Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

João Miguel Malva Freitas

Aprendizagem Evolutiva de Controladores Difusos

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores.

Coimbra

Maio de 2018

Universidade de Coimbra

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

Aprendizagem Evolutiva de Controladores Difusos

por

João Miguel Malva Freitas

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores, ramo de especialização em Automação.

Orientador: Prof. Doutor Rui Alexandre de Matos Araújo
Co-Orientador: Prof. Doutor Jérôme Amaro Pires Mendes

Júri:
Presidente: Prof. Doutor Manuel Marques Crisóstomo
Vogais: Prof. Doutor Rui Pedro Duarte Cortesão
Prof. Doutor Rui Alexandre de Matos Araújo

Maio de 2018

Para todos os que me ensinaram algo na vida.

Agradecimentos

Ao Professor Doutor Rui Araújo pelo desafio e oportunidade de realizar a minha tese de mestrado no Laboratório de Controlo Inteligente e Robótica (LCIR) do Instituto de Sistemas e Robótica (ISR).

Ao Doutor Jérôme Mendes pelos ensinamentos e por me ajudar a ultrapassar os obstáculos que este trabalho proporcionou.

Ao Núcleo de Estudantes de Engenharia Electrotécnica e de Computadores, por ter sido uma escola e por me ter dada a oportunidade de fazer mais pela comunidade do Departamento de Engenharia Electrotécnica e de Computadores.

Ao BEST Coimbra, por me ter ensinado que é na diversidade de pessoas, cursos e culturas que aprendemos mais. Nunca esquecerei as aventuras e as amizades que tive a sorte de fazer.

Aos Ledzener e companhia, por serem companheiros na vida académica desde o início, pelas aventuras e jantaradas eternas.

Aos meus colegas e amigos, por me fazerem uma pessoa melhor e feliz, uma aventura académica sem vocês teria sido uma aventura sem nenhum sabor.

À minha família, por ser o eterno porto de abrigo, por acreditarem em mim e, acima de tudo, pelos valores e educação que me deram e que tanto orgulho tenho em possuir.

À minha Sara, por tornares a minha vida muito mais emocionante.

A todos, obrigado!

Resumo

Controladores inteligentes são hoje um importante aspecto no controlo de processos industriais e, particularmente, o controlador de lógica difusa com capacidades de aprendizagem é um caso de estudo muito interessante, devido ao seu formato e características únicas.

Com o intuito de controlar sistemas com múltiplas entradas e uma saída foram estudados algoritmos de controlo difuso com uma componente adaptativa, por outras palavras, com a capacidade de adaptar a regras e parâmetros existentes no controlador e com uma componente evolutiva, por outras palavras, com a capacidade de modificar a estrutura do controlador com a adição de novas regras, obtidas através do uso de informação da saída e entradas do sistema. Adicionalmente, o controlador deve ser capaz de alterar a sua estrutura ao mesmo tempo que controla o sistema, sem necessidade de treino prévio, e também controlar sistemas desconhecidos sem conhecimento do modelo e dinâmica do sistema. Após algumas pesquisas foi escolhido um algoritmo com as características mencionadas que serviu de base para o algoritmo apresentado nesta dissertação.

Neste trabalho são apresentados os fundamentos de Controladores Difusos, a arquitectura e funcionamento do algoritmo proposto, sendo mencionado as melhorias às falhas detectadas do algoritmo original que foi estudado. A importância e influência de vários parâmetros do algoritmo proposto são também analisados em detalhe.

De forma a validar e demonstrar a capacidade do algoritmo proposto, foi testado e analisado o seu desempenho no controlo de diversos sistemas simulados com múltiplas entradas e uma saída, assim como num sistema real composto por dois motores DC acoplados. Em todos os sistemas testados foram induzidas perturbações, tendo sido analisada a resposta do algoritmo proposto.

Palavras-chave: Controlo Difuso, Aprendizagem Online, Controlo Difuso Adaptativo, Controlo Difuso Evolutivo, Inteligência Computacional.

Abstract

Nowadays, intelligent controllers are an important aspect in the control of industrial processes and the particular Fuzzy Logic Controller with learning capabilities are a specially interesting subject of study, due to its format and characteristics.

In order to control systems with multiple inputs and one output it was studied fuzzy control algorithms with an adaptive component, in other words, with the capacity to adapt the existing controller rules and parameters and with an evolving component, in other words, with the capacity to modify the controller structure with the addition of new rules, using the historical data about the controlled system. Furthermore, the control system must be able to change its structure at the same time is controlling the system, don't need to do offline training and also be able to control unknown systems without previous knowledge of the model and dynamics of the systems. After some research, an algorithm with the mentioned characteristics was chosen and served as the basis for the algorithm proposed in this dissertation.

In this work are presented the concepts of Fuzzy Controllers, the architecture and structure of the proposed algorithm, being mentioned the improvements to the detected faults of the original algorithm that was studied. The importance and influence of several parameters of the proposed algorithm are also analysed in detail.

In order to validate and demonstrate the capacity of the proposed algorithm, it was tested and analysed its performance in the control of several simulated systems with multiple inputs and one output, as well as in a real non-linear system based on two-coupled DC motors. All tested system were also subjected to perturbations, being analysed the response of the proposed algorithm.

Keywords: Fuzzy Control, Online Learning, Adaptive Fuzzy Control, Evolving Fuzzy Control, Computational Intelligence.

Abbreviations and Symbols

Acronyms List

MISO	*Multiple Inputs Single Output*
FL	*Fuzzy Logic*
BL	*Boolean Logic*
FRB	*Fuzzy Rule-Based*
T-S	*Takagi-Sugeno*
ANYA	*Angelov and Yager Method*
MF	*Membership Function*
FLC	*Fuzzy Logic Controller*
SPARC	*Self-Evolving Parameter-Free fuzzy Rule-Based Controller*
OSEFC	*Online self-evolving Fuzzy Controller*
MIMO	*Multiple Inputs Multiple Output*
PL	*Parameter Learning*
SE	*Self-Evolving*
FCS	*Fuzzy Control System*
FWHM	*Full width at half maximum*
FIE	*Fuzzy Inference Engine*
IFO	*Inferred Fuzzy Output*
FIE	*Fuzzy Inference Engine*
IFO	*Inferred Fuzzy Output*
MSE	*Mean Square Error*
CSTR	*Continuous-Stirred Tank Reactor*

Symbols List

R_j	j-th Fuzzy Rule
x_i	value of the i-th variable
A_i^j	antecedent of the j-th for the i-th variable
u_j	output of the j-th rule
Q_j	consequent of the j-th rule
μ_A	degree of membership of the antecedent A
M	size of stored memory
u_m	stored data of the command signal
x_m	stored data of the input variables
k	iteration
$u(k)$	command signal in iteration k
$\phi(k)$	Fuzzy parameters in iteration k
$\alpha_i(k)$	conjugated strength of all activation degrees of the i-th rule in iteration k
N_r	Number of rules
N_v	Number of variables
$G(x(k), \phi(k))$	Fuzzy logic controller in iteration k
$\Delta Q_i(k)$	Variation of the i-th consequent in iteration k
$signM$	signal of monotonicity
$e(k)$	tracking error $(r(k-1) - y(k))$
Δu	variation of the command signal
Δr	variation of the reference signal
G	gain parameter
$threshod1$	threshold use to prevent the creation of rules
G_{aux}^j	Auxiliary FLC for the j-th variable
IR_j	Index of Responsibility
n^*	number of auxiliary MFs
X_j^s	s-th interval of the j-th input
ϕ_s	proposed new center
$index$	index of the proposed new center

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The necessity of a better efficiency and autonomy in industrial process management requires control algorithms that are robust, autonomous and with a considerable performance when confronted with uncertainties and perturbations. The autonomy is evaluated by the capacity of self-government, in other words, the adaptation of its parameters and evolution of its structure, without external intervention. The robustness is related with the stability of the controller when the controlled system is subject to perturbations, and the performance corresponds to the quickness in achieving, steadily, the referenced values, even when subjected to perturbations.

Adaptive fuzzy control algorithms uses Fuzzy Logic (FL), a multi-value logic that uses the concept of partial truth. Unlike the Boolean Logic (BL), the truth value is a real number between 0 and 1 and describes the degree of truth of a statement, making the FL excel at representing human reasoning [Wang, 1999]. Another aspect is the use of linguistic variables, easily comprehensible to a human operator, to describe inputs and output values. Due to this characteristics, the FL control algorithms are an important and interesting field in the new industrial revolution, the so called industry 4.0, where one of the objectives is the existence of intelligent control algorithms which are perceived by human operators [Lasi *et al.*, 2014].

An important aspect of a controller is the capacity to control a system without knowledge about its model and dynamics, since one of the most exhaustive procedure on the control design is the study of a system dynamics and model that sometimes aren't accurate enough. Another important requirement of a control system is its capability of working online, in other words, the capability of changing its parameters at the same time it's controlling a system.

With this assumptions this dissertation presents, analyses and deconstructs a fuzzy evolving control algorithm with learning capabilities.

1.1 State of the Art

Lotfi A. Zadeh introduced the concepts of Fuzzy Logic (FL) and fuzzy sets in 1965 [Zadeh, 1965]. A fuzzy set is a set where values have degrees of membership, making it possible to elements belong to several sets with different degrees of membership. The degrees are multi-value, this contrast with the classical set theory that use a bivalent condition, in other words, an element either belongs or not to a set. Bellman and Zadeh continue the work using FL, specifically in the decision making [Bellman and Zadeh, 1970] and fuzzy ordering in [Zadeh, 1971].

In 1973, Zadeh published a paper that would become the foundation of fuzzy control, called "Outline of a new approach to the analysis of complex systems and decision processes" [Zadeh, 1973]. In this paper, the proposed fuzzy controller can be considered a rule-based controller because the command value of the controller is obtained by the combination of fuzzy rules. The format of the fuzzy rules presented in the same paper is composed by two parts: the antecedent and the consequent. The antecedent, based on the values of the inputs of a system, assigns strength to the different rules and the consequent contributes to the final value of the command signal. Thus a common used nomenclature for the fuzzy rules is the IF-THEN rules: IF a antecedent is verified THEN a consequent will contribute to the command signal.

To define the formats of the antecedents and consequents there are three different Fuzzy Rule-Based (FRB) methods: Mamdani, Takagi-Sugeno (T-S) and, more recently, the ANYA, proposed by Plamen Angelov and Ronald Yager.

The Mamdani was the first method introduced, in 1975, and the most easily associated to a conventional fuzzy controller [Mamdani and Assilian, 1975]. It use for both the antecedent and consequent the same type of format, the fuzzy set, a linguistic variable. The input variables will have a degree of membership for each fuzzy set, obtained through a Membership Functions (MFs) using operations describe in Chapter 2, to determine the strength of the output fuzzy sets. An example of a rule using this method to define the behaviour of an air conditioner system could be: IF Temperature is *Low* THEN ACmode = *Heating*, being *Low* and *Heating* two linguistic variables representing two fuzzy sets, Temperature an input variable and ACmode an output variable.

The Takagi Sugeno (T-S) is similar to Mamdani, using the same type of format in the antecedent, a fuzzy set. The difference is in the format of the consequent, instead of a fuzzy set it uses functions. An example of a rule using this method and, again, an air conditioner system could be: IF Temperature is *Low* THEN ACfanspeed = $a_1 * Temperature + a_0$, being a_1 and a_0 two coefficients, *Low* a fuzzy set, Temperature an input and ACfanspeed an output.

The ANYA uses a different concept for the antecedent, instead of using fuzzy sets uses data clouds that are created by the distribution of variables. This type of antecedent compare the local density between the input values and the cloud samples, with that value it's possible

to determine the degree of membership of the variable to the different clouds, this works like a MF for the fuzzy sets. For the consequent, ANYA can use fuzzy sets, like the Mamdani method, or Functions, like T-S method. An example of a rule using this method, using the same example, could be: IF Temperature $\in Cloud_1$ THEN ACmode = *Heating*, being $Cloud_1$ a cloud of variables of the input Temperature and *Heating* a possible fuzzy set for the output ACmode.

The Table 1.1 summarizes the three FRB methods.

Table 1.1: Comparison between the Fuzzy Rule-Based Methods

Fuzzy Rule-Based Method	**Antecedent format**	**Consequent format**
Mamdami	Fuzzy set	Fuzzy set
Takagi-Sugeno	Fuzzy set	Function
ANYA	Cloud	Fuzzy set or Function

The first test in a system happen in 1975, where Mamdani and Assilian applied a fuzzy controller [Mamdani and Assilian, 1975], since then, were made several studies that show the capacity of fuzzy control in complex non-linear processes. fuzzy controllers also gain support after their implementation in real world systems, specially by japanese, like washing machines, automatic transmission in cars and even in large systems like a subway train, in Japan [Wang, 1999]. These implementations and wide use of Fuzzy Logic Controlers (FLC) gave credibility to these types of control that, in the beginning were met with some scepticism by the scientific community.

Conventionally, the rules of a fuzzy controller are defined *a priori* using the knowledge of the dynamic of a plant or the human knowledge about the system, however, when the dynamics aren't known there is need of a more intelligent method version to design fuzzy controlers capable to adapt to several unknown systems. In the last decade were researched and presented several solutions of intelligent FLCs.

The intelligent FLCs can be divided in two main areas, the adaptive controllers, that basically don't add new rules, but instead adapt the existing ones, and the evolving controllers, that altered the structure of the controller, through the addition of new rules.

In [Mucientes and Casillas, 2007], [Mingzhi *et al.*, 2009], [Chen *et al.*, 2009], [Lin and Xu, 2006], [Li and Lee, 2003], [Wang *et al.*, 2008a] and [Hoffmann and Nelles, 2001] are presented adaptive algorithms that allow the users to choose the number of fuzzy sets for each input and use pre-training to adapt the fuzzy rules to be able to control systems, in other words, simulated the behaviour of the controller in training sets composed by examples that cover values of each variable. For this aspect, the algorithm is an off-line algorithm, i. e., can't control and adapt at the same time, needing to first adapt to possible values and only then can control efficiently a system.

In [Phan and Gale, 2008], [Park *et al.*, 2005] and [Gao and Er, 2003] are presented

algorithms that, apart the adaptive component, also have evolving component, in other words, they can change the structure of the FLC through the addition of new rules and adapt the existing rules. However this process is off-line, needing to pre-test the controller performance before controlling a system.

In [Wang *et al.*, 2008b], [Rojas *et al.*, 2006] and [Pomares *et al.*, 2002] there was an advance to the adaptative algorithms because they are online algorithms, i. e., do the adaptation of the existing controller's structure happens at the same time of the control for a system. This characteristic was an important improvement to the existing algorithms because the pre-training needed for the off-line algorithms some times didn't cover the unpredictable behaviour of non-linear systems. Is important to refer that the algorithms could adapt but still could change the structure of the controller with the addition of new rules, so they lack the evolving component.

Two early algorithms [Lin *et al.*, 1995] and [Angelov, 2004] proposed an online approach to adapt the FLC rules , as well the modification of the controller structure using information of every new sample of a system under control, however these algorithms are highly susceptible to noise and perturbations, creating a problem since corrupted data can originate "wrong" structures. They also need some learning time, and could only control the plants after that time.

There are two algorithms of adaptive and evolving fuzzy control: the Self-Evolving Parameter-Free fuzzy Rule-Based Controller (SPARC) [Sadeghi-Tehran *et al.*, 2012] and the Online Self-Evolving Fuzzy Controller (OSEFC) [Cara *et al.*, 2010]. Both can works without any previous knowledge about the plant, can start work without any rule and do online adaptation and evolution, that means it can change the structure of the control when controlling the plant. The two algorithms differ in the Free Rule-Based (FRB) method used, the SPARC uses the ANYA and the OSEFC uses T-S, due the considerable different format of the antecedents the algorithms differ in the addition of new rules, the adaptation process is similar.

In summary, the SPARC, like the OSEFC, uses the tracking error to adapt the consequent value, however since it uses the ANYA FRB method the addition of new rules is completely different, in each iteration is verified if the new values are close to the existing clouds, if a large number of values are distant enough of the existing clouds and close enough of each other is creating a new cloud and hence new rules.

The OSEFC has the same adaptation component as the SPARC, as was mentioned before. However, the evolving component is different and, in summary, uses the memory of the results of the plant to detect the best positions for the MFs of the new fuzzy sets.

Since the ANYA isn't yet widely used in the scientific community, the OSEFC is the chosen adaptive and evolving FLC algorithm to implement, however the final version of the algorithm present and used in this dissertation has some improvements to the problems detected in the OSEFC.

1.2 Motivation

The main motivation for this dissertation was its connection to industry 4.0. The industry 4.0 is a term created in 2011 to describe the 4th industrial revolution, it presents numerous goals, but the one related to this dissertation is the goal to increase productivity and efficiency of the workers using automation and machine learning techniques to complement their work. The connection of the dissertation to this goal is due to two facts:

- The first is the use of a Fuzzy Logic Controller (FLC), a controller which uses logic similar to human logic and use fuzzy sets that are linguistic values, easily interpreted by operators;

- The second is the fact that the algorithm create rules to assure an efficient response to changes in the reference signal or perturbations. Doing an analogy to a conventional FLC, which the MFs are obtained using human knowledge and trial error strategy, the MFs are determinated, in the algorithm, through mathematical methods that detect which values are causing more errors in the controlling process, this assures that the chosen MFs will have, probably, a better impact than the MFs chosen using human knowledge.

Another motivation is the importance and numerous real life applications of control systems, they are almost everywhere and deepening knowledge in this fascinating area of the automation branch is a great opportunity for a future engineer. The final aspect is the fact that area of intelligent control is a growing field of interested in research.

1.3 Objetives and Developed Work

The main objective of this dissertation is control Multiple Inputs Single Output (MISO) systems with adaptive and online FLC algorithms with learning capabilities without previous knowledge of the plant. For that purpose, it was study the state-of-the-art of Self-evolving fuzzy control algorithms and identified the most advanced.

To analysed the adaptation and scalability of the algorithm were researched plants with different characteristics, being the only assumption the existence of only one output. The researched and chosen plants were also subject to perturbations with the purpose of evaluate the algorithm robustness.

With the goal of analysed the importance of algorithm parameters was evaluated their influence in algorithm performance. The algorithm has two main features, the change of existing rules, known as Parameter Learning (PL), and the addition of rules, known as Self-Evolving (SE). It was realized tests with only the PL enabled, and tests with the both features enabled.

The results are discussed and is done a detailed analysis of the overall performance of the proposed algorithm, there also suggested improvements to the detected weaknesses of the original algorithm, the OSEFC.

1.4 Dissertation Structure

The dissertation is organized in five chapter, where in:

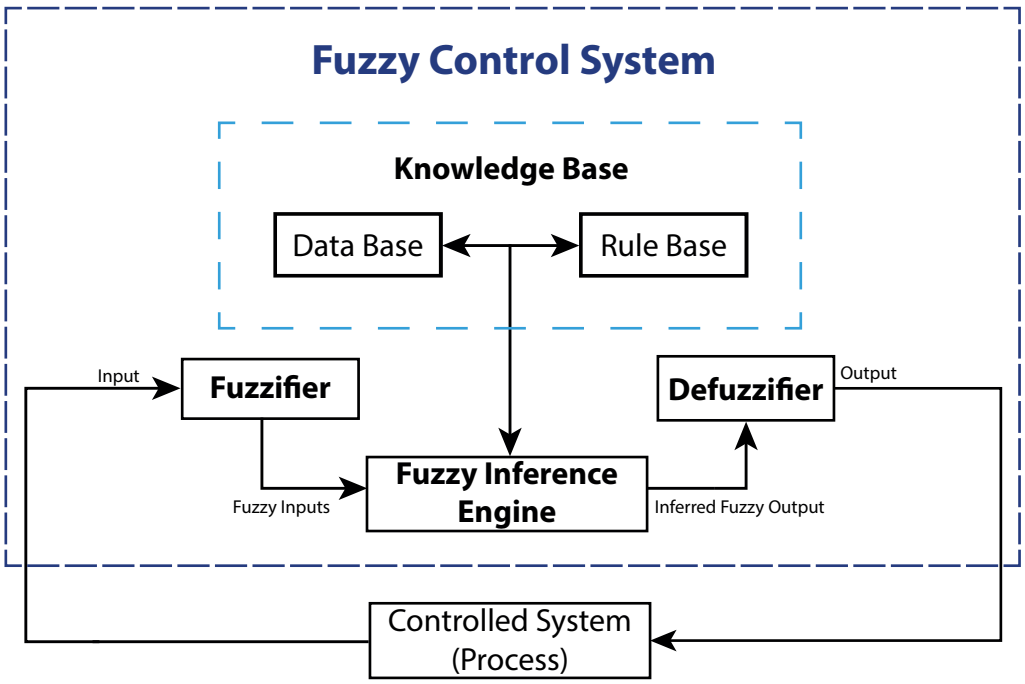
- Chapter 1 - introduces the dissertation main subject, motivation, state of the art and objectives;
- Chapter 2 - presents the main concepts about Fuzzy Control Systems;
- Chapter 3 - analyses in detail the proposed Algorithm;
- Chapter 4 - presents the plants where the algorithms were tested, the respective results and analyses the influence of the algorithm parameters;
- Chapter 5 - presents conclusions and future work.

Chapter 2

Fuzzy Control Systems

A Fuzzy Control System (FCS) can be divided in four phases: fuzzifier, knowledge base, fuzzy interference engine) and the defuzzifier, which they are analysed in more detail in the next sections. They are connected as Figure 2.1 shows.

Figure 2.1: Structure of a Fuzzy Control System.



The FCS is an excellent alternative to conventional control systems because it doesn't need a mathematical model of a plant and can be equally applied to linear or non-linear systems, which some conventional control systems can have performance issues [Ying, 2000].

2.1 Knowledge Base

"Fuzzy systems are Knowledge-based or rule-based systems. (...) A fuzzy system is constructed from a collection of fuzzy IF-Then Rules"[Zadeh, 1965]. As mentioned in the previ-

ous chapter, these fuzzy rules are divided in two parts: the antecedent and the consequent. The idea is IF a condition, described in the antecedent part, is verified THEN a command signal will be generated. So the rules of a fuzzy system can be represented in the form:

$$R_j : \text{IF } x_1 \text{ is } A_1^j \text{ and } ... \text{ and } x_n \text{ is } A_n^j \text{ THEN } u_j = Q_j \tag{2.1}$$

where $j \in [1, ..., N_r]$, N_r is the number of fuzzy rules, $x_i \in [x_1, ..., x_n]$ being x_i the i-th input variable of a total number of n input variables, $A_i^j \in [A_1^j, ..., A_n^j]$ are the fuzzy sets for each input variable x_i, and u_j and Q_j are, respectively, the output and consequent of the j-th rule that can have an different format depending of the chosen Fuzzy Rule-Based (FRB) method, as mentioned in section 1.1.

The applications of fuzzy control system in consumer products by japanese engineers was one the most vital aspects for the increasing use and interest of fuzzy control systems [Wang, 1999]. One of the first products was the control of a air conditioner, that could be represented by a Multi-input Multi-output (MIMO) system with four inputs and four outputs [Sobhy and Khedr, 2015]. Using a simplified format of this system as example, a fuzzy rule of this FCS using a Mamdani FRB method could be:

$$R_j : \text{IF } UserTemperature \text{ is } Low \text{ THEN } ACmode \text{ is } Heating \tag{2.2}$$

where $UserTemperature$ and $ACmode$ are, respectively, the input and output variables and Low and $Heating$ are possible fuzzy sets.

2.1.1 Membership Function

"A fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between zero and one" [Zadeh, 1965].

The most known MFs are the triangular, trapezoidal and gaussian, which are, respectively, represented in the next pairs of figures and equations.

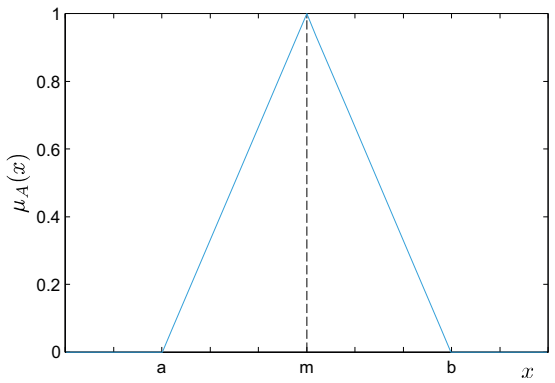
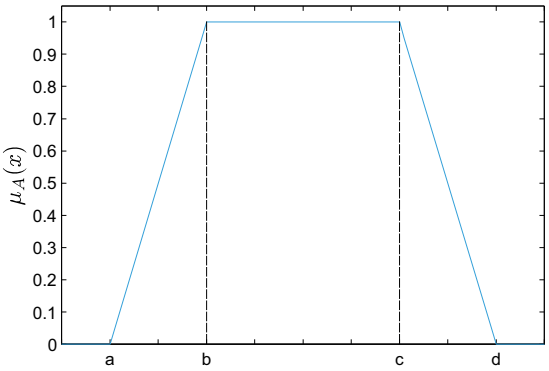


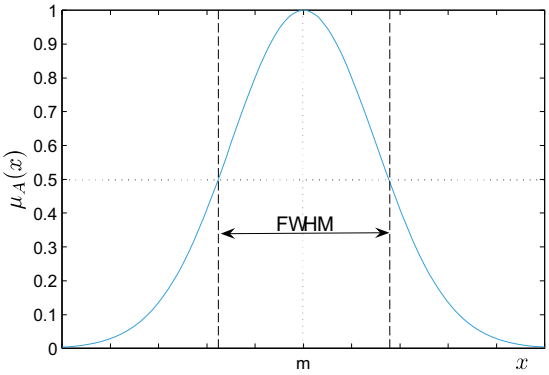
Figure 2.2: Triangular Function.

$$\mu_A(x) = \begin{cases} 0, & \text{if } x \leq a \\ \frac{x-a}{m-a}, & \text{if } a < x \leq m \\ \frac{b-x}{b-m}, & \text{if } m < x \leq b \\ 0, & \text{if } x \geq b \end{cases} \tag{2.3}$$



$$\mu_A(x) = \begin{cases} 0, & \text{if } x \leq a \\ \frac{x-a}{b-a}, & \text{if } a < x \leq m \\ 1, & \text{if } b < x \leq c \\ \frac{d-x}{d-c}, & \text{if } m < x \leq b \\ 0, & \text{if } x \geq b \end{cases} \tag{2.4}$$

Figure 2.3: Triangular Function.



$$\mu_A(x) = e^{-\frac{x-m}{2c^2}}, \; c = \frac{FWHM}{2\sqrt{2ln(2)}} \tag{2.5}$$

Figure 2.4: Gaussian Function that uses the full width at half maximum (FWHM).

Using the AC example, there are three fuzzy sets for the input variable $UserTemperature$, where the corresponding membership functions are represented in Figure 2.5.

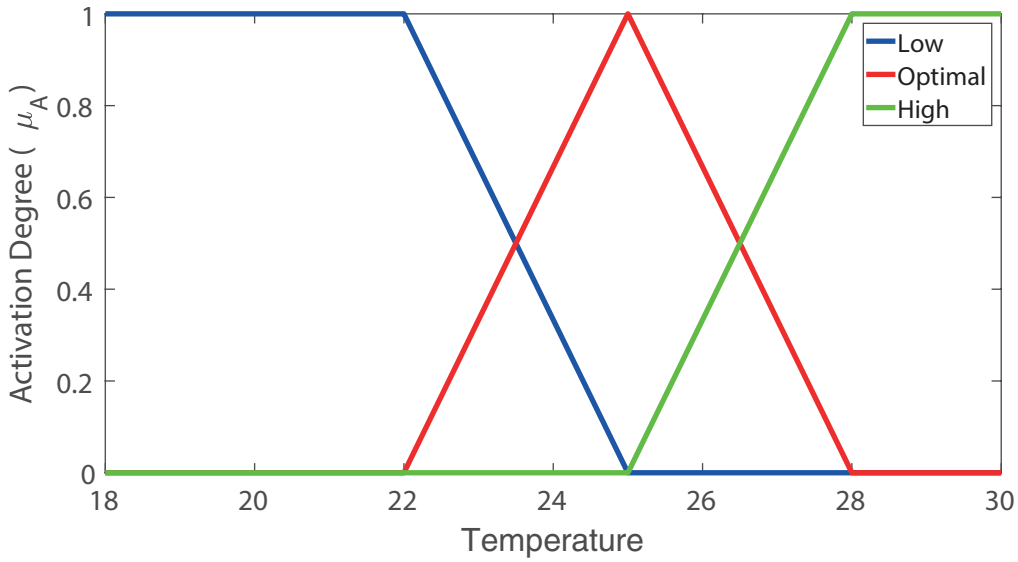


Figure 2.5: Membership Function for three fuzzy sets of $UserTemperature$ variable.

2.2 Fuzzifier

In this phase, the real values of a input variable are mapped into fuzzy inputs that are later associated to the existing fuzzy sets using the membership functions. The most common fuzzifier is the singleton fuzzifier, due to being easily implemented and also because it "greatly simplifies the computation involved in the fuzzy inference engine for any type of membership functions"[Wang, 1999]. For this reasons the singleton fuzzifier is implemented in the proposed algorithm of this dissertation.

The singleton fuzifier maps a real value $x^* \in U$ into a fuzzy singleton as Equation 2.2 shows.

$$\mu_A(x) = \begin{cases} 1, & \text{if } x = x^* \\ 0, & otherwise \end{cases} \tag{2.6}$$

More fuziffiers can be consulted in chapter 8 at [Wang, 1999].

2.3 Fuzzy Inference Engine

The Fuzzy Inference Engine (FIE) uses the fuzzy inputs obtained by the fuzzifier's transformation process and the fuzzy rules to obtain the activated consequent Q_j, then they are combined and it's calculated the inferred fuzzy output. In systems with more than one input variable it is necessary operations that can process the degree of membership of the different input variables. In the FIE there are three main operations to conjugate the antecedent part of the rules: intersection ($A_1 \cap A_2$), union ($A_1 \cup A_2$), and complement (complement of A_1, $C = \overline{A_1}$), examples of these operations with two different fuzzy sets can be consulted in Table 2.1.

Table 2.1: Comparison between the Norm Operators.

T norm - AND	
Minimum	$min(\mu_{A_1}(x_1), \mu_{A_2}(x_2))$
Algebraic product	$\mu_{A_1}(x_1).\mu_{A_2}(x_2)$
Bounded product	$max(0, \mu_{A_1}(x_1) + \mu_{A_2}(x_2) - 1)$
S norm - OR	
Maximum	$max(\mu_{A_1}(x_1), \mu_{A_2}(x_2))$
Algebraic Sum	$\mu_{A_1}(x_1) + \mu_{A_2}(x_2) - \mu_{A_1}(x_1).\mu_{A_2}(x_2)$
Bounded Sum	$min(1, \mu_{A_1}(x_1) + \mu_{A_2}(x_2))$
C norm	
Complement	$C(\mu_{A_1}(x_1)) = 1 - \mu_{A_1}(x_1)$

After obtain the result of the operators, it is calculated the implication of the result value (A) in the consequent (B), this process ($A \rightarrow B$) uses implication operator ($\rightarrow$). The most common methods of implication are minimum and the product, which can be consulted in Table 2.2.

Table 2.2: Comparison between the Implication Operators.

Implication norm ($\rightarrow$)	
Minimum	$min(\mu_{A_1}(x_1), \mu_{A_2}(x_2))$
Product	$\mu_{A_1}(x_1).\mu_{A_2}(x_2)$

After apply the implication operator occurs the aggregation of all outputs determined by the implication operation, using an aggregation operator. The output obtain after this process is denominated Inferred Fuzzy Output (IFO) and it is used in defuzzifier process. The aggregation operator use different methods as Table 2.3.

Table 2.3: Comparison between the Aggregation Operators.

Aggregation norm ($\bigtriangledown$)	
Bounded Sum	$min(\mu_{A_1}(x_1) + \mu_{A_2}(x_2), 1)$
Maximum	$max(\mu_{A_1}(x_1), \mu_{A_2}(x_2))$
Normalized Sum	$\frac{\mu_{A_1}(x_1)+\mu_{A_2}(x_2)}{min(\mu_{A_1}(x_1)+\mu_{A_2}(x_2),1)}$

Using the example of the air conditioner presented in previous sections, if the $UserTemperature$ is 23.5, as can be seen in Figure 2.6, the fuzzy sets *Low* and *Optimal* will be activated by

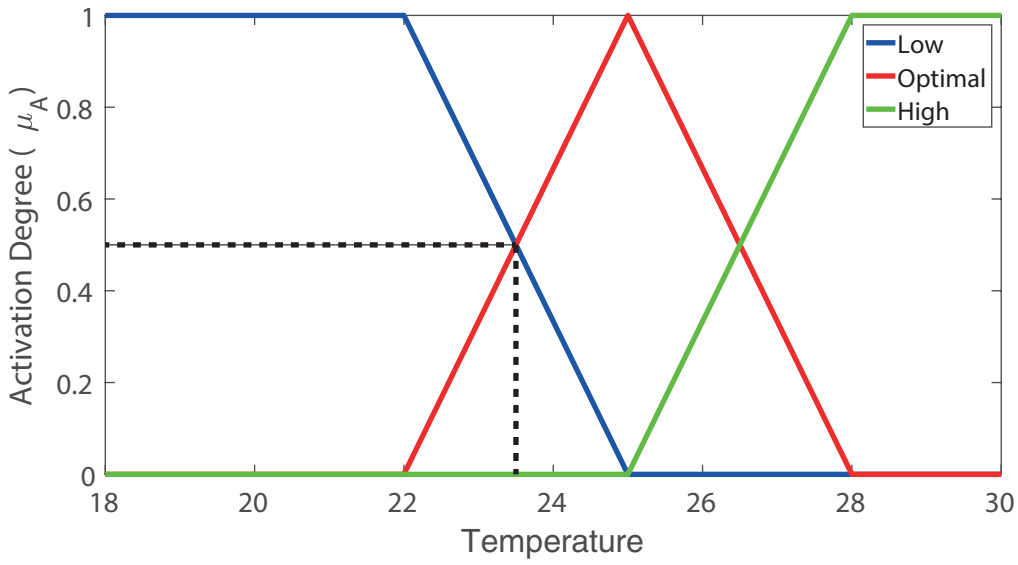


Figure 2.6: Three Membership Functions for each fuzzy set of $UserTemperature$ variable.

0.5, on other hand, fuzzy set *High* will have the value 0. After this process, the rules are subjected to the different operations introduced in this section. Since the example only has an input variable there is no need of a norm operation, so the next step is the implication, in this case it's used the minimum operator, the result of the operation is observed in Figure 2.7.

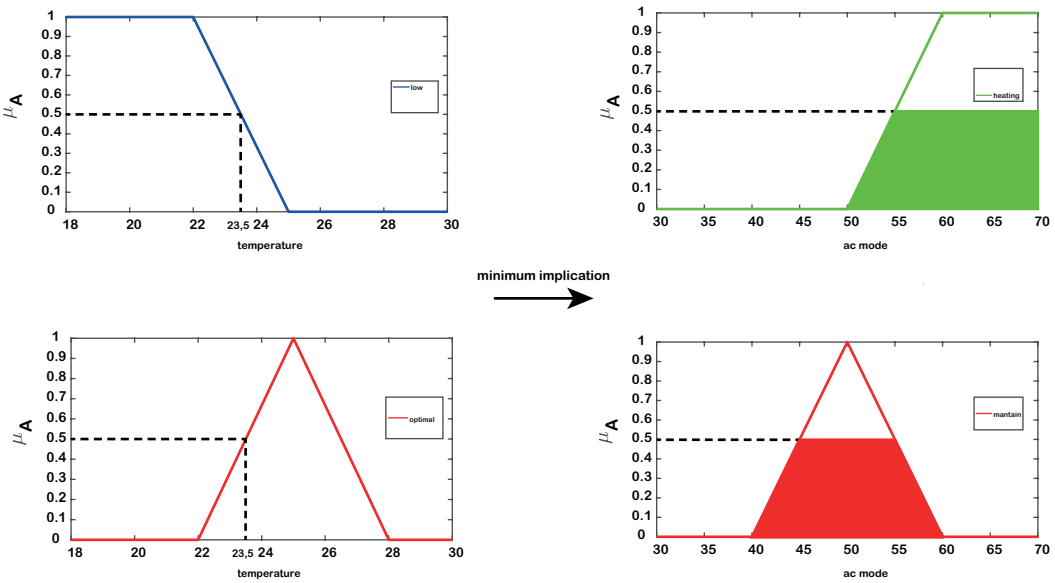


Figure 2.7: Effects of the minimum implication operator.

It's possible to verified, in Figure 2.7, that the implication process occurs in output fuzzy sets associated, by the fuzzy rules, to activated input fuzzy sets. For example, the 0.5 degree of membership of the input fuzzy set *Low* make the associated output fuzzy set *Heating* be activated by a degree of 0.5. After the implication process the resulting of all rules are combined using an aggregation operator, in this case the maximum, as Figure 2.8 shows.

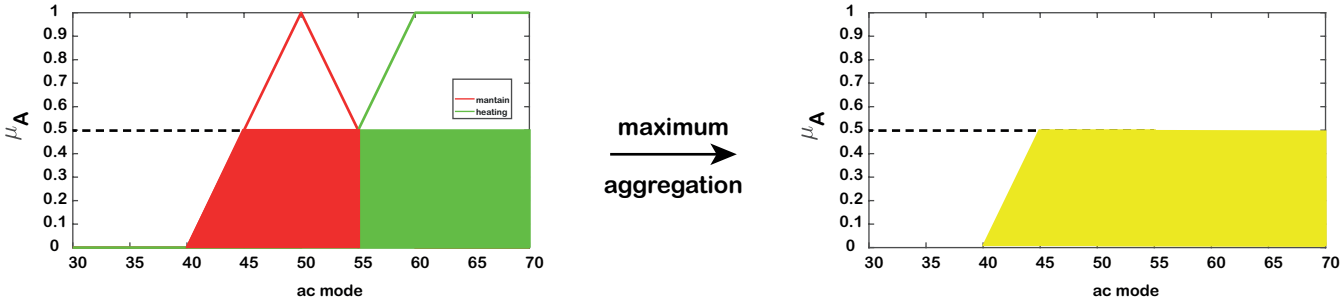


Figure 2.8: Exemplification of the aggregation operation in Figure 2.7 results.

2.4 Defuzzifier

The last piece of the puzzle is the defuzzifier where the Inferred Fuzzy Output (IFO) obtained in the FIE is transform in real valued output. The most known defuzification methods are

the center of gravity, first of maxima and last of maxima, as shown in Table 2.4, where $hgt(Y) = \{y \in v | \mu_Y(y) = \sup_{y \in v} \mu_Y(y)\}$.

Table 2.4: Comparison between the defuzification methods.

Defuzification methods	
Center of Gravity	$y^* = \frac{\int_{max}^{min} y\mu_Y(y)dy}{\int_{max}^{min} \mu_Y(y)dy}$
First of Maxima	$y^* = inf\{y \in hgt(Y)\}$
Last of Maxima	$y^* = sup\{y \in hgt(Y)\}$

Using the IFO of the example in the Figure 2.8 is possible to see the final values in Figure 2.9 using the three defuzzification methods presented in Table 2.4

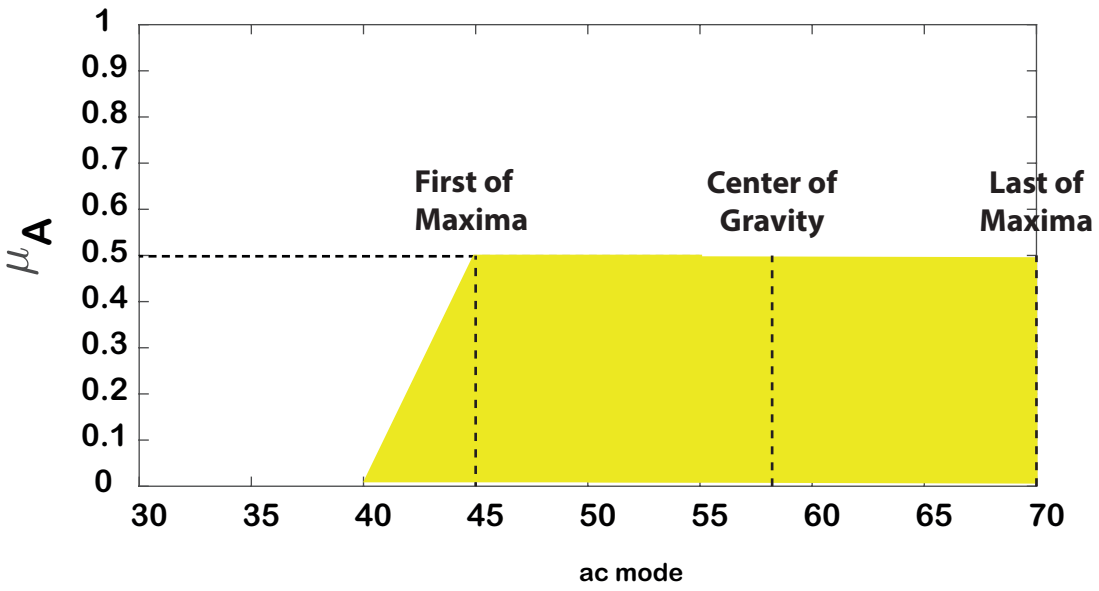


Figure 2.9: Application of defuzzification methods in example's Infered Fuzzy Output.

Chapter 3

Adaptive and Evolving Online Fuzzy Control Algorithm

In this chapter is analysed the algorithm implemented to control unknown MISO systems. The main properties of the implemented algorithm are the following:

- Online adaptation of control parameters;
- Online modification of control structure;
- Use historical information to adapt the control parameters and modified the control structure;
- No model of the plant needed;
- No previous knowledge about the control policy, in other words, the algorithm can start working from an empty set of rules.

The online adaptation and evolving is the capacity of changing, respectively, the control parameters and structure at the same time it is controlling a system and also means that, before starting to control a system, there is no need of a training process, using previously obtained data. The use of historical information, specially in the evolving process when adding rules, assures the identification of the best new centers for the new MFs. The fact the algorithm doesn't need the model of the controlled plant makes the algorithm adaptable to different unknown MISO systems.

However the algorithm requires some information from the controlled system, specifically, the range of its inputs and command signal, because they are needed to determine the gain parameter and make it adaptable to the controlled system, to initialize new MFs and the consequents of the newly created rules. The value of this range doesn't need to be precise, since the adaptation and evolving process will compensate the possible errors in the estimation of the inputs ranges.

3.1 Architecture

The implemented algorithm is composed by a Fuzzy Control System (FCS) that is altered by an adaptation mechanism using the system's stored. The FCS is then applied in the controlled system, as shown in Figure 3.1.

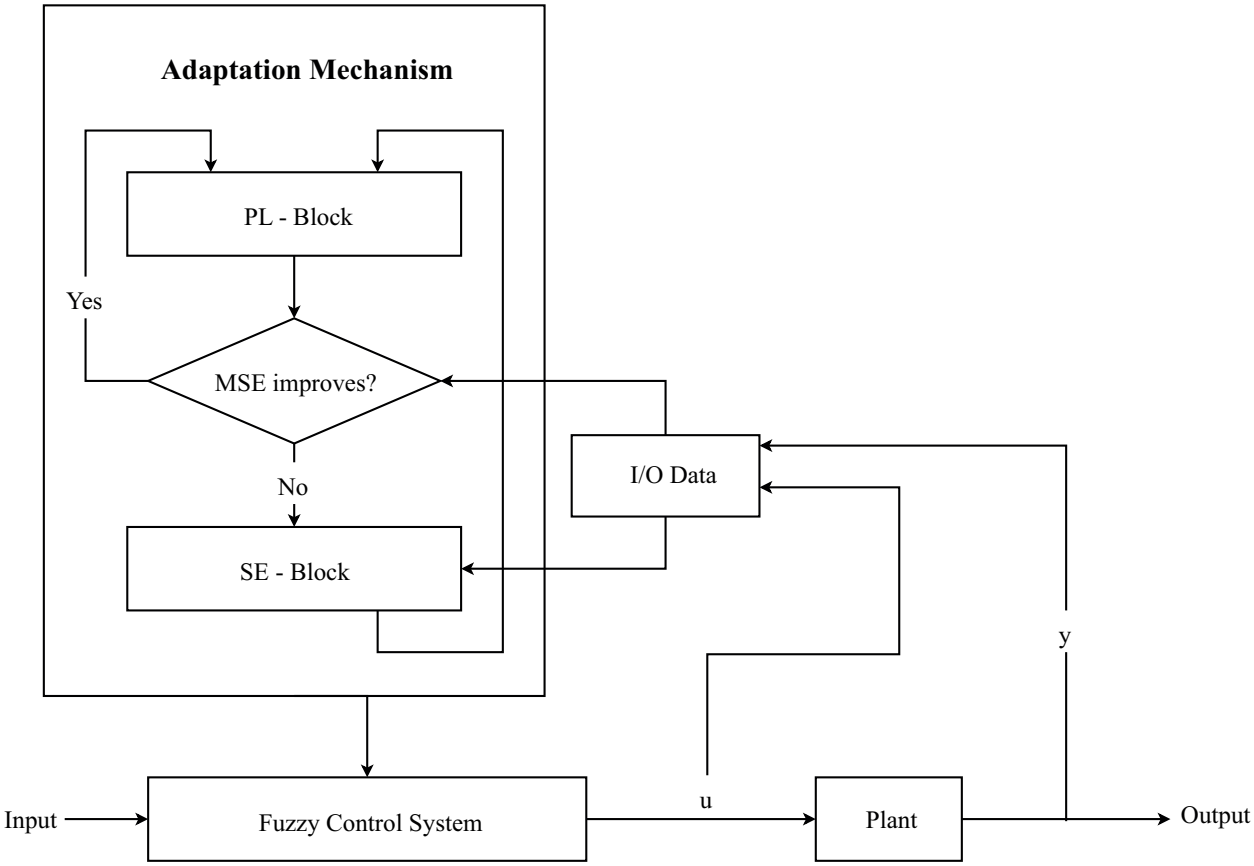


Figure 3.1: Representation of the algorithm architecture.

The adaptation mechanism is composed by two main blocks the Parameter Learning (PL) block and the Self Evolving (SE) block. The PL-block is responsible for the adaptation process, where the consequents of existing rules are changed, and the SE-block responsible for the evolving process, where can be added a new MF, and consequently new rules, to the FCS, using the I/O information of the plant under control.

The algorithm 3.1 represents the pseudo-code of the adaptation mechanism (illustrated on Figure 3.1), and it is possible to observe the comparison of the actual Mean Square Error (MSE) with the previous one which decide the activation or not of the SE-block. Another important aspect is the manipulation of I/O Data, when a MF is added in the SE is important that the data used previoulsy don't compromise the addition of a new MF, for that purpose when a new MF is added, for M iterations, the SE-block is never activated, as the pseudo-code shows. This verification is an improvement to the original algorithm OSEFC [Cara *et al.*, 2010].

Algorithm 3.1 Adaptation Mechanism

1: Calculation of MSE and error
2: Activate: `PL_Block`
3: **if** $(MSE(k) > MSE(k-1))\&\&(k > M)$ **then** ▷ k represents the iteration and M the Memory's size
4: Activate: `SE_Block`
5: **if** Is created a new MF **then**
6: k is reset
7: **end if**
8: **end if**

3.2 FCS Structure

As the architecture of the algorithm shows in Section 3.1 the adaptation mechanism changes a FCS, in this section is identified the FCS characteristics.

The knowledge base of the FCS is a special case of a Takagi Sugeno (T-S) fuzzy system, the zero order T-S fuzzy system, which uses fuzzy sets in the antecedent part, and a scalar value in consequents part.

Since we are working in a MISO system, there is need of a conjunction operation between the values of membership degrees of the different inputs, to that purpose is used the T-norm algebraic product present in Table 2.1.

The chosen defuzzification method is the weighted average that translates in the result output of the FCS being:

$$u(k) = G(\mathbf{x}(k), \phi(k)) = \frac{\sum_{i=1}^{N_r} Q_i \ \cdot \ \alpha_i(x(k))}{\sum_{i=1}^{N_r} \alpha_i(x(k))}, \tag{3.1}$$

where $x(k)$ is the vector of the input values at time k, $\phi(k)$ is the set of fuzzy parameters at time k, Q_i is the consequent of the ith rule and α_i is the conjugated strength of the activation degrees of the different MFs of the ith rule, which is calculated using, as mentioned before, a T-norm algebraic product as shown in the next equation:

$$\alpha_i(\mathbf{x}) = \prod_{j=1}^{N_v} \mu_j^i(x_j), \tag{3.2}$$

where $\mu_j^i(x_j)$ is the activation degree of the j-th input in the related MF of the i-th rule.

3.2.1 MF Structure

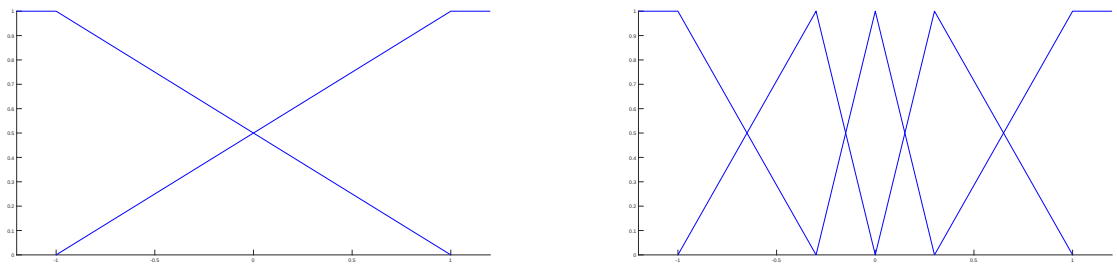
The chosen MF is a triangular function normalized, in other words, the frontier points of a triangular MF will be the centers of consecutive triangular MFs as the examples in Figure 3.2 shows. This aspect "provide better transparency and interpretability to the fuzzy system"

[Wang *et al.*, 2008a], makes only necessary to save the centers of the triangular functions and make the sum of all activation degrees of the MFs equal to one, in any iteration. So, the use of triangular normalized MF gives three advantages:

- The facilitation of new rules creation process, because the frontier points of the existing and the new rule can easily adapt to the new created centers;
- The reduction of computing complexity, since triangular functions are, relatively, easy to manage and the normalization guarantee that the maximum number of MFs for each input activated at any time is only two, the others MFs have all zero activation degree;
- The simplification of the calculation of the resulting command signal in Equation 3.1, due to the fact that for any input value the sum of all activation degrees is one, i.e. $\sum_{i=1}^{N_r} \alpha_i(x(k)) = 1$. So the resulting and formulae used by the algorithm to calculated the command signal is presented in Equation 3.2.1

$$u(k) = G(x(k), \phi(k)) = \sum_{i=1}^{N_r} Q_i \ \cdot \ \alpha_i(x(k)) \tag{3.3}$$

It is also important to refer that are created two initial MFs, which use as center the limits of the range of the related input variable. Another characteristic of the initial MFs is that they are trapezoidal to guaranteed that data beyond the range defined by the user has an activation degree equal to one, as the example in Figure 3.2a shows. The added MFs are located between these two initial MFs and are a triangular function normalized, as shown by the same example in Figure 3.2b that also shows the change in the location of the frontier points of the initial MFs.



(a) (b)

Figure 3.2: Examples of a composition of MFs: initial composition (a) and a final composition after using the proposed algorithm (b).

3.2.2 Consequent values

Since the algorithm initializes with two MFs for each input, there will be 2^{N_v} rules, being N_v the number of inputs of the system. These rules need an initial consequent value, there are two main considerations in the initial value, it must be related to the command range and it should be equal to create balanced rules. To achieve this, the consequents are initialize with the lowest value possible of the command signal range that also guarantee no troubles in most controlled systems since, usually, the lowest value corresponds to a stationary and resting state.

However since there are systems that this doesn't applied is given the liberty to the user define the initial command value.

3.3 Parameter Learning

The Parameter Learning block occurs in every iteration, and is responsible for changing the consequents of the activated fuzzy rules, using the tracking error, i.e. the difference between a reference and the value of the system's output variable. It is also analysed the monotonicity of the system output with respect to the the command signal as well as a gain parameter, that will counteract the usual difference of the magnitude between the command signal and output, and with that purpose the gain will depend of their ranges. Another key aspect is the need of only change the activated rules at a k iteration, for that purpose is used the antecedent value obtained using the Equation 3.2. These assumptions result in Equation 3.4.

$$\Delta Q_i(k) = G \cdot signM \cdot \alpha_i(k-1) \cdot e(k), \tag{3.4}$$

where ΔQ_i represents the proposed variation for the value of the consequent of the ith rule, G is the positive learning gain, $signM$ is the signal that represents the monotonicity, $\alpha_i(k-1)$ represents the strength of the ith rule at the previous iteration, $e(k)$ is the error obtained by the difference between the reference and actual output value($e(k) = r(k-1) - y(k)$) and k is the current iteration. The equation is equal to the one presented in recent papers [Cara *et al.*, 2010], [Sadeghi-Tehran *et al.*, 2012].

3.3.1 Gain parameter

The gain parameter is an essential parameter of equation 3.4, and has the important role to give an appropriate strength to the adaptation of consequents. In some observed plants the discrepancy of magnitude between command signal and the reference is significant, and if a gain is not applied the change in the consequents can create problems. In fact, the results obtained in plants with significant discrepancies between the magnitude of the command

signal (Δu, where $\Delta u = |u_{max} - u_{min}|$) and the magnitude of the reference (Δr, where $\Delta r = |r_{max} - r_{min}|$), and where the gain wasn't applied, showed two main situations:

- When $\Delta u >> \Delta r$: the changes in consequent would never be sufficient to make the variable achieve the referenced value, since without the gain the ΔQ_i would be too low to effectively influence the consequent value;
- When $\Delta r >> \Delta u$: the changes in consequent would be too strong and would create instability, since the ΔQ_i would be too high when compare the normal values of the consequent;

So with these observations and to make the variation of the consequent adaptable to different systems is necessary to implement a gain that use in its calculation the Δu and Δr. The proposed equation to calculate the gain is the same as the proposed in some litterature ([Rojas *et al.*, 2006] and [Cara *et al.*, 2010]):

$$Gain = \frac{\Delta u}{\Delta r} = \frac{|u_{max} - u_{min}|}{|r_{max} - r_{min}|} \tag{3.5}$$

However, in realized tests was observed that the use of equation 3.5 to calculate the gain, in spite of being theoretically a faster solution to make a variable achieve a reference, can easily create instability and for that reason that value was softened through a division of the value of Equation 3.5 by a positive constant (C) as Equation 3.6 shows.

$$G = \frac{Gain}{C} \tag{3.6}$$

The realized tests showed that a constant between $15-25$ would have good performance in every plant/system tested, though the best performances in the different plants had different values of C.

The influence of this gain in regulating the strength of the ΔQ make it an important value for a user, if the variation is too fast the user should reduce the gain, increasing the C, if the adaptation is not being sufficiently fast, the user should decrease C, increasing the gain.

3.3.2 Monotonicity parameter

In most plants/systems an increasement of the command signal translate in an increasement of the plants output, however there are systems that are the opposite and a adaptive algorithm must be prepare to either case. To achieve this, is necessary to analyse the monotonicity of the plant and the command signal, through the controllability condition, presented in Equation 3.7, which is used to guarantee that the plant output always depend on the control signal [Phan and Gale, 2008].

$$\frac{\partial f(x,u)}{\partial u} \neq 0 \quad \forall x \in \Omega, \forall u \in \mathbb{R}. \tag{3.7}$$

Since the equation 3.7 can't be zero, there are two main situations: when it is positive, which means that an increasement on the command signal translates an increasement on the plant output; and when it is negative, an increasement in the command signal translates a decreasement on the plant output.

It is only evaluated the signal of equation 3.7 being this the Monotonicity parameter in equation 3.4 ($signM$). In practice, the Monotonicity parameter is calculated by [Cara *et al.*, 2010]:

$$signM = signal(\frac{y(k) - y(k-1)}{u(k) - u(k-1)}). \tag{3.8}$$

On the tests, two problems about $signM$ (eq. 3.8), were discovered:

- When the tracking error ($e(k) = r(k-1) - y(k) = 0$) is zero, the consecutive values of $y(k)$ and $r(k)$ are constant, so the result of equation 3.8 is undefined;
- When plants are subjected to perturbations, even with the change of the command signal the perturbation can make the output has a result opposite to the expected, and that induces the monotonicity parameter to has the signal opposed to the pretended, empowering the perturbation effect.

To solve both problems, a solution was developed using the calculation of the monotonicity in the initial iterations and maintain that value through the control process. This solution presented good results in the tested plants and it was used in the proposed algorithm, though isn't perfect due to the fact that can exist a plant dynamic where the monotonicity invert, which make the solution create problems in the rules adaptation process. However in tested plants, the inversion of monotonicity never happen.

3.3.3 Additional conditions

The PL block adapts the rules to make the output value achieve the reference, however, most controlled systems are limited in their operation range and some adaptations can make the controller surpass the permitted values for the command signal, to prevent this it is suggested, in [Cara *et al.*, 2010], an additional condition to limit the variation of the consequents, as shown by the Equation 3.9.

$$\Delta Q_i(k) = \begin{cases} 0, & \text{if } u(k-1) = u_{min} \ \ \& \ \ \Delta Q_i(k) < 0 \\ 0, & \text{if } u(k-1) = u_{max} \ \ \& \ \ \Delta Q_i(k) > 0 \\ \Delta Q_i(k), & \text{otherwise.} \end{cases} \tag{3.9}$$

However, the applied equation in the proposed algorithm is different, the first modification was change the conditions $u(k-1) = u_{min}$ and $u(k-1) = u_{max}$ to $u(k-1) <= u_{min}$ and $u(k-1) >= u_{max}$, respectively. The result of the original conditions (eq. 3.9) could easily be wrongly checked if in a previous iteration the ΔQ_i made $u(k)$ inferior than u_{min} or greater

than u_{max}. After this small change in the conditions, it was observed that the consequents of some rules could be uncompensated in some iterations, and it would be better they be within range of the command signal $[u_{min}, u_{max}]$, so after this observations the final format of the consequents variation condition in the proposed algorithm is given by Equation 3.10.

$$\Delta Q_i(k) = \begin{cases} Q_i(k) = u_{min}, & \text{if } Q_i(k-1) \leq u_{min} \ \ \& \ \ \Delta Q_i(k) < 0 \\ Q_i(k) = u_{max}, & \text{if } Q_i(k-1) \geq u_{max} \ \ \& \ \ \Delta Q_i(k) > 0 \\ Q_i(k) = Q_i(k-1) + \Delta Q_i(k), & \text{otherwise.} \end{cases} \tag{3.10}$$

3.3.4 Pseudo-Code

So, in conclusion, the PL-Block can be described by the pseudo-code presented on Algorithm 3.2.

Algorithm 3.2 Parameter Learning Block

procedure INITIALIZATION
  Determinate $\Delta u = |u_{max} - u_{min}|$
  Determinate $\Delta r = |r_{max} - r_{min}|$
  Determinate the Gain $= \frac{\Delta u}{\Delta r}$
    G $= \frac{Gain}{C}$ ▷ C is the constant that soften the gain
  Determinate the Monotonicity Parameter
    $u(1) = u_{min} + \Delta u \cdot 0.2$
    $u(2) = u_{min} + \Delta u \cdot 0.4$
    $y(1) = f(x(k), u_1)$ ▷ $f(x(k), u(k))$ represents the plant dynamics
    $y(2) = f(x(k), u_2)$
    $SignM = signal(\frac{y(2)-y(1)}{u(2)-u(1)})$
end procedure
procedure PARAMETER LEARNING BLOCK
  Determinate error $e(k) = r(k-1) - y(k)$
  for each rule **do**
    $\Delta Q_i(k) = G \cdot signM \cdot \alpha_i(k-1) \cdot e(k)$
    if $(Q_i(k-1) \leq u_{min})\&\&(DeltaQ_i(k) < 0)$ **then**
      $Q_i(k) = u_{min}$
    else if $(Q_i(k-1) \geq u_{max})\&\&(DeltaQ_i(k) > 0)$ **then**
      $Q_i(k) = u_{max}$
    else
      $Q_i(k) = Q_i(k-1) + \Delta Q_i(k)$
    end if
  end for
end procedure

3.4 Self-Evolving

The Self-Evolving Block is responsible for the evolution of the FCS topology, in other words, the addition of new MFs and fuzzy rules and can be divided in three main elements:

- First, an input variable is selected to add a new MF. It isn't practically adding MF to all inputs since the number of rules would increase drastically [Pomares *et al.*, 2002], so it is analysed, using the I/O data collected, which input are contributing more to the approximation error.
- After the selection of the input variable, it is selected the best location for the center of the MF to be added, through the analysis of the error distribution.
- To prevent performance drop issues, it is important to initialize the new rules assuring that the new generated command signal is similar to the previous command signal.

The flowchart that represents the SE-Block is shown in figure 3.3.

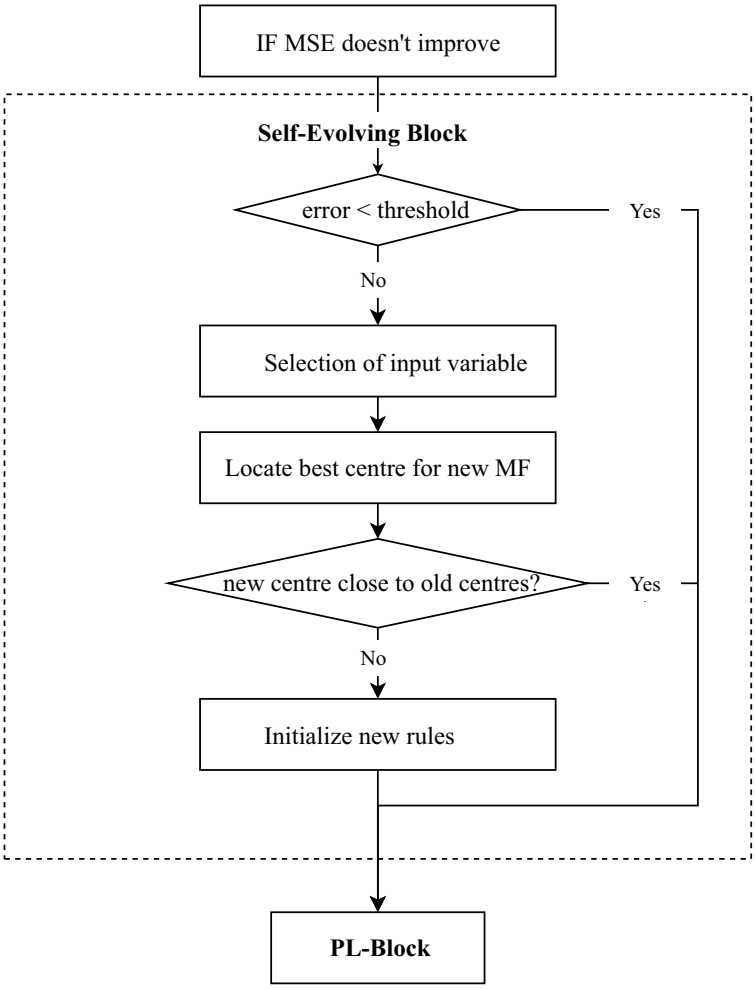


Figure 3.3: Representation of the Self-Evolving Block.

One of the main problems, shown by the tests, it is an uncontrollable growth of MFs and rules that, consequently, increase the computation time and destabilize the control process.

To counteract these aspects it was necessary the definition of two conditions that prevent the overgrowth. The first condition is a threshold that prevents the creation of new MFs if the tracking error is too small, basically if the error is smaller than the threshold the SE-block is bypassed, as the flowchart of Figure 3.3 shows. To make the threshold adaptable to all plants, it must depend of the operational range of the inputs, as Equation 3.11 shows.

$$threshold1 = F * \Delta e, \quad \text{with } 0 < F < 1. \tag{3.11}$$

where F works as a percentage and Δe is the range of the error that is basically equal to the range of the input. This condition is present in the original algorithm [Cara *et al.*, 2010].

The second condition occurs after the selection of the input and the location of the proposed center and its main goal is to prevent the creation of new MFs with centers relatively close to the existing ones, this condition is analysed in detail in subsection 3.4.3. This second condition is an improvement to the original algorithm.

3.4.1 Selection of the input variable

The main idea of the selection process is analyse the contribution of each input variable to the approximation error. So, for each input variable (j) is applied the following process [Cara *et al.*, 2010]:

- Construction of the set parameters (Φ_j) of an auxiliary Fuzzy Control System (G^j_{aux}) using the MFs of the analysed variable and creating auxiliary MFs for the other inputs:
  - The number of auxiliary MFs, created for the other inputs, is determined by the equation 3.12, as proposed in [Cara *et al.*, 2010]. In the proposed algorithm the only improvement is the use of the close rounded up integer, since most cases the result of 3.12 won't be a integer.
  - The auxiliary MFs will be created using the range of the inputs to make sure the centers will be distributed uniformly, being equidistant between themselves.
  - To complete the G^j_{aux} is necessary to calculate the consequent, for that purpose and since [Cara *et al.*, 2010] doesn't present a clear solution, it was used the equation 3.14 from [Mendes *et al.*, 2014].
- Determination of the command signal of the auxiliary Fuzzy Control using input data stored in M.
- Calculation of the Index of Responsibility for input j (IR_j) as equation 3.13 shows.
- The input with the largest value of IR_j is the selected input to locate a new MF.

$$n^* = \frac{\sqrt[N_v]{\#M}}{2} \tag{3.12}$$

where n^* is the number of auxiliary MFs, N_v the number of inputs, $\#M$ the total number of data stored. This expression is the result of "under the assumption that memory M provides a uniform distribution of the data in the state-space, (...) the number of data points along one axis is indeed the n-th root of the total number" [Cara *et al.*, 2010].

$$IR_j = \sum_{m=1}^{\#M} (u_m - G^v_{aux}(x_m; \Phi_j))^2 \tag{3.13}$$

where IR_j is the Index of Responsibility for a variable j, u_m is the stored command signal and the $G^j_{aux}(x_m; \Phi_j)$ is the command signal created by the auxiliary FCS with the parameters Φ_j and value stored of the variables x_m . Basically, this equation is a comparison between the stored command signal and the command signal generated by G^j_{aux}.

$$Q_i = \frac{\sum_{m=1}^{M} u_m \cdot \alpha_i(x_m)}{\sum_{m=1}^{M} \alpha_i(x_m)} \tag{3.14}$$

where Q_i is the consequent of the ith rule, u_m is the stored command signal, $\alpha_i(x_m)$ is the value of the antecend result of a T-norm alebraic product operation of the activation degree of the stored input data x_m in the MFs for the ith rule. The formulae is presented in paper [Mendes *et al.*, 2014].

3.4.2 Location of the new membership function

After selecting the input variable is necessary to locate the new center, to that purpose is analysed the error distribution through the stored data related to the chosen input. [Cara *et al.*, 2010] proposes the division of the chosen input possible values in K intervals, for that purpose the input range is divided in K intervals.

For each interval is calculated the mean square error (MSE), using equation 3.16 for each interval X^s_j defined by equation 3.15 .

$$X^s_l = \{x \| x_s \in [x_{s_{min}} + (l-1) \cdot \Delta x_s, x_{s_{min}} + l \cdot \Delta x_s]\} \tag{3.15}$$

where X^s_j is the interval for the input s, $x_{s_{min}}$ is the minimum value possible of the input s, Δx_s is the width of each interval and $l \in 1 : K$, being K the number of intervals.

$$\bar{e^2}(X^s_l) = \frac{\sum_{m=1}^{M} (u_m - G(x_m, \Phi))^2}{M}, \quad \forall\ m \text{ where } x_m \in X^s_l \tag{3.16}$$

where M is the number of data stored, u_m is the stored command signal and x_m the stored input values for the m iteration, $G(x_m, \Phi)$ is the actual fuzzy control applied in the previous input data stored and X^s_j represents the interval. The condition means that in the

equation 3.16 is only verified the data where the chosen input data belongs to the defined interval X_j^s.

After this process, the gravity center of the error distribution is calculated through formulae 3.17, and will be the new proposed center.

$$\phi_s = \frac{\sum_{l=1}^{K} c_l \cdot \bar{e^2}(X_l^s)}{\sum_{l=1}^{K} \bar{e^2}(X_l^s)} \tag{3.17}$$

where ϕ_s is the new center, $\bar{e^2}(X_j^s)$ is the MSE from the data belonging to the interval X_l^s and c_l is the center of the l interval.

The final proposed algorithm differs a little because the computation time and process of the method present in [Cara *et al.*, 2010] is costly. So instead of divide in intervals is just analysed the data collected, using the next formulae, which gives the index of the new center in the data collected:

$$index = \frac{\sum_{m=1}^{M} m \cdot \bar{e^2}(u_m)}{\sum_{l=1}^{K} \bar{e^2}(u_m)} \tag{3.18}$$

where $\bar{e^2}(u_m)$ is calculated using formulae 3.16, and m is the index of each datum. This equation represent a method that gives strength to the index where the MSE is stronger, since the consecutive results are close to each other, this method will choose the index of the more troublemaker data. Basically, this formula allows to identified the specific data that is contributing to the error, for example if the data between the 120 and 150 samples are responsible for a large amount of error the Equation 3.18 will "push" the index towards the samples of 120 and 150. An interesting aspect of this formulae is that analyses the contribution for the tracking error of all the samples, if for example were used a method that only choosed the value with the largest contribution, the algoritm could choose an anomalous result as a center, using the contribution of all data makes this formulae more reliable and capable to deal with possible anomalies in the results.

3.4.3 Evaluation of the new center

The original algorithm after the addition of a new center didn't have any verification of the new center, as proposed by [Cara *et al.*, 2010], and some results showed that the new centers were sometimes too close to previous ones, creating instability and unnecessary rules. To prevent this the new center and all existing centers are compared, then the comparison is verified if is greater than a defined threshold, if it is a new MF is created, if not, the SE-block is bypassed, as Figure 3.3 shows. To make the verification adaptable to any system the threshold is a percentage of the related input range. The pseudo-code related to this verification is:

The value of percentage could be adjusted to allow the possibility of more or less rules, the default value is 10%. This evaluation is a great improvement of the original algorithm

Algorithm 3.3 Self Evolving Pseudo-code

```
for each rule i do
    if (newcenter-center(i))<(percentage*InputRange) then
        The new center is too close to an existing center and the MF won't be created
    end if
end for
```

proposed by [Cara *et al.*, 2010], since there wasn't any verification of the center proximity.

3.4.4 Initialization of the new Fuzzy Rules

The addition of a new MF results in a new set of rules, if the new rules' consequents are random the originated command signal can cause a sudden decrease in performance. To prevent this, it is important make sure that the FLC performance isn't affected, in other words, it is necessary to assure that the new command signal creates a similar response as the previous command signal.

To achieve this purpose, the consequents of the rules defined by the previous MFs are maintained and the consequents of the rules with the new MF are calculated. In the proposed algorithm were tested three different options for the the consequents of the rules with the new MF:

- First, a rudimentary solution was created, the consequents would have the minimum value of the command signal;
- Second, the consequents would have the value of the previous command signal, as suggested in [Cara *et al.*, 2010];
- Third, is used the equation 3.14 to determine the new consequents.

The analyse of the results is in chapter 4, and shows that the third option presented the best results, using the formulae 3.14 present in paper [Mendes *et al.*, 2014].

Since the number of rules resulted from the combination of all MFs for all inputs through formulae $N_{rules} = \prod_{j=1}^{N_{var}} N_{MF}^{j}$, when a new MF is added the new number of rules will be the result of the number of existing MFs related to all the inputs except the selected input.

3.4.5 Pseudo-Code

So, the Self Evolving Block can be described by the pseudo-code 3.4.

Algorithm 3.4 Self Evolving Pseudo-code

```
1: procedure SELF EVOLVING BLOCK
2:     if error>threshold1 then
3:         procedure SELECT INPUT VARIABLE
4:             for each input variable do
5:                 Create an auxiliary Fuzzy Control G^v_aux
6:                 Calculate the index of responsibility IR_v summing the comparisons between the auxiliary G^v_aux for a stored x_m to the corresponding original command signal u_m
7:             end for
8:             The variable with the highest IR_v is the chosen variable
9:         end procedure
10:        procedure CALCULATE THE NEW MF'S CENTER
11:            Use equation 3.18 to determine the index of the data which contains the new-center
12:        end procedure
13:        if (newcenter-oldcenters)>threshold2 then
14:            procedure INITIALIZE NEW RULES
15:                Maintain the consequents of rules with previous combinations of MFs
16:                Calculate the consequents of rules with the new MFs using formulae 3.14
17:            end procedure
18:        end if
19:    end if
20: end procedure
```

Chapter 4

Results and Discussion

In this chapter, it is analysed the different systems controlled by the proposed algorithm and the corresponding results, being divided in three categories:

- First is analysed results when the PL-Block is activated and the SE-Block is deactivated, in other words, only the change of existing rules is activated;
- Second is analysed when both the PL-Block and SE-block are activated, in other words, there is change in the existing rules and new rules can also be added;
- Third is analysed the perturbations and the proposed algorithm capacity of response, evaluating in the process its robustness.

In the last segment, is analysed the influence of the different parameters like the Gain, Memory size, value of the different algorithm thresholds and is also evaluated the results of the different initialization methods presented in 3.4.4. The results sustain the choices and the vulnerabilities of the proposed algorithm mentioned in chapter 3.

4.1 Continuous-Stirred Tank Reactor (CSTR)

The Continuous-Stirred Tank Reactor (CSTR), is one of the tested plant, and consist in a process that "consists of an irreversible, exothermic reaction $A \rightarrow B$ in a constant volume reactor cooled by a single coolant stream" [Morningred *et al.*, 1992]. The equations that describe the process are presented in Equation 4.1 and the related variables meaning can be consulted in Table 4.1

$$\frac{\partial C_A(t+d_c)}{\partial t} = \frac{q(t)}{V}(C_{A0}(t) - C_A(t+d_c)) - k_0 C_A(t+d_c) exp\left(-\frac{E}{RT(t))}\right) \quad (4.1)$$

$$\frac{\partial T}{\partial t} = \frac{q(t)}{V}(T_0(t) - T(t)) - \frac{-\Delta H k_0 C_A(t+d_c)}{\rho C_p} exp\left(-\frac{E}{RT(t))}\right) + \frac{\rho_c C_{pc}}{\rho C_p} q_c(t)\left[1 - exp\left(\frac{-hA}{q_c(t)\rho_c C_{pc}}\right)\right](T_{c0}(t) - T(t)) \quad (4.2)$$

$$y = C_A(t), \quad u(t) = q_c(t). \quad (4.3)$$

Table 4.1: Nominal CSTR parameter values

Measured product concentration	C_A	0.1 $[mol/l]$
Reactor temperature	T	438.54 $[K]$
Coolant flow rate	q_c	103.41 $[l/min]$
Process flow rate	q	100$[l/min]$
Feed concentration	C_{A0}	1 $[mol/l]$
Feed temperature	T_0	350 $[K]$
Inlet coolant temperature	T_{c0}	350 $[K]$
CSTR Volume	V	100
Heat transfer term	h_A	7×10^5 $[calmin^{-1}K^{-1}]$
Reaction rate constant	k_0	7.2×10^{10} $[min^{-1}]$
Activation energy term	E/R	1×10^4 $[K]$
Heat of reaction	ΔH	-2×10^5 $[cal/mol]$
Liquid densities	ρ, ρ_c	1×10^3 $[g/l]$
Specific heats	C_p, C_{pc}	1 $[calg^{-1}K^{-1}]$

The results in [Morningred *et al.*, 1992] when changing the command variable, in this case, the coolant flow rate show that the CSTR plant have highly nonlinear characteristics which make it a perfect test subject to our algorithm capabilities, since one of the objectives was the control of non-linear systems.

As mentioned in Chapter 3 the algorithm only needs the operating range of the inputs and output wich can be consulted in the next Table 4.2. The Fuzzy Control System used to control the CSTR plant will be a MISO system, with two inputs, the reference and output of the CSTR plant which is feedback and works as a input, and one output, the command signal which works as a input in the plant process, as shown in Figure 3.1, except in the figure the output of the plant isn't feedbacked.

Table 4.2: Nominal CSTR parameter values

Reference minimum value	0.06
Reference maximum value	0.13
Reference Range	0.07
Concentration minimum value	0.06
Concentration maximum value	0.13
Concentration Range	0.07
Command minimum value	95
Command maximum value	120
Command Range	25

4.2 Two coupled DC motors

The chosen Real-world system to test the proposed algorithm was a experimental system formed by two similar DC Motors coupled by a shaft, as figure 4.1 shows. The first motor is controlled using Pulse Width Modulation (PWM) generated by the micro-controller (μ_C) wich use the command value obtained by the proposed algorithm. The second motor works a generator, however is conected to a variable load, which can also be controlled by the user to create pertubations in the system. The goal is to control the velocity of the first motor, which is determinated by the motors encoders both with a encoder of 64 Counts Per Revolution (CPR) and for the maximum of 12 [V] they can achieve 11000 Rotations per minute (RPM) [Mendes *et al.*, 2017]. The unit used for the velocity in the controller is pulses per 100 miliseconds (pp/(100ms)).

The controlled motor receives energy of a motor drive which, in its turn, received the needed energy from a external power source and the controlled PWM signals used to define the ammount of energy given to the motor. The encoders of the motor give data to the μ_C with the purpose of obtained the velocity of the motor.

The generator motor is conected to a variable load which, in practice, is an electronic circuit from Arachnid Labs labelled Re:load, this device needs an analog voltage so it was built an RC filter and a voltage divider to make the controlled signal using PWM signals of the μ_C possible.

The μ_C is a Texas Instruments Tiva C microcontroller and, as mentioned before, receive the encoders data and gives the PWM signals for the load and the controlled motor drive, through serial communication. The μ_C is also conected to the computer that uses MATLAB to applied the algorithm to the system and transmits to the computer the velocity obtained by the encoders and receives an integer command that is transform in a PWM signal.

This Real-World system has noise, electro-magnetic perturbations, friction and other phenomena, typical in a non-linear system, these aspects are common in real-world systems, making this sytem perfect to test the proposed algorithm.

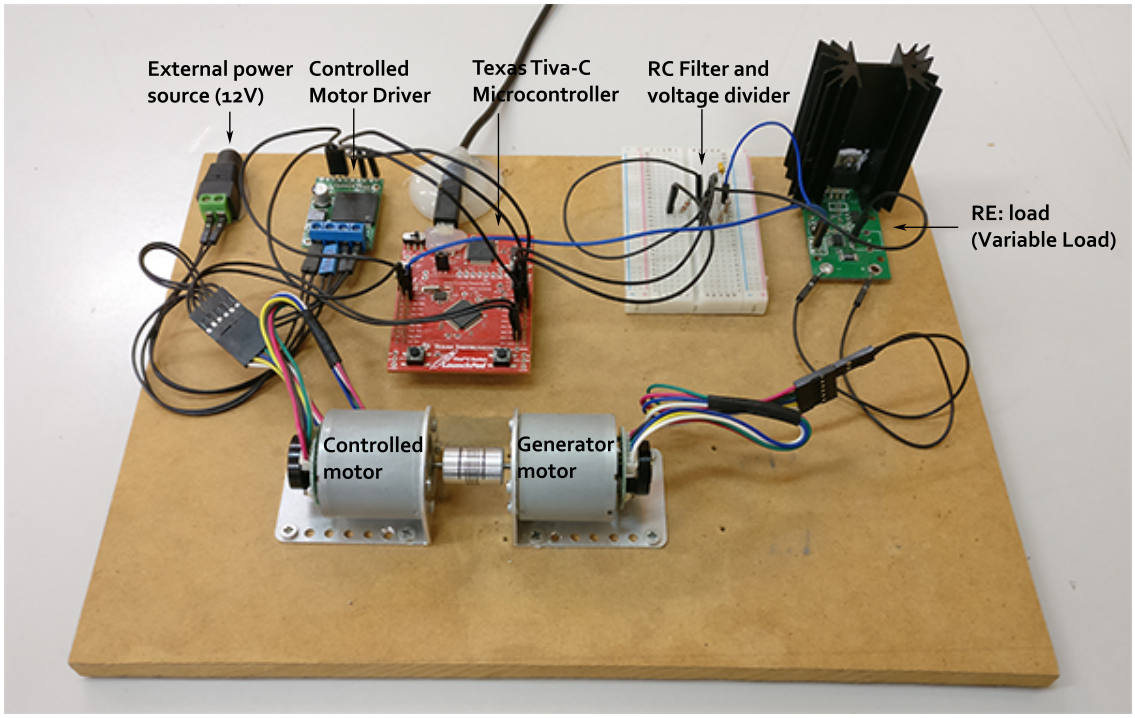


Figure 4.1: Representation of the Self-Evolving Block

The proposed algorithm needs two values the inputs range, the reference and output, the command range and, being a real world system, is also important define a sampling time (T_s). The inputs range are equal for the two inputs which are the velocity of the controlled motor and the reference of that velocity, which range value is between 0 and 1200, they aren't precise limits but are close to reality. The command signal has a range between 0 and 4096 (since is used the value given to the μ_C), and has a positive monotonocity, that means an increase of command signal translates in a increase of the velocity, the load has also the same range. Speaking of the sampling time, since the velocity is pulses per 100 ms, makes sense the sampling time (T_s) be 0.1 s. So the used values can be consulted in table 4.3.

Table 4.3: Two coupled DC Motors parameter values

Reference minimum value	0
Reference maximum value	1200
Reference Range	1200
Velocity minimum value	0
Velocity maximum value	1200
Velocity Range	1200
Command minimum value	0
Command maximum value	4096
Command Range	4096
Sampling Time	0.1[s]

4.3 Additional Plants

The first tested plants were more simple than the two previous plants and work as a control group to evaluate the perfomance of the proposed algorithm. This plants were obtained through papers mentioned in the state of art and were annalysed their equations, inputs' range, output range and command range.

The first plant shows "nonlinear behaviour with respect to both the control signal and the variable to be controlled" and a command signal of 0 don't garantee a stationary output [Cara *et al.*, 2011]. The plant can be defined by the equation:

$$y(k+1) = -0.075sin(y(k)) + \frac{u+u^3}{4} \tag{4.4}$$

The reference signal ($r(k)$) and the pant output ($y(k)$) that works as an input to the proposed algorithm have a operation range of [-1, 1] and the command signal has a range of [-1, 1].

The second plant also shows the same tendencies of the first one [Cara *et al.*, 2010] and can be described by the equation:

$$y(k+1) = \frac{1.5y(k+1)y(k)}{1+y^2(k-1)+y^2(k)} + 0.35sin(y(k-1)+y(k)) + 1.2u(k)) \tag{4.5}$$

The reference signal ($r(k)$) and the pant output ($y(k)$) that works as an input to the proposed algorithm have a operation range of [-1, 1] and the command signal has a range of [-1, 1].

4.4 Results with PL only

One strategy of control is create a set of MFs and adapt them along the time. In this section, it will be analysed the results, perfomance and problems of this strategy. It will be used the real world plant and the CSTR plant.

The analysed results will be represented by four images and are divided in:

- First image: The inputs value: in all the cases is the reference and the output of the system to be controlled;
- Second image: The MSE error for each iteration, the MSE error gives a good ideia of the controller's perfomance;
- Third image: The evolution of the consequents, which shows the effects of the adaptation mechanism;
- Four image: The control signal evolution;

The MFs created and adapted are two for each input with centers corresponding to the limits of the inputs range.

4.4.1 CSTR Plant

Using the data about the plant is created initially two trapezoidal MFs for each input that are online adapted, with the vertice of the trapezoid used as "center" of the MF corresponding to the limits of each input. Since the range of the inputs are the same, the centers are 0.06 and 0.13. There are two inputs with two MFs so there will be four rules. The results of the adaptation of this four rules are presented in figure 4.2.

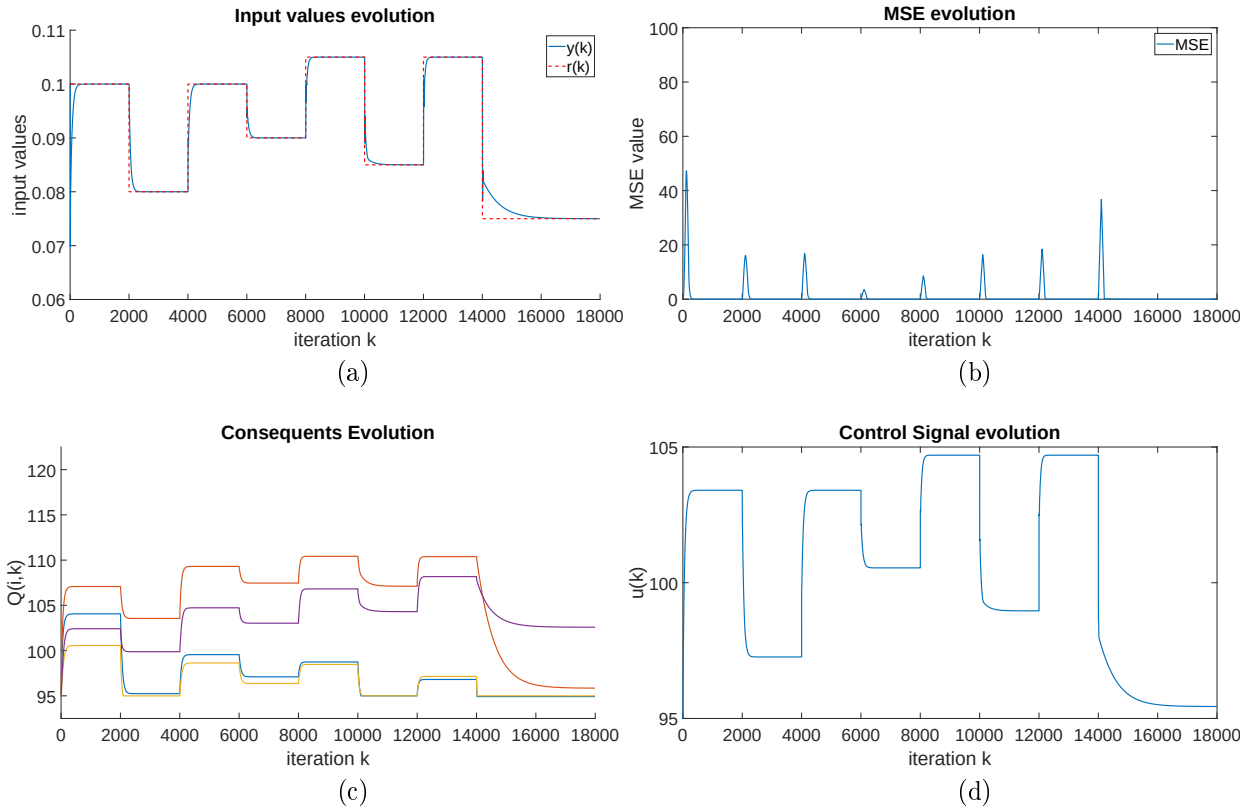


Figure 4.2: Results using only the PL block in the CSTR Plant: the inputs evolution (a), MSE evolution (b), the consequents evolution (c) and the command signal evolution (d).

The input values evolution show a slow reaction and adaptability using only the PL block, due to the fact that there were only four rules. However the system was more stable, as the evolution of the signals shows and also the MSE shows. The MSE has peaks when the reference change reflecting the fact that the existing rules are not yet ready for the new reference value.

The third and four graphics of figure 4.2 show the results of the consequents and command signal and is possible to verified the connection between them. Also, due the fact that there is only four rules and aren't added new rules, all rules are changed through the control process because in every iteration all rules are activated and adapted.

4.4.2 Two Coupled DC Motor

For the Two Coupled DC Motor are also created initially two trapezoidal MFs for each input that are online adapted, with the vertice of the trapezoid used as "center" of the MF corresponding to he limits of each input which are 0 and 1200. There are two inputs with two MFs so there will be 4 rules. The results of the adaptation of this four rules are presented in figure 4.3.

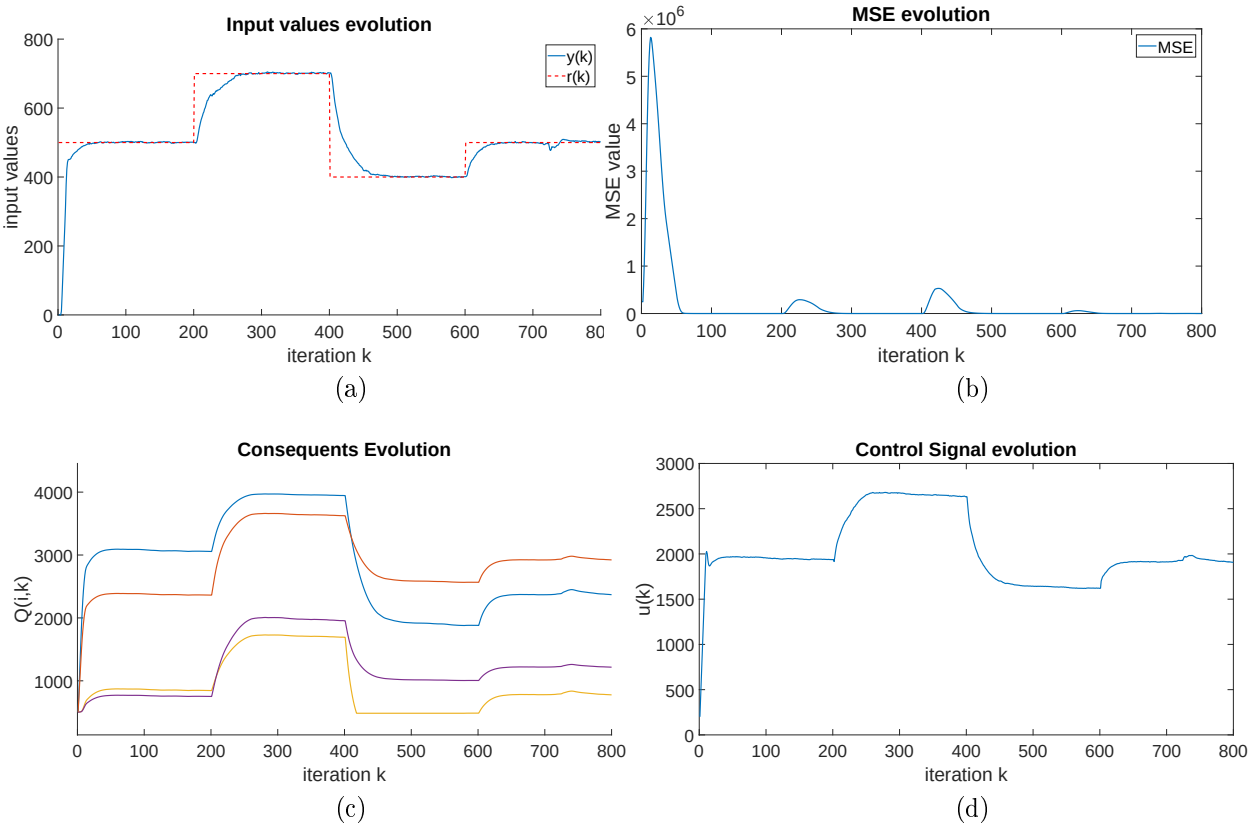


Figure 4.3: Results using only the PL block in the Two coupled DC motor: the inputs evolution (a), MSE evolution (b), the consequents evolution (c) and the command signal evolution (d).

The same conclusions of the CSTR plant control can be written for the real world plant. The figure 4.3 shows that a FCS with only a PL block can control and react to a real world plant, as the response of the controller to non-linear behaviour shows, for example between the iteration 700-750.

4.5 Results with the PL and SE blocks

In this section are presented the results with both the PL and SE blocks activated, in other words, the adaptation and addition of new rules mechanisms activated. The results will be

structured the same way as the previous section, however there will be a additional figure corresponding to the final MF distribution.

4.5.1 CSTR Plant

The initial data is the same as the previous section. The iterations corresponding to moments of the addition of a new rule are signalled by a marker, which also have the value of the new center's location, as figure 4.4 shows.

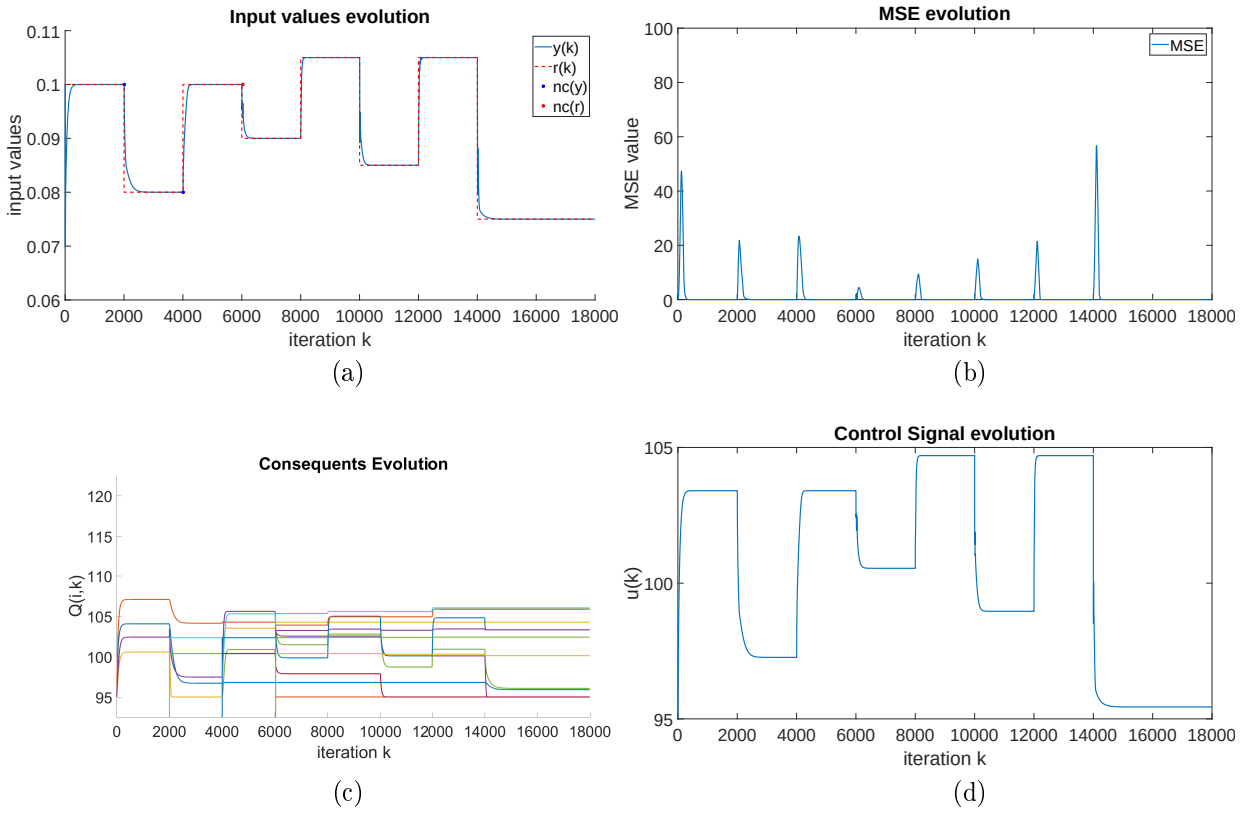


Figure 4.4: Results using the PL block and the SE Block in the CSTR Plant: the inputs evolution (a), MSE evolution (b), the consequents evolution (c) and the command signal evolution (d).

Comparing the results is possible to write two main conclusions related to the inputs and MSE evolution: The first and more important is the comparative quickness of adaptation specially visualized in the last changes in reference, after the all necessary rules are created. The second conclusion is related with the susceptibility to unstable behaviour, since the activated rules are always changing, with the change in the reference values, and they can be descompensated in the initial iterations that they are activated.

The results related to the consequents evolution are also interesting comparing with only a PL block activated. Due to the distribution of MFs, when there are more than two MFs for an input, the consequents in rules with the MFs not activated are constant. Due to

this aspect, contrary to the PL block only, there isn't need of change all the rules, only a maximum of 2^{N_v} rules, for each iteration k. Another aspect is the centers of MFs are more close to the varibales value, this makes a change in the MFs with that centers much more efficient and with faster results.

The MFs distribution is possible to see in figure 4.5 and the iteration and value is present in table 4.4. Is possible to see that the moments corresponding to the creation of new MFs are closely connected with the changes in the reference.

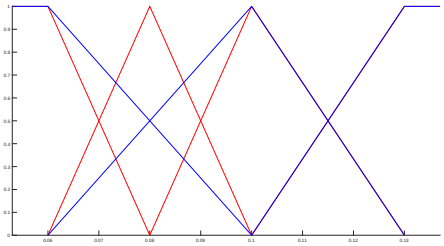


Figure 4.5: MF distribution in the CSTR plan.

Input type	Iteration	Value
Output	2009	0.1
Output	4009	0.08
Reference	6020	0.1
Reference	12013	0.085

Table 4.4: Location of the new centers in the CSTR plan.

4.5.2 Two Coupled DC Motor

The initial data is the same as the previous section. The iterations corresponding to moments of the addition of a new rule are signalled by a marker, which also have the value of the new center's location, as figure 4.6 shows.

The results of the real world plant reinforce the conclusions of the CSTR plant and shows a higher quickness in achieving the established reference signal, however to this quickness is associated an instability.

The MFs distribution is possible to see in figure 4.7 and the iteration and value is present in table 4.5. Is possible to see that the moments corresponding to the creation of new MFs are closely connected with the changes in the inputs.

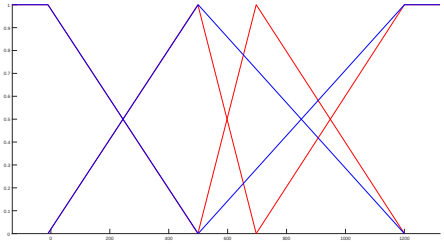


Figure 4.7: MF distribution in the real world system.

Input type	Iteration	Value
Output	86	499
Reference	93	500
Output	280	697

Table 4.5: Location of the new centers in the real world system.

4.6 Results with perturbations

An important aspect to evaluate the perfomance of the algorithm is its ability to respond well to pertubations. The chosen types of pertubations, for the simulated systems, to evaluate the algorithm perfomance were an input and output pertubation. The input pertubation

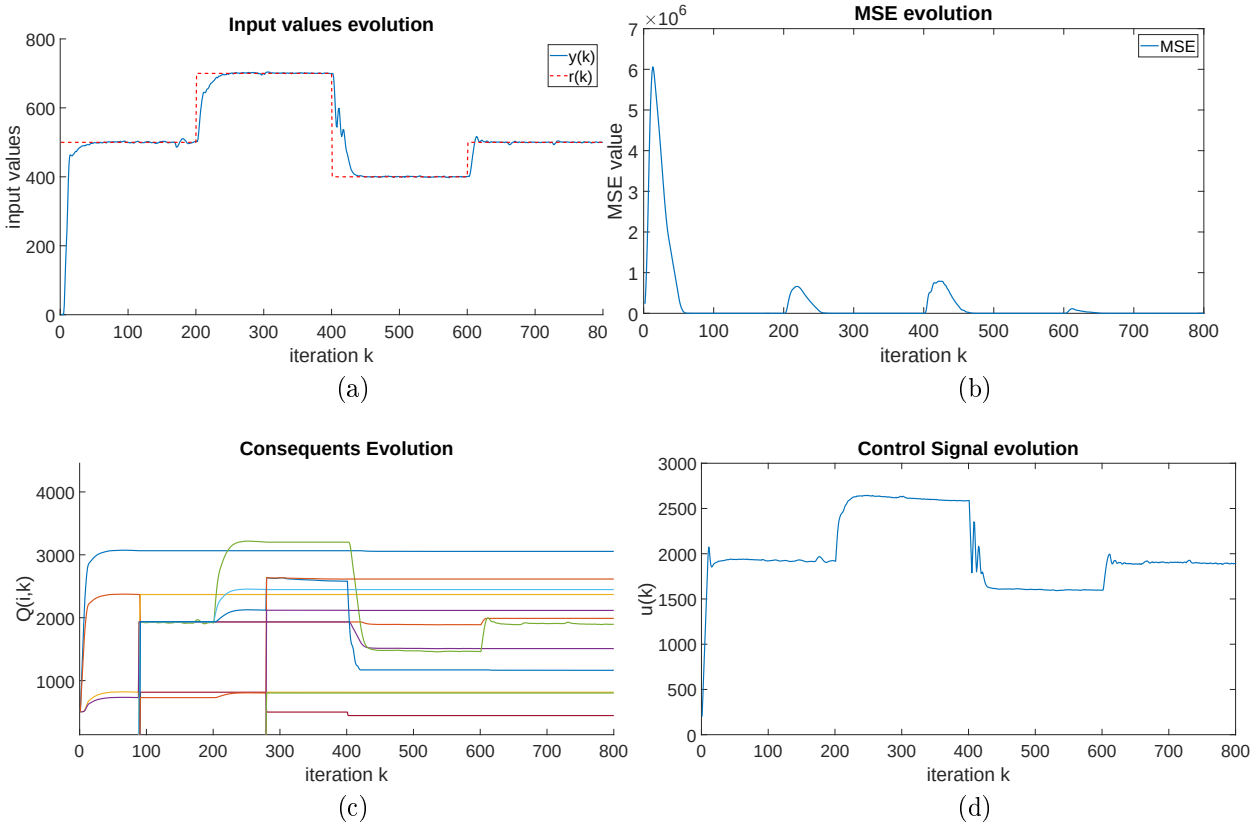


Figure 4.6: Results using the PL block and the SE Block in the Two coupled DC motor: the inputs evolution (a), MSE evolution (b), the consequents evolution (c) and the command signal evolution (d).

corresponds to an addition to the command signal, for that reason is a percentage of the command signal range. The output pertubation is an addition to the value of the ouput, for that reason is a percentage of the output range. Both perturbations are a step-type signal.

In the real world system it was induced a load using the generator motor that works as a pertubation, the range of operation of the load is the same as the command signal given to the u_c.

4.6.1 CSTR Plant

The CSTR is subjected to step input pertubation corresponding to $5\%\Delta u$ in iterarion 5000 and the symmetrical step input pertubation in iteration 7000. The output pertubation is also a step function with the value corresponding to $5\%\Delta y$ in iteration 9000 and the opposite value in iteration 13000.

The results show that the algorithm can react well to pertubations, however with both blocks activated the perfomance of the algorithm response is much more efficient.

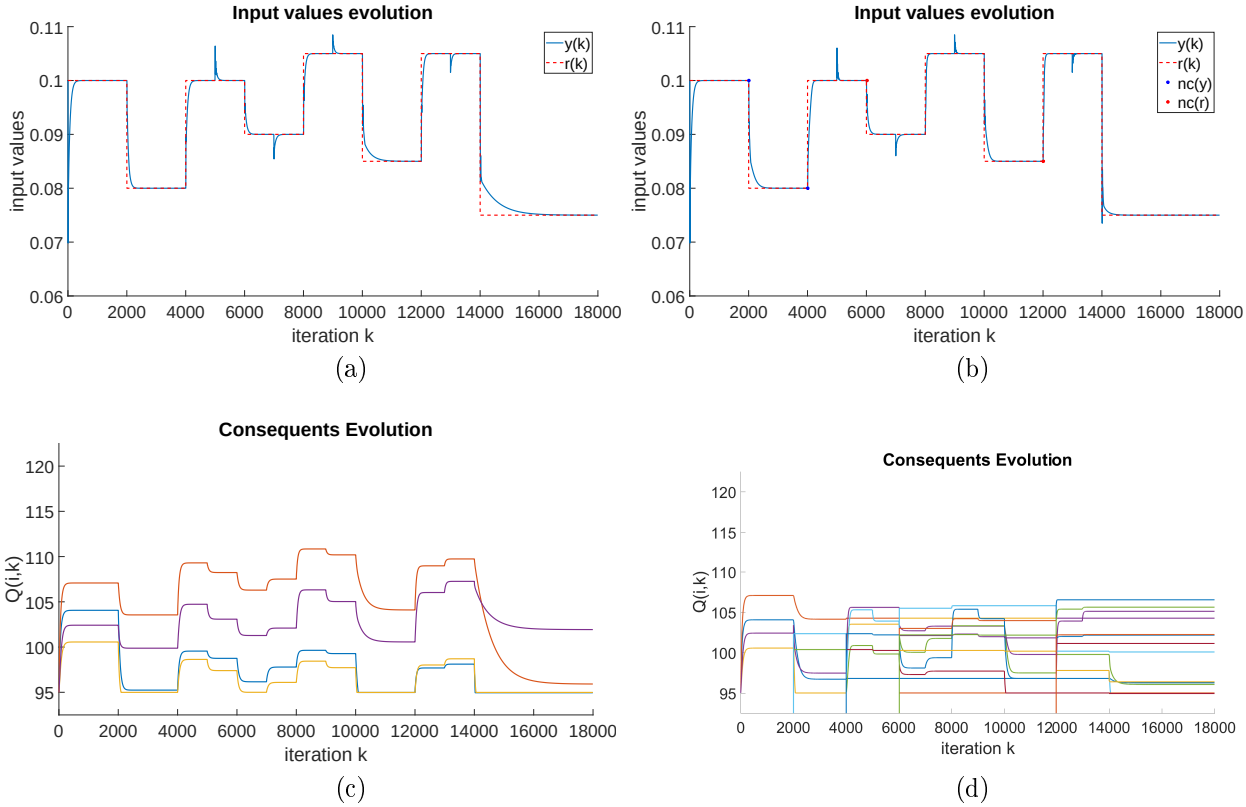


Figure 4.8: Results of the control process when subjected to pertubations using the PL block and the pert Block in the CSTR Plant: the inputs evolution (a), Mpert evolution (b), the consequents evolution (c) and the command signal evolution (d).

4.6.2 Two Coupled DC Motor

As mentioned in section 4.2 there is a variable load which can be configure by the user to simulate an pertubation. This is the only type of pertubation used for this plant, the value used was 700 between the iterations 300 and 500.

The results show the capacity of response of the proposed algorithm and also that with the two blocks activated the responser is quicker and more efficient.

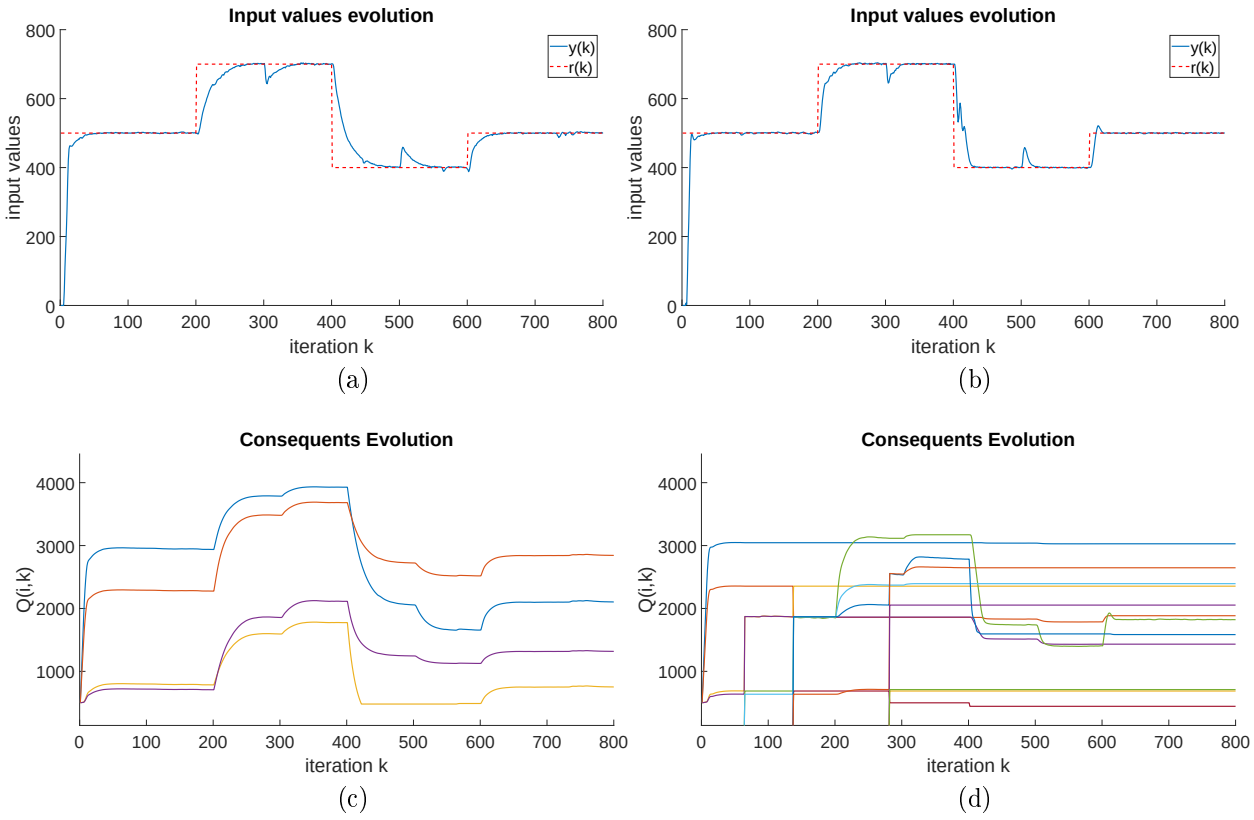


Figure 4.9: Results of the control process when subjected to pertubations using the PL block and the pert Block in the Two-coupled DC motor: the inputs evolution (a), Mpert evolution (b), the consequents evolution (c) and the command signal evolution (d).

4.6.3 Additional plants

As mentioned in section 4.3, there were also tested two additional plants, since the conclusions are similar to the results of the two-coupled DC motor and the CSTR plant, are only showed the results with both SE-block and PL-block activated and with perturbations that are originated with the same method as the CSTR plant. So the two plants are subjected to a step input pertubation corresponding to $5\%\Delta u$ in iterarion 5000 and the symmetrical step input pertubation in iteration 7000. The output pertubation is also a step function with the value corresponding to $5\%\Delta y$ in iteration 9000 and the opposite value in iteration 13000.

The results of the first additional plant can be visualized in Figure 4.10 and the results of the second additional plant can be visualized in Figure 4.11. The results of both test show capacity of the algorithm control several different systems.

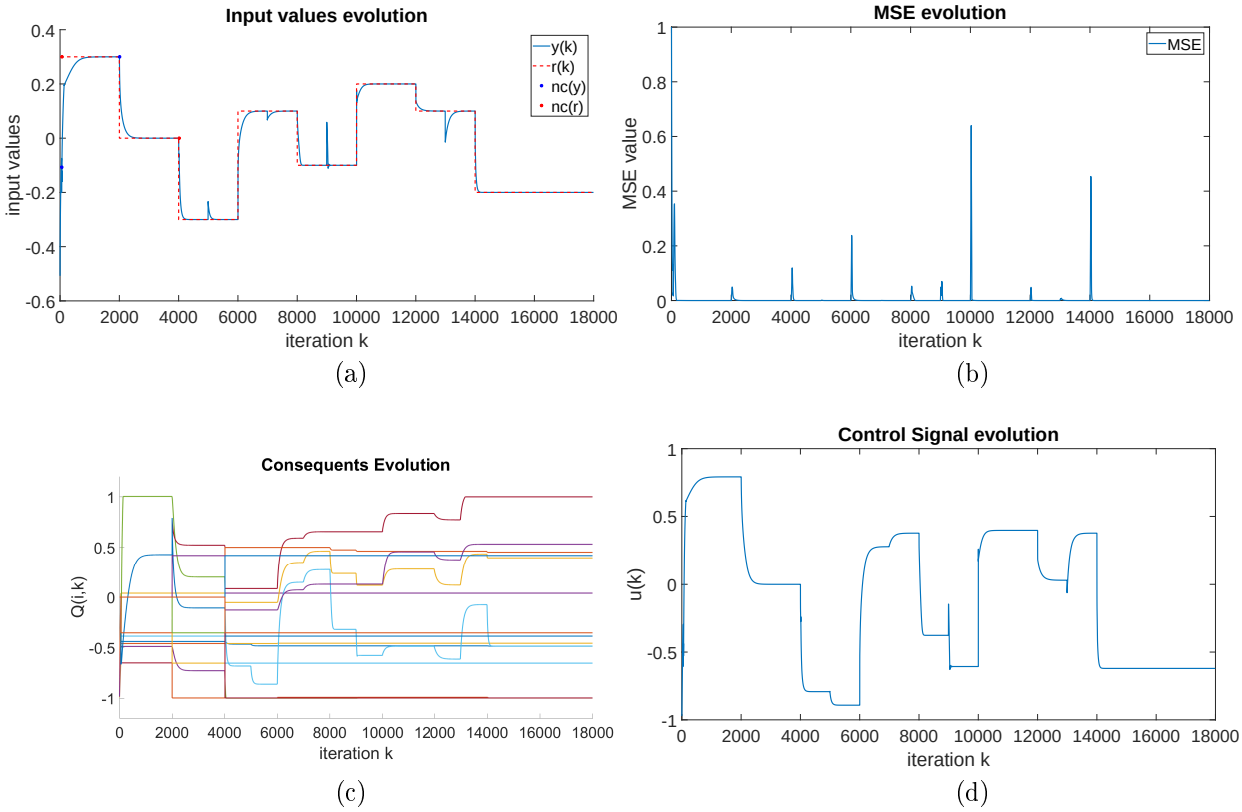


Figure 4.10: Results using the PL block and the SE Block in the first additional plant: the inputs evolution (a), MSE evolution (b), the consequents evolution (c) and the command signal evolution (d).

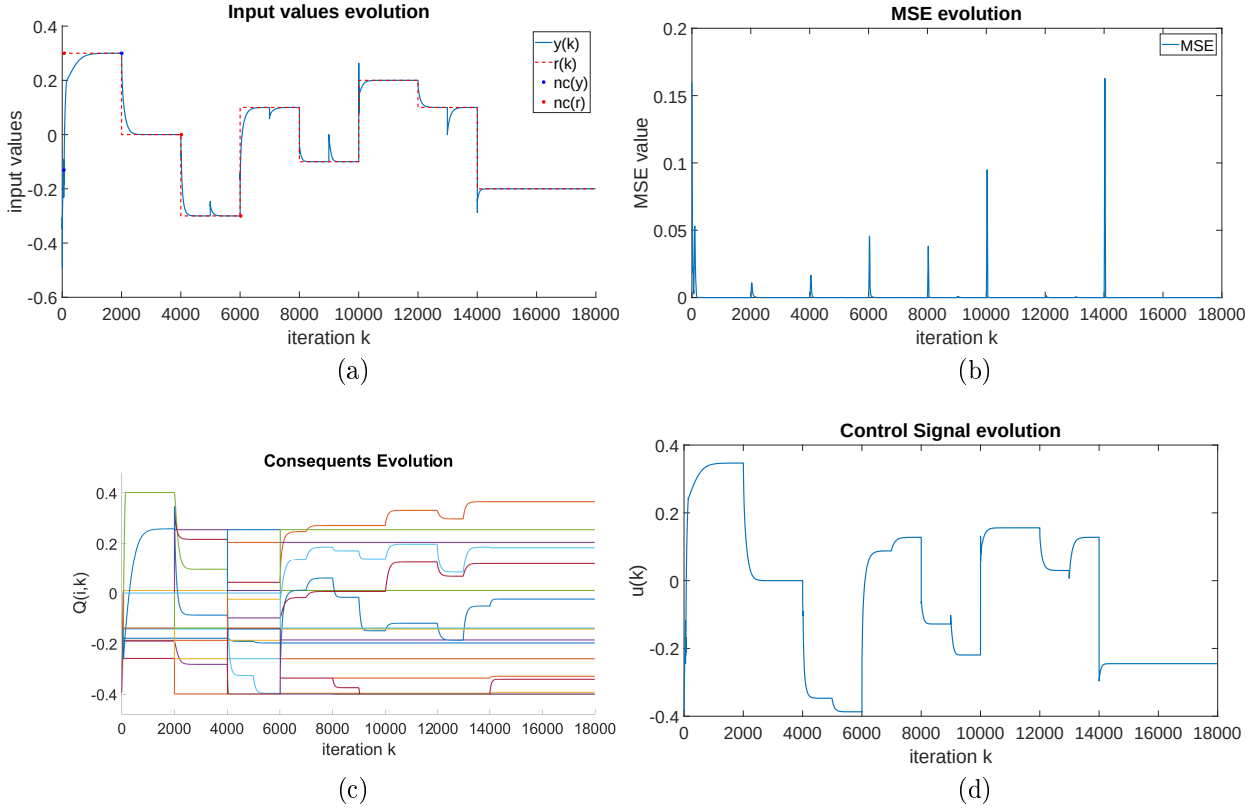


Figure 4.11: Results using the PL block and the SE Block in the second additional plant: the inputs evolution (a), MSE evolution (b), the consequents evolution (c) and the command signal evolution (d).

4.7 Influence of variables

4.7.1 Gain

The Gain is one of the most influential parameters, a high gain makes the adaptation faster at cost of stability has figure 4.12. In fact higher values creates such instability that makes impracticable the use of such values. On the other hand, lower values make the controller slower to react and is also observed instability, specially with both blocks activated.

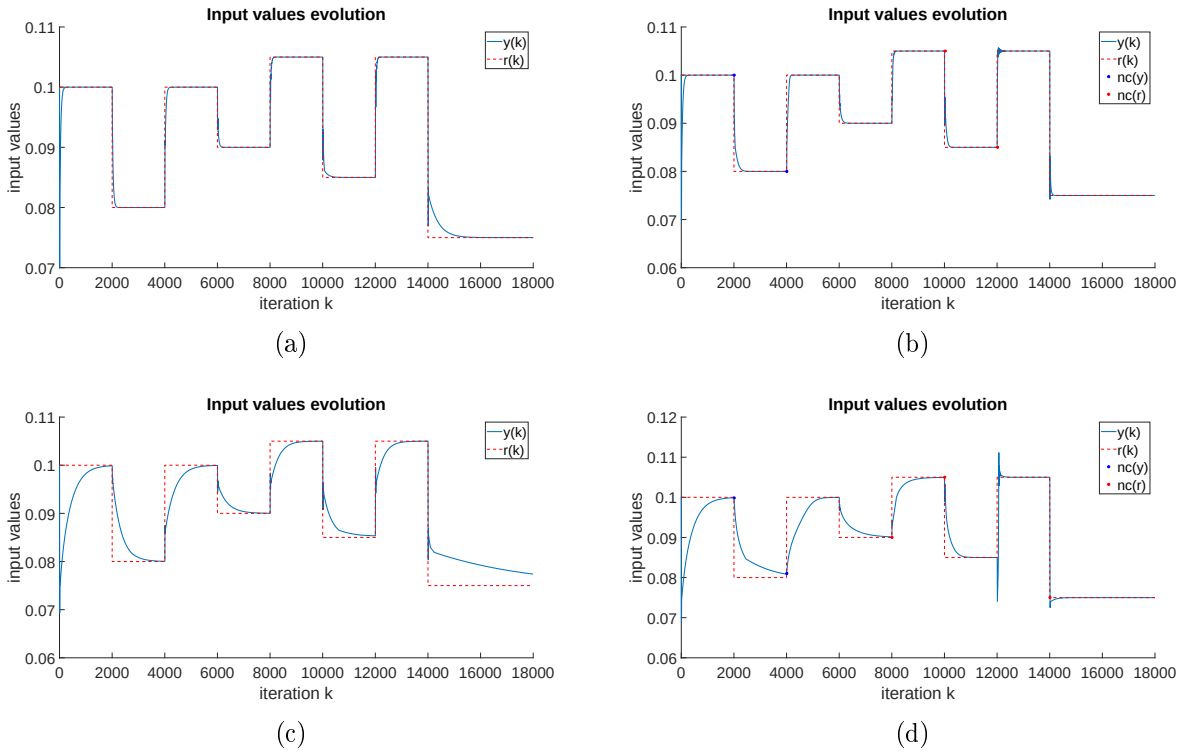


Figure 4.12: Results with a high gain using only the PL block (a) and using both blocks (b) and also with a low gain using only the PL block (c) and using both blocks (d).

Such volatility show one of the main weakness of the algorithm, if the gain is wrongly calculated can create disastrous outcomes. However, there is no need of a precise gain, results show that there is a considerable range of gains that work well, for example the CSTR plant works well with gains from 12 to 50. The default value of the gain is, as mentioned in section 3.3.1, obtained dividing $\frac{|u_{max}-u_{min}|}{|r_{max}-r_{min}|}$ for 15, value used in all results presented in the previous sections.

4.7.2 Memory

The memory parameter is the size of the memory data and influence the following aspects:

- The SE block can only be activated after the memory data is fullfill, for example, if the parameter is 100, the SE block is only activated 100 iterations after the beginning

or after a new change in the structure.

- When a new MF is created the center is obtained by the error distribution in the memory, so the memory parameter defines the number of possible choices.
- The memory parameter have a large influence in the computation cost of the algorithm.

Figure 4.13 shows the mentioned aspects, is possible to see that the MSE evolution is also different, which makes sense because of the MSE is calculated using the memory data and with a larger memory the influence of iterations with considerable error is extended.

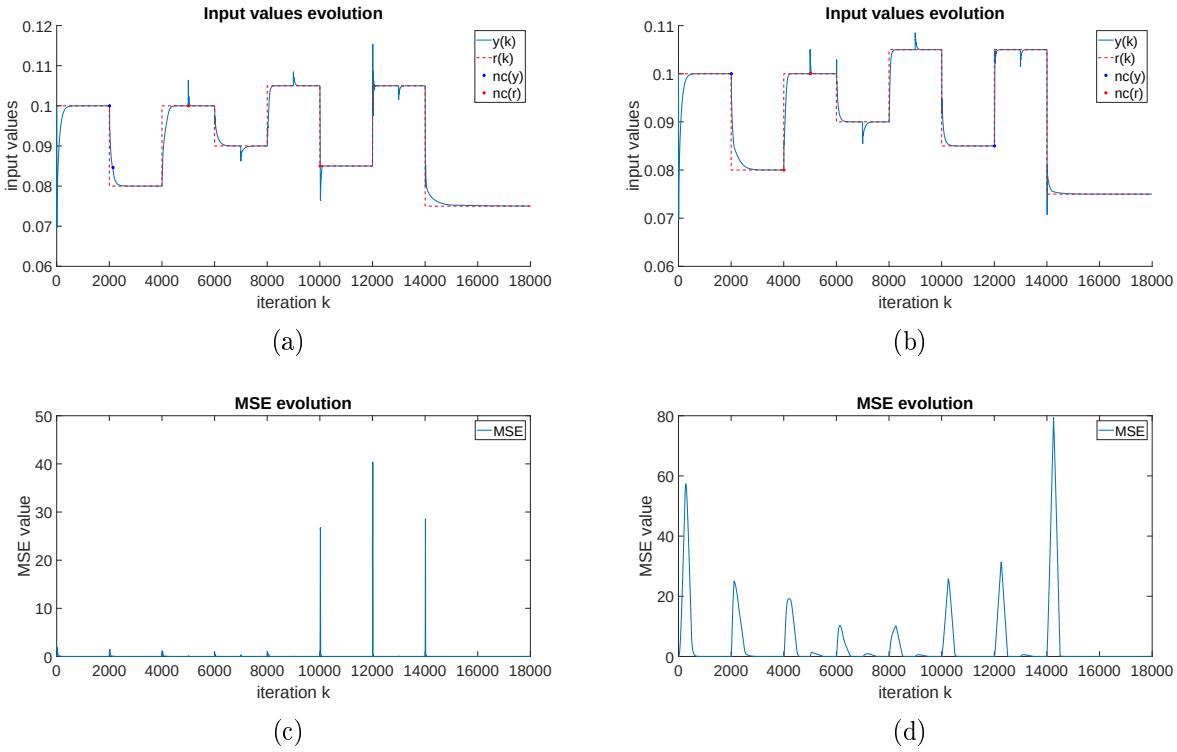


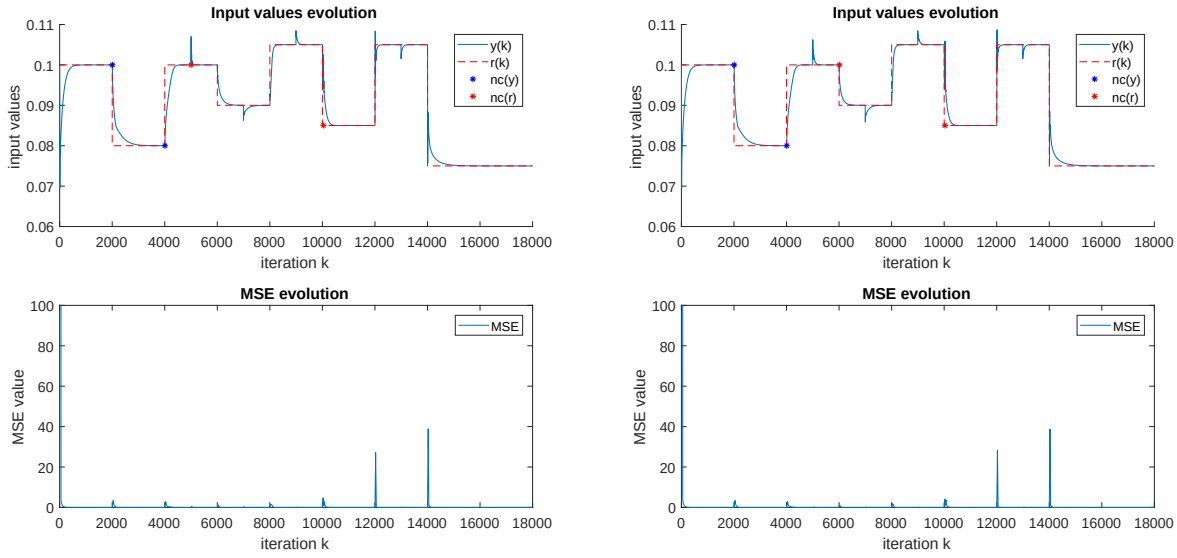
Figure 4.13: Results with a low value of memory size, specifically, the input evolution (a) and the MSE evolution (c) and with a high value of memory size, specifically, the input evolution (b) and the MSE evolution (d).

The effect of the size of memory can be really useful when well manipulated, for example if the tested plant is subjected to a lot of quick changes, a large value can have bad effects, because the larger this parameter the longer the influence of the errors will persist in the following iterations, on other hand, a small value could make anomalies, resulted by non-linear behaviour, much more impactful then if it was used a large value for memory, because a larger value softens the influence of this anomalies.

4.7.3 Thresholds

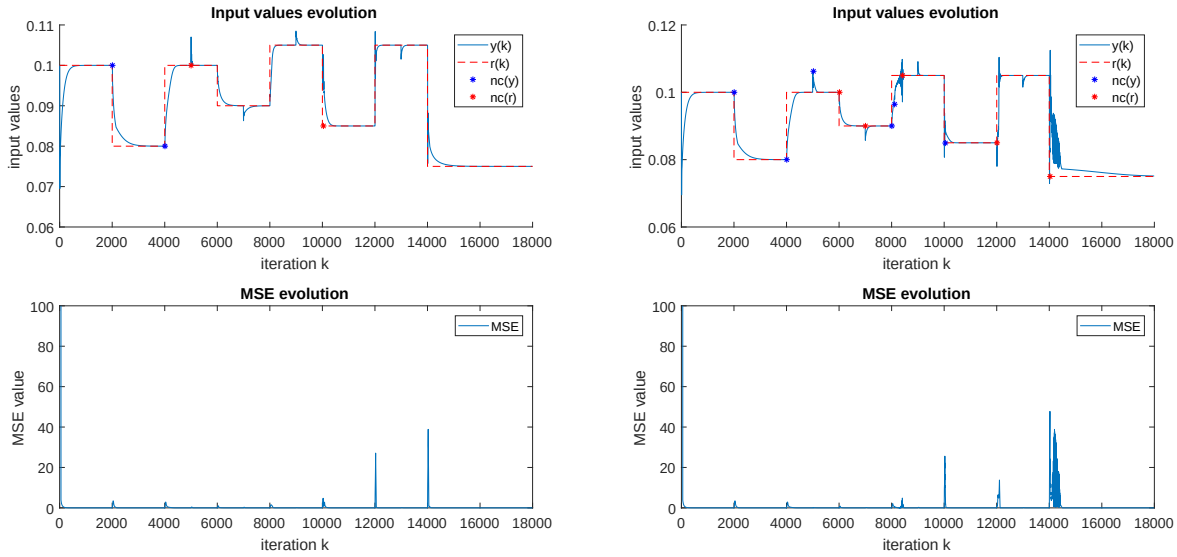
In chapter 3 is mentioned the use of two thresholds in the SE block, the first is used in the entry condition that can activate the block and the second is used in the condition to verify

the proximity of the new center to existing centers, as figure 3.3 shows. Both thresholds are percentages multiply by the range of the variables related.



(a) (b)

Figure 4.14: Results with the default value of the first threshold (a) and with an insignificant value of the first threshold (b).



(a) (b)

Figure 4.15: Results with the default value of the second threshold (a) and with an insignificant value of the second threshold (b).

The value of the first threshold guarantee that when the error is small and the controller is doing well there isn't need of a new MF. As figure 4.14 shows, this threshold prevents the

creation of rules when the error is small. The default value of the threshold used was 5% of the error range.

The second threshold is one of the most important conditions of the algorithm because it assures some stability in the structure and prevents the creation of rules close to others. Is important to remind that with more rules the controller have more computational cost and a large number of rules creates instability, as Figure 4.15b show. The default threshold is 10% of each input range.

4.7.4 Initialization of new rules

As mentioned in section 3.2.2 there are three methods to calculated the consequents of the new rules, the results of each method is presented in figure 4.16. It's possible to see the influence of a wrong initialization in the figure 4.16a, where the new rules' consequents are initialized with the minimum value of the consequent possible and this aspect makes the controller have a break in performance and consequently creates instability. The other two methods show similar results, but the initialization using the formulae in [Mendes *et al.*, 2014] is, comparably, softer, for that reason is the default method.

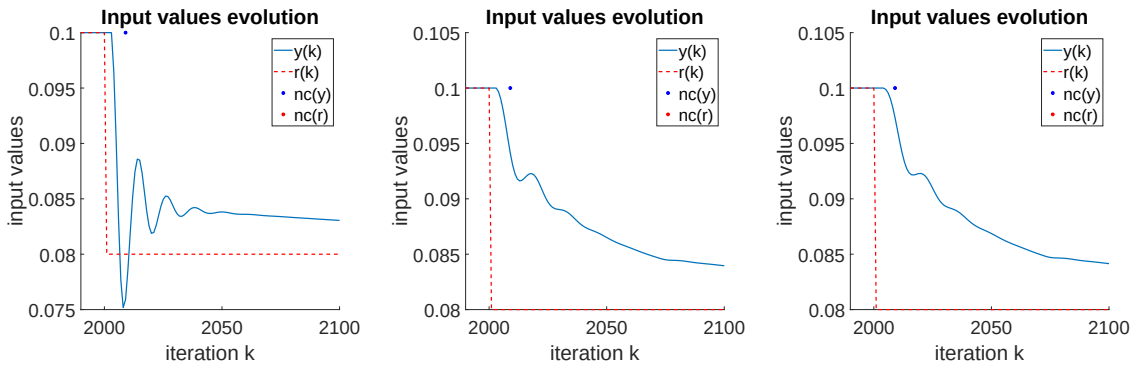


Figure 4.16: Results where the initialization of the consequents of the new rules is obtained using: the minimum consequent (a), the previous iteration command signal (b) and the equation in [Mendes *et al.*, 2014] (c).

Chapter 5

Conclusion

This dissertation propose an evolving Fuzzy Control algorithm with learning capabilities that is able to control an industrial process without knowledge of its structure or behaviour. The algorithm is constituted by two parts, an adaptive, that change existing rules, and an evolving, that change the structure, creating new rules. The learning process happens using the I/O data obtained while controlling the system, for this reason the algorithm is considered an online algorithm. Is possible to say that the objectives proposed in this dissertation are fulfilled.

In order to validate the adaptation to different type of systems, the algorithm was used in four different systems, and in all of them was able to adapt and control. It was also subjected to perturbations and non-linear behaviour, specially in a real-world plant. The results are satisfying, however it's noticed that some parameters can have a huge impact in the performance of the algorithm, and in some cases is necessary to regulated this parameters, specially the gain parameter, responsible, if too high, for creating instability and, if too low, for a slow adaptation. So if a user is planning to use the proposed algorithm, should be careful with the choice of the gain parameter and change it if the results are not being satisfactory.

There also other parameters that the algorithm allow the user to change, however their impact in the controller performance are not so strong as the gain. The user can change the Memory size, the influence of this parameter is described in section 4.7.2 and, summarizing, the value determine how long the influence of errors persist in the algorithm, for example if the plant has a lot of changes in the reference values the memory value should be lower to make sure that the errors don't influence the algorithm for too long, another example is when the plant have some frequent anomalies, due to non-linear behaviour, is advised to have a higher value of Memory so that the influence of the anomalies can be soften.

Other parameter are the thresholds, there are two main thresholds that the user can manipulated, both are related to conditions that determine the addition of new rules. The values manipulated by the user are percentages to be adaptable to different systems. The first threshold defines the minimum value of error to add new rules, this prevents changes

in the structures when the controller is being already efficient. The second threshold defines the minimum distance between the centers of the MFs, and basically works as a safeguard to the creation of a large number of MFs, and subsequently, rules.

For future work is suggested methods to manipulated and study of a better gain for a specific plant, the value used in this dissertation works to all tested plants, however for each plant the best results are obtained with different gains, so the creation of methods to calculate the gain for a specific plant would improve the proposed algorithm.

It would also be interesting test the algorithm in other type of plants, like plants whose monotony relation between the command signal and output change along the time, Multiple Inputs Multiple Outputs (MIMO) systems would also be interesting and, of course, more real-world plants.

Another interesting work would be the use of the ANYA FRB, that contrary the proposed algorithm that use fuzzy sets to define the antecedent, the method ANYA uses clouds, and the inputs values have a degree of membership to each existing cloud. This, relatively, recent proposed method of Fuzzy Rules as yet to be widely used in the scientifically community and algorithms using it could be a fine contribution to the research field of evolving Fuzzy Control Algorithms, the implementation of the [Sadeghi-Tehran *et al.*, 2012] algorithm could be a good start to achieve this goal.

Bibliography

[Angelov, 2004] Plamen Angelov. A fuzzy controller with evolving structure. *Information Sciences*, vol. 161, no. 1-2, pp. 21–35, 2004. (Citado na página 4).

[Bellman and Zadeh, 1970] Richard E Bellman and Lotfi Asker Zadeh. Decision-making in a fuzzy environment. *Management science*, vol. 17, no. 4, pp. B–141, 1970. (Citado na página 2).

[Cara *et al.*, 2010] Ana Belén Cara, Héctor Pomares, Ignacio Rojas, Zsófia Lendek, and Robert Babuška. Online self-evolving fuzzy controller with global learning capabilities. *Evolving Systems*, vol. 1, no. 4, pp. 225–239, 2010. (Citado nas páginas 4, 16, 19, 20, 21, 24, 25, 26, 27, e 33).

[Cara *et al.*, 2011] Ana Belén Cara, Héctor Pomares, and Ignacio Rojas. A new methodology for the online adaptation of fuzzy self-structuring controllers. *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 3, pp. 449–464, 2011. (Citado na página 33).

[Chen *et al.*, 2009] Cheng-Hung Chen, Cheng-Jian Lin, and Chin-Teng Lin. Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 4, pp. 459–473, 2009. (Citado na página 3).

[Gao and Er, 2003] Yang Gao and Meng Joo Er. Online adaptive fuzzy neural identification and control of a class of MIMO nonlinear systems. *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 4, pp. 462–477, 2003. (Citado na página 3).

[Hoffmann and Nelles, 2001] Frank Hoffmann and Oliver Nelles. Genetic programming for model selection of TSK-fuzzy systems. *Information Sciences*, vol. 136, no. 1-4, pp. 7–28, 2001. (Citado na página 3).

[Lasi *et al.*, 2014] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 2014. (Citado na página 1).

[Li and Lee, 2003] Chunshien Li and Chun-Yi Lee. Self-organizing neuro-fuzzy system for control of unknown plants. *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 1, pp. 135–150, 2003. (Citado na página 3).

[Lin and Xu, 2006] Cheng-Jian Lin and Yong-Ji Xu. A novel genetic reinforcement learning for nonlinear fuzzy control problems. *Neurocomputing*, vol. 69, no. 16-18, pp. 2078–2089, 2006. (Citado na página 3).

[Lin *et al.*, 1995] Chin-Teng Lin, Cheng-Jian Lin, and CS George Lee. Fuzzy adaptive learning control network with on-line neural learning. *fuzzy sets and systems*, vol. 71, no. 1, pp. 25–45, 1995. (Citado na página 4).

[Mamdani and Assilian, 1975] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man Mach. Studies*, vol. 7, p. 1–13, January 1975. (Citado nas páginas 2, e 3).

[Mendes *et al.*, 2014] Jérôme Mendes, Rui Araújo, Tiago Matias, Ricardo Seco, and Carlos Belchior. Evolutionary learning of a fuzzy controller for industrial processes. In: *Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE*, pp. 139–145. IEEE, 2014. (Citado nas páginas xii, 24, 25, 27, e 46).

[Mendes *et al.*, 2017] Jérôme Mendes, Luís Osório, and Rui Araújo. Self-tuning PID controllers in pursuit of plug and play capacity. *Control Engineering Practice*, vol. 69, pp. 73–84, 2017. (Citado na página 31).

[Mingzhi *et al.*, 2009] Huang Mingzhi, Wan Jinquan, Ma Yongwen, Wang Yan, Li Weijiang, and Sun Xiaofei. Control rules of aeration in a submerged biofilm wastewater treatment process using fuzzy neural networks. *Expert Systems with Applications*, vol. 36, no. 7, pp. 10428–10437, 2009. (Citado na página 3).

[Morningred *et al.*, 1992] J Duane Morningred, Bradley E Paden, Dale E Seborg, and Duncan A Mellichamp. An adaptive nonlinear predictive controller. *Chemical Engineering Science*, vol. 47, no. 4, pp. 755–762, 1992. (Citado nas páginas 29, e 30).

[Mucientes and Casillas, 2007] Manuel Mucientes and Jorge Casillas. Quick design of fuzzy controllers with good interpretability in mobile robotics. *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 4, pp. 636–651, 2007. (Citado na página 3).

[Park *et al.*, 2005] Jang-Hyun Park, Gwi-Tae Park, Seong-Hwan Kim, and Chae-Joo Moon. Direct adaptive self-structuring fuzzy controller for nonaffine nonlinear system. *fuzzy sets and systems*, vol. 153, no. 3, pp. 429–445, 2005. (Citado na página 3).

[Phan and Gale, 2008] Phi Anh Phan and Timothy J Gale. Direct adaptive fuzzy control with a self-structuring algorithm. *fuzzy sets and Systems*, vol. 159, no. 8, pp. 871–899, 2008. (Citado nas páginas 3, e 20).

[Pomares *et al.*, 2002] Héctor Pomares, Ignacio Rojas, Jesús González, Fernando Rojas, Miguel Damas, and FJ Fernández. A two-stage approach to self-learning direct fuzzy

controllers. *International Journal of Approximate Reasoning*, vol. 29, no. 3, pp. 267–289, 2002. (Citado nas páginas 4, e 23).

[Rojas *et al.*, 2006] Ignacio Rojas, Héctor Pomares, Jesús Gonzalez, Luis Javier Herrera, Alberto Guillén, F Rojas, and Olga Valenzuela. Adaptive fuzzy controller: Application to the control of the temperature of a dynamic room in real time. *fuzzy sets and Systems*, vol. 157, no. 16, pp. 2241–2258, 2006. (Citado nas páginas 4, e 20).

[Sadeghi-Tehran *et al.*, 2012] Pouria Sadeghi-Tehran, Ana Belén Cara, Plamen Angelov, Héctor Pomares, Ignacio Rojas, and Alberto Prieto. Self-evolving parameter-free rule-based controller. In: *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pp. 1–8. IEEE, 2012. (Citado nas páginas 4, 19, e 48).

[Sobhy and Khedr, 2015] Sameh Mohamed Sobhy and Wael Mohamed Khedr. Developing of fuzzy logic controller for air condition system. *International Journal of Computer Applications*, vol. 126, no. 15, 2015. (Citado na página 8).

[Wang *et al.*, 2008a] Di Wang, Xiao-Jun Zeng, and John A Keane. An incremental construction learning algorithm for identification of TS Fuzzy Systems. In: *Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on*, pp. 1660–1666. IEEE, 2008. (Citado nas páginas 3, e 18).

[Wang, 1999] Li-Xin Wang. *A course in fuzzy systems*. Prentice-Hall press, USA, 1999. (Citado nas páginas 1, 3, 8, e 10).

[Wang *et al.*, 2008b] Wei-Yen Wang, Yi-Hsing Chien, and I-Hsum Li. An On-Line Robust and Adaptive TS Fuzzy-Neural Controller for More General Unknown Systems. *International Journal of Fuzzy Systems*, vol. 10, no. 1, 2008. (Citado na página 4).

[Ying, 2000] Hao Ying. *Fuzzy control and modeling: analytical foundations and applications*. Wiley-IEEE Press, 2000. (Citado na página 7).

[Zadeh, 1965] L. A. Zadeh. fuzzy sets. *Information and Control*, vol. 8, pp. 338–353, June 1965. (Citado nas páginas 2, 7, e 8).

[Zadeh, 1973] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision process. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 28–44, January 1973. (Citado na página 2).

[Zadeh, 1971] Lotfi A Zadeh. Similarity relations and fuzzy orderings. *Information sciences*, vol. 3, no. 2, pp. 177–200, 1971. (Citado na página 2).