Universidade de Aveiro
2014
Departamento de Eletrónica, Telecomunicações e Informática

Daniel Valente Valentim

Controlador Fuzzy baseado em ARM Cortex M3

Fuzzy Controller based on ARM Cortex M3

Universidade de Aveiro
2014
Departamento de Eletrónica, Telecomunicações e Informática

Daniel Valente Valentim

Controlador Fuzzy baseado em ARM Cortex M3

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Professor Dr. Alexandre Manuel Moutela Nunes da Mota e do Professor Dr. Telmo Reis Cunha, ambos do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

agradecimentos /
acknowledgements

Dedico este trabalho à minha família e amigos, por me orientarem e auxiliarem em todo este trajeto.

the jury

Presidente / President	Professora Doutora Pétia Georgieva Georgieva Professora Auxiliar, Universidade de Aveiro
Vogais / Examiners Committee	Prof. Doutor José António Barros Vieira Professor Adjunto, Instituto Politécnico de Castelo Branco
	Prof. Doutor Alexandre Manuel Moutela Nunes da Mota Professor Associado, Universidade de Aveiro

Key-words

Fuzzy Logic, Digital Control, Microcontrollers

Abstract

Fuzzy logic is the base for fuzzy controllers. The fuzzy logic is a concept that defies the normal boolean logic, used mostly everywhere, where everything belongs to a defined state and jumps to other state. The fuzzy logic confines that “middle” space, undefined for the usual logic. In fuzzy logic, there is a continuous growth in the variable between states, and that continuous path is defined and interpreted.

This concept is applied to digital controllers and, with the strong development of microcontrollers and digital control overall, the fuzzy theory can be applied with great performance results. Through digital control it is simple to converge both fuzzy logic and mathematics methods, like PID, into the same system, improving the performance of the control mechanism.

In this dissertation, there is an approach to the basics of fuzzy theory and fuzzy controllers and the development of this area in digital control for microcontrollers. To achieve several control requirements, it was implemented using real-time methods. The tests were made on a DC motor, for velocity control, and compared with default tests using a PID controller.

Palavras-chave Lógica difusa, Controlo Digital, Microcontroladores

Resumo

Lógica difusa é a base dos controladores difusos. A lógica difusa é um conceito que desafia a lógica Boleana, onde tudo pertence a um estado definido e salta para outro estado. A lógica difusa define e interpreta este espaço entre estados, não definido pela lógica boleana, onde não existe um crescimento contínuo das váriáveis entre estados.

Este conceito é aplicado em controladores digitais e, com o forte desenvolvimento na área dos microcontroladores e controlo digital, a lógica difusa pode ser aplicada para atingir melhores desempenhos. Através do controlo digital é possível juntar a lógica difusa e métodos matemáticos, como PID, no mesmo sistema, aumentando o desempenho do mecanismo de controlo.

Nesta dissertação é feita uma introdução à lógica difusa e controladores difusos e o seu desenvolvimento na área de controlo digital para microcontroladores. Todo o sistema foi implementado utilizando ferramentas de sistemas de tempo real. Os testes foram realizados num motor DC, com o objectivo de controlar a sua velocidade e comparado com testes utilizando um controlador PID.

Contents

Image list

Table List

List of Acronyms

QEI	Quadrature Encoder Interface
RTOS	Real Time Operating System
PID	Proportional, Integral, Derivative
PWM	Pulse-Width Modulation
MCPWM	Motor Control PWM

1 Introduction

1.1 Motivation

Since the beginning of development of computer systems and processing units the idea of using these to control systems has been a constant achievement. In modern times, with the proliferation of microprocessors, digital control systems offer a wide range of new possibilities for almost every system in the world. It became an opportunity to research alternative control methods and developing new digital control systems to better adjust with the reality. This new challenge will be addressed at this dissertation, using fuzzy control logic in these new microprocessors and compare its results with the so-far dominant PID approach.

1.2 Objectives

Introduce fuzzy logic theory with its implementation and development on a digital controller.

Develop a fuzzy controller based on an ARM Cortex M3 processor for velocity control using a DC motor as test and comparison with a PID controller. Infer the quality of the controllers using active load changes with several setups.

Develop the controller software using free RTOS tools.

Create a Matlab interface between the hardware and the software that communicates with the microcontroller monitoring the whole structure. It should be able to perform the basic operations on the controller, change the system input (set point), enable and control the active load and show the real time plot of both the motor speed and control signal.

1.3 Structure

This document is organized in five chapters:

- Introduction;
- Fuzzy Control;

- Fuzzy Controller design;
- Results;
- Conclusion and future work.

In the first chapter it is presented the dissertation objectives, its structure and the motivation towards the theme.

In the second chapter there is an overview on fuzzy logic and its implementation on fuzzy control. Then the fuzzy control method is present with constant discussion for better insight about the theme.

In the third chapter the fuzzy control design applied to the dissertation is presented. It starts with an overview of the control design for the work environment, explaining the selected microcontroller and the controllable object – a DC motor. It follows by explaining the thought process in the construction of the fuzzy controller, explaining each phase and decision and summing up with the integration of the whole process in digital control on microcontrollers.

In the fourth chapter are presented the results for each controller built. The results are discussed based on the performance, complexity and possible improvements. There is also a comparison to a standard PID controller.

In the fifth chapter the dissertation concludes with an overall discussion of the dissertation results and future work to be developed on the fuzzy control area.

1.4 Methodology

The dissertation consists in developing a fuzzy controller based on an ARM Cortex M3 processor. The fuzzy system is applied to control the velocity of a DC motor.

The work developed in this dissertation follows a previous dissertation work (2), which developed all the conditions for a controllable system. It included a power control module and a DC motor coupled to another motor working as generator. Starting at this point, the workflow began with adapting the power control module into the microprocessor and retrieving the input of the fuzzy controller, the velocity.

The next step used the selected microprocessor, an LPC1957 with ARM Cortex M3, which is a unique unit specialized to work with motor control (3). Both the velocity and position are measured by the Quadrature Encoder Interface of the microcontroller. The

output, duty cycle of a PWM wave, is directed to the power control module using the motor control PWM module. For optimal reasons and better control guarantees the developed code used free RTOS (4). The fuzzy control logic was developed in the microcontroller, followed by testing both systems and a PID controller for results comparison.

Finally it was built an interface using Matlab to isolate the hardware from the software, granting the option to operate the control system and overview the system data in real time.

2 Fuzzy Control

2.1 Introduction

"Fuzzy control provides a formal methodology for representing, manipulating and implementing human's heuristic knowledge about how to control a system" (1 p. 10). Fuzzy logic is a technique to embody human like thinking into a control system. A fuzzy controller can be designed to emulate human deductive thinking, that is, the process people use to infer conclusions from what they know. Traditional control approach requires formal modeling of the physical reality. Fuzzy Logic Control has proven to be an excellent choice for many control systems applications since it mimics human control logic. It can be built into anything from small, handmade products to large computerized process control systems. Fuzzy basic control has become highly competitive due to its better performance, high reliability and robustness. The thinking process involved in the fuzzy realm is not complex; it is simple, elegant and easy to apply.

Fuzzy logic control is one of the methodologies for solving control system problems. Its implementation in control systems ranges from simple, small, embedded micro controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software or a combination of both.

Fuzzy logic controllers provide a simple way to reach at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. Fuzzy logic controllers approach to control problems mimics how a person would make decisions at a faster rate. It is inherently robust since it does not require precise noise free inputs. The output control is a smooth control function despite a wide range of input variations. This also allows a different approach to control complex systems. Fuzzy logic focuses on what the systems should do rather than trying to model how it works. One can concentrate on solving the problem rather than trying to model the system mathematically, if that is even possible. On the other hand, the fuzzy approach requires a sufficient expert knowledge for the formulation of some of the fuzzy controller main structure parts - the rule base, the combination of the sets and the defuzzification. "If you can't explain it to a six year old, you don't understand it yourself" (5).

The fuzzy controller block diagram is represented in Figure 2.1. It is a fuzzy controller applied in a closed-loop control system. The plant outputs are denoted by y(t), its inputs are denoted by u(t), and the reference input to the fuzzy controller is denoted by r(t).

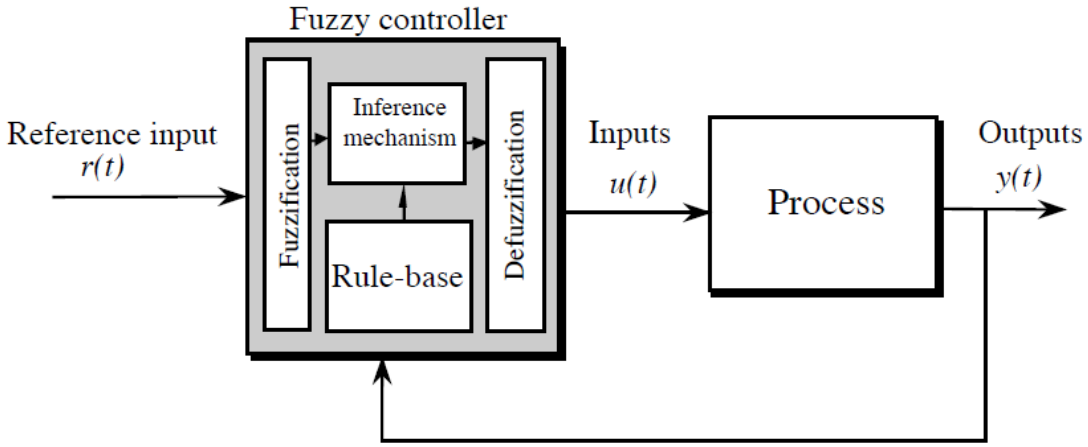


Figure 2.1: Fuzzy controller architecture (1 p. 11)

The fuzzy controller consists of four major phases/blocks. The first phase is the fuzzification, which works as an interface for the fuzzy logic. It simply transforms the inputs so that they can be interpreted by the inference mechanism and related to the rules in the rule-base block.

The second phase is the inference mechanism, which evaluates the content in the rule-base block and decides which control rules are relevant based on the information gathered from the inputs and then decides what the input to the plant should be.

The third is the rule-base that contains the knowledge required to control the system, in the form of a ruled-set.

The last phase is the defuzzification. It is an interface that converts the conclusions inferred from the rule-base into the inputs of the plant. Basically, the fuzzy controller acts as an artificial decision maker that operates in a closed-loop system in real time. It gathers the plant output data y(t), compares it to the reference input r(t), and then decides what the plant input u(t) should be in order to achieve the performance objectives.

To design the fuzzy controller, the control engineer must gather information on how the artificial decision maker should act in the closed-loop system. Depending on the controllable system, this information can come from a human expert at the controllable task. It must be the control engineer to understand the plant dynamics and formulate a set of rules about how to control the system. This quantification process of the knowledge gathered from the controllable system to the rules is of major importance. It is quite natural to realize that how much better the expert on the controllable task is, the better performance it is to expect from the fuzzy controller. The importance of the control engineer is still very much as relevant, as it is on his own skills the ability to correctly import

the information gathered to the system from the expert. It is a crucial task, because only the control engineer knows both sides, and his ability to correctly pass the information gathered from an expert to the fuzzy controller and knowing the controller limitations can be a very hard task to perform.

As this information is loaded into the rule-base, and an inference strategy is chosen, then the system is ready to be tested to see if the closed-loop performance objectives are met. Knowing this, the ability to create a fuzzy controller is very much different for every designer.

Fuzzy control system design essentially selects the fuzzy controller inputs and outputs, choose the preprocessing that is required for the controller inputs and possibly post processing that is required for the outputs and finally designing each of the four components of the fuzzy controller explained above.

2.2 Fuzzy inputs and outputs

There are several important aspects to keep in mind when choosing the correct inputs for a fuzzy system. In general, the inputs should grant enough information about the current state of the system that allows the fuzzy controller to correctly decide which action to take that best serves the performance objectives. In a fuzzy system, it is wise to carefully choose the number of inputs. If the expert on the controllable task only bases his predictions on a single reference, then adding more inputs would only cause potential unexpected results. When the expert relies on many inputs, even if some are not of major importance, it is really important to keep them. The fuzzy controller can easily adapt different scales of importance due to its inference process. It is really important to keep in mind that sometimes the human expert does not know correctly how many aspects he is evaluating to make his decisions, and therefore sometimes it is a hard choice for the control engineer to choose the inputs. In a simple example, a motor velocity can be evaluated by its sound. Naturally you could assume that the correct input for a motor velocity control would be the error or change of error. But as a fuzzy controller it is really important to keep the correct input that better relates the knowledge gathered from the expert to the rules. It could still be possible to use the error of the velocity if you could relate it to the sound it makes, but that is much harder than choosing to implement a sensor that evaluates the sound.

The thought process behind choosing an output follows the same reasons, while its importance might not be of such value as the inputs, it is still optimal for the defuzzification

process if the chosen output can relate with the controller rule-set and the controller actuator in a simple way.

2.3 Linguistic variables

To better define the knowledge given by the expert, fuzzy control uses a set of linguistic variables. These variables describe each of the time-varying fuzzy inputs and outputs. There are many choices of the linguistic description of variables. Often the control engineer can choose for a more complete description of the variable and use long linguistic descriptions, but these choices do not reflect in any way in how the controller operates. It is simply a notation that helps the construction of the fuzzy controller using fuzzy logic. These linguistic variables change dynamically over time, so after defining those, some sort of dynamic identification is also required. For instance, for the input “e(t)” the linguistic variable “error” can be used and to identify its dynamics “negative small” , “zero”, “positive small”. Other kind of descriptions can be used, even numbers, but they only act as a representation of its dynamic value and not the value itself. An example is shown in the figure 2.2.

The input “height” represents the height of the senior human population, which ranges from 1.25m to 2.25m. So “height” is the linguistic variable that represents the input of the fuzzy system. The crisp values are fuzzified to three dynamic variables Short, Medium and Tall. Each one defining a specific part of the input and together covering the entire input range.

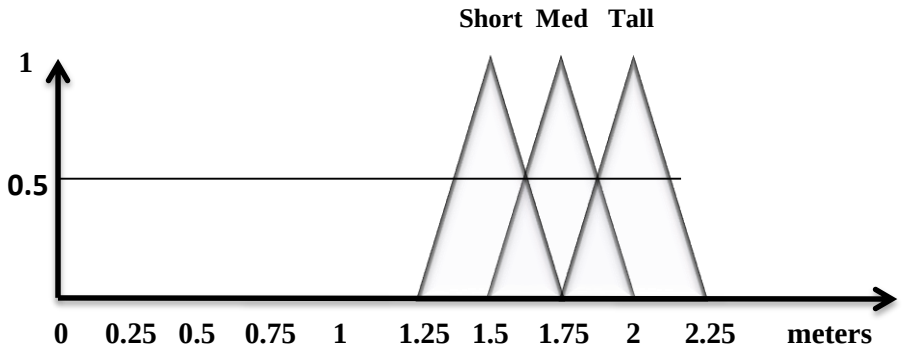


Figure 2.2: Fuzzy linguistic variables representation

This linguistic variables and dynamic values provide a language for the expert to express his knowledge about the decision-making process established in the fuzzy

controller, given by its inputs and outputs. Basically the linguistic variables are a representation of the inputs and linguistic values the quantification of dynamics. This aggregate created by the control engineer to fuzzify the system is called universe of discourse, which also includes the way these variables relate to each other, expressed in the fuzzy rules.

In the classic approach, the difficulty of correctly defining the margins between sets is an evident problem. Defining tall as two meters and not tall as less than two meters means people with 1.75 meters aren't tall, but people with 1.76 meters are. This is a very weak approach as 'objects' with such a small difference are labeled so differently and, as such, treated differently. This also means in the same set we have a wide array of values that are treated in the same way, which might be imprecise and worse, it makes it very weak towards errors near the set margins, known as rigid boundaries. In contrast, fuzzy sets have soft boundaries.

2.4 Rules

Using the linguistic variables, it is possible to create a set of rules that mimics the decision-making of the expert. The process relies on creating a rule for each possible state of fuzzy inputs, so that the control system always knows how to best perform at each situation to achieve the system objectives. These rules are called the linguistic rules, since they are created based only on linguistic variables. The figure 2.3 shows an example of linguistic variables for a fuzzy system that tries to predict the best position for each person based on two inputs, height and reaction time.

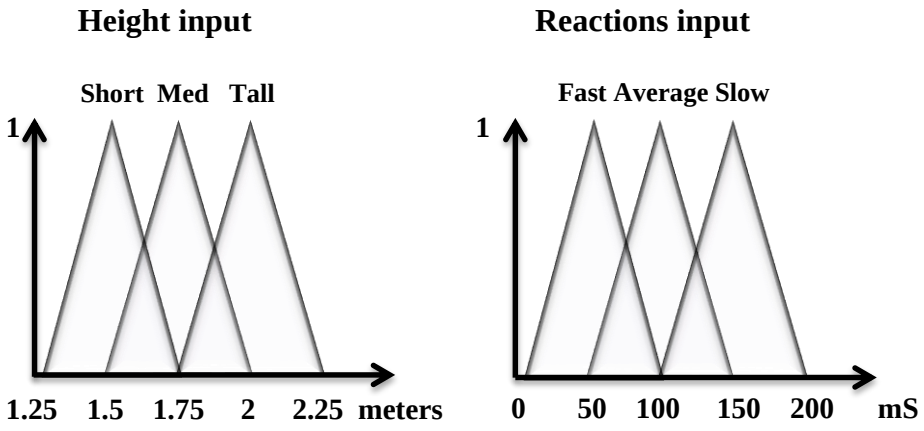


Figure 2.3: Fuzzy linguistic variables

With three dynamic variables (Short, Medium, Tall) for the linguistic variable "Height" and three dynamic variables (Fast, Average, Slow) for the "reaction time" inputs, there is a possible number of nine different rules that govern the entire possibility of inputs combination.

The form that the inputs relate to each other to generate an output is shown below. It is up to the designer to select the best logic to relate the inputs, in many systems the logic "or" is best suited for instance. The nine different rules for the fuzzy set represented in the figure 2.3 can be described as:

If the height is **"short"** *and* the reaction time is **"fast",** then the control output is **"Defender"**.

If the height is **"short"** *and* the reaction time is **"average",** then the control output is **"Midfielder"**.

If the height is **"short"** *and* the reaction time is **"slow",** then the control output is **"Support"**.

If the height is **"med"** *and* the reaction time is **"fast",** then the control output is **"Midfielder"**.

If the height is **"med"** *and* the reaction time is **"average",** then the control output is **"Midfielder"**.

If the height is **"med"** *and* the reaction time is **"slow",** then the control output is **"Support"**.

If the height is **"tall"** *and* the reaction time is **"fast",** then the control output is **"Striker"**.

If the height is **"tall"** *and* the reaction time is **"average",** then the control output is **"Defender"**.

If the height is **"tall"** *and* the reaction time is **"slow",** then the control output is **"Defender"**.

These rules offer an abstract level that humans are often comfortable with in terms of specifying how to control a process, but these rules are not yet the precise representation of the quantities used to actually control the system. These rules are used to build the rule-based table, which is the representation of every possible case of inputs and its expected output.

Since the fuzzy logic controller processes user defined rules governing the target control systems, it can be modified and tweaked easily to improve or alter the system

performance drastically. New sensors can easily be incorporated into the systems simply by generating appropriate governing rules. Because of the rule-based operation, any reasonable number of inputs can be processed and numerous outputs can be generated.

As it is possible to understand from the example bellow, defining the rules becomes complex if too many inputs and outputs are chosen for a single implementation. In those cases it is better to break the control systems into smaller parts and use several smaller fuzzy logic controllers that can be distributed through the system.

2.5 Fuzzy quantification of knowledge

At this point, the system inputs and control logic is fuzzified and the generic concept of how to control the system has been done. The next step is to quantify the universe of discourse created in this fuzzification process in order to automate and control the system, using fuzzy logic. The linguistic values are represented/quantified using membership functions. These functions must confine the certainty of a linguistic variable in a continuous manner and can take many shapes, usually triangles, trapezoids or bell-shaped functions (6). This continuous manner can relate to a probability that varies from each membership function shape.

An example of two active triangular membership functions is shown in the figure 2.4.

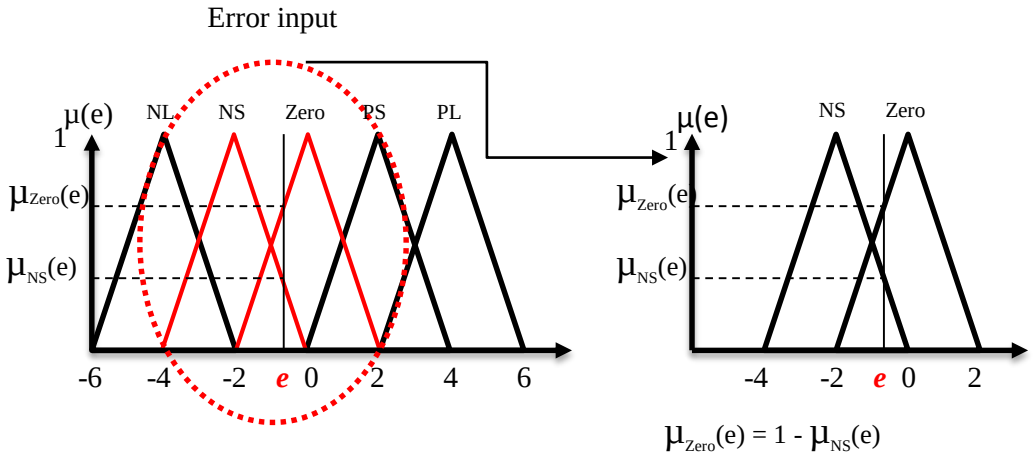


Figure 2.4: Selection of two membership functions

A system with 5 membership functions defines the linguistic variable “error”. For the input “error”, both the membership functions ‘negative small’ (NS) and ‘Zero’ are active and μ_x represents the certainty level of that membership function. Since triangular

membership functions represent a linear change of certainty, for the input *e,* that activates both 'NS' and 'Zero', $\mu_{Zero}(e)$ (certainty of membership function 'Zero' for the input *e*) is 2/3 and $\mu_{NS}(e)$ is 1/3.

Each linguistic variable is represented by membership functions and for a stated input it specifies which membership function that input is a member of. A good definition of the shape of the membership function is a crucial step to a good fuzzy controller.

Creating a basic and simple shape that quantifies in a manner that makes sense for the designer is always the best step. In later stages, to achieve high performance objectives it can be required to tune the shapes of the membership functions, but usually they only recreate a subjective approach.

2.6 Inference process

The inference process involves two important steps, the matching process and the conclusions.

In the matching process the premises of all the rules are compared to the controller inputs in order to determinate which rules are valid to the current situation. This matching process requires knowledge about the certainty of each rule, so that the more relevant the rule, the more impact it has in the current situation. Again, the process to match the rules can be very complex, there is a wide array of choices for the designer to match the rules, the simplest way is using the logic "and"/"or", but in the fuzzy universe, it is up to the designer to choose the best for each situation. The figure 2.6 shows the quantification of the "and" operation as an example of the matching process for a fuzzy system with 2 inputs, *e* and *de*, each with five triangular membership functions as shown in figure 2.5.

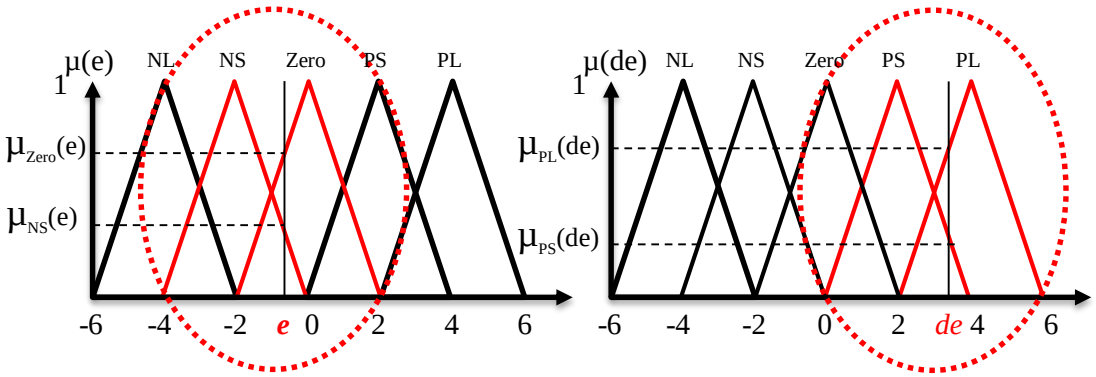


Figure 2.5: Set of membership functions for error (left) and change of error (right)

The matching process is identifying which rule is active. As shown in figure 2.6, there are four active membership functions, two for the error input 'Negative Small, Zero' and two for the change of error 'Positive Small, Positive Large'. These active membership functions 'match' in the fuzzy rule table, highlighted in red, selecting four different rules. The variable u_f represents a generic output membership function variable, for instance in motor speed control it can be "power".

Notice that, depending of the number of active membership functions, the number of rules can go from one to four.

Fuzzy Rule Table

E / dE	NL	NS	Z	PS	PL
NL	C_{00}	C_{01}	C_{02}	C_{03}	C_{04}
NS	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}
Z	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}
PS	C_{30}	C_{31}	C_{32}	C_{33}	C_{34}
PL	C_{40}	C_{41}	C_{42}	C_{43}	C_{44}

Fuzzy Inference and Output:

Rule 1: ***error*** is NegativeSmall *and* ***de*** is PositiveSmall then u_f is C_{31}

Rule 2: ***error*** is NegativeSmall *and* ***de*** is PositiveLarge then u_f is C_{41}

Rule 3: ***error*** is Zero *and* ***de*** is PositiveSmall then u_f is C_{32}

Rule 4: ***error*** is Zero *and* ***de*** is PositiveLarge then u_f is C_{42}

Figure 2.6: Fuzzy matching process

After the matching process is complete, the conclusion process starts. The matching process goes through the active membership functions and returns a set of rules. It is now required to analyze these rules and set a single conclusion for the output. For this, each rule must be analyzed independently. In the end, all single rules recommendations are combined to conclude the final output.

The figure 2.7 represents the conclusion process for each rule, using the product between matched membership functions values.

Rule 1: ***error*** is NegativeSmall *and* ***de*** is PositiveSmall then u_f is C_{31}

$\mu_{NS}(\boldsymbol{e}) * \mu_{PS}(\boldsymbol{de}) = D_{13}$

Rule 2: ***error*** is NegativeSmall *and* ***de*** is PositiveLarge then u_f is C_{41}

$\mu_{NS}(\boldsymbol{e}) * \mu_{PL}(\boldsymbol{de}) = D_{14}$

Rule 3: ***error*** is Zero *and* ***de*** is PositiveSmall then u_f is C_{32}

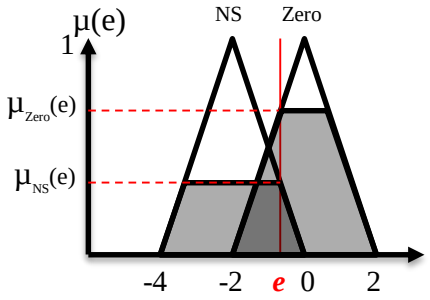
$\mu_{Zero}(\boldsymbol{e}) * \mu_{PS}(\boldsymbol{de}) = D_{23}$

Rule 4: ***error*** is Zero *and* ***de*** is PositiveLarge then u_f is C_{42}

$\mu_{Zero}(\boldsymbol{e}) * \mu_{PL}(\boldsymbol{de}) = D_{24}$

a)

$D_{13} + D_{14} + D_{23} + D_{24} = 1$



b)

Figure 2.7: a) conclusion phase, b) membership function certainty

The μ represents the change of certainty of the respective membership function and in this particular shape it matches the probability. As it is possible to see in the figure 2.7 b), the $\mu_{Zero}(e)$ is 2/3 and $\mu_{NS}(e)$ is 1/3. Each rule contribution is calculated separately (figure 2.7 a)) to a value D_x that represents the implied fuzzy set for that rule.

At this point, the fuzzy system has taken information from the inputs, quantified it to fuzzy logic, applied knowledge from the expert to it by using the fuzzy rules and got its conclusions in terms of fuzzy outputs. The only step left is to defuzzify these conclusions.

2.7 Defuzzification

The defuzzification phase uses the information produced by the inference process into numeric fuzzy controller outputs. Some authors relate the defuzzification part to the conclusion phase of the inference process. The ambiguity is clear, since most of the times, if the outputs are chosen correctly, the final value returned from the conclusion process is often the direct value to be applied to the system under control, and, when so, the defuzzification process is “hidden” in the conclusion phase.

The idea about the defuzzification phase is to know that while all the values inside the fuzzy controller work with fuzzy logic, at the end of the defuzzification process, all values in the output must be specific (crisp values) and without fuzzy logic implied.

The most usual defuzzification method is the COG (Center of Gravity) (7). The COG is calculated manually, since the membership functions used (2.8 b)) are simple. The COG method can be described as:

$$u = \frac{\sum_i C_i * \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \qquad (2.1)$$

C_i represents the output value, as shown in the fuzzy rule table in the figure 2.6. $\int \mu_i$ is the area under the membership function.

For the example at the figure 2.7, the defuzzification using the COG method is:

$$u = \frac{C_{31} * D_{13} + C_{41} * D_{14} + C_{32} * D_{23} + C_{42} * D_{24}}{D_{13} + D_{14} + D_{23} + D_{24}} \qquad (2.2)$$

2.8 Tuning the fuzzy controller

Often the designer comes to a point where the basic structure of the fuzzy controller is completed and the next step is to improve performance or test for better results. In a fuzzy controller, there are many ways to change the way that the controller works and therefore its results.

2.8.1 Tuning via scaling the universe of discourse

The more intuitive way to tune a fuzzy controller is to look at the fuzzy logic interpretations. The universe of discourse the designer built, using fuzzy logic, often provides soft boundaries between fuzzy sets. The designer can change these boundaries, for instance scale the fuzzy definition of linguistic variable or adapt the fuzzy boundaries. The designer should always be aware to keep the basic fuzzy rules applied, in this case, keep the entire input range defined. The figure 2.8 represents a fuzzy controller with different scaling gains for the inputs and output.

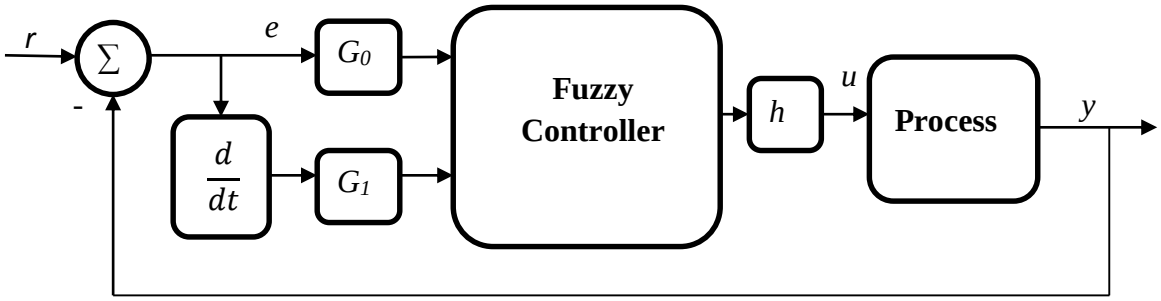


Figure 2.8: Fuzzy controller with scaling gain G1, G2 and h

It is really important to notice that scaling the gain via G1 has the same effect of changing the derivative axis membership function axis. In fact, choosing a scaling gain G1 = 0.5 is equivalent to have the change observed in the figure 2.9.

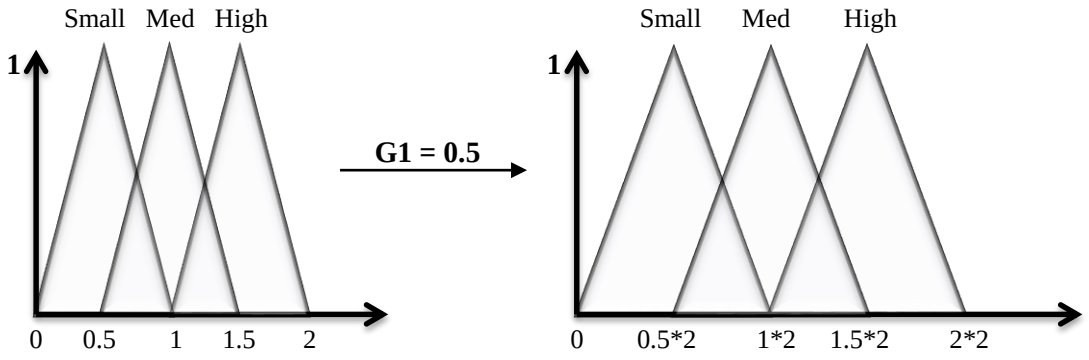


Figure 2.9: Scaling the input

The figure 2.9 shows that the choice of a scaling gain G1=0.5 results in scaling the horizontal axis of the membership functions by 1 / G1. In general, it is possible to get some conclusions about scaling G1 and its effects on the membership functions.

If the scaling gain G1 < 1, the membership functions are uniformly spread out by a factor of 1/G1, like the figure 2.10 represents. The opposite effect happens when the scaling gain G1 > 1 as the membership functions are uniformly contracted. The same effect would occur to the other input (error) for the G0 gain. However, the effect on the meaning of the linguistics that form the definition of the fuzzy controller change when scaling their respective input gains. In particular, when G1 <1, as the membership functions are uniformly spread out, it changes the relative meaning of the linguistics so that, for example " high" is now a membership function that represents higher values. The opposite happens when G1 > 1 as the membership functions contract, "high" now represents smaller values then before.

Similar statements can be made for all the other membership functions and respective associated linguistic values. The input scaling factors have an inverse relationship in terms of their effect on the scaling. A G1 factor greater than 1 corresponds of changing the meaning of the linguistics so that they quantify smaller numbers. While this inverse relationship exists for the input, scaling the output gain has the opposite effect as it is shown in the figure 2.10:

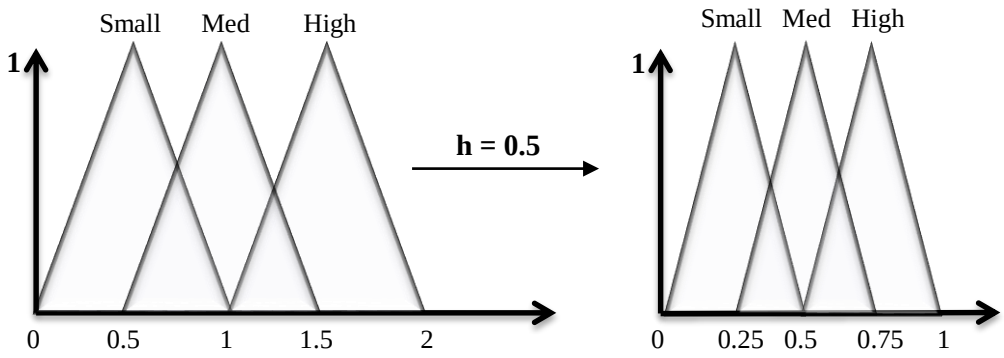


Figure 2.10: Scaling the output

There is a proportional effect between scaling h and the output membership functions. If h<1, the output membership functions contract, hence making the meaning of their associated linguistic quantify smaller numbers. If h>1, the output membership functions spread out, hence making the meaning of their associated linguistics quantify larger numbers.

2.8.2 Tuning membership functions

Sometimes it is not possible to achieve the desired performance tuning only the scaling gains. Often, it is required a more careful consideration of how to specify additional rules or better membership functions. The major problem is that there are often too many parameters to tune like membership function shapes, positioning, number and type of rules. In fact, for most cases there is not a clear connection between the design objectives (rise-time, overshoot) and a reasonable method that should be used to tune these parameters.

A simple method to tune output membership functions is to change their positioning. To do so, it is required to move their center. An easy and understandable example to tune membership functions is shown in the next figure 2.11:

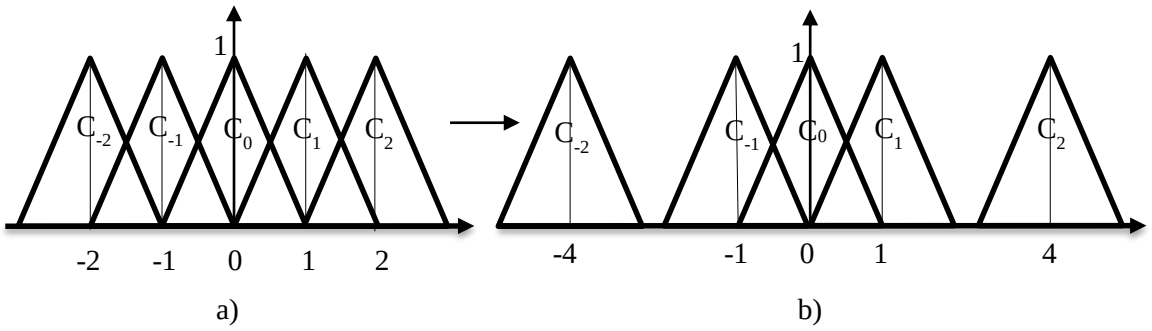


Figure 2.11: Nonlinear output membership functions

In the figure 2.11 a) it is represented a common linear equally spread set of output membership functions. Imagine the fuzzy system described in the figure 2.8. When the inputs error and change of error are small, likely the system is close to its set point, the output values are near zero. For the figure 2.11 a), even if the inputs change, they linearly increase or decrease the response from the output membership functions. If the designer feels that the system, for instance, should keep a linear and close to zero response when close to the set point and a much stronger output when far away from the set point. This change is represented in the figure 2.11 b), by replacing the centers of the output membership functions, using $C_i = \text{sign}(i)i^2$, where C_i is the center of the respective membership function.

If the systems inputs are near where they should be, then the signal applied to the system is small, but if the inputs change, then the signal applied to the system is much higher.

While there are some methods for tuning the logic premises used in the inference process ('and/or') or in the conclusion phase ('product/minimum'), these methods often do not provide insights into how these parameters affect the performance that we are trying to achieve (hence it is difficult to know how to tune them to get the desired performance). It often lends to a process of trial and error, which results must be carefully validated and interpreted by the designer.

The tuning process overall should be an interactive process after the first fuzzy requirements are met. The designer should feel that the tuning process should be adding more knowledge to the system or to better work with the knowledge it already has. Generally, if the designer is having difficulties building a good fuzzy controller, he probably needs more knowledge of the physics of the process to control, and then get that knowledge to properly affect the plant dynamics into the fuzzy controller.

3 Fuzzy Controller Design

3.1 Introduction

In order to implement the fuzzy controller it was chosen a DC motor RS210L (8) with the objective to control its velocity. It is a permanent magnet DC motor from Parvex (8 p. 2), using 24V supply and definition speed of 3000rpm. The tests setup used a previous work from Luís Terra that includes the LPC1759 and the power control module (2 p. 61) (figure 3.1).

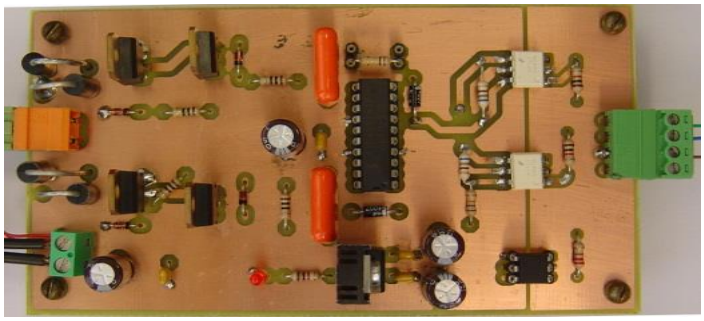


Figure 3.1: Power control module

Luís Terra developed two boards, one for the LPC2129 used in his work and the LPC1759 after its release that year. The LPC1759 (2 p. 106) (figure 3.2) is a significant advance from the LPC2129, doubling its performance. It also provides a Motor Control PWM (MCPWM) and a Quadrature Encoder Interface (QEI) module.

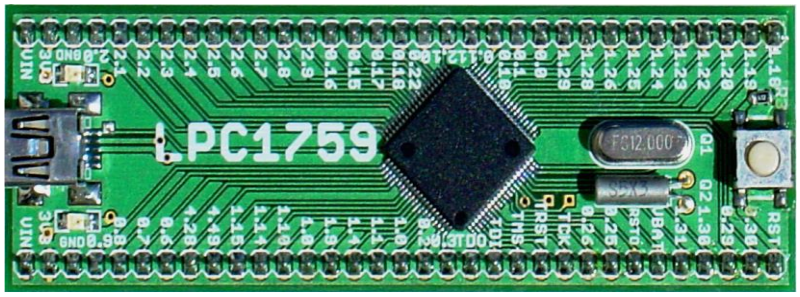


Figure 3.2: LPC1759

The DC motor is mechanically coupled with a parallel excited motor (load), as shown in the figure 3.3. This motor, unlike the permanent magnet DC motor has a very nonlinear behaviour and, even at open loop, behaves like a load that increases with velocity. If there is a constant current supply at the terminals of the generator, this current will not imply a constant torque for different velocities, different from the permanent magnet DC motors. This behavior is used to test the quality of the controllers.

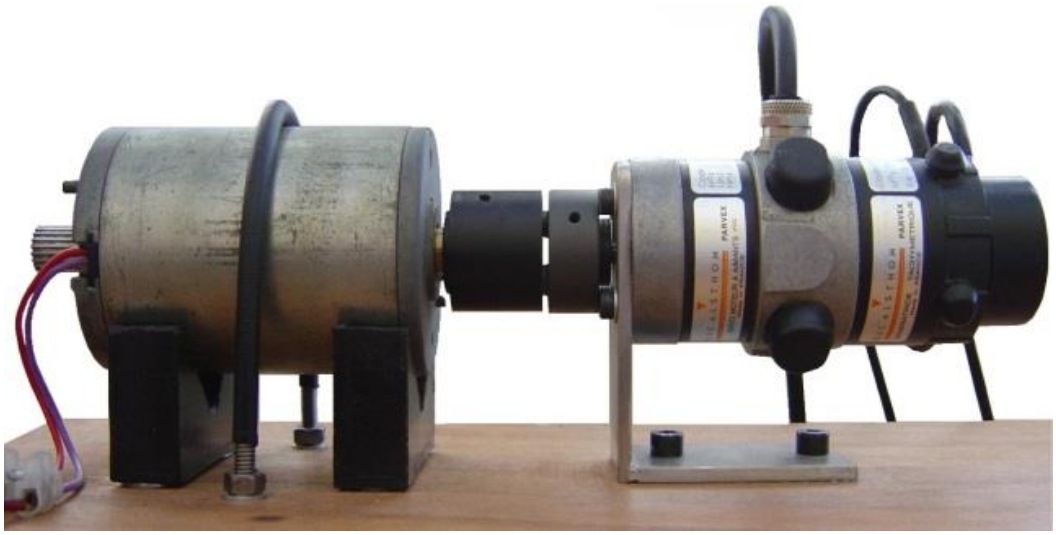


Figure 3.3: DC-motor (right) and generator (left)

The power supply was connected to the power control module with 3A maximum current and a 24V supply voltage.

A generic overview of the controller environment is shown in the figure 3.4. The fuzzy inputs are the velocity error and the error rate. These signals were provided by using the incremental encoder attached to the motor and the Quadrature Encoder Interface provided by the selected microcontroller (3 p. 554). The signal consisted in two phase signals and an index. The index counted revolutions while the phase signals were used to know the motor speed and direction. The selected output for the fuzzy controller is the PWM duty cycle, using the MCPWM module provided that generates a square wave with a programmed duty-cycle. The PWM is then directed to the power control module, which converts it to a DC voltage and applies it to the DC-motor. The microcontroller used two different UART units, one for programming the memory and other for communicating with Matlab.

Each module was tested separately, the QEI used to get information about the motor speed was compared with the tachometer values, and the MCPWM was compared with a square wave form created using the DAC.

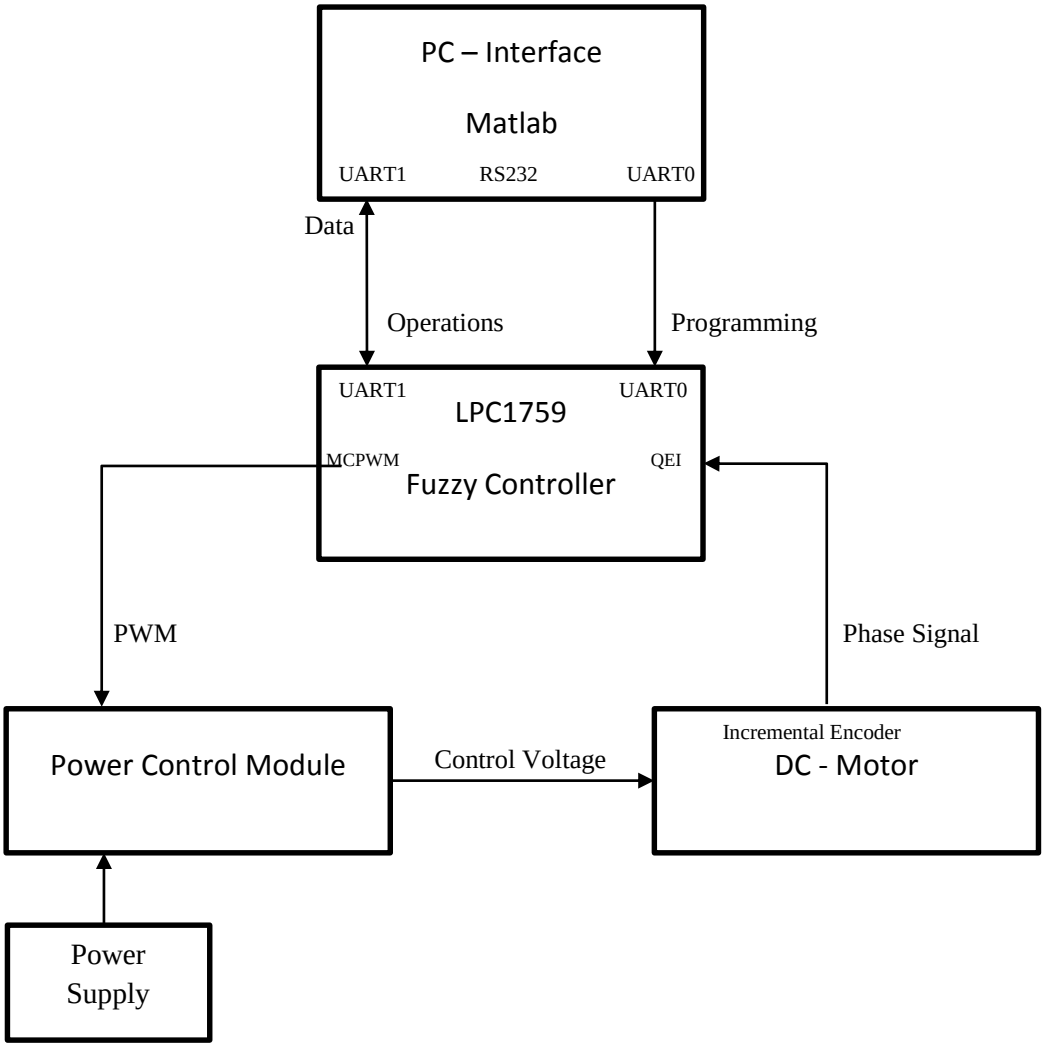


Figure 3.4: Controller environment

3.2 Interface

The methodology to operate the whole system had some problems. To better check and validate the system performance it was required to power externally the microcontroller, connect it to one PC using RS232, change the program, flash it to the processor memory, send the data using serial communication and print it to a file in the PC. Then the user could read the file using Matlab, plot the variables and take offline conclusions on the fuzzy controller performance. This workflow was not optimal for several reasons. The whole process was slow, the code was developed in Virtual machine Linux operating system so the serial communication used to flash the processor memory and then to write the data (system output, control signal, etc.) could not use the same

communication port, as one was only accessible on each operating system at once (Matlab was running on Windows). When the user was evaluating the data, it was an offline view of the motor performance, which limits the fuzzy controller designer options.

In order to solve these problems and add new functionalities to the whole system, it was developed an interface using Matlab. It separated the programming of the memory, using different UARTs for the data communication and program flash. The microcontroller power supply was provided directly from the PC-Com.

From the system interface the user could select the input wave, start/stop the controller, watch the real time plot of the system output and the control signal, save any plot at any given time and change the active load value. These options allowed a faster tuning and evaluation of the fuzzy controller, which majorly improved the fuzzy control development.

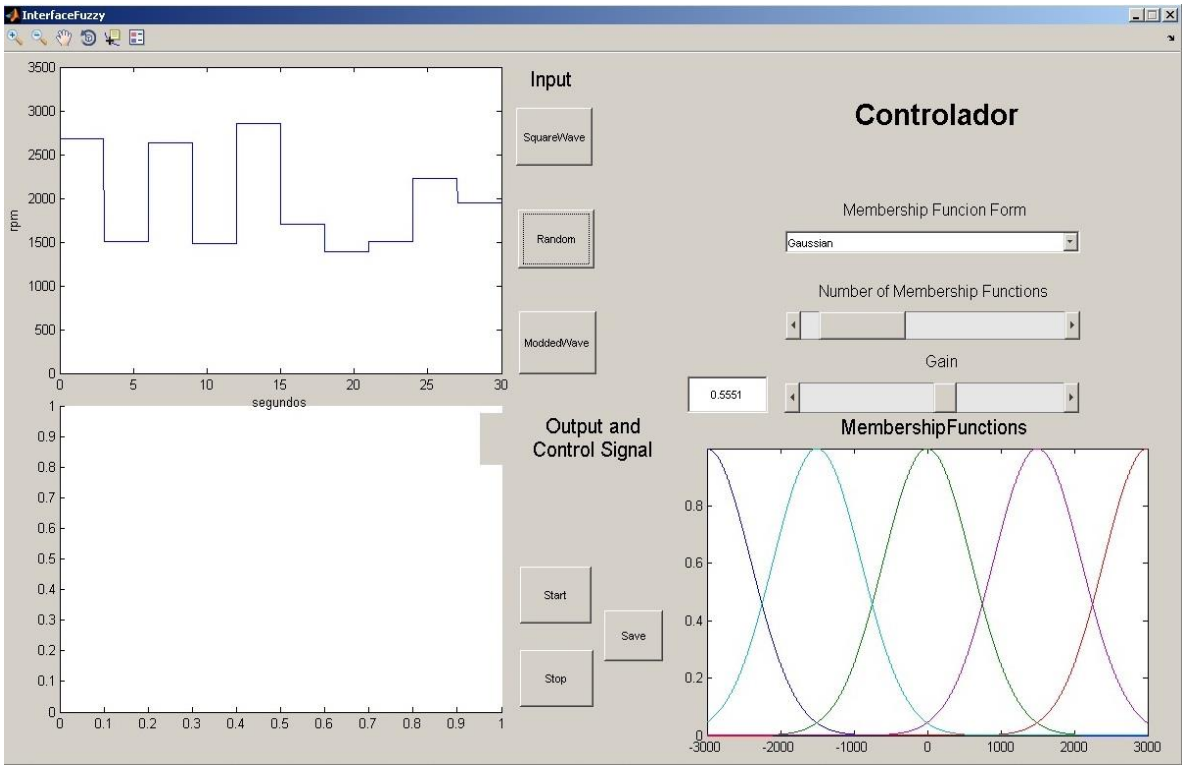


Figure 3.5: Fuzzy Interface

An image of the current fuzzy interface is shown at the figure 3.5. If the user wants to save a specific plot, it must be selected previously by clicking on the plot then pressing save. The start/stop buttons only affect the communication between the microcontroller and the fuzzy interface, it does not shut down the motor. The moddedwave button is the standard input created for testing the controllers. The random button creates a random 10 step square wave as it is represented in the figure.

The membership function form is only available with triangular shape and Gaussian shape, but it could not be implemented since it required floating point. The number of membership function actually changes the number of both inputs, and it ranges from 3 to 21. This also allows the construction of the rule set table, which will not be tuned, and it will follow a base gain between the numbers of membership function, for instance, it would be something like the figure 3.6.

	f_1	f_2	f_3	f_4	f_5	f_6	
t_1	0.6	0.4	0	1	0	0	
t_2	0.4	0.6	0	0.1	0.9	0	
t_3	0.6	0.4	0	0.2	0.8	0	
t_4	0	1	0	0	0	1	
t_5	0.8	0.2	0	0	0.4	0.6	
t_6	0	0.6	0.4	0	1	0	x gain

Figure 3.6: Automatic rule set table

The interface really helped the development of the controllers due to the real time plot of the system variables and changing the set point. However, there were a lot of significant improvements that could be done to help the designer to build better fuzzy controllers. Some options were still tested and disabled after, due to some complexity problems with the time restrains the program in the microcontroller had.

3.3 Knowledge gathering

The amount of information the designer possesses about the controllable system majorly dictates his success on developing a controller. Some basic performance tests were made to the motor in that sense. The first test (figure 3.7) evaluates the motor response to a 0 to 100% duty-cycle variation and after reaching the maximum speed, back to 0% duty-cycle. The motor vein was attached to another DC motor working as generator but without load applied to it. The velocity was measured using the QEI module.

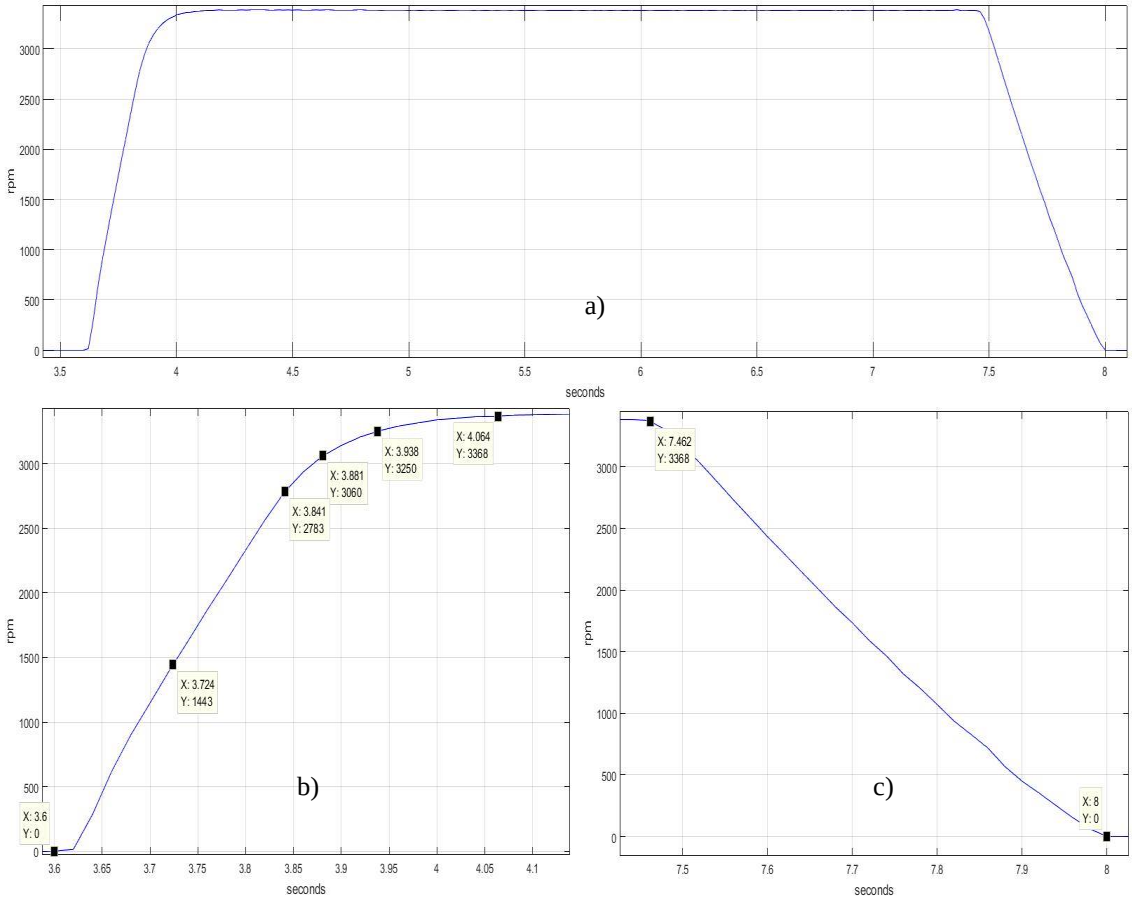


Figure 3.7: a) DC-Motor velocity (rpm), b) 0-100% Duty-cycle c) 100-0% Duty-cycle

This test insights the controller designer about the motor behavior at maximum power. It is possible to identify several different slopes that define the rate of change of the motor speed. It takes around 130ms to make the first 1500rpm and then 150ms to reach 3000rpm. Over the 3000rpm mark the motor slows a lot the rate of change taking 120ms to reach the maximum speed. This information is important to get a reference about the maximum rate of change possible for this setup. Since the motor is receiving the maximum power, this times reflect the minimum times for rpm variations.

Over to the other side, when stopping, the motor follows a close to linear slope, taking around 500ms to stop. This difference should be important and reflect in different gains in the rule-set table for the membership functions that define negative error.

The second test (figure 3.8) was made under the same setup and is a one second paced increment of the PWM wave duty-cycle that controls the motor velocity.

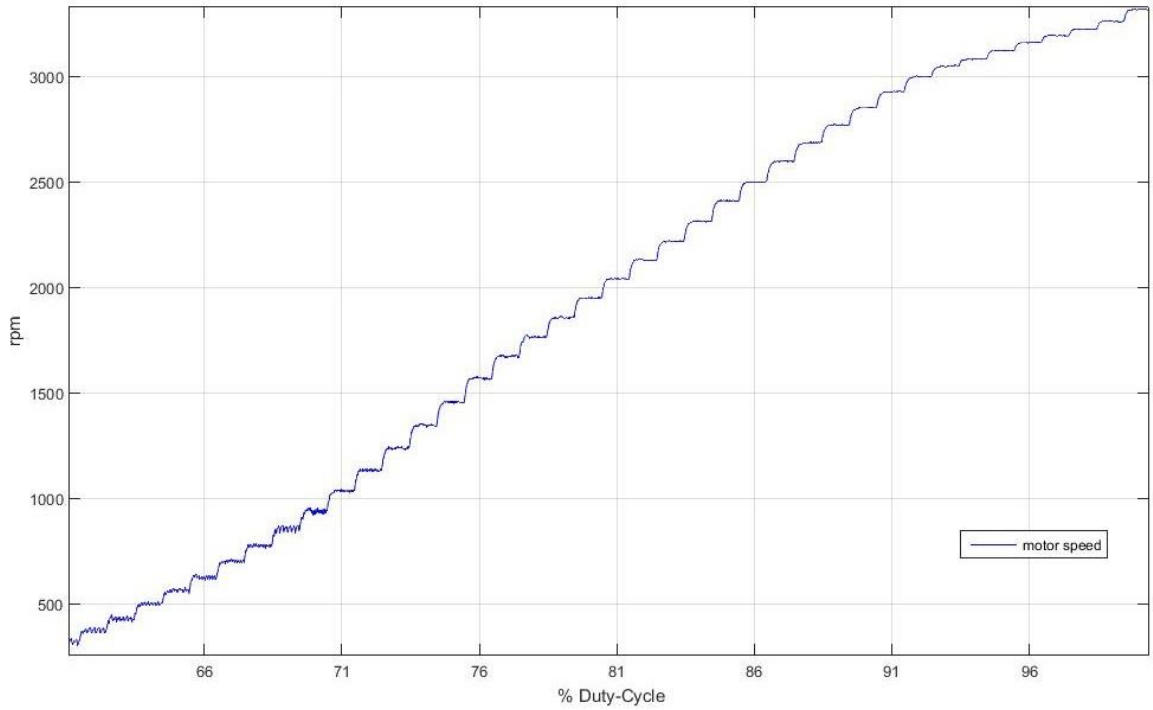


Figure 3.8: Paced increment of the PWM wave

The main goal of this test is to better tune the controller response when near the set point. This graphic shows the different rpm “jumps” of the motor speed. This was achieved by keeping the duty-cycle increment fixed (1%) and evaluating the response during the 60-100% duty-cycle range.

The graphic only shows the variation between 60-100% duty-cycle since the acquisition of the motor speed at lower speeds was imprecise due to error. This error is due to the counting of the phase edges on the QEI on the microcontroller which at lower rates a difference of 1-2 phase edges would be significant.

Having the basic information about the motor, having the hardware design ready, the inputs and the output selected, the next step is to create the fuzzy control system.

3.4 Fuzzy design

The controller environment shown in the figure 3.4 represents the overview of the hardware. The fuzzy design however, relates to the hardware in the form of the output it provides to the system, in this case, the PWM wave sent to the power control module and the inputs it receives, the wave form from the incremental encoder attached to the motor.

The motor speed was calculated in the LPC1759 QEI module after counting the phase edges from the incremental encoder in a period of time.

The difference from the set point and the actual speed (error) was the primary input for the fuzzy system. It was the most natural way to gather information from the controllable system and by selecting the error as an input, it can be easily extracted knowledge from the membership functions to the inference system and use it in the fuzzy rules.

For the second input it was selected the change of error. It is a source of information from the first input and also simple to interpret to better tune the fuzzy controller performance.

The reasons for the output selection of the fuzzy controller are quite natural, as it needed to be a variable that could closely relate the knowledge gathered in the fuzzy rules from the membership functions and the actual objective of the system – velocity control. Therefore the duty-cycle provides a way to relate with the motor velocity, as it controls the voltage applied. It is also important to note the way that the system changes the output, the duty-cycle percentage. The microcontroller changes the duty cycle by dividing two 32 bits register (3 p. 518), one representing the processor frequency and the other the frequency of the PWM wave. In order to keep a high velocity resolution, this is, the least incremental gain possible from the controller change the motor velocity in the smallest way, the frequency of the PWM wave needed to be inside the human ear frequency sound. Since the software used fixed point, the resolution required for the duty-cycle constant to change 1rpm should be around 15000. To diminish the error from the fixed point, it was used 100000 to represent 100% duty-cycle. Since the processor frequency is 120MHz, the maximum frequency for the PWM wave is 1200Hz. Even using the constant at 15000 to represent 100% duty-cycle, it would mean 8000Hz. This options bring some noise. To better accommodate both, the frequency was set to 20 000 Hz, giving a much more smooth rotation to the motor, with a small noise and with a resolution of three rpm.

The sampling period was set to 20ms. With the information from the figure 3.7, the maximum velocity change for 20ms is around 250rpm. Since the motor speed can go up to 3300rpm and further tests should be realized with active loads, that lower this rate, the 20ms sampling period should prove sufficient for this motor behavior. To guarantee the sampling time, the whole software was built upon FreeRTOS. The FreeRTOS scheduler (9) used only two tasks. The highest priority task, that sets the controller increment to the motor, ran at 20ms. The other task, at the same rate, computed all the controller arithmetic and the communication with the Matlab. This way, the variation of time that the controller arithmetic took to run did not affect the controller output step.

The whole system used fixed point, avoiding the floating point unit. This was the only computationally viable choice, since the fuzzy controller arithmetic consumed a lot of time resources plus the communication with the interface, reading sensors, etc. However, it set a maximum precision value, since all numbers had to be multiplied to bring some sort of precision to the system. Other consequence of using fixed point is that without the floating point unit, for instance, it was not possible to build bell shaped membership functions.

It was developed a startup process lasting 0.1 seconds to start up the motor at its minimal PWM command in open loop. As in digital control, this action prevents a lot of unwanted factors and the first measurements for the inputs were only done after this startup, so that the control system already deals with a functional controllable system. The basic control design idea is to have the fuzzy controller built based on proportional, derivative and integral components.

The next sub-chapter explains the construction of the fuzzy controller and the thought process behind it.

3.5 Fuzzy structure

3.5.1 Membership functions

Membership functions are more subjective than objective. This means that the information they quantify reflect the engineer view of the system and others might quantify it in a different manner.

3.5.1.1 Number

The selection of the number of membership functions was not an obvious choice from the start. Since the primary source of information was the first input (error) it seemed reasonable to emphasize that choice on the number of membership functions. The choice ended up being eleven membership functions for the error and three for the change of error.

The eleven functions divided the possible range of error, having one centered at zero error, and the other five symmetrically represent high error and small error, positive and negative. For the change of error, one centered in zero and one representing positive change of error and the other one negative.

The number of membership functions to use in this design was by far the most important and most conscious decision between complexity/simplicity, execution time and performance. The thought process behind it is explained separately since most of the problems came across during the fuzzy design implementation, the performance tuning and testing.

The first approach towards the number of membership functions is to look at the inputs. The error input, range [-3000, +3000] rpm and the change of error [-1000, +1000] rpm/step, represent the fuzzy inputs and must be divided by membership functions. Each of these membership functions will confine a part of the input range and interpret the input in a way to be used in the rule set table. The number of membership functions then reflects how many different phases the controller will have and the boundaries between each. The first problem adjacent to this decision is the input range versus actual input.

In most fuzzy logic controllers, initial membership functions are normally laid evenly all across the universes of discourse that represent fuzzy control inputs. However, for evenly distributed membership functions, there might be a problem that may adversely affect the control performance. If the actual inputs are not equally distributed, but instead concentrated within a certain interval that is only part of the entire input area, this will result in two negative effects. On one hand, the membership functions in the dense input area will not be sufficient for a precise response to the inputs, because these inputs are too close to each other compared to the membership functions in this area. The same fuzzy control output could be triggered for several different inputs.

On the other hand, some of the membership functions assigned for the sparse-input area are never active or just trigger once, being technically a waste of resources.

Both these problems are quite difficult to solve, especially in the early stages of a fuzzy control system. The first problem could be solved by adding more membership functions which would decrease the operating range of each, making them more precise. But this is not really a valid solution. In this system, having 11 membership functions would result in 500rpm difference between each. To change this value by half, it would be necessary to increase the number of membership functions to 23. It is important to remember that for the motor speed, evaluating high error, should only happen a few times between set point changes, and only on large set point changes. Usually the controller will have much more effort for the near to zero error membership functions, where the controller should require a more clustered zone of membership functions.

The other problem is the fact that by increasing the number of membership functions, the complexity of the rule set table also increases to a maximum of n x n. If both the inputs increase from 3 membership functions to 5, then the rule set table will grow

from 3 x 3 to 5 x 5. In fact, these choices proved even more challenging when trying to improve the controller performance.

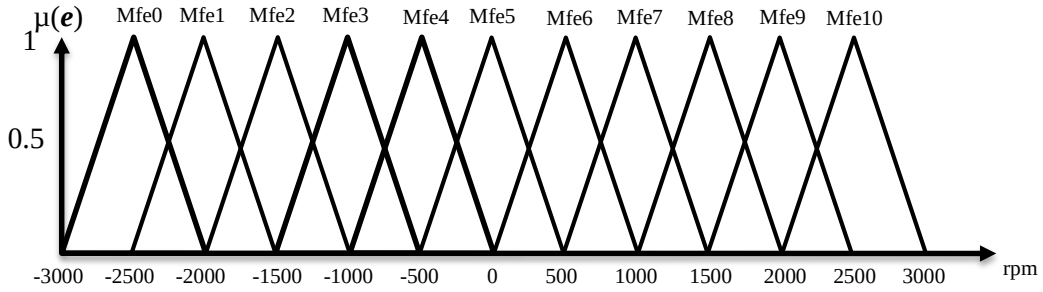
The final number of membership functions ended up being 11 for the error input, spacing the error for 500rpm between each function, from Positive Very High to Negative Very High. For the change of error, only 3 membership functions were built, enough to identify positive, zero and negative change of error. These choices would create 33 different phases of action in the rule set table which, by analyzing the figures 3.7 and 3.8, is a bit low. The main problem of using 11x3, is that often the same membership function will be active for several consecutive controller steps, which is not optimal when reaching the set point (zero error) value. In the change of error input, this is even more highlighted, where the central membership function will always be active.

3.5.1.2 Shape

The fuzzy controller used triangular shape of membership functions with interception at half curve. This way the conception about the certainty of the linguistic variable became linear by phases, which not only simplifies the through process and analysis of the membership functions as it makes more sense after the information gathered in the figure 3.7. It is possible to identify several linear phases of the motor speed during the tests. More complex shapes of membership functions would require floating point, which is out of this controller capabilities, so when using fixed point, the triangular membership functions are simple to implement and computationally fast.

The fuzzy controller number and shape of the input membership functions is shown in the figure 3.9. The change of error membership functions range from [-1000, 1000] rpm/step. The data from the figure 3.7 shows that without external interference with the system, the velocity cannot change quicker than 350 rpm/step. Even so, it was selected 1000, as the system would be tested using active load changes. These loads when applied/detached will drastically change the motor speed. By having a larger range in the membership functions, the controller can have a better response. The downside of this larger range is that in the setup without active load the controller could be less precise in the [-350, 350] rpm range. This actually happens because of the linear change behavior of the triangular membership functions. Since the change of error only uses three membership functions, expanding their range forces the linear behavior of the left and right membership functions to continue for a larger change of error, which is the good part. The downside is that the central membership function is now always active [-500, 500] rpm/step.

Error Membership Functions



Change of Error Membership Functions

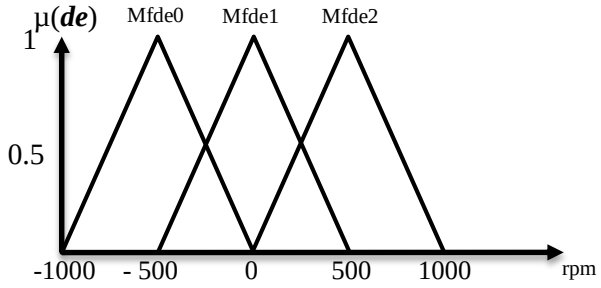


Figure 3.9: Fuzzy Controller input membership functions

During the development of the fuzzy controller it was quite difficult to have the perception when would a different shape of membership function, this is, a different change of certainty, be required or optimal for speed control on the motor. In fact, many times, the simplistic and linear approach that the triangular membership functions offered the system was often a safe liability. Maybe due to the lack of experience, it proved easier to add nonlinear behavior to the fuzzy control by adapting the rule-base set than changing the shape of the membership functions.

3.5.2 Inference mechanism and defuzzification

The inference mechanism starts by identifying all active membership functions for the respective input and their certainty level. The software always finds the (active) most left membership function first and calculates its probability (P0). Then, as the system uses triangular membership functions evenly spaced and the edges of each function are equal to the centers of its adjacent functions, the next membership function is always active and

with probability 1-P0. The software proceeds to match the rules using “and” implication. With only two inputs, there can only be 1, 2 or 4 active rules. It will be selected only one rule if both the inputs have a membership function with 100% probability, two rules if one of the inputs membership function have a probability of 100% and four rules if both inputs have two active membership functions with probabilities different than 100%.

The software always assumes there are four active rules but, when the probability is 0% their contribution is 0, as it is possible to see in the expressions bellow, where μ_{Mf} represents the probability associated with its respective membership function.

$$\mu_{Mfe1}(\boldsymbol{e}) * \mu_{Mfde1}(\boldsymbol{de}) = D_{11} \qquad \mu_{Mfe1}(\boldsymbol{e}) * \mu_{Mfde2}(\boldsymbol{de}) = D_{12}$$

$$\mu_{Mfe2}(\boldsymbol{e}) * \mu_{Mfde1}(\boldsymbol{de}) = D_{21} \qquad \mu_{Mde2}(\boldsymbol{e}) * \mu_{Mfde2}(\boldsymbol{de}) = D_{22}$$

In the conclusion phase, the software uses the matched information (D_{xx}) and the rule set table to get the conclusions for the COG method in the defuzzification process.

The rule set table is shown in the figure 3.10, where the value “k” represents a constant to multiply since the software uses fixed point.

de \ e	Mfe0	Mfe1	Mfe2	Mfe3	Mfe4	Mfe5	Mfe6	Mfe7	Mfe8	Mfe9	Mfe10
Mfde0	-165k	-135k	-125k	-120k	-75k	-45k	-75k	-120k	-125k	-135k	-165k
Mfde1	-110k	-90k	-57k5	-42k5	-25k	0	25k	42k5	57k5	90k	110k
Mfde2	165k	135k	125k	120k	75k	45k	75k	120k	125k	135k	165k

Figure 3.10: Rule-set table with the inference and defuzzification phases

This is an example of the rule-set table used in control the motor speed. The error membership functions are on top and the change of error on the side. The relation between the gains presented and the output PWM is 10k = 1% duty-cycle. With the information from the figure 3.8, 1% DC is around 100 rpm.

As it is possible to see, looking only to the error membership functions, the incremental gain between functions is around 20k to 30k, which represents a careful approach, especially when reaching some kind of stationary speed.

The change of error is 3 times more aggressive, especially at Mf5-Mde0/2. It is really important to note that like it was previously explained, the range of membership in the change of error input it is quite large. This forces most of the inputs to be in the central membership function and with high probability, making its attached rules more influent. One way to decrease this effect is to force the attached rule for the Mfde0/2 to be more impactful, making it so that even with low probability, they still influence the system enough.

3.6 Performance

It is possible to evaluate beforehand the expected performance of a designed fuzzy controller. Knowing its basic structure allows the designer to predict in a certain way how the control mechanism will behave under specific circumstances that are always very helpful for further tuning or understanding the results given.

The overall performance of a fuzzy controller can be measured by analyzing the membership functions. Their shape defines the certainty they influence the output value and the inferred value the way they reflect it. With multiple membership functions, it is possible to define different control behaviors to better adapt for different plant behaviors. This way, the fuzzy controller performance can change between membership functions and its actual expected behavior is a match between one or more membership functions, depending on the number of active membership functions.

For this specific design, there are some very important aspects to mention, that are relevant for understanding the performance analysis. The fuzzy controller uses an integral of the control variable in the output, which means that every new plant input will be the sum of the current output value given by the fuzzy controller plus the last output. Looking at the fuzzy rule set table at figure 3.10, it is possible to make some further conclusions on how the incremental gain will affect the system. As the figure 3.9 shows, the membership functions are spaced by 500rpm. Imagine that the set point changes 1000rpm, making it so that the error is 1000, selecting Mf7 and Mfde2. The output would then be 120k, which from the knowledge gathered from the motor behavior at the figure 3.8 will be enough (if given time) to change the speed around 12*83 = 996rpm (1%DC = 83rpm). This increment would be sufficient to reach the desired new set point but, as the figure 3.7 shows, the motor dynamics will not allow such high speed variation in one step time. At maximum power it would still take around 6 steps for the motor to change 1000rpm. The next time steps, the controller will continue to reinforce the control signal, but at a much lower rate, still, all the increments after the first should only try to improve the rise time and then softening down the gain towards a stationary point.

This kind of analysis is always possible but it gets difficult to predict with the ongoing change of speed towards the set point. For this reason it is really important to have a good source of information about the plant behavior, in this case, the motor. If it would be possible to know the effects of each increment at each speed, it would be possible to make a model for the motor using the membership functions and the rule set table that would inherit the motor dynamics.

Other point that helps this kind of analysis is the shape of membership function. With triangular shape, the change of certainty is linear and, if the rule set table uses linear gains then the controller would be linear and would only require a minimum number of membership functions to perform. Since the DC-motor-Generator setup has a nonlinear response, it is wise to tune nonlinear gains for each membership function on the rule-set table that better represent the motor behavior.

To control the motor speed using error and change of error inputs, the central membership function gains increased importance. It represents the last increments of power to control the motor and which the control model converges to, since it is the only membership function that will always be active at some point in this type of control. While this perspective of the central membership function is important, the practical design can change it. There will always be a combination of two membership functions that define the controller output for each input and this combination, or at least the grade of influence of each membership function, is directly given by the range of each function.

Outside of the fuzzy control performance there are general control measurements that are useful for comparison between controllers. In the results chapter the following metric (9) will be used to compare the fuzzy controller and the PID.

Integral of the Absolute Error (IAE) where:

$$IAE = \sum_{k=1}^{n} | e(kh)| \tag{3.1}$$

gives basic information about the overall error of the system.

Integral of the Square Error (ISE) where:

$$ISE = \sum_{k=1}^{n} e^2(kh) \tag{3.2}$$

gives enhanced information about greater errors, since errors are squared.

Integral of the Time -Weighted Absolute Error (ITAE) where:

$$ITAE = \sum_{k=1}^{n} k\,|e(kh)| \tag{3.3}$$

improves the information given by IAE, adding more importance towards errors that persist over time.

Integral of the Time-Weighted Square Error or Mean Square Error (MSE) where:

$$MSE = \sum_{k=1}^{n} k\,e^{2}(kh) \tag{3.4}$$

Mean square error is really important, since it prioritizes large errors, normal in the initial system response but since it multiplies with time, large errors that persist for long will be more relevant (10).

4 Results

4.1 Introduction

The velocity control was tested using the developed fuzzy controller explained previously and a PID controller which will be explained further in this chapter. Since there were no specified performance objectives, it was opted to build a generic fuzzy controller.

This dissertation objectives are based on understanding the digital control towards fuzzy controllers, being aware of which parts could improve the controller performance, what to do to make the controller more precise and understanding the problems that each step revealed.

The precision of the controller was the first evident problem. From the hardware perspective, being able to gather the velocity by counting the phase edges directly from the incremental encoder mounted on the motor shaft proved to be a significant increase in the input resolution. The other option of using the tachometer had a resolution of 6Volts/1000rpm. This meant a 0.6mV for each rpm that was highly susceptible of error. Even so, counting the phase edges from the phase signal in the QEI for a step time (20ms) brought some problems at lower speed. For instance, if the motor is rotating at 500rpm, a single edge count means a variation of 4rpm. This means that even if the control signal does not change, there might be a slight velocity variation due to that. At higher speed, over 2000rpm, a single phase edge count will barely be noticed.

For the software of the controller, the precision was majorly dictated by the use of fixed point. By only using integers there is always a precision loss. In fact, as it will be shown in the subchapter 4.3, the minimal increment to the system was of 3rpm.

4.2 PID Controller

To compare and validate the fuzzy controller performance and results it was selected the PID controller for comparison. To develop the PID controller it was selected the Matlab PID Tuner toolbox with plant identification.

The plant identification required the data from a step command, like the one shown in the figure 3.7. After identifying the plant it was possible to estimate the PID parameters and tune them towards slower-faster behaviors.

After some modest selection of the PID parameters it was tested on the microcontroller. The control results were far from optimal, with around 50% overshoot. The system was further manually tuned to reach the set point without overshoot and the final results are shown in the next subchapter.

4.3 Experimental Results

The experimental results shown in this subchapter all used the same reference signal. It is a square wave that ranges from 1500rpm to 3000rpm with several different size steps used in between as it is shown in the figure 4.1. The main purpose of using the square wave form is that it emulates a perfect variation, only digitally possible, which can be reflected as the best case possible.

The close loop output should be analyzed as it tries to follow this input wave as closely as possible, in all its different steps, each separated by 3 seconds.

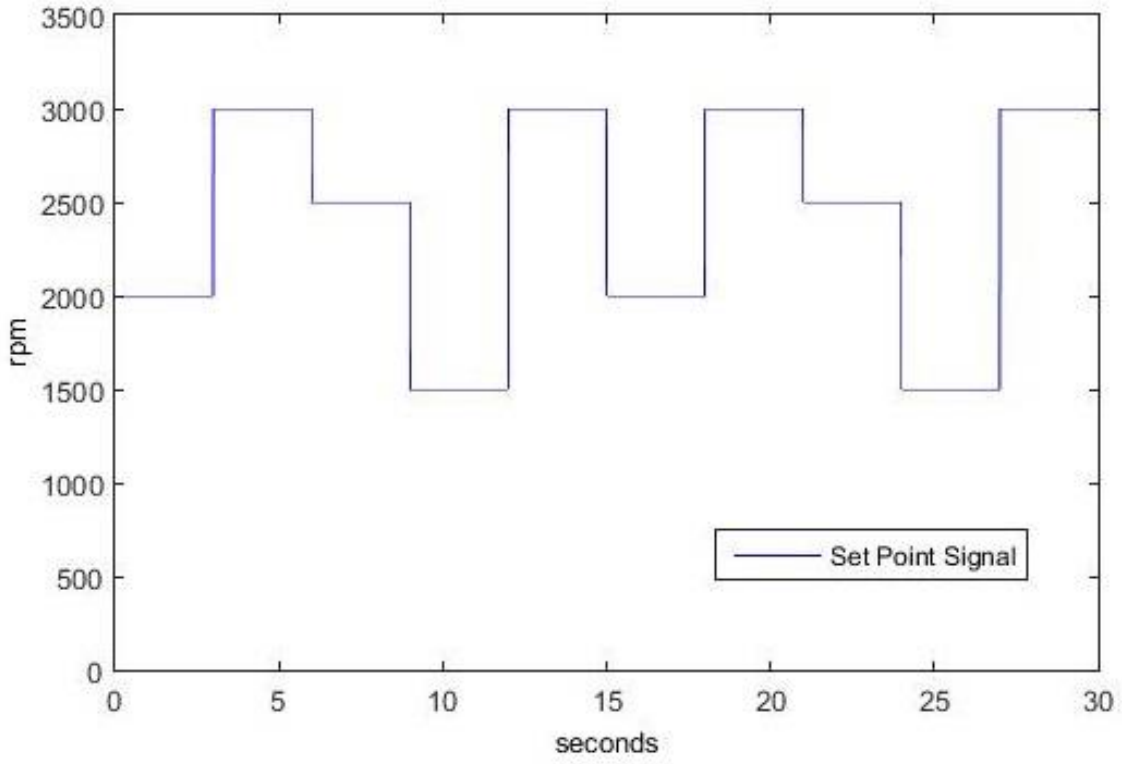


Figure 4.1 Tests reference signal

4.3.1 PID and Fuzzy controllers without load applied

The first tests with the controllers used the setup shown in the figure 4.2. The DC motor has it vein attached to the other motor (used as generator) but without any load applied to its terminals. At the right of the motor is the power control module that receives the PWM wave from the microcontroller using the MCPWM module. The reference signal is received through RS232 from the Matlab, using the Uart1.

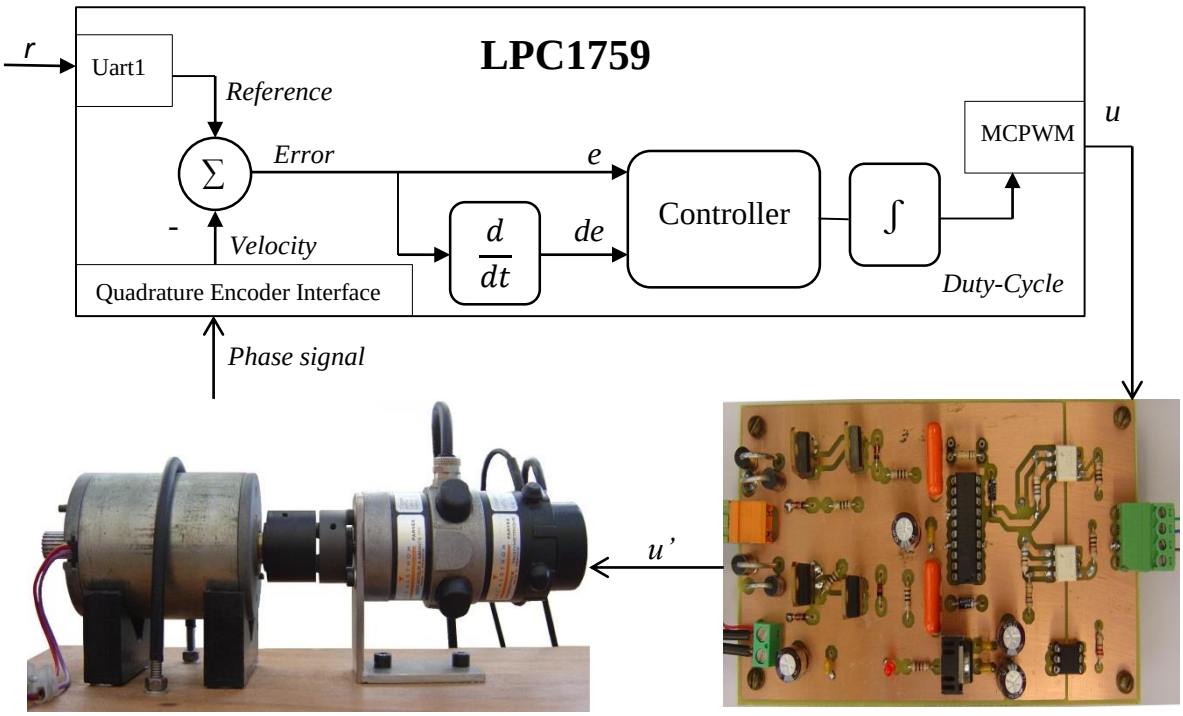


Figure 4.2: Motor speed set up without load

The figure 4.3 represents the output for the PID controller for the figure 4.1 reference signal. The signal repeats twice, for the course of 60 seconds. The PID controller has no overshoot. High errors are quickly recovered but without overshoot the controller is slower at small errors, taking some time to reach the steady state value.

At lower speed the motor tends to have a small oscillation, which comes in line with the way the motor speed input is gathered. At lower velocity, there are fewer impulses counted, whereas the difference between few impulses can generate a resolution error in speed that does not reflect the actual state of the motor. It is possible to notice this effect gradually getting nullified as the motor speed increases, being quite noticeable at 1500rpm, slightly noticeable at 2000rpm and barely seen at 3000rpm. This effect is shared between both controllers due to its source being from the input and not from the respective controller influence.

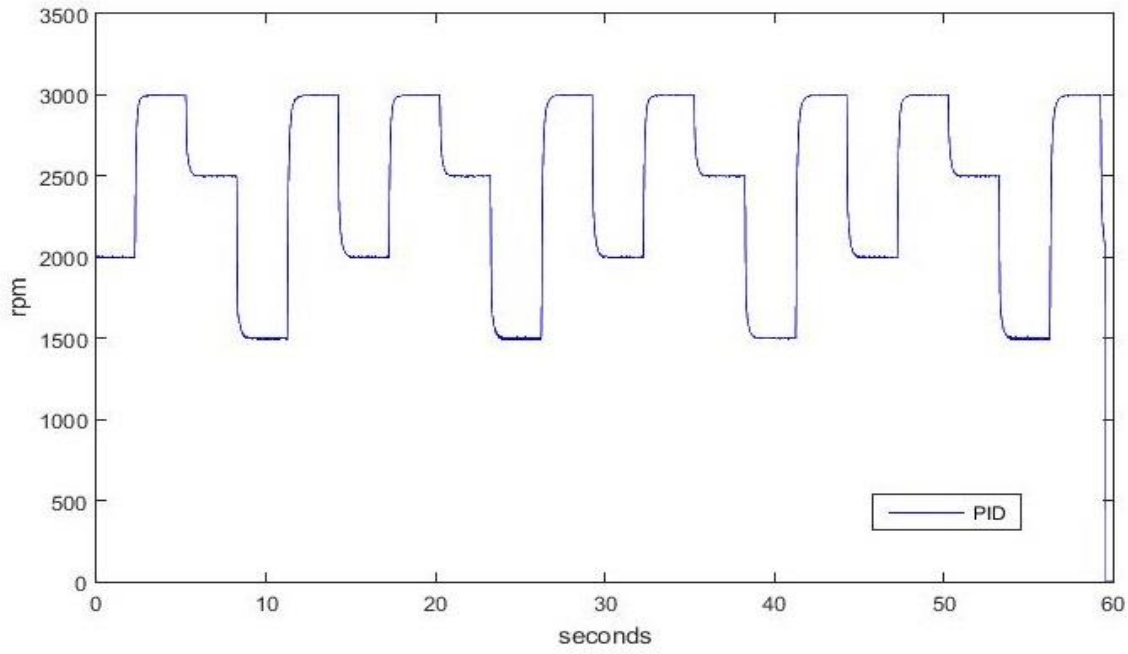


Figure 4.3: Motor speed for the PID Controller

The control signal for this output is shown in the figure 4.4, where it is possible to notice the usage of the controller signal the duty-cycle wave sent to the power control module. When the set point changes the control signal goes close to 100%, responding to the large error input. When the set point changes from 1500rpm to 3000rpm, the control signal actually reaches 100% and as the motor speed approaches the set point value, the control signal drops and, to avoid overshoot, even drops slightly more than the final value. In the 1000rpm set point changes this behavior is barely noticeable.

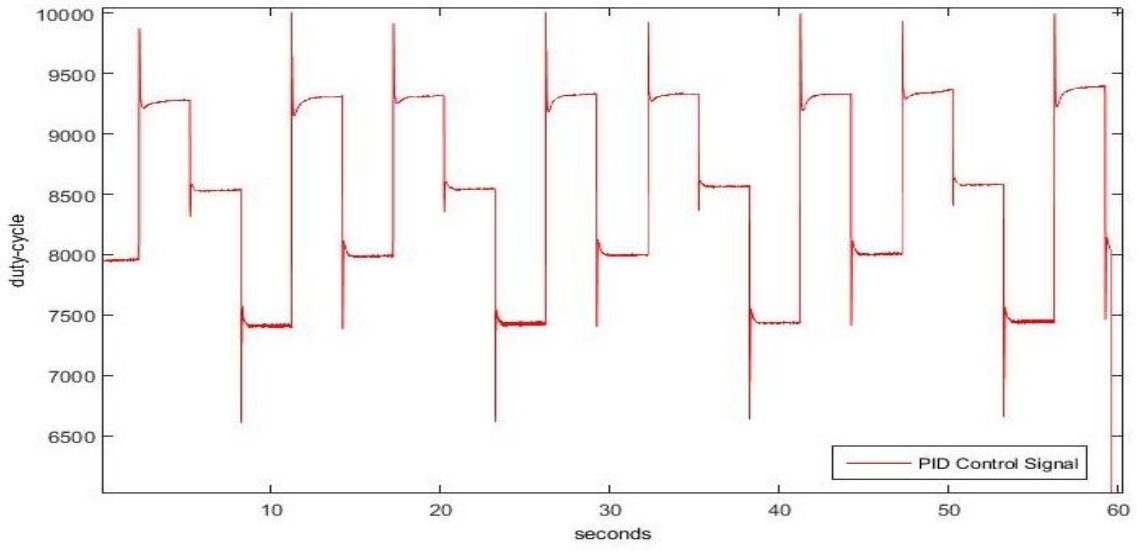


Figure 4.4: PID Control signal

The next figure shows the output signal for the fuzzy controller.

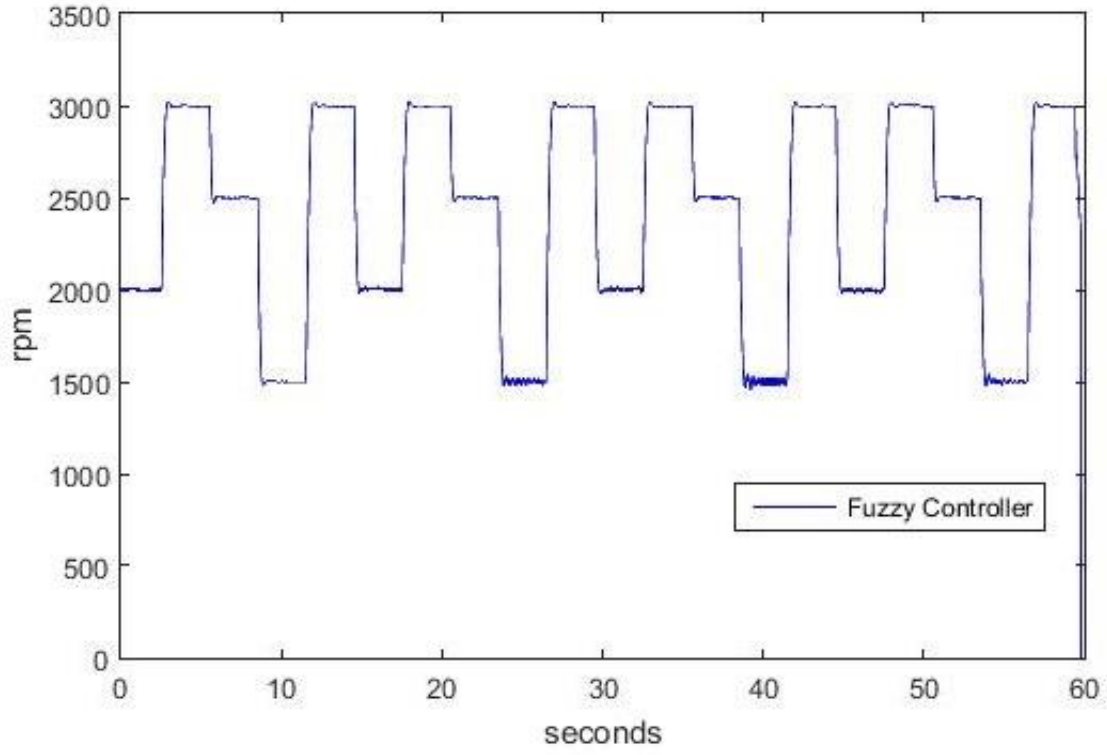


Figure 4.5: Motor speed for the Fuzzy Controller

At the figure 4.5 it is possible to notice the overall fuzzy controller time performance for the set point wave shown at figure 4.1. The system output follows the input really closely, with barely any overshot. The rise time (0%-100%) for the 1500rpm to 3000rpm is 0.42s, for the 2000rpm to 3000rpm is 0.34s. The time for the 3000rpm to 2500rpm is 0.2s, for the 2500rpm to the 1500rpm is 0.28s and from 3000rpm to 2000rpm is 0.3s. The fuzzy controller approach towards every new set point is slightly different. While in PID controller it follows a predictable variation towards the various input changes, the fuzzy controller can have slight differences. Since the triangular membership functions offer a linear by phases change of certainty but the rule set table values change non linearly, depending on which membership functions are active and which percentage, it can generate different behaviors.

The control signal behavior, presented in the figure 4.6 is not as aggressive as the PID control signal when the set point changes, which slows down the fuzzy controller at the start. When both control signals reach its peak value, the fuzzy control signal decreases until the set point while the PID control signal decreases to the actual speed control signal value and raises form there to the set point.

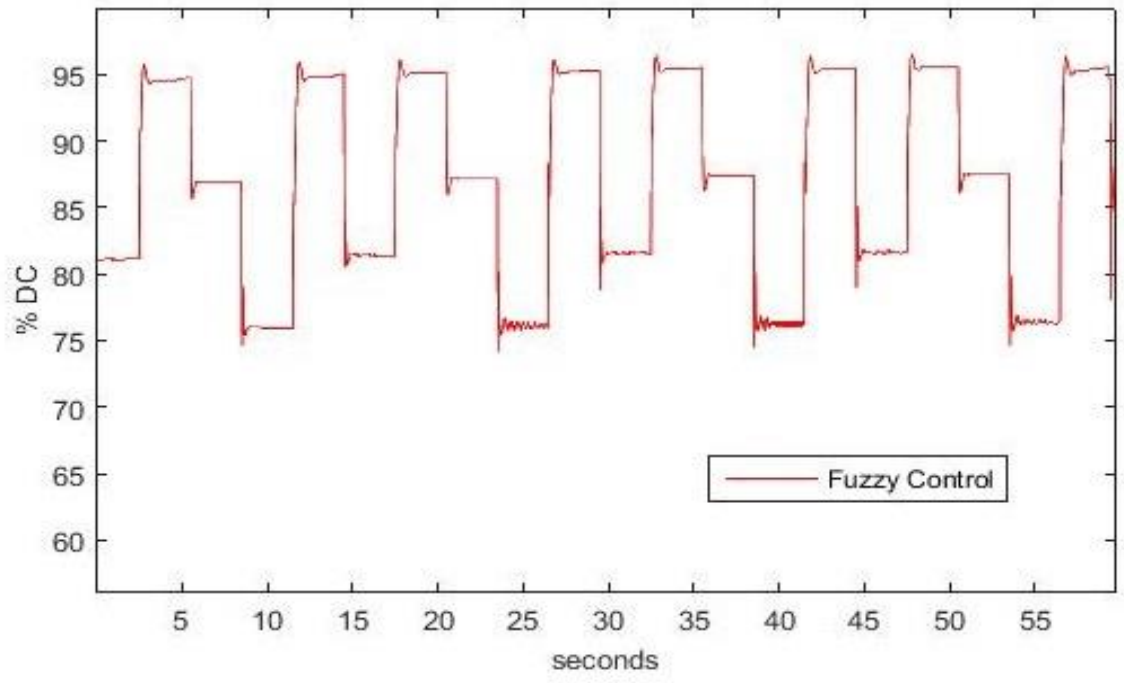


Figure 4.6: Fuzzy Control Signal

To take more conclusions on both controllers behavior it is shown a closer look at the 2000rpm to 3000rpm set point variation.

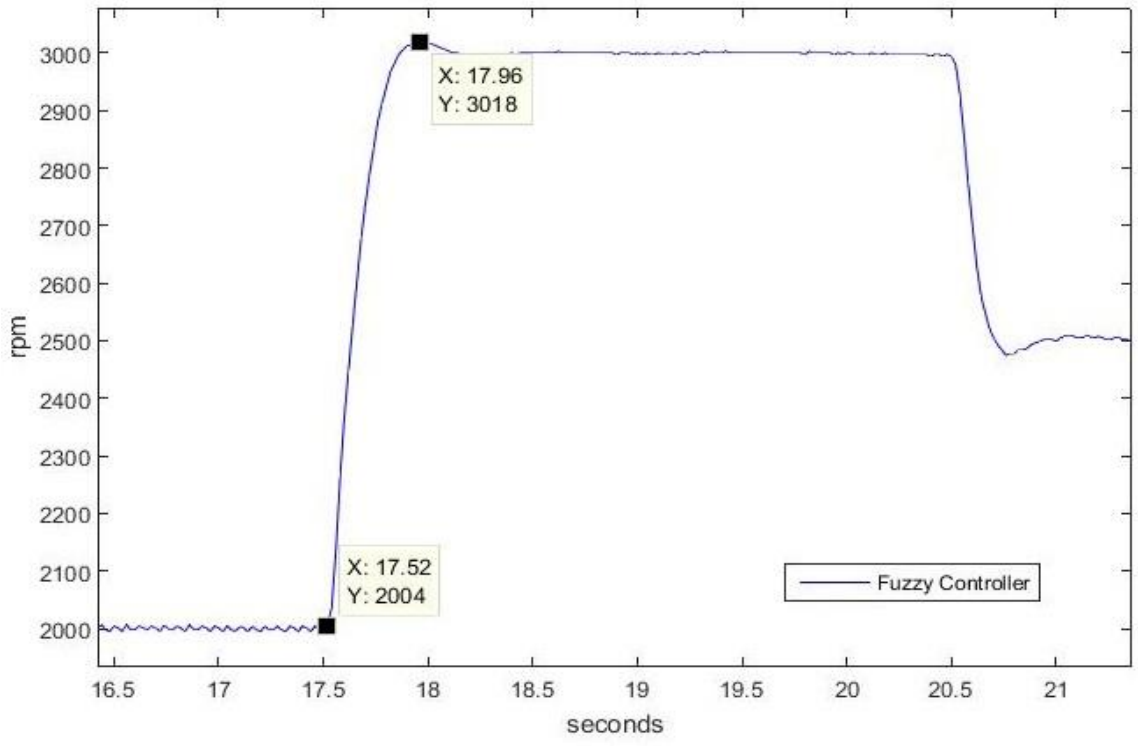


Figure 4.7: Fuzzy controller response to a set point variation

The figure 4.7 shows a closer look to the fuzzy controller behavior. Several important aspects should be addressed, the initial small oscillation at 2000rpm, which is a problem addressed in the figure 4.11, the less than 5% overshoot and the rise time.

Even though the output actually shows a small overshoot, it is barely noticeable as it is less than 3% of the final value. The rise time achieved was 0.34 seconds. This time represents 17 controller time steps to reach the final value. From the knowledge gathered for the motor, it was possible to change the motor speed from 2000rpm to 3000rpm in 0.14 seconds. Having in mind that the controller objective is to change set points in the least time possible without overshoot and with some degree of precision, the overall performance towards the maximum possible in terms of rise time was quite good. This value could be better if it was not the problem shown in the control signal for this same output, at the figure 4.8:

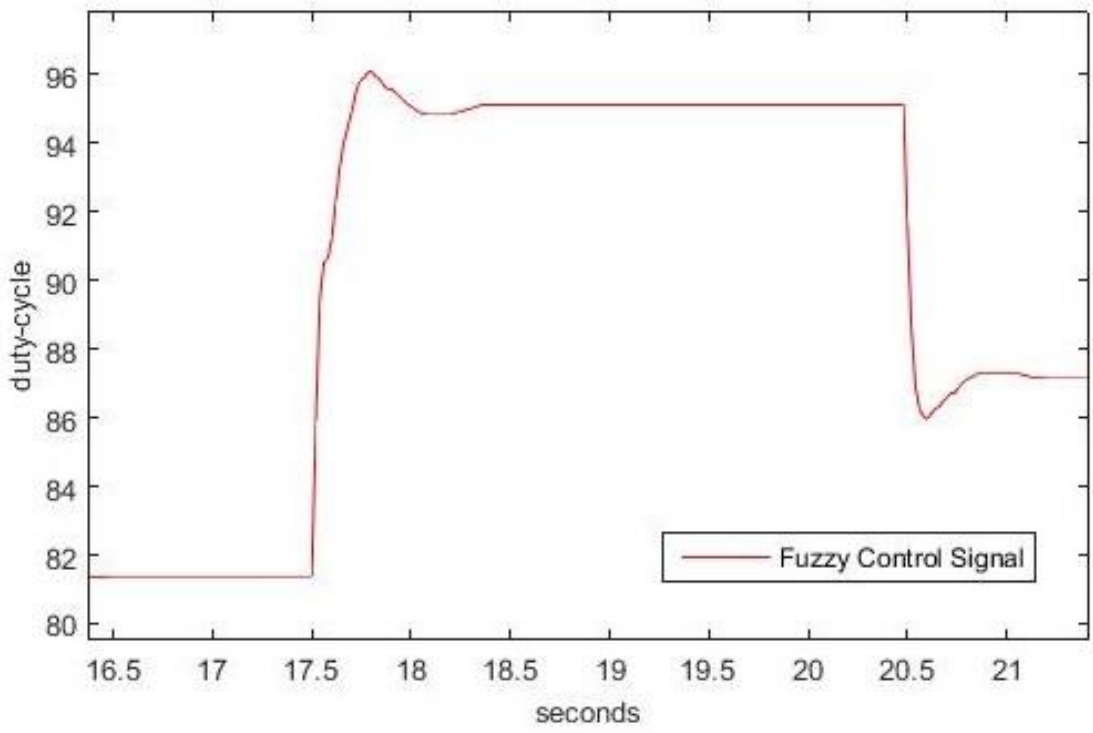


Figure 4.8: Fuzzy control signal

The fuzzy controller control signal was actually raising at a very high rate, but due to the fact the derivative component was so aggressive it held down the control signal loosing 3 control steps correcting that. It then resumes increasing the control signal at a slower rate since it was closer to the set point. The main reason this happens is because the fuzzy system was only using three membership functions for the change of error input, which represents the derivative gain. When using the active load setup this effect disappears, which reinforces this statement. Back in the chapter 3 it was explained how the

change of error membership functions were centered, to improve the system response in an active load setup and, due to that choice, decreasing the quality of the controller response on this setup.

The choice of using only three membership functions for the change of error limits the number of control options. If more membership functions were used, it would be possible to tune a less aggressive behavior when the error was large and the change of error was low – medium, and a more aggressive impact when the error was low and the change of error was high, which happens for faster control systems. Given that only three membership functions were used to the change of error, it was preferred to keep the no overshoot performance objective over the rise time and, as such, the aggressive behavior on the derivative gain.

The PID controller response for the same variation is how in the figure 4.9.

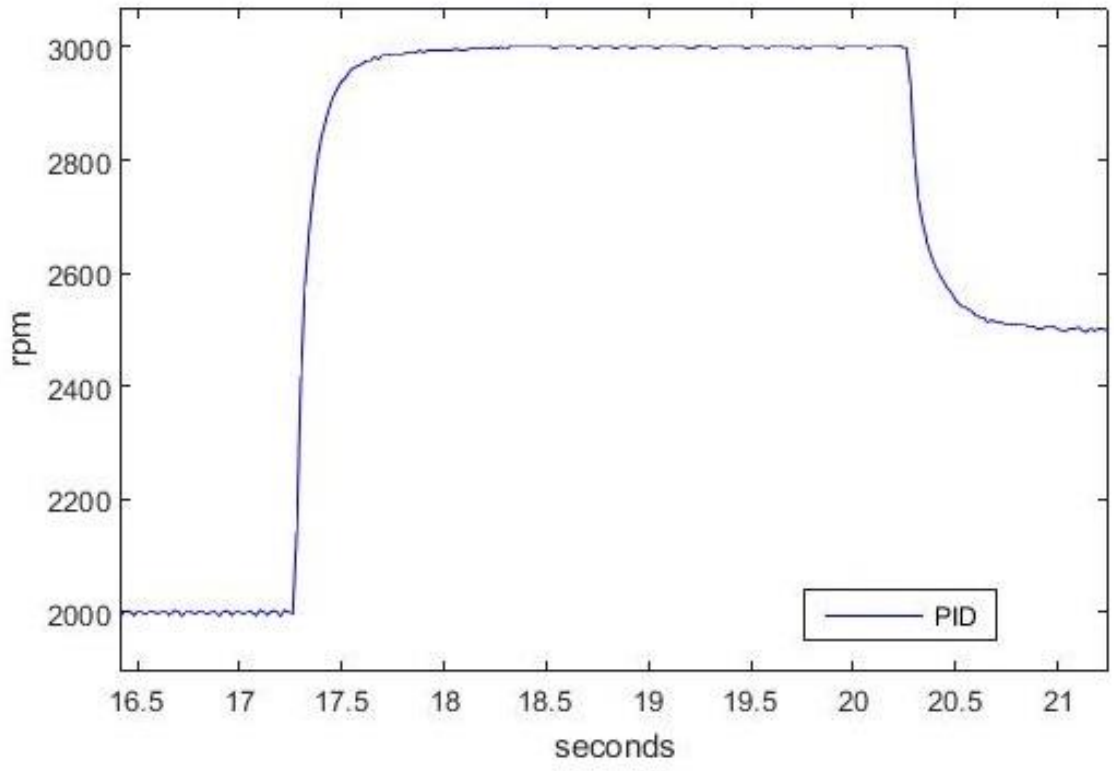


Figure 4.9: PID controller response to a set point variation

The PID controller shows no overshoot and the rise time is similar of the fuzzy controller. The more impactful difference is at the initial behavior towards the set point change. The PID controller changes 750 rpm in 0.12s while the fuzzy controller takes 0.18s. This represents a difference of 3 controller steps between the controllers, which confirms the previous analysis on the fuzzy control signal. The PID controller then slows significantly,

taking a very cautious approach to the set point, completely avoiding any overshoot. The fuzzy controller is then slightly faster on the last 250rpm.

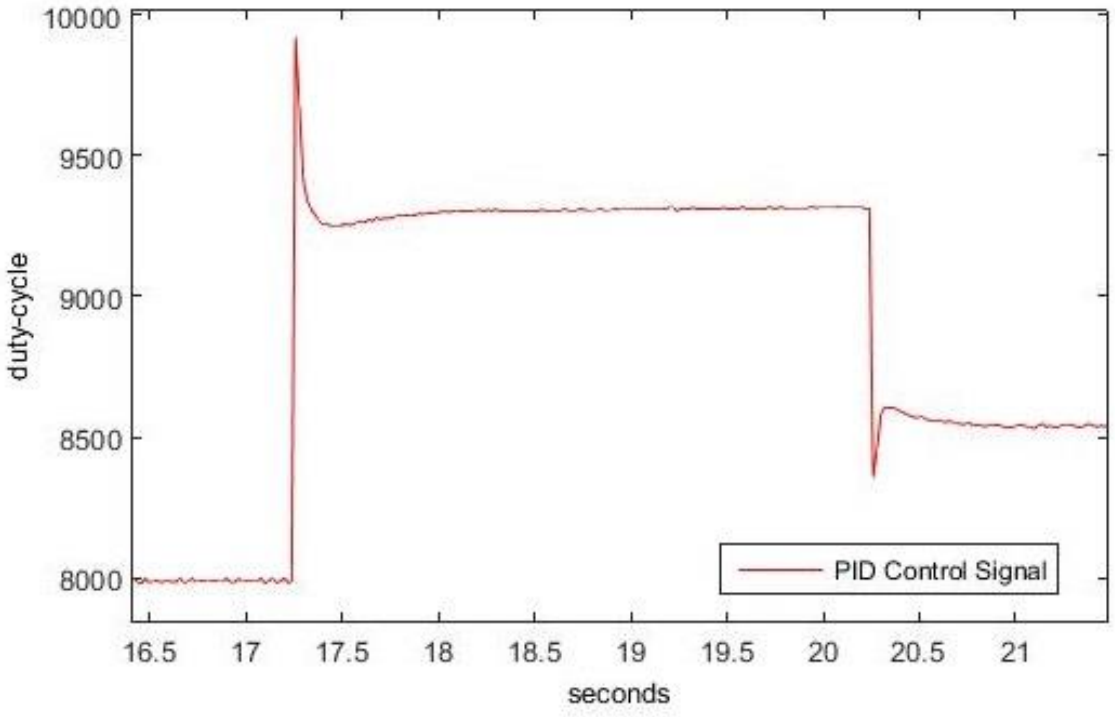


Figure 4.10: PID control signal

The figure 4.10 shows the PID control signal behavior. The initial response to the set point variation uses up to 98% of the control signal which is almost optimal. As expected from the careful approach towards the set point shown in the figure 4.9, the control signal changes slowly near the set point.

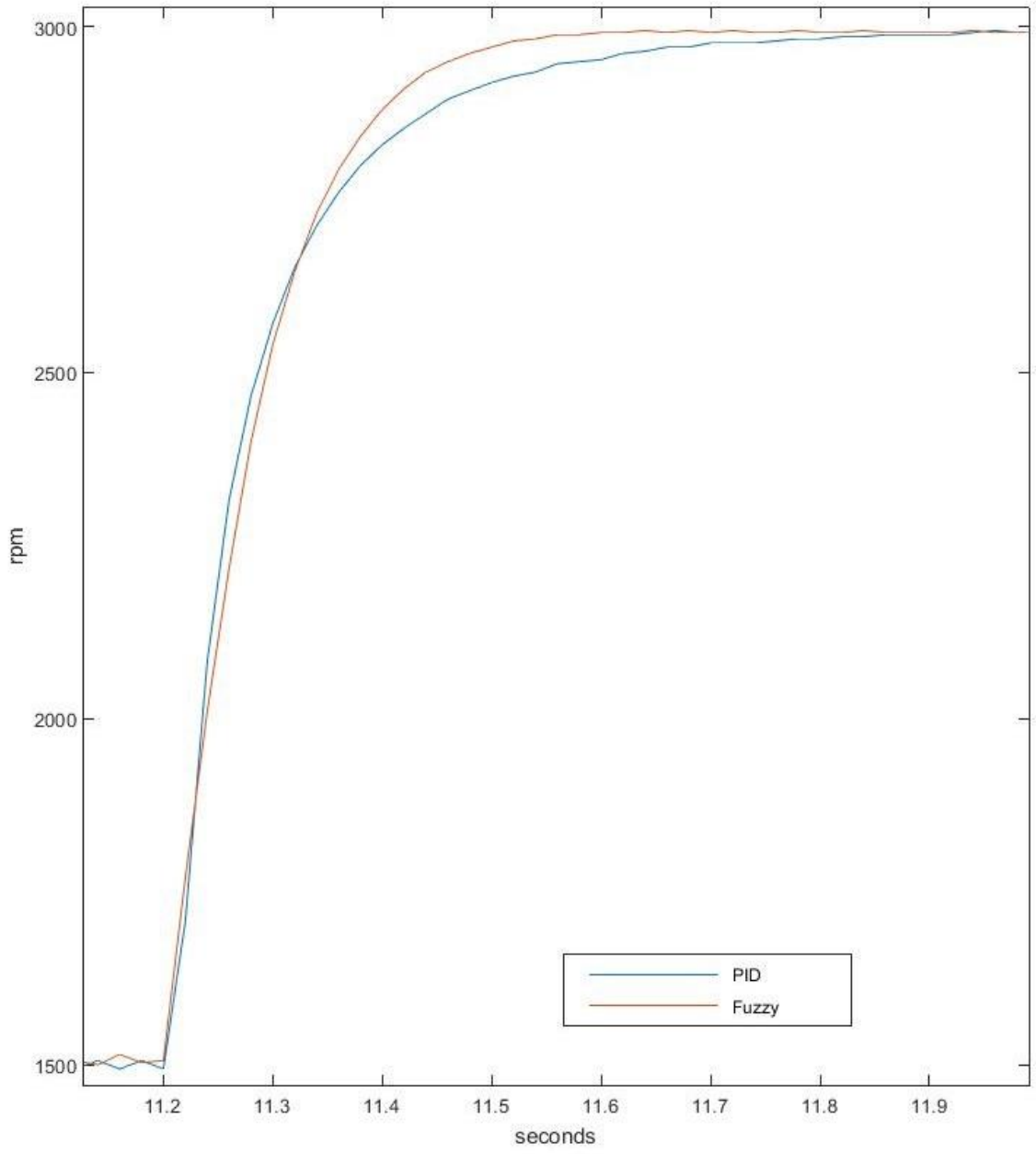


Figure 4.11: PID versus Fuzzy

The figure 4.11 represents the comparison of the PID and Fuzzy controllers.

The figure 4.12 shows the effects of the imprecision caused by the minimum increment of the controller.

With the minimum increment to be held at 3 rpm there is a random problem. When the control gains that add up to the final membership function do not make it possible to reach the actual correspondence of the control gain – controller speed at the desired set

point, the controller will oscillate around the minimal incremental value around the set point but never reaching it.

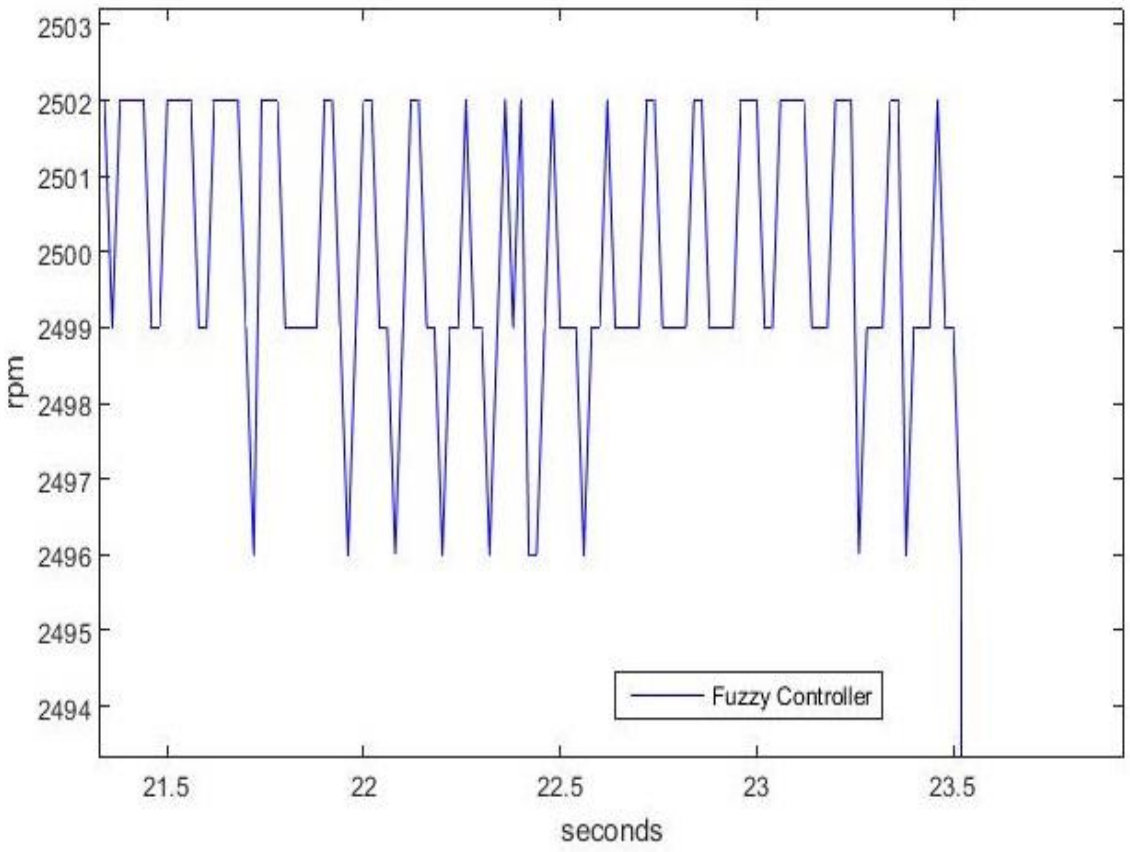


Figure 4.12: Controller imprecision at the set point

4.3.2 PID and Fuzzy Controllers with active load applied

To check both controllers quality it was used a setup with an active load applied to the motor working as generator. It is the same setup as the figure 4.2, with the following changes on the generator (figure 4.13):

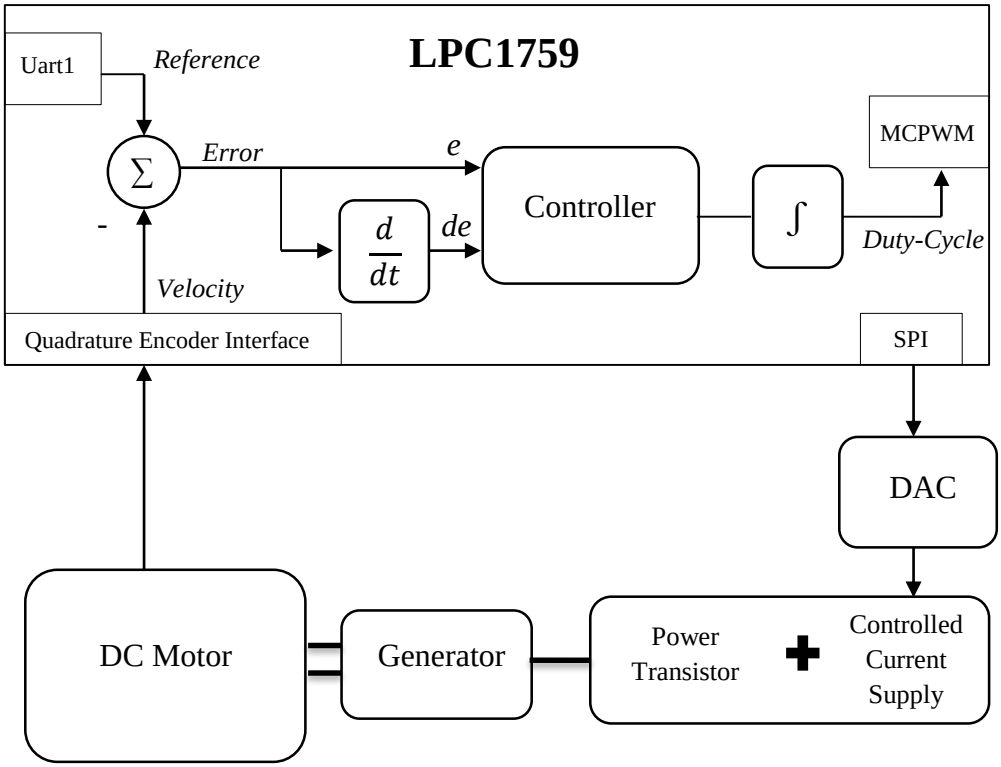


Figure 4.13: Active load setup

The current was set using a reference voltage, which was controlled by the microcontroller using an external DAC. The current supply had a factor of 1:1, so that the current in amperes passing through the power transistor is equal to the reference voltage from the DAC module.

The electric schematic of the controlled current supply is shown in the figure 4.14 and the figure 4.15 shows the power transistor with the respective dissipater.

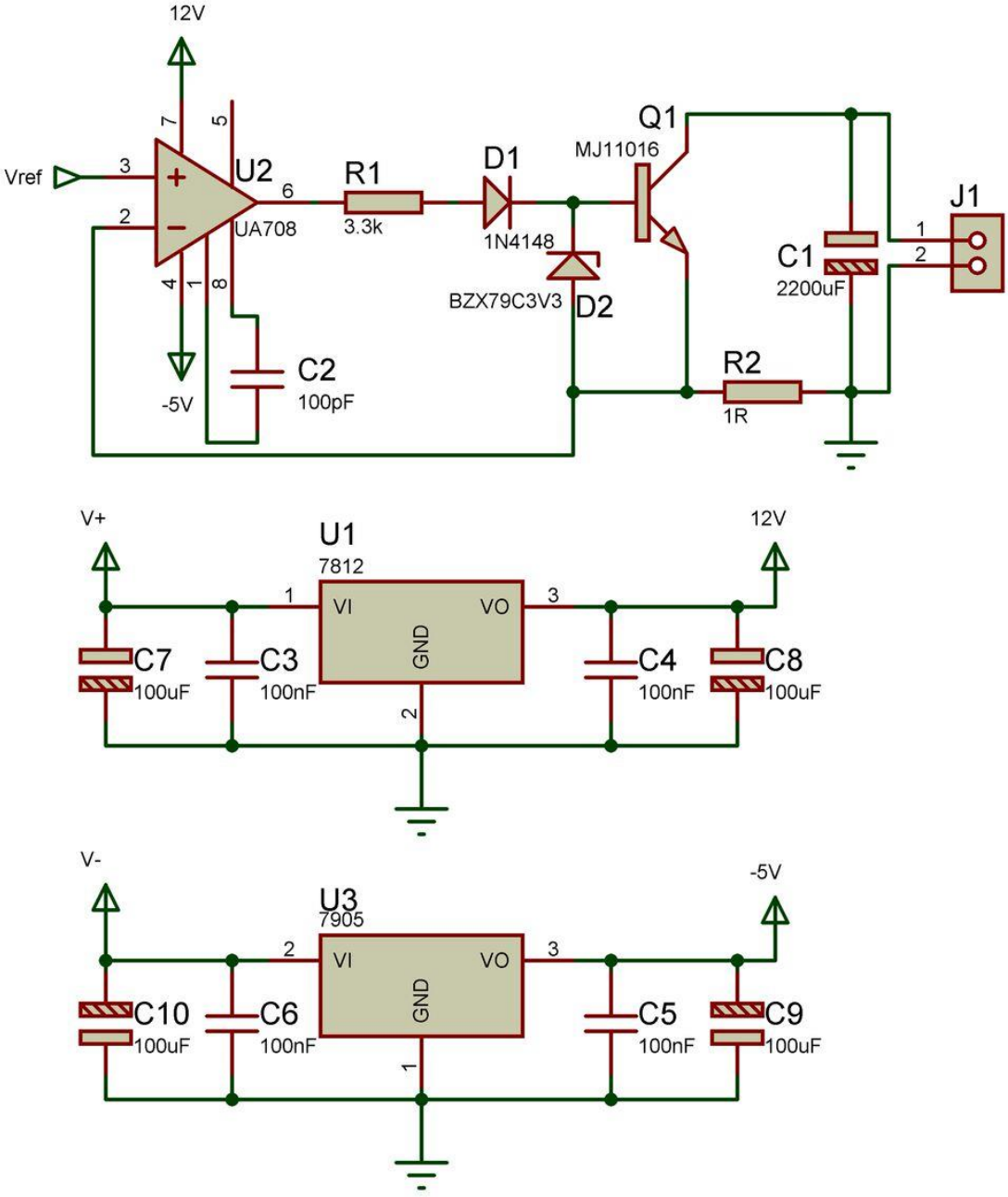


Figure 4.14: Electric schematic of the controlled current supply

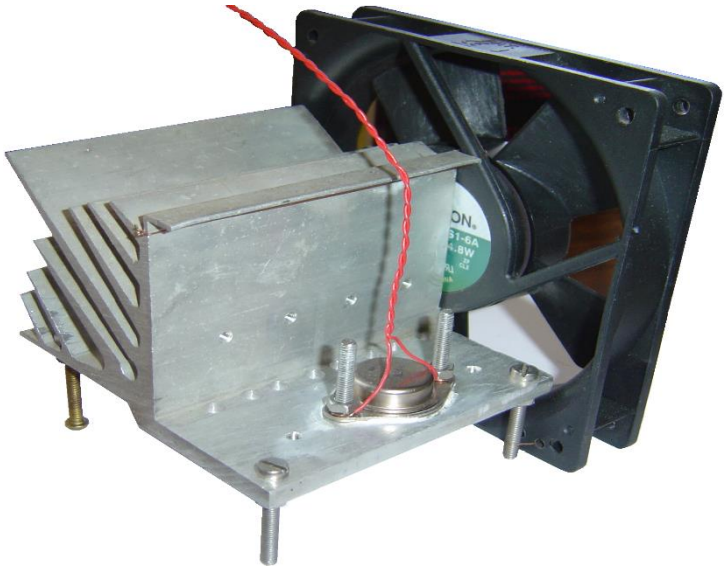


Figure 4.15: Power transistor

To check the impact and the operating range of the motor speed with the load applied it was first tested with a constant current applied of 0.5A (current that the fixed current supply pulls from the generator).

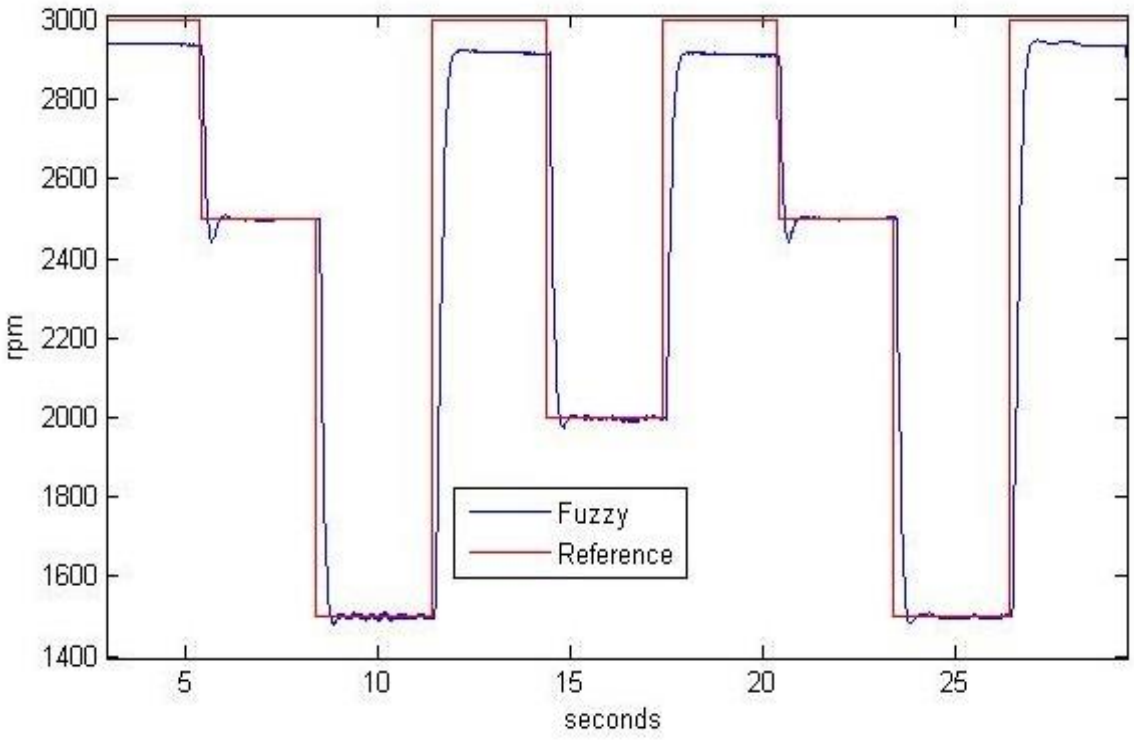


Figure 4.16: Fuzzy controller response

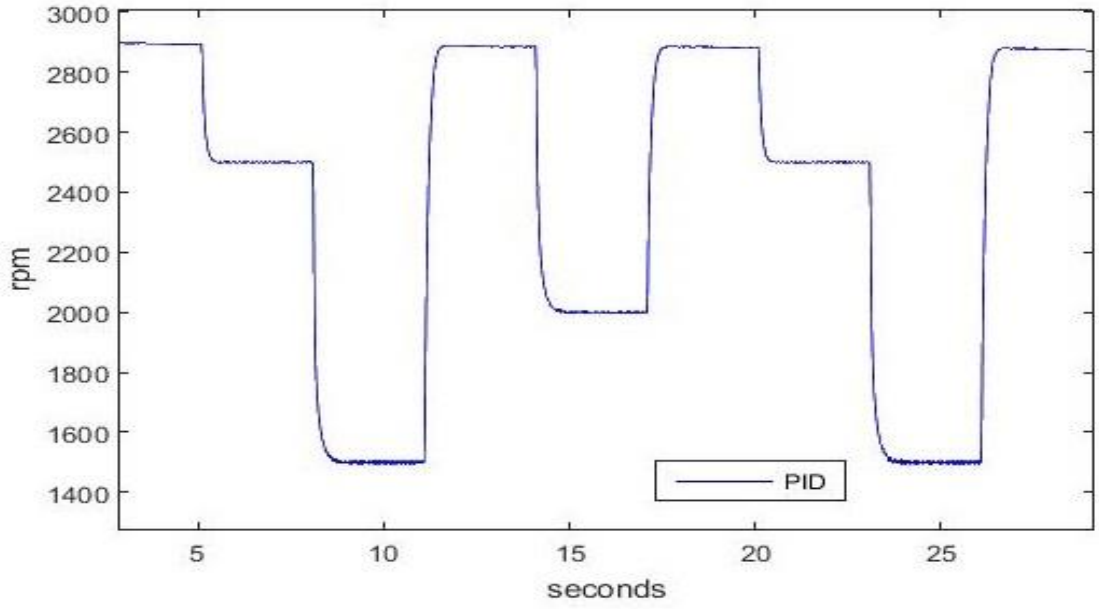


Figure 4.17: PID controller response

The figure 4.16 and 4.17 shows both controllers results in controlling the motor speed (reference signal) with a constant load of 0.5A. Both controllers cannot reach the 3000rpm, since the control signal is already at 100%, as shown at the figures 4.18 and 4.19.

Both controllers show a similar response as without static load (figures 4.3 and 4.5), with the PID controller being slightly slower. The control signals shown at the figures 4.18 and 4.19 help understand this behaviour.

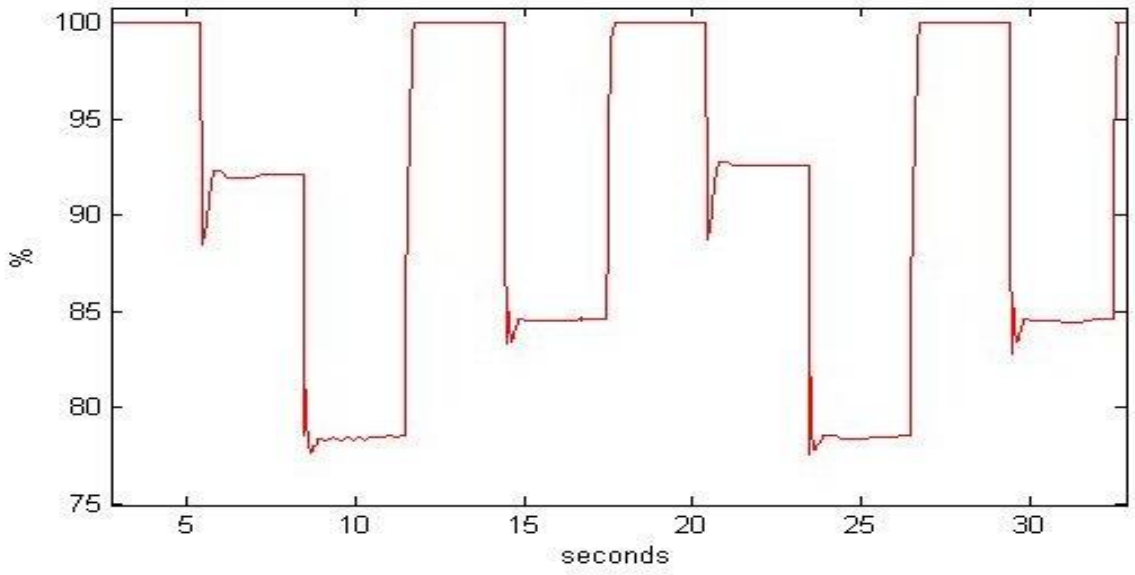


Figure 4.18: Fuzzy control signal

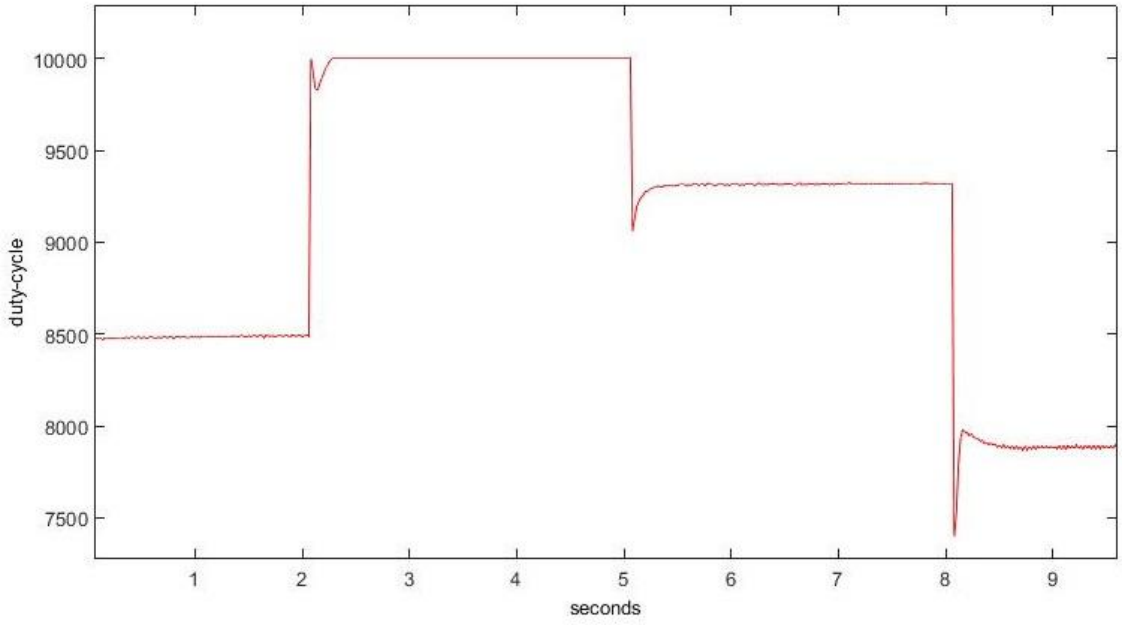


Figure 4.19: PID control signal

The control signals are much higher when compared with the precious setup and the PID controller struggles more with that. The PID controller had a response towards the set point change that used 98% of the control signal even though it would fall back a few iterations after. Even so, the PID controller when dropping the control signal would have the error diminish, being closer to the set point. Now, since the control signals are higher, that early gain is lower.

The figure 4.20 shows the comparison of the PID and Fuzzy controllers for the static load setup in the same graphic. The small gap in the top of the graphic between both controllers is due to a small offset variation on the external DAC that indirectly controlled the current that was being pulled from the generator.

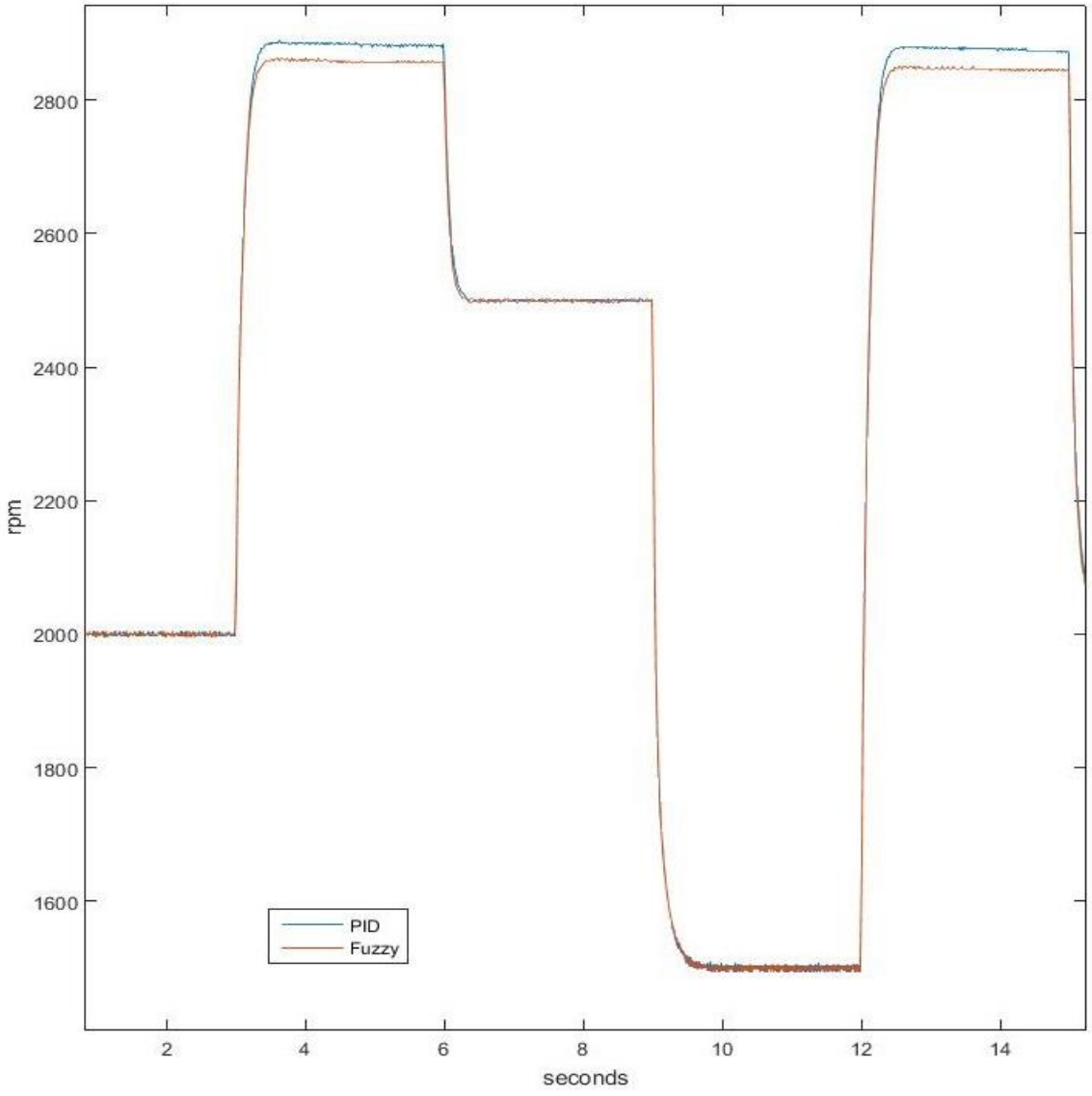


Figure 4.20: PID versus Fuzzy (static load)

The response of the controllers to the motor active load variations is shown in the figures 4.21 and 4.23 and their respective control signals in 4.22 and 4.24.

The set point was fixed at 2000rpm. During 10 seconds it was applied 3 different loads and cancelled before the next load being applied with 2 seconds interval. The first load is around 250mA, the second 500mA and the last 1A, like the next table shows:

Time(s)	0	0.5	2.5	4.5	6.5	8.5	10.5
Current(A)	0	0.25	0	0.5	0	1	0

Table 4.1: Active load variation profile

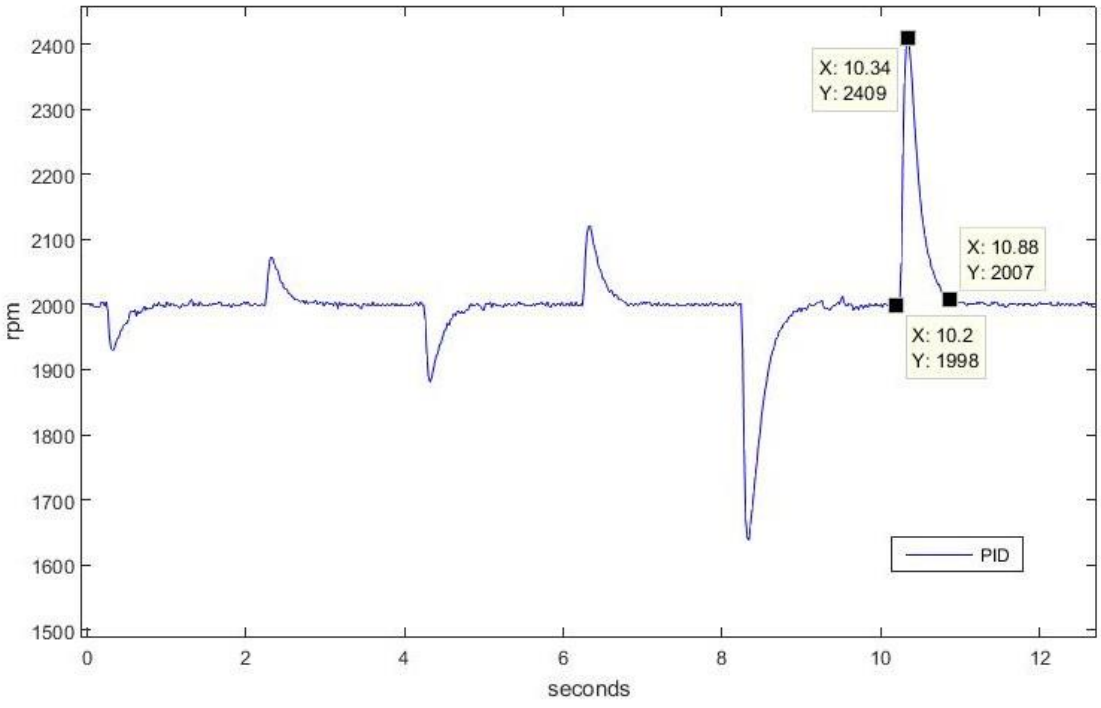


Figure 4.21: PID Controller response for active load variations

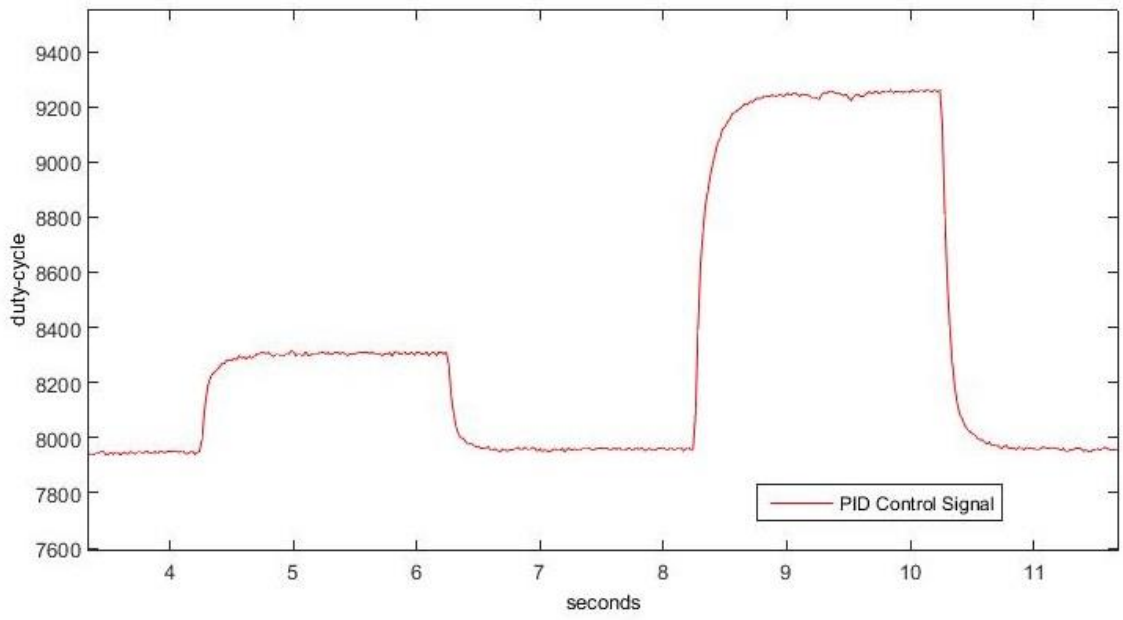


Figure 4.22: PID control signal for active load variations

The PID controller response for the active load variations showed in the figure 4.19 reflects a decrease of the controller quality in this setup. By analysing the control signal at the figure 4.20, it is possible to notice how the control signal increases slowly. This effect make the PID response slower, taking nearly 0.6s to recover from the 1A variation.

The fuzzy controller response if shown in the figure 4.21. It is possible to see that the fuzzy controller keeps a close behaviour to the previous setups. By analysing the control signal shown in the figure 4.22, it is possible to see the similarities with the figure 4.8 but without the control signal setback. This time, the fuzzy control signal went up to 98% and then decreased until reaching the set point value.

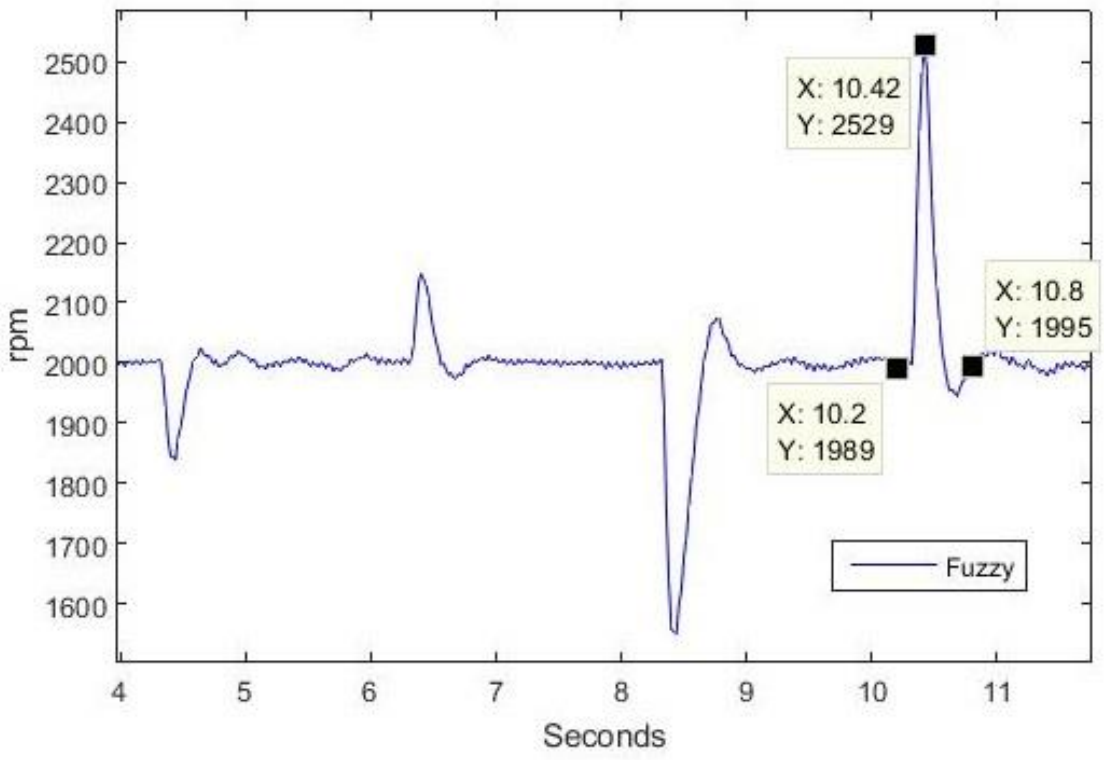


Figure 4.23: Fuzzy Controller for Active load variations

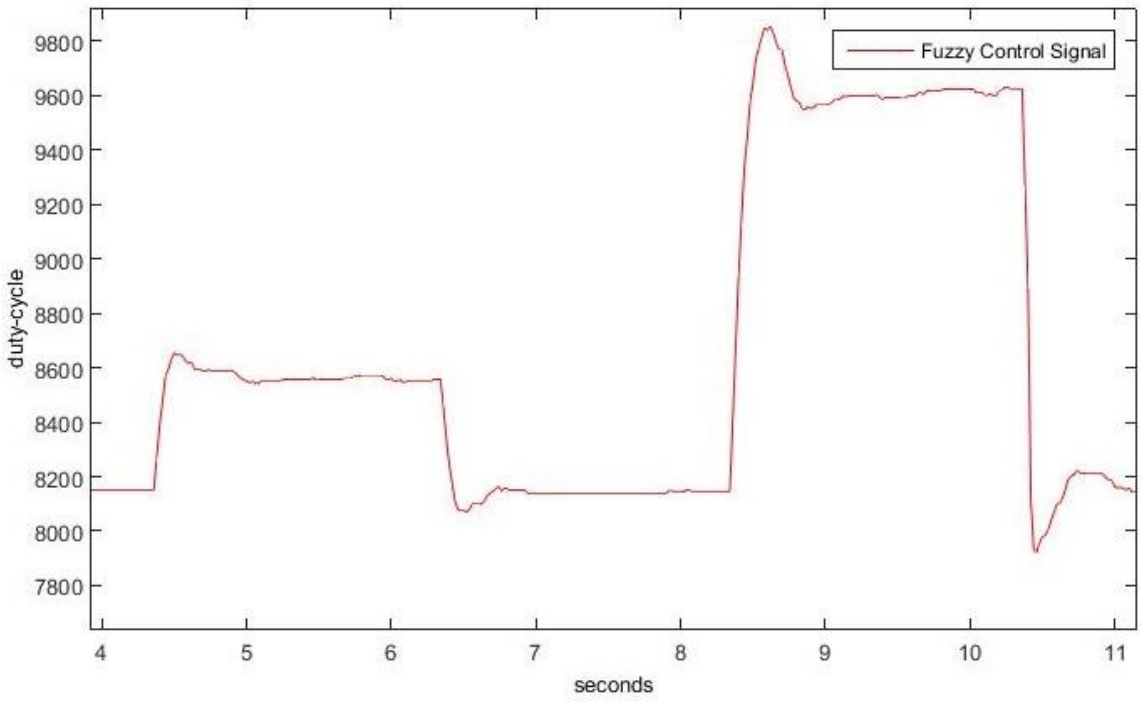


Figure 4.24: Fuzzy control signal for active load variations

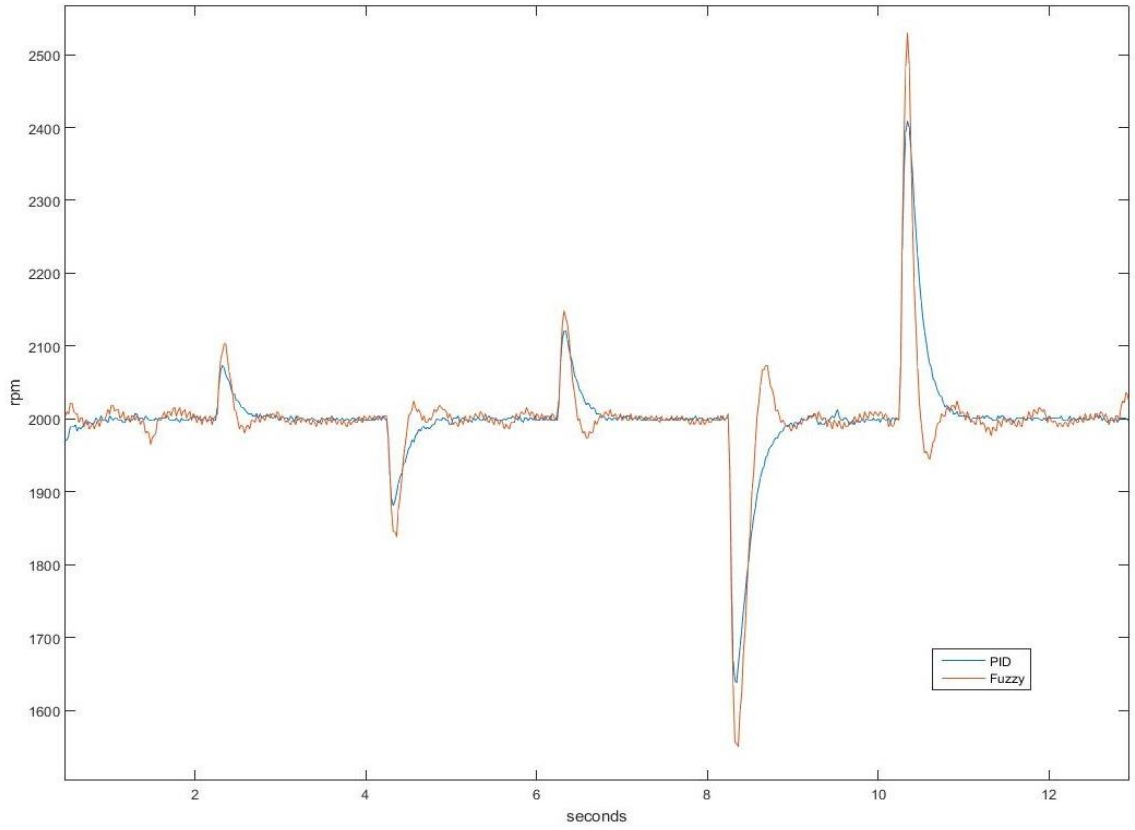


Figure 4.25: PID versus Fuzzy (active load)

The figure 4.25 shows the comparison of the PID and Fuzzy controllers on the active load setup.

4.3.3 Controllers performance

To measure the controllers performance it was used the metric IAE, ISE, ITAE and MSE. The first results on both controllers use a setup without load applied to the generator, with a set point variation of 750rpm (table 4.3). The measurements for the static load used 500mA and a set point variation of 750 rpm (table 4.4). For the active load variation is was used a single measurement with 1A variation at 2250rpm (table 4.5).

In addition to the two controllers explained it was also used one extra fuzzy controller with a different configuration.

The additional fuzzy controller used triangular membership functions in a configuration with 5 membership functions for each input. The rule-set table is presented in the table 4.3. The fuzzification and defuzzification methods were the same used on the previous fuzzy controller setup. The gains presented in the rule set table are the duty-cycle, whereas 1000 = 1% duty-cycle.

The new fuzzy controller will be denoted as fuzzy2 and it will be displayed the graphics for the fuzzy2 response on the tests used to calculate the metric IAE, ISE, ITAE and MSE

e / *de*	Negative high	Negative small	Zero	Positive small	Positive high
Negative high	-200000	-180000	-140000	-50000	0
Negative small	-150000	-140000	-75000	0	50000
Zero	-100000	-75000	0	75000	100000
Positive small	-50000	0	75000	140000	150000
Positive high	0	50000	140000	180000	200000

Table 4.2: Rule set table fuzzy2

The fuzzy2 response towards the first metric setup is shown in the figure 4.26 and its respective control signal in the figure 4.27.

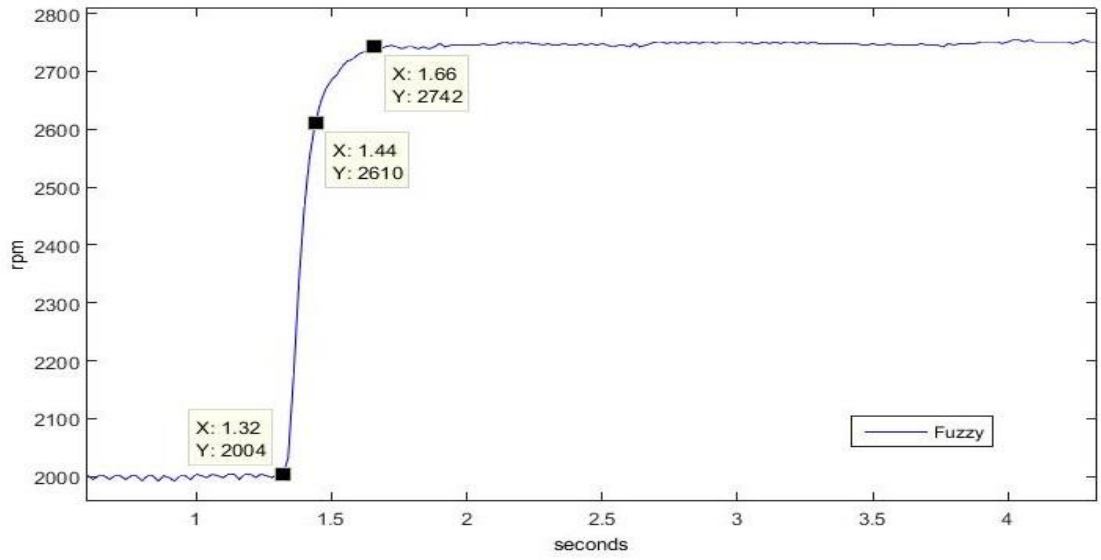


Figure 4.26: Fuzzy2 response for the first setup

The fuzzy2 controller had a faster response than the first fuzzy setup, majorly because of the addition of two more membership functions in the change of error input. Even with the error input passing from 11 membership functions to 5, it was not as relevant. This is explained because most of the error membership functions were in an error input range that was almost never used. The number of active membership functions on the error input almost did not change with this. It is also now possible to see the control signal following two different behaviours, as each control signal figure show (figures 4.27, 4.29 and 4.31.

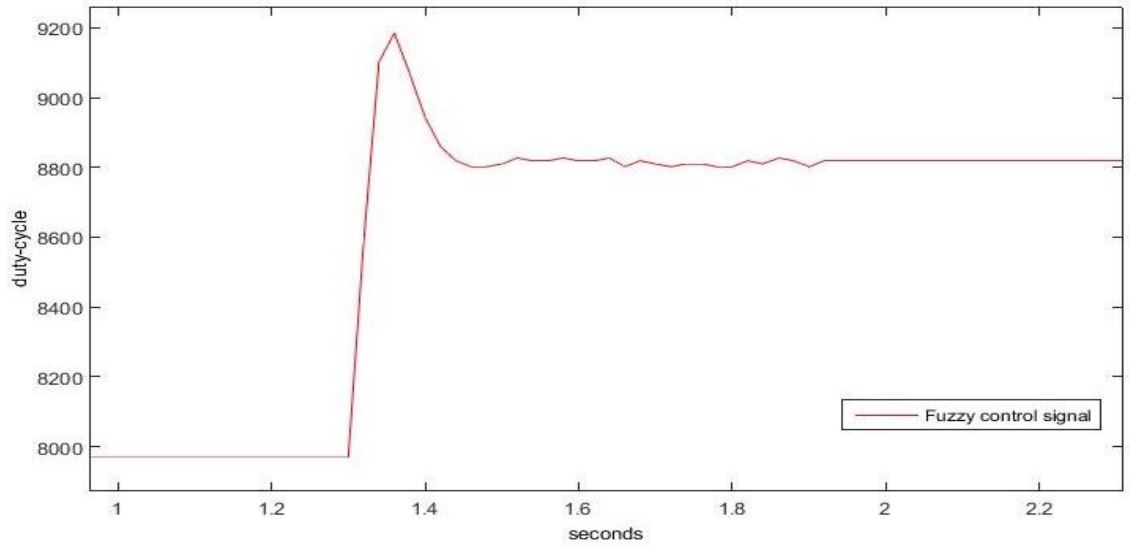


Figure 4.27: Fuzzy2 control signal for the first setup

The figure 4.28 shows a similar behaviour with the figure 4.26 as the control signals are mostly equal.

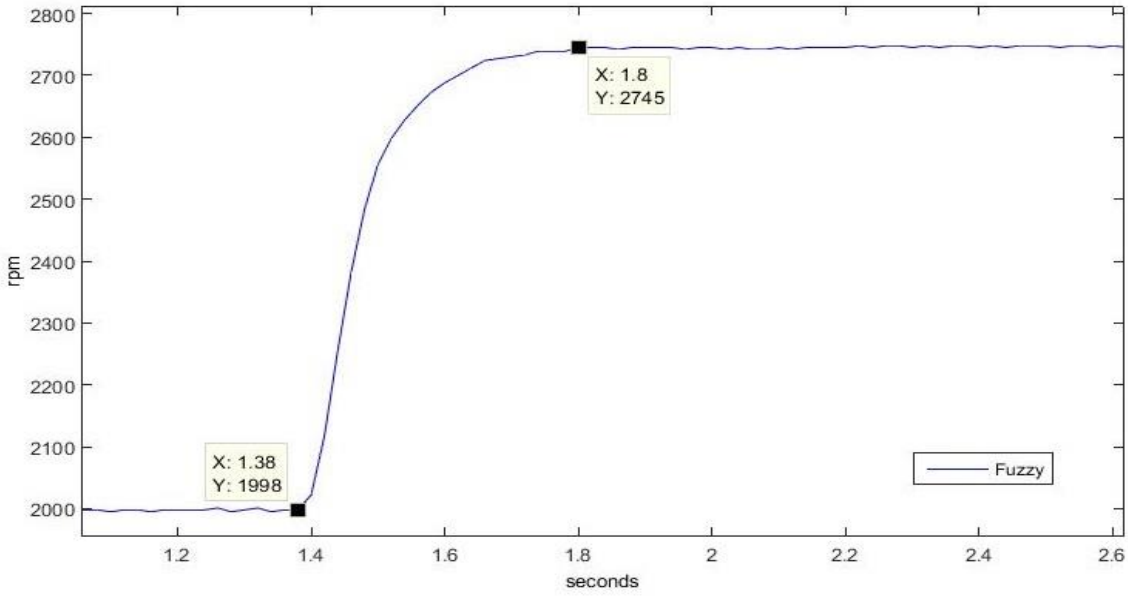


Figure 4.28: Fuzzy2 response for the second setup

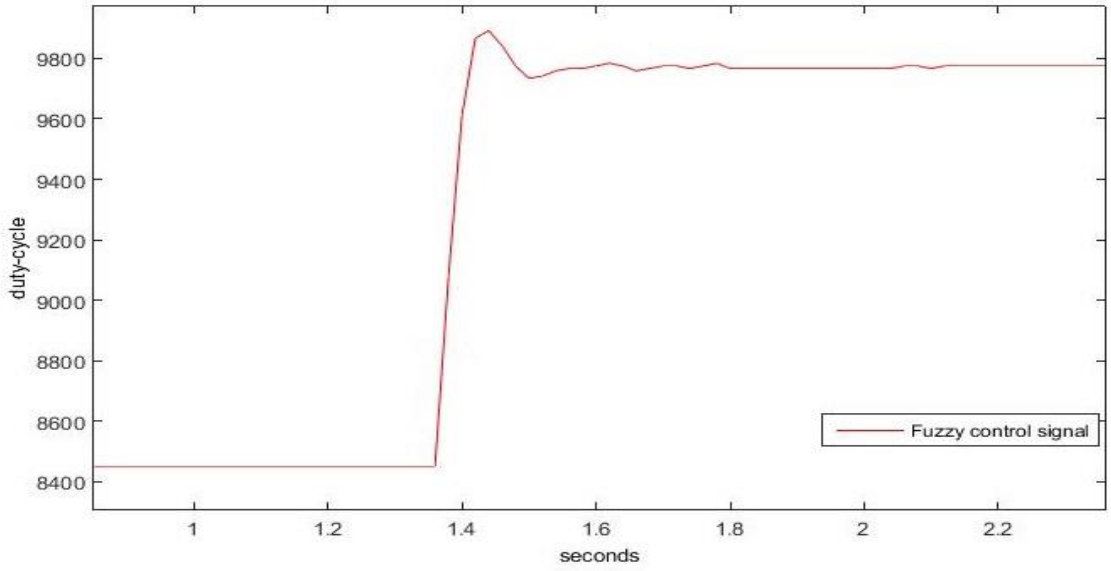


Figure 4.29: Fuzzy2 control signal for the second setup

For the last setup, the fuzzy2 response is shown in the figure 4.30.

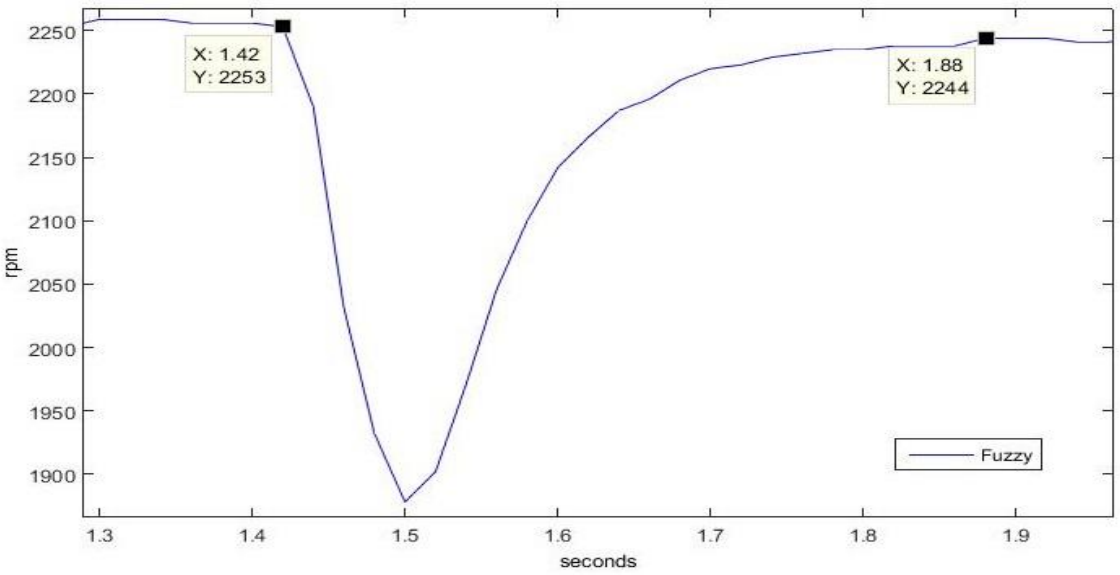


Figure 4.30: Fuzzy2 response for the third setup

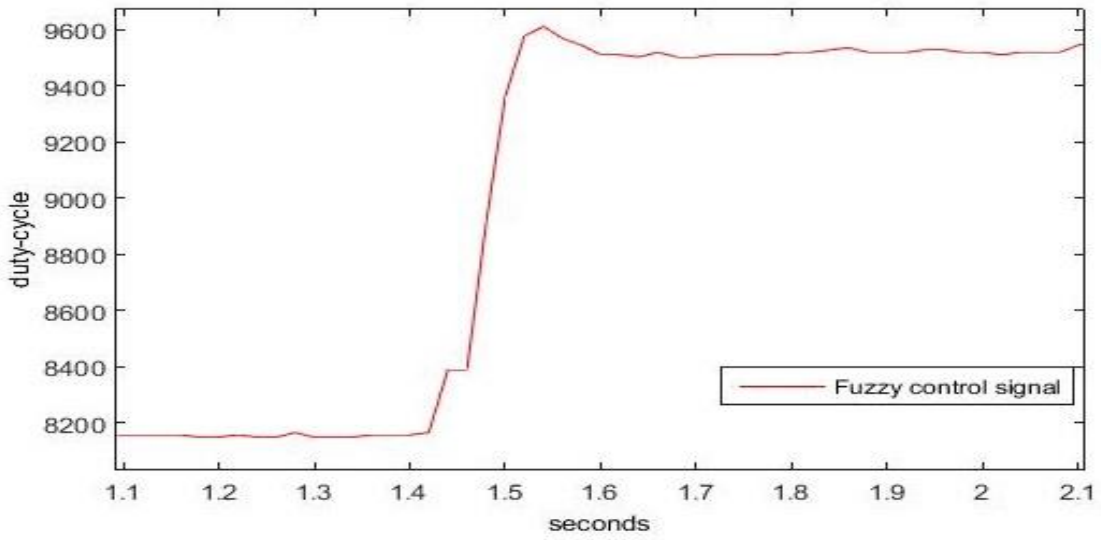


Figure 4.31: Fuzzy2 control signal for the third setup

The results on all controllers to the metrics is shown on the tables bellow.

The first setup results:

	ITAE	IAE	ISE	MSE
Fuzzy	571	3843	1658251	58025
PID	476	3258	1425766	33957
Fuzzy2	677	3208	1192054	30236

Table 4.3: Performance for PID and Fuzzy controllers without load

The second setup results:

	ITAE	IAE	ISE	MSE
Fuzzy	1089	4938	2091448	97588
PID	793	4843	1848247	83821
Fuzzy2	830	3844	1482982	47787

Table 4.4: Performance for PID and Fuzzy controllers with Static load

The third setup results:

	ITAE	IAE	ISE	MSE
Fuzzy	1129	4719	1568853	167080
PID	1323	5226	1252296	198400
Fuzzy2	1448	3228	590634	65343

Table 4.5: Performance for PID and Fuzzy controllers to active load variations

Based on this results the PID controller showed better results than the first fuzzy controller except in the last setup, where the fuzzy controller showed better results except in the ISE. The second fuzzy controller meanwhile showed better results in every setup, with the exception of the ITAE index. This means the second fuzzy controller had more small errors over time since the MSE index was normally very good, that reflects a very good response towards large errors and the initial response.

5 Conclusion and Future Work

5.1 Conclusion

The work developed granted a very insightful experience in fuzzy control, digital control and embedded programming. It was possible to realize that fuzzy controllers offer a wide array of opportunities in digital control to achieve very good performances objectives. In comparison with the general PID controllers, the fuzzy systems can achieve better results and still offer more control options. The simplicity of PID controllers stands as a relevant fact in this comparison but even a simple fuzzy system can have very good performance results. The more complex fuzzy controllers can achieve and outpace the performance of the PID controllers, given the versatility that the fuzzy systems have that the PID systems does not. A very important fact is that every digital fuzzy controller can be upgraded, without changing the hardware. The amount of tuning process that can be applied in the fuzzy systems is immense, which is a very good remark to acknowledge when choosing between the two types of controllers. This options and the fact that the fuzzy controllers can easily adapt to nonlinear systems, since fuzzy controllers can have nonlinear change of certainty in the membership functions shape or have a nonlinear rule set table, makes the fuzzy controller a choice to be acknowledge.

The selected microcontroller – NXP LPC1759, which stands in a new type of microcontrollers that specific offer better solutions for embedded controllers, proved to be a valuable option. Both the Quadrature Encoder Interface and the Motor Control PWM modules offered a very good option to simplify the hardware and still grant very good results. The tests on the QEI velocity reading versus the tachometer attached to the motor only had a slight difference of 1% maximum. The MCPWM could be slightly better, as if it is required a frequency higher than 20000Hz, the resolution for the PWM constant gets really low, which can become a problem.

Using FreeRTOS also proved to be an excellent choice. In digital control is very important to keep a perfect notion on every action, when each segment of code happens and, most of all, to be able to control that. FreeRTOS tools offered this possibility with great results and also being simple to use and to control.

The workflow was full of step backs with practical problems that took a long time to overcome. The path towards a functional controller involved a lot of choices that mostly were an agreement between something good and something bad. Some examples of this is the choice between fixed point and floating point or the number and shape of membership functions, This process was of extreme importance, as it made possible to

acknowledge at least some problems and solutions methods that might be important in the years ahead.

5.2 Future Work

Following the conclusions of this work, the next step should be to evaluate the advantages of fuzzy systems over other controllers. It would be interesting to build an adaptive fuzzy controller. As it was discussed previously, it is very hard for the designer to know which membership functions or rule-sets to pick to achieve a certain level of performance. By giving the controller the ability to adapt these and automatic tune to achieve certain levels of performance it is a great improvement overall. In many cases, even if the performance is achieved, there could be always some new factors that could change the system in a way that a non-adaptive controller ca not answer. For this type of project it would be recommended to use C++ to interpret the membership functions as objects to have move options and better processing times.

There are various improvements that can be made to the fuzzy controller interface as well. The interface has its basics abilities but it is far from achieving great versatility. The most important factor could be adding the ability to directly make or change the rule set table form the interface. Adding the ability to make different number of membership functions and more shapes and even selecting the number of inputs should be possible. It would be recommended to use a more powerful processor for this case and being able to work with floating point would vastly increase the controller performance.

Bibliography

1. **M.Passino, Kevin and Yurkovich, Stephen.** Fuzzy Control System Design. *Fuzzy Control.* s.l. : Prentice-Hall, 1998.

2. **Ferreira, Luís Filipe Terra.** *Controlo Adaptativo de um motor DC.* Aveiro : Universidade Averio, 2010.

3. **NXP.** LPC17xx User Manual. s.l. : NXP, 2010.

4. **Real Time Engineers Ltd.** Quality RTOS & Embedded Software. [Online] Real Time Engineers Ltd, 2010-2013. http://www.freertos.org/.

6. **Zhang, Huaguang, Liu, Derong.** *Fuzzy Modeling and Fuzzy Control.* Chicago : Birkhäuser, 2006. ISBN-10 0-8176-4491-1.

7. **Tanaka, Kazuo.** *An Introduction to Fuzzy Logic for Practical Applications.* New York : Springer, 1996. 0387948074, 9780387948072.

8. **PARVEX.** DC SERVOMOTORS. FRANCE : PARVEX SA, 2003.

9. **Dr. Feng Xia, Prof. Youxian Sun.** *Control and Sheduling Codesign.* Hangzhou : Springer, 2008. ISBN 978-7-308-05765-3.

10. *Comparison of Tuning Methods of PID Controllers for FOPTD System.* **K. Mohamed Hussain, R. Allwyn Rajendran Zepherin, M. Shantha Kuma.** 3, Trichy : INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING, 2014, Vol. II. ISSN (Online) 23 21 – 2004 / ISSN (Print) 2321 – 5526.

11. **Kung, Ying-Shieh, Huang, Chung-Chun and Huang, Liang-Chiao.** *FPGA-Based Motion Control IC for Linear Motor Drive X-Y Table Using Adaptive Fuzzy Control.* [Article] 2012. ISBN 978-953-51-0759-0.