URL: https://www.researchgate.net/figure/A-classification-of-fuzzy-controllers_fig1_255567860.

[47] George K. I. Mann Bao-Gang Hu and Raymond G. Gosine. "A systematic study of fuzzy PID controllers - Function-based evaluation approach". In: *IEEE Transactions on Fuzzy Systems* 9 (5 Oct. 2001), pp. 699–712. ISSN: 10636706. DOI: 10.1109/91.963756.

[48] Zdenko Kovači´ Kovači´c and Stjepan Bogdan. *Fuzzy Controller Design Theory and Applications*. 2006.

[49] Bernard Widrow. *Intelligent Control Nazmul Siddique A Hybrid Approach Based on Fuzzy Logic, Neural Networks and Genetic Algorithms Foreword by*. URL: http://www.springer.com/series/7092.

[50] Karl J.. Åström and Tore. Hägglund. *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society, 2006, p. 460. ISBN: 1556179421.

[51] Xin Lan Li, Jong Gyu Park, and Hwi Beom Shin. "Comparison and evaluation of anti-windup PI controllers". In: *Journal of Power Electronics* 11 (1 2011), pp. 45–50. ISSN: 15982092. DOI: 10.6113/JPE.2011.11.1.045.

[52] Donna L. Mohr, William J. Wilson, and Rudolf J. Freund. *Statistical Methods*. 4th. Academic Press, 2021. ISBN: 9780128202346.

[53] Richard C. Dorf and Robert H. Bishop. *Modern Control Systems*. 13th. Pearson, 2016. ISBN: 9780134407623.