DECSAI Department of Computer Science
and Artificial Intelligence

A Practical Study on the Implementation of Fuzzy Logic Controllers

O. Cordón, F. Herrera, A. Peregrín

Technical Report #DECSAI-98107
July, 1998

Available at URL: http://decsai.ugr.es/~ocordon/public.html
ftp site: ftp://decsai.ugr.es/pub/arai/tech_rep/ga_fl/tr-98107.ps.Z

ETS de Ingeniería Informática. Universidad de Granada.
18071 Granada - Spain - Phone +34.58.244019, Fax +34.58.243317

A Practical Study on the Implementation of Fuzzy Logic Controllers

Summary

A PRACTICAL STUDY ON THE IMPLEMENTATION OF FUZZY LOGIC CONTROLLERS*

OSCAR CORDÓN, FRANCISCO HERRERA

Departament of Computer Sciences and Artificial Intelligence
Escuela Técnica Superior de Ingeniería Informática. University of Granada. 18071 – Granada – Spain
e-mail: ocordon, herrera@decsai.ugr.es

ANTONIO PEREGRÍN

Departament of Electronic Engineering, Computer Systems and Automatics
Escuela Politécnica Superior de La Rábida. University of Huelva. 21819 – Huelva – Spain
e-mail: peregrin@uhu.es

Since the end of seventies, Fuzzy Logic Controllers (FLCs) have enjoyed a good place in intelligent and automatic control systems, mainly though their good practical results. In this paper, we describe the basis of fuzzy control and we cautiously study practical software implementations of FLCs that can be easily incorporated into real systems. We study the implementation of a practice Mono-operator FLC and compare two methods, the Exact and Approximate ones, for an advanced Multi-operator FLC that allows us to choose from several fuzzy operators in order to select the one best adapted to a specific application.

Keywords : fuzzy logic controller, implementing fuzzy logic controllers, FLC application, fuzzy operators.

1. Introduction

Fuzzy logic based systems are suitable for engineering because their inputs and outputs are real-valued variables, mapped with a non-linear function. Fuzzy logic based systems, when used in control, receive the name of Fuzzy Logic Controllers (FLCs).[19,20] FLCs achieve an alternative for those applications where classical control strategies do not achieve good results. In many cases these systems have two characteristics: the need for human operator experience, and a strong non linearity, where it is not possible to obtain a mathematical model.

Usually, FLCs are not used in those problems where another classical control strategy, a direct digital or computed response are applicable. FLCs are suitable to use where there is a lot of information that is difficult to handle, a low precision environment and vague information. The summary conclusion about FLC preferred application areas are:

- non linearity systems,
- systems with no predictable disturbance, or low accurate sensors, and
- systems where it is necessary to incorporate human experience.

FLCs had to wait for the low cost of microprocessor technology and modern software development to give them promotion in the industrial processes and consumer products world. At the present time, there are a lot of real-world applications of FLCs like intelligent suspension systems, mobile robot navigation, wind energy converter control, air conditioning controllers, video and photograph camera autofocus and imaging stabilizer, anti-sway control for cranes, and a lot of industrial automation applications.[14]

This paper deals on how to implement FLCs in practice, covering the zone that exists between the theoretical studies about different fuzzy operators and the works that describe the results in particular applications. Usually, neither of them explain the way used to implement FLC. We will cover methods for big computers with high level languages interpreters and compilers as well as for small microcontrollers commonly used in industry, showing the typical different options employed in practice. This work is aimed at people who already know some programming language and want to construct FLCs.

We begin in Section 2 condensing the FLC principles and making recommendations about its design. In Section 3, the design questions will be shown, presenting several selected characteristic fuzzy operators and some criteria for choosing them. Section

*This research has been supported by CICYT TIC96-0778 and TIC96-1393-C06-04

4 presents the framework for implementing FLCs. Then, in Section 5 and 6 we will discuss software implementations, data structures, algorithms and so on, for a specific practical FLC and two software implementations of controllers with a large number of fuzzy operators available. The comparative study of these two methods will be performed in Section 7. Finally, Section 8 presents the concluding remarks. Besides, we add three Appendixes, the first one dedicated to the graphical representation of the selected fuzzy implication operators, and the other describing the applications used for the comparative study and results.

2. Fuzzy Logic Controllers

The world of automatic control has taken advantage of the flow in digital and computer technology in two main ways: sequence controllers and analog device controllers. Sequence controllers are implemented with a little theory, initially based on relays and pneumatic logic. Customarily, analog devices were controlled by analog controllers produced using mathematics as the main tool.

Expert Control is a field of Artificial Intelligence that has become a research topic in the domain of process control, with its purpose being to avoid the drawbacks mentioned in the introduction with respect to classical control strategies. Fuzzy Logic Control is one of the topics within Expert Control.

FLCs, as initiated by Mamdani and Assilian[19,20], are now considered as one of the most important applications of Fuzzy Set Theory proposed by Zadeh[32] in 1965. This theory is based on the notion of fuzzy sets as a generalization of the ordinary set characterized by a membership function μ that takes values in the interval [0,1] representing degrees of membership to the set. FLCs typically define a non-linear mapping from the system's state space to the control space. Thus, it is possible to consider the results of an FLC as a non-linear control surface reflecting the process of the operator's prior knowledge.

Figure 1 shows the generic structure of an FLC. An FLC is a kind of Fuzzy Rule Based System which is composed of a *Knowledge Base* that comprises the information used by the expert operator in the form of linguistic control rules, a *Fuzzification Interface*, that transforms the crisp values of the input variables into fuzzy sets that will be used in the fuzzy inference process, an *Inference System* that uses the fuzzy values from the Fuzzification Interface and the information from the Knowledge Base and performs the reasoning process, and the *Defuzzification Interface*, which takes the fuzzy action from the inference process and translates it into crisp values for the control variables.

The Knowledge Base is comprised by two components: the *Data Base* and the *Rule Base*. The Data Base contains the definitions of the linguistic labels, that is, the membership functions for the fuzzy sets. The *Rule Base* is a collection of fuzzy control rules representing the expert knowledge from the controlled system.

The rules of the ***Knowledge Base*** are conditional statements of the type

IF antecedent *THEN* consequent

connected by the *also* connective that it is modeled by an operator G.

There are different types of rules regarding the expression of the consequent:

- Mamdani rules[19,20], where the consequent is a linguistic variable as the ones in the antecedent:

  $$IF\ x_1\ is\ A_1\ and \ldots and\ x_n\ is\ A_n\ THEN\ y\ is\ B$$

  where the x_i are the input or state variables and y_i is the output or control variable. A_i and B are linguistic labels associated with fuzzy concepts, that is, linguistic terms related to the input and output variables respectively. In this paper, we will consider a Rule Base constituted by Mamdani type fuzzy control rules.

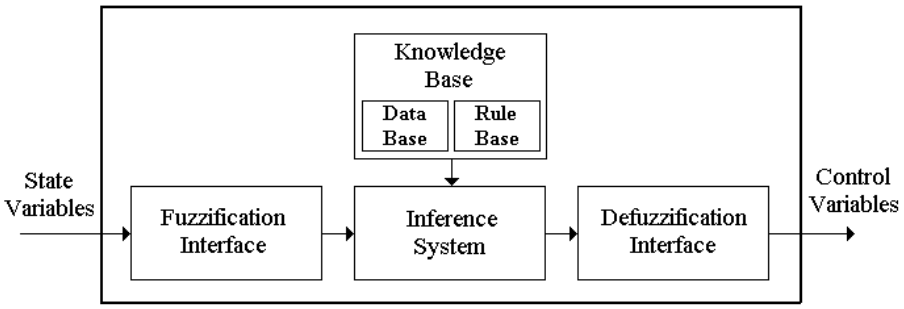

Fig. 1. Generic structure of an FLC.

- TSK rules[25], where the consequent of the rule is a linear function of the inputs:

$$IF\ x_1\ is\ A_1\ and\ ...\ and\ x_n\ is\ A_n\ THEN\ y = p_0 + p_1X_1 + ... + p_nX_n$$

or, more generally, any function of the input parameters:

$$IF\ x_1\ is\ A_1\ and\ ...\ and\ x_n\ is\ A_n\ THEN\ y = f(X_1,...,X_n)$$

The ***Fuzzification Interface*** establishes an application between each precise value of the input variable and a fuzzy set defined in the universe of the corresponding variable. Then, the Fuzzification Interface works as follows:

$$A' = F(x_0) \quad (2.1)$$

where x_0 is a precise value defined in **U**, A' is a fuzzy set defined on the same universe **U** and F is a fuzzifier operator.

There are two possibilities for selecting F:

1. *Singleton Fuzzification:* A' is built like a singleton fuzzy set with support x_0, that is, with the following membership function:

$$A'(x) = \begin{cases} 1, & if \quad x = x_0 \\ 0, & otherwise \end{cases} \quad (2.2)$$

2. *Non-Singleton Fuzzification or Approximate Fuzzification.* In this case, when $x = x_0$, $F(x_0) = 1$, and the membership of the rest of the values for **U** decrease while moving away from x_0.

The first way is the one most used and we will use it always in this paper.

The ***Inference System*** or *Fuzzy Inference Engine* is based on the application of the Generalized Modus Ponens (GMP), an extension of the classical Modus Ponens, proposed by Zadeh[33] as follows:

$$\frac{\begin{array}{l} If\ X\ is\ A\ then\ Y\ is\ B \\ X\ is\ A' \end{array}}{Y\ is\ B'}$$

The fuzzy conditional statement *If X is A then Y is B* (X and Y being linguistic variables, and A and B being fuzzy sets) represents a fuzzy relation between A and B, defined in $\mathbf{U} \times \mathbf{V}$, ($\mathbf{U} = \mathbf{U_1} \times \mathbf{U_2} \times ... \times \mathbf{U_n}$ and **V** being the universes of the input variables $X_1,...,X_n$ and the output Y, respectively). The fuzzy relation is expressed by a fuzzy set R whose membership function $\mu_R(x,y)$ is given by:

$$\forall x \in \mathbf{U}, y \in \mathbf{V}: \mu_R(x,y) = I(\mu_A(x), \mu_B(y)), \quad (2.3)$$

with $\mu_A(x)$ and $\mu_B(y)$ being the membership functions of the fuzzy sets A and B, respectively, and I a fuzzy operator modeling the fuzzy relation.

The membership function of the fuzzy set B' in the consequent, obtained from the GMP, is deduced by projection on **V** by means of the Compositional Rule of Inference (CRI) (introduced by Zadeh[33]) given by the following expression:

$$\mu_{B'}(y) = Sup_{x \in U} \{ T'(\mu_{A'}(x), I(\mu_A(x), \mu_B(y))) \} \quad (2.4)$$

where $\mu_{A'}(x) = T(\mu_{A'1}(x),..., \mu_{A'n}(x))$, $\mu_A(x) = T(\mu_{A1}(x),..., \mu_{An}(x))$, with T and T' being t-norms and I an implication operator.

When Singleton Fuzzification is considered, the fuzzy set A' is a singleton, that is, $\mu_{A'}(x) = 1$ *if* $x = x_o$, and $\mu_{A'}(x) = 0$ *if* $x \neq x_o$. Thus, the CRI is reduced to the following expression:

$$\mu_{B'}(y) = I(\mu_A(x_0), \mu_B(y)) \quad (2.5)$$

Hence, it is found that it directly depends on the fuzzy implication operator selected. In the specific literature, it is proposed a huge amount of operators that can be used as implication operators in the fuzzy control inference process. Many studies that add information in order to select this operator have been published.[3,4,6,7,8,,9,12,15,17,21,23,27]

As has been commented, the calculation of $\mu_A(x_0)$ consists of the application of a conjunctive operator on $\mu_{Ai}(x_i)$:

$$\mu_A(x_0) = T(\mu_{A1}(x_1), ..., \mu_{An}(x_n)) \quad (2.6)$$

The result of the application of the T connective operator is commonly called *matching*, and represents the matches between the values presented in the inputs and the fuzzy sets of the rule antecedent. We will note it by h.

The Inference System produces the same amount of output fuzzy sets as the number of rules collected in the Knowledge Base. These groups of fuzzy sets must be transformed into crisp values for the control variables. This is the goal of the ***Defuzzification Interface.*** We denote by B_i' the fuzzy set obtained as output when performing inference on rule R_i, and by y_0 the global output of the FLC for an input x_0.

There are two types of defuzzification methods[1,7,31] according to the way in which the individual fuzzy sets B_i' are aggregated in order to put into effect the function of the connective *also*, G:

- *Mode A : Aggregation First, Defuzzification After:* The Defuzzification Interface performs the aggregation of the individual fuzzy sets inferred, B_i', to obtain the final output fuzzy set B'.

$$\mu_{B'}(y) = G\{ \mu_{B'1}(y), \mu_{B'2}(y), ..., \mu_{B'n}(y) \} \quad (2.7)$$

Usually, the aggregation operator modeling the connective *also* (G) is the minimum or the maximum. After that, the fuzzy set *B'* is defuzzified using any strategy *D*, like Middle of Maxima, or the Center of Gravity mostly.

$$\mu_0 = D\,(\mu_{B'}\,(y)) \qquad (2.8)$$

- *Mode B : Defuzzification First, Aggregation After:* It avoids the computation of the final fuzzy set *B'* by considering the contribution of each rule output individually, obtaining the final control action by taking a calculation (an average, a weighted sum or a selection of one of them) of a concrete crisp characteristic value associated to each of them.

Historically, mode A was firstly proposed and used in the initial approximations to fuzzy control.[19,20] On the other hand, mode B is the one most used today on top in real-time systems which need the quickest response and in many other kinds of systems due to its simplicity.

3. Design of Fuzzy Logic Controllers

In this section, we are going to discuss the two FLC design tasks, the derivation of the Knowledge Base and the selection of the operators and methods that the controller will use to perform the fuzzy inference process.

3.1. *Obtaining the Knowledge Base*

The Knowledge Base is the only component of the FLC that directly depends on the specific application. The accuracy of the controller is very related to it. There are four modes of derivation for the fuzzy control rules that are not mutually exclusive[2,16]:

1. *Expert Experience and Control Engineering Knowledge*. It is the most widely used and it is effective when the human operator is able to linguistically express the control rules he uses to control the system. These rules are normally of Mamdani type.
2. *Modeling of the Operator´s Control Actions.* The control action is formed making a model of the operator actions without interviewing him.
3. *One based on the Fuzzy Model of a Process.* It is based on developing a fuzzy model of the system and constructing the fuzzy rules of the Knowledge Base from it. This approach is similar to that traditionally used in Control Theory. Hence, structure and parameter identification are needed.[24]
4. *One based on Learning and Self-Organization*. This method is based on the ability for creating and modifying the fuzzy control rules in order to improve the controller performance by means of automatic methods.

3.2. *Selecting the Fuzzy Operators*

The engineer must consider some factors that have a significant influence on the FLC:

- The choice of the connective operator for the antecedents (*T*).
- The choice of the fuzzy implication operator (*I*).
- The choice of the mathematical definition of the composition of fuzzy relations existing in the CRI.
- The choice of the connective *also* that connects the rules in the Knowledge Base (*G*).
- The choice of the defuzzification operator mode and the defuzzification method (*D*).

The connective operator for the antecedents has a low influence[7] in the FLC accuracy. It is a t-norm when the antecedents are connected with the connective *and*, and a t-conorm when the connective *or* is used.

As regards, the fuzzy implication and defuzzification operators, different decisions strongly influence the accuracy of the FLC. In the specific literature, there are a lot of possible choices for the implication operator as well as for the defuzzification method. We are going to show a classification of them into families with representative examples, and we will shed some light on how to choose the implication operators and defuzzification methods to obtain good controllers in the sense of the best behaviour.

3.2.1. *Fuzzy Implication Operators*

A classification of the fuzzy implication operators is proposed by Dujet and Vincent[11] considering the extension that they perform with respect to boolean logic:

- *Those extending the boolean implication.* Within this group, fuzzy implication functions are found.[26] They satisfy the following truth table:

a \ b	0	1
0	1	1
1	0	1

- *Those extending the boolean conjunction.* Force Implications[11] and T-norms when used as implication operators[12] are included in this group satisfying the truth table:

a \ b	0	1
0	0	0
1	0	1

There are many implication operators that do not belong to any of these two families. In the following, we are going to show some examples of implication operators[7,8,12,22] belonging to these three groups.

Appendix A shows the graphical representation of the membership functions of the inferred fuzzy sets for the implication operators presented.

3.2.1.1. *Boolean Implication Extension Operators*

Fuzzy implication functions[26] are the most well known implication operators that extend the boolean implication.

A continuous function $I: [0,1]\times[0,1] \rightarrow [0,1]$ is a *fuzzy implication function* iff $\forall\, x, x', y, y', z \in [0,1]$ verifies the following properties:[26]

1.- If $x \leq x'$ then $I(x,y) \geq I(x',y)$
2.- If $y \leq y'$ then $I(x,y) \leq I(x,y')$
3.- Falsehood Principle: $I(0,x) = 1$
4.- Neutrality Principle: $I(1,x) = x$
5.- Interchange Principle: $I(x,I(y,z))=I(y,I(x,z))$

They are classified into different families:[26,27]

- **Strong Implications (S-implications)**: Corresponding to the definition of implication in classical Boolean Logic : $A \rightarrow B = \neg A \vee B$. They present the form: $I(x,y) = S(N(a),b)$, with S being a t-conorm and N a negation function.
- **Residual Implications (R-implications)**: Obtained by residuation of a t-norm T, as follows $I(x,y) = Sup\ \{ c : c \in [0,1] / T(c,x) \leq y \}$.

The implication functions selected for use in this paper are the ones which showed the best behaviour from the preceding families in our previous contributions:[6,7]

S-Implications :
Dubois-Prade :

$$\mathbf{I_1\,(x,y)} = \begin{cases} 1 - x, & if\ y = 0 \\ y, & if\ x = 1 \\ 1, & otherwise \end{cases} \qquad (3.1)$$

R-Implications :
Goguen :

$$\mathbf{I_2\,(x,y)} = \begin{cases} 1, & if\ y \leq x \\ y, & otherwise \end{cases} \qquad (3.2)$$

S and R-Implications :
Lukasiewicz :

$$\mathbf{I_3\,(x,y)} = Min\,(1, 1-x+y) \qquad (3.3)$$

Other Extensions of the boolean implication:
There are some implication operators that extend the boolean implication but do not verify the properties of the implication functions. We have selected the following one according to its good behaviour in our previous studies:[7,8]

$$\mathbf{I_4\,(x,y)} = \begin{cases} 1, & if\ x = y \\ y, & otherwise \end{cases} \qquad (3.4)$$

3.2.1.2. *Boolean Conjunction Extension Operators*

(a) *T-norms:*
A function $T: [0,1]\times[0,1] \rightarrow [0,1]$ is a t-norm $\forall\, x, y, z \in [0,1]$ if it verifies the following properties:[12,22]

1.- Existence of unit element 1: $T(1,x) = x$
2.- Monotonicity: If $x \leq y$ then $T(x,z) \leq T(y,z)$
3.- Commutativity : $T(x,y) = T(y,x)$
4.- Associativity: $T(x,T(y,z)) = T(T(x,y),z)$
5.- $T(0,x) = 0$

The most typical t-norms used in FLCs are:

Logical Product (Minimum) :

$$\mathbf{I_5\,(x,y)} = Min\,(x, y) \qquad (3.5)$$

Algebraic Product :

$$\mathbf{I_6\,(x,y)} = x \cdot y \qquad (3.6)$$

(b) *Force-Implication Operators:*
Force implication operators were introduced for "combining the aim to model human reasoning in a more natural way with the need to achieve an implication".[11]

There are two different groups of force implications depending on the way in which they are built:

(b.1) *Force implications based on indistinguishability operators:*
They are formed with this expression:

$$I(x,y) = T(x\ ,\ E(x,y)) \qquad (3.7)$$

where T is a t-norm, and E is an indistinguishability operator,

$$E = T'(I'(x\ ,y),\ I'(y,x)) \qquad (3.8)$$

with T' being a t-norm, and I' an implication function.

There are three different kinds of indistinguishability operators depending on the t-norm used to define them:[26]

- *Similarity Relations*: $T'(x,y) = Min(x,y)$.
- *Probabilistic Relations*: $T'(x,y) = x \cdot y$.
- *Likeness Relations*: $T'(x,y) = Max(0, x+y-1)$.

We are going to show the three implication operators that presented the best behaviour of the force implication based on the indistinguishability family in a previous work.[8] They have been obtained by means of the $E_{Gödel}$ indistinguishability operator[26] and three t-norms: logical, algebraic and bounded products. Their expressions are:

$$\mathbf{I_7\,(x,y)} = Min\,(x, E_{Gödel}(x,y)) \qquad (3.9)$$

where $E_{Gödel}(x,y) = \begin{cases} 1, & \text{if } x = y \\ Min(x,y), & \text{otherwise} \end{cases}$

$$\mathbf{I_8\ (x,y)} = x \cdot E_{Gödel}(x,y) \tag{3.10}$$

$$\mathbf{I_9\ (x,y)} = Max\ (x + E_{Gödel}(x,y) - 1,\ 0) \tag{3.11}$$

(b.2) *Force implications based on distances:*
This second group follows the expression:

$$I(x,y) = T(x,\ 1 - d(x,y)) \tag{3.12}$$

where T is a t-norm, and *d* is a distance.

We will consider a force implication operator, which showed good behaviour in previous work[8], based on the t-norm bounded product, for which the expression is:

$$\mathbf{I_{10}\ (x,y)} = Max\ (x - |x-y|,\ 0) \tag{3.13}$$

3.2.2. *Defuzzification Methods*

In the following, we introduce the *Importance Degrees* and *Characteristic Values*, used for defining defuzzification methods.

Importance Degrees of a rule R_i :[7]

- *Area* of a fuzzy set *B'*:

$$s = \int_V \mu_{B'}(u)\, du \tag{3.14}$$

- *Matching* of a rule R_i,:

$$h_i = T(\mu_{A1}(x_1), \ldots \mu_{An}(x_n)) \tag{3.15}$$

with $A_1,\ldots A_n$ being fuzzy sets in the antecedent of the rule, and $x_1,\ldots x_n$ the values of the input variables.

- *Height* of a fuzzy set *B'*:

$$y = \sup_{x \in V} \mu_{B'}(x) \tag{3.16}$$

Characteristic Values:[7]

- *Maximum Value* of a fuzzy set *B'*:

$$G = \{x \in V \mid \mu_{B'}(x) = y\} \tag{3.17}$$

Sometimes, if there is more than one value that verifies the condition, the maximum value can be selected as the first one, the last one, or the middle of them, with the latter usually being chosen.[10]

- *Center of Gravity* of a fuzzy set *B'*:

$$W_i = \frac{\int_V y \cdot \mu_{B'_i}(y)\, dy}{\int_V \mu_{B'_i}(y)\, dy} \tag{3.18}$$

As we mentioned in Section 2, the Defuzzification Interface may operate in two ways[1,7,31]:

(a) ***Mode A:*** *Aggregation first, and defuzzification after.* First, the Defuzzification Interface performs the aggregation to accomplish the *also* operator, of the individual fuzzy sets inferred, B_i', to obtain the final output fuzzy set *B'*. The aggregation operator modeling that connective *also* ordinarily is the minimum or the maximum. To perform the defuzzification of the fuzzy final set *B'*, any of the following strategies may be used:

- Middle of Maxima (usually called MOM):

$$y_0 = \frac{y_1 + y_2}{2} \tag{3.19}$$

when $y_1 = Min\ \{z / \mu_{B'}(z) = Max\ \mu_{B'}(y)\}$ and $y_2 = Max\ \{z / \mu_{B'}(z) = Max\ \mu_{B'}(y)\}$

- Center of Gravity, for which the expression was shown before.

Combining the two possibilities expressed for the *also* connective and these two defuzzifiers we obtain four methods:

- $\mathbf{D_1}$: *Middle of Maxima* of the fuzzy set *B'*, result of the aggregation of the individual fuzzy sets B'_i with *also* connective *Minimum*.
- $\mathbf{D_2}$: *Center of Gravity* of the fuzzy set *B'*, result of the aggregation of the individual fuzzy sets B'_i with *also* connective *Minimum*.
- $\mathbf{D_3}$: *Middle of Maxima* of the fuzzy set *B'*, result of the aggregation of the individual fuzzy sets B'_i with *also* connective *Maximum*.
- $\mathbf{D_4}$: *Center of Gravity* of the fuzzy set *B'*, result of the aggregation of the individual fuzzy sets B'_i with *also* connective *Maximum*.

(b) ***Mode B****: Defuzzification first, and aggregation after:* This mode avoids the sometimes complex computation of the final fuzzy set *B'* by considering the contribution of each rule output individually, obtaining the final control action by taking a calculation (an average, a weighted sum or a selection of one of them) of a concrete crisp characteristic value associated to each of them.

There is a large group of defuzzification methods in Mode B in the specific literature. We have selected four of them which showed the best behaviour in previous works. They can be classified in different families depending on the kind of calculation performed:

- *Sums weighted by importance degrees.*[7,13,28] For example:
  - $\mathbf{D_5}$: Maximum Value weighted by the matching:

$$y_0 = \frac{\sum_i h_i \cdot G_i}{\sum_i h_i} \tag{3.20}$$

- *Based on the fuzzy set with the greatest importance value*.
  - **D_6**: Maximum Value of the fuzzy set with the greatest matching:[7]

$$B'_k = \{ B'_i \mid h_i = Max(h_t),\ \forall t \in \{1,...,m\} \}$$

$$y_0 = G_k \qquad (3.21)$$

- *Others*:
  - **D_7** : Middle of the Maximum Values:

$$y_0 = \frac{\sum_i G_i}{m} \qquad (3.22)$$

where *m* represents the non empty fuzzy sets obtained from the inference process.

  - **D_8** : Center of Sums:[10,13]

$$y_0 = \frac{\sum_i \int_V y \cdot \mu_{B'i}(y)\, dy}{\sum_i \int_V \mu_{B'i}(y)\, dy} \qquad (3.23)$$

3.2.3. *Fuzzy Operators: Robust Implication Operators*

As we said, the connective operator for the antecedents has a low influence,[7] in the behaviour of the resulting FLC. The most important choice corresponds to the operators in the Inference System and the Defuzzification Interface. Particular combinations of implication operators and defuzzification methods may offer excellent as well as bad results.

We found three basic properties for a *robust* implication operator.[8] We used the word *robust* in the sense of good average behaviour with different applications and different defuzzification methods. These three basic properties are:

(a) $I(h,0)=0,\ \forall h \in [0,1]$
(b) $I(h,1)>0,\ \forall h \in (0,1)$ *and* $I(1,1)=1$
(c) $I(0,y)=0, \forall y \in [0,1]$ (3.24)

If we use a fuzzy implication operator that verifies these three properties, we can select any defuzzification method with warranted good behaviour. The implication operators selected in this paper that verify the three properties are I_5, I_6, I_7, I_8, I_9 *and* I_{10}.

4. Framework for Implementing Fuzzy Logic Controllers

In this section, we are going to show some FLC classifications based on the host hardware used, software implementation method and number of fuzzy operators implemented for the same task.

4.1. *Design Methods*

There are two main ways to design an FLC for a specific application according to the host hardware:

(a) Using concrete electronic devices for fuzzy control applications of one of these two types:

- Fuzzy coprocessors. They are specific electronics devices to perform fuzzy inference operations being subordinated to generic purpose microprocessors or microcontrollers, e.g., VY86C570 from Togai InfraLogic, Inc.
- Microcontrollers that have special instructions and registers to carry out fuzzy inference, e.g., Motorola 68HC12 device.

(b) Using generic purpose microprocessors or microcontrollers that implement the FLC via software. We find the following options:

- On the market there are programs called *fuzzy shells* which take the information about the Knowledge Base and the fuzzy operators selected by the user from the overall data used, and automatically generate code for several microprocessors or microcontrollers.
- Taking improvement from libraries specially designed for programming applications with FLCs.
- *FLC Based on a Table*, where the fuzzy inference is not performed in real-time. A table with the pairs of input and output variables presented by an FLC is built. Then, the hardware generic controller only performs an interpolation between the values of the table. This solution is carried out when the fuzzy inference process is too slow in the control computer and the application needs the quickest response. The scheme of the FLC that fills the table may be accomplished with any of the two previous methods mentioned.
- Making all the code of user's own. In this paper, we are going to help the software implementation of the FLCs for use with generic devices. This option will mainly produce FLCs with lower speed response than the ones based on a specific hardware. Despite that, this way is more flexible from the FLC designer point of view, obtaining better adapted controllers in a general scientific point of view, and to study the behaviour of the different design options of the controller. This way is the preferable one, because it lets you test many options by changing only code instead of any kind of electronic devices.

4.2. *Software Implementation Methods*

- *Exact Method:* It works by first calculating the parametrical representation of the fuzzy sets inferred for every implication operator that we want to use.

The result of this study must be archived on a data structure. The main drawback of this method is that it needs the prior calculation of the parametric expressions of the fuzzy sets before implementing the FLC.

- *Approximate Method*: Due to the cited disadvantages, people use to avoid performing these previous calculations discretizing the universe of the consequent in a predefined number of points. This method will be slower than the exact one from a computational point of view, and it will use more memory. The precision will be in line with the granularity in the discretization and the lower the efficiency and speed response is, the higher this discretization level is. On a other hand, its advantages are that there is no need to calculate the parametrical representation of the output and his ability to deal with implication operators that describe fuzzy sets with curved zones. These ones could be directly incorporated into this approximate method while the exact method needs additional complexities because it has no way of knowing whether the line that connects two points is a curve or not.

4.3 *Types of Fuzzy Logic Controllers*

We may distinguish two kinds of FLCs according to the number of fuzzy operators implemented for the same task:

- *Mono-operator FLCs*: Controllers with a fuzzy operator per task. This is the conventional FLC used for specific applications (e.g., embedded controllers). Their data structures are exclusively adapted for the fuzzy operators used. The algorithms are the simplest and sometimes different components of the FLC are condensed in the same code.
- *Multi-operator FLCs*: Controllers that have some available user selectable fuzzy operators for the same task in the implementation. Data structures must be generic. Different component codes are clearly distinguished. These FLCs let us compare the behaviour of different FLCs built with different fuzzy operators. A flexible Multi-operator FLC could be used to generate several tables of input / output pairs to implement embedded FLCs *based on a table*.

At this point, it is important to mention that Mono-operator and Multi-operator FLCs could be implemented using Exact or Approximate Methods. But normally, during the design, if we are using a single operator for the Inference System and a single operator for the Defuzzification Interface (Mono-operator FLC), we will study the resulting expressions in order to condense and simplify the data structures and algorithms. Simplifications must be *wired* in the program code. This is coherent with the philosophy of practical and embedded FLCs in which the control computer is a small system where resources must be cautiously optimised. By other hand, Multi-operators need well separate data structures and algorithms in other to assemble different inference operators and defuzzification methods.

5. Software Implementation of Fuzzy Logic Controllers: Data Structures

The choice of the data structures is not a unique solution and sometimes depends on the compiler or interpreter used. In this paper, we are going to describe the more usual data structures to implement generic FLCs. In the following, we shall discuss the Knowledge Base and Inference System data structures.

5.1. *Knowledge Base Data Structures*

As said in previous sections, the Knowledge Base data structure must involve the conceptual information classified into Data Base and Rule Base. Generally, except for specific applications (adaptive FLCs that modify their behaviour norms depending on the results obtained by the control action previously administered), the Knowledge Base is a static data structure with a predefined size and fixed contents. The Knowledge Base data structure and its corresponding information must be specified before the FLC begins to work. The information about the fuzzy sets associated to the linguistic terms employed in the fuzzy rules compounds the Data Base. In order to simplify the calculations, linear piecewise membership functions are considered and every fuzzy set is usually described by three or four points x_0,x_1,x_2,x_3 (Figure 2), that is, depending on the membership function shape: triangular or trapezoidal fuzzy sets.

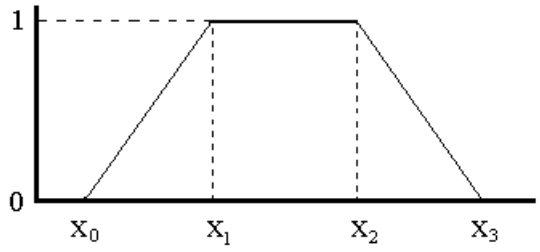

Fig. 2. Trapezoidal Fuzzy Set.

The values of these three or four definition points in the fuzzy set correspond to the place that it has in the universe of the corresponding variable. Figure 3 shows an example of a variable universe split into five trapezoidal fuzzy sets.

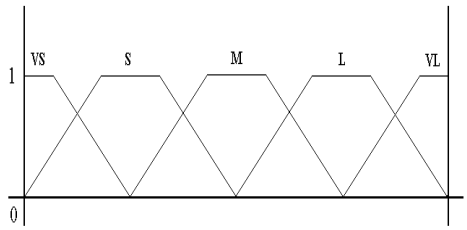

Fig. 3. Fuzzy partition with fuzzy sets VS (very small), S (small), M (medium), L (large) and VL (very large).

Therefore, each fuzzy set from each one in the variable fuzzy partitions must be described and stored in this way, so forming the Data Base. The Rule Base information stored is a set of rules where each one refers to the corresponding fuzzy sets in the Data Base.

We are going to propose two possible data structures to store the Knowledge Base that will be used depending on the capabilities of the coding language used:

(a) For an advanced compiler, an optimal data structure may consist of an array of records, where each record could be a rule in the Rule Base having some fields, the variables of the rule (antecedents and consequents), which would be of numeric or pointer type. These numbers or pointers will be the reference to another array of fuzzy sets, i.e., an array of records which again has three or four fields associated depending on the form of the fuzzy sets, whether triangular or trapezoidal as mentioned. This second group of fields would be of a real or a floating numerical type. Figure 4 shows this data structure in the case of rules with two antecedents, a_1 and a_2 , and a single consequent c.

Sometimes it may be useful to consider a variable number of rules or fuzzy sets for several Knowledge Bases. In those cases, static arrays can be replaced by lists on dynamic memory.

(b) When we use microcontrollers, it is usual to have a simple language compiler (e.g., any ANSI C subset) or a language interpreter (e.g., Intel MCS BASIC-52) that only allows us to use plain data structures and not to manage records. Thus, it is not possible to use the previously introduced data structure. In these cases, the data structure shown in Figure 5 may be considered. It only uses arrays of real or integer numbers.

The structure is based on considering as many arrays as antecedents, consequents, and definitions points used in the fuzzy sets. Figure 5 shows an example for trapezoidal fuzzy sets.

The Data Base is stored on the arrays of real numbers X0, X1, X2 and X3. Each cell in these arrays has the information related to a corresponding definition point. In Figure 5, they are marked with the legend a_i, fs_j, x_k standing for the information of the antecedent i (i takes values from 1 to 2 if we have rules with two antecedents), fuzzy set j (j takes values from 1 to the total number of different fuzzy sets defined for the antecedents and consequents), and obviously to the definition point k of the fuzzy set. The dimension of these arrays is the total number of definition points which is easy to compute as the product between the number or rules and the sum of antecedents and consequents.

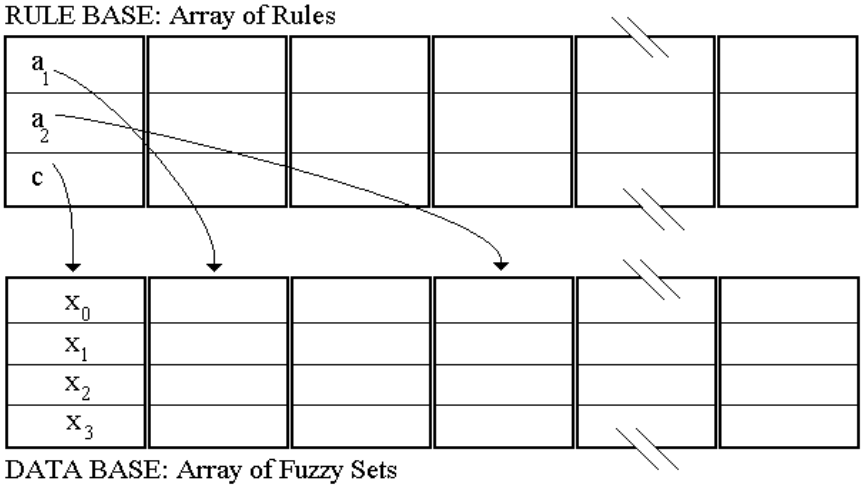

Fig. 4. A possible data structure for the Knowledge Base.

The Rule Base is stored on the three arrays A1, A2 and C. The dimension of these arrays is the number of rules in the Rule Base. The rules allude to fuzzy sets that are stored on the Data Base (represented by means of arrows in Figure 5). Thus, the integer number stored on each cell of these arrays points to a fuzzy set on the arrays used to store the Data Base, X0, X1, X2 and X3.

For example: We have a Rule Base with seven rules with two antecedents and a single consequent. The antecedent fuzzy partition has 10 and 3 fuzzy sets, respectively, whilst the consequent fuzzy partition has 5 fuzzy sets. The Data Base presents trapezoidal fuzzy sets (four definition points). Then, the array declaration for the Intel MCS BASIC-52 code will be:

```
100  REM Rule Base with 7 rules.
110  DIM A1(7)120  DIM A2(7)130  DIM C(7)
140  REM Data Base with 10+3+5=18 fuzzy sets.130
DIM X0(18)140  DIM X1(18)150  DIM X2(18)
160  DIM X3(18)
```

The result is a very easy to use structure, e.g., if we need the second definition point of the consequent on rule *i*, we will use X1(C(i))

This data structure can also be easily mapped for assembler coding in the amount of addressed memory on the microcontroller based system.

5.2. *Inference Process Data Structures*

The inference process needs two data structures:

(a) A *Matching Vector*: It is a list of real numbers that will temporarily store the matching for each rule of the Knowledge Base. Usually, it is implemented as a static array because the number of cells is usually small and previously known. For example, the array declaration for the Intel MCS BASIC-52 code will be:

```
170  DIM H(7)
```

when we are using seven rules in the Knowledge Base.

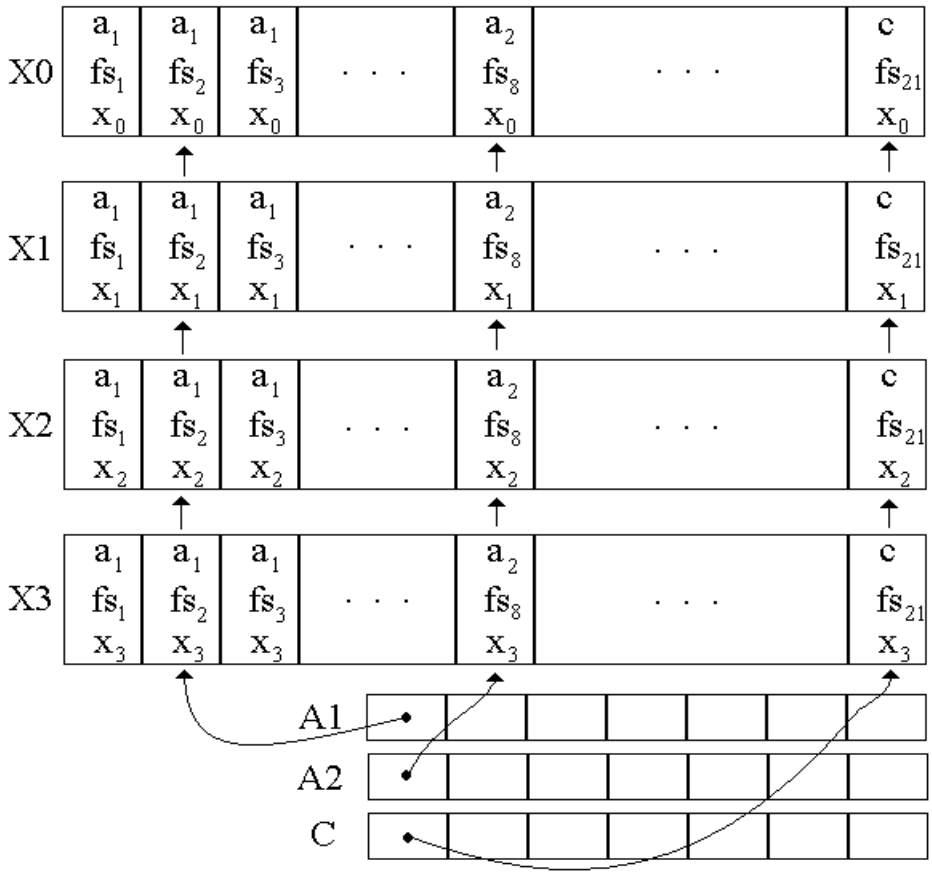

Fig. 5. Another data structure for a Knowledge Base.

(b) *Consequent Vectors* : They store the final result of the inference process. These vectors are taken as input by the Defuzzification Interface. The data structure depends on the kind of software implementation of FLC:

- When they are implemented with the Exact Method: Some lists of description points will be used. Each list will contain the inferred fuzzy set proceeding from a rule. The points has two coordinates, that is, two real numbers. The length of the lists is variable. They must be allocated on dynamical memory.
- When they are implemented with the Approximate Method: *N* lists or arrays of points will be used (being *N* the number of rules in the Knowledge Base). The length is fixed and equal to the number of points of the discretization.

In Mono-operator FLCs, when a Mode B defuzzification method is employed, Inference System and defuzzification method may be condensed in a single algorithm, then Consequent Vectors are not needed. We shall deal with this in following sections.

6. Software Implementation of Fuzzy Logic Controllers: Algorithms

This subsection shows the algorithms for performing the different components of an FLC: Fuzzification Interface, Inference System and Defuzzification Interface, pointing out the Mono-operator and Multi-operator FLCs differences when they are required.

6.1. *Fuzzification Interface*

The most widely used and simplest way to perform the Fuzzification Interface in fuzzy control is by applying the singleton fuzzifier, as we said in the preceding section. The singleton fuzzification does not need any operation because the crisp values of the input variables are the only ones in which the fuzzy sets take value 1, with it being 0 otherwise. That is, the values of the input variables directly represent the singleton fuzzy sets centered on them in the implementation.

$$A'(x) = \begin{cases} 1, & if \quad x = x_0 \\ 0, & otherwise \end{cases} \qquad (6.1)$$

6.2. *Inference System*

The matching calculation is carried out by going through each rule in the Knowledge Base and calculating the intersection point h_i between the singleton fuzzy set A' (obtained from the input x_j) and the fuzzy set of the rule. If the rules have more than one antecedent, the corresponding intersections of the other fuzzy sets must be computed with their respective singleton fuzzy sets, also obtaining as many h_{ij} values as antecedents.

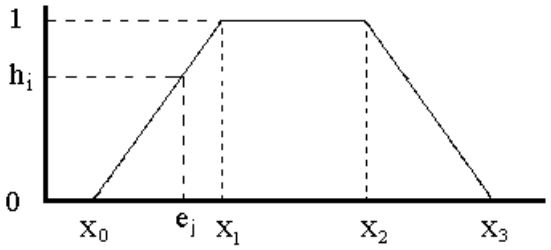

Fig. 6. Computing the matching, h_i.

The different zones existing in the fuzzy set should be observed in order to compute the matching values. For example, in Figure 6, five possible zones must be considered. Here are these zones and the expressions of the height of the corresponding input value:

$$\begin{aligned}
&\text{(i)} && e_j' < x_0 : h_{ij} = 0 \\
&\text{(ii)} && x_0 < e_j < x_1 : h_{ij} = \frac{e_j - x_0}{x_1 - x_0} \\
&\text{(iii)} && x_1 < e_j < x_2 : h_{ij} = 1 \\
&\text{(iv)} && x_2 < e_j < x_3 : h_{ij} = \frac{e_j - x_3}{x_2 - x_3} \\
&\text{(v)} && x_3 < e_j' : h_{ij} = 0
\end{aligned} \qquad (6.2)$$

where e_j is the current value for the input variable x_j.

After that, the connective operator must be applied in order to obtain the ith matching value h_i, with $i=1$ to N, with N being the number of rules in the Knowledge Base. If the connective operator of the rules is a conjunction *AND*, then the function applied will be a t-norm (e.g., the minimum). But, if the connective operator is a disjunction *OR*, then the function applied will be a t-conorm (e.g., the maximum). Figure 7 graphically shows the application of the minimum t-norm resulting $h_i = Min\ (h_{i1}, h_{i2})$.

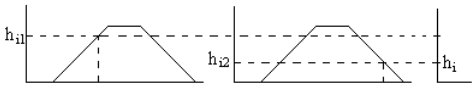

Fig. 7. The minimum as conjunction connective *AND*.

We showed[7] that the choice of the t-norm that implement the connective operator conjunction does not significantly affect the accuracy for the response of the FLC. The h_i values must be stored in an array or list of real numbers with size N (the Matching Vector).

For example, here is the Intel MCS BASIC-52 code to compute the Matching Vector *H*. We use the Data Base and Inference System Structure shown in the preceding subsections, seven rules, two antecedents and trapezoidal fuzzy sets. The connective operator *AND* is applied with the minimum t-norm:

```
260 FOR i=1 TO 7
270   REM Antecedent 1
280   IF e1<X0(A1(i)) THEN 290 ELSE 300
290     min=0
300   IF e1>X0(A1(i)) AND e1<X1(A1(i)) THEN 310 ELSE 320
310     min=(e1-X0(A1(i)))/(X1(A1(i))-X0(A1(i)))
320   IF e1>X1(A1(i)) AND e1<X2(A1(i)) THEN 330 ELSE 340
330     min=1
340   IF e1>X2(A1(i)) AND e1<X3(A1(i)) THEN 350 ELSE 360
350     min=(e1-X3(A1(i)))/(X2(A1(i))-X3(A1(i)))
360   IF e1<X3(A1(i)) THEN 370 ELSE 380
370     min=0
380   H(i)=min
390   REM Antecedent 2
400   IF e2<X0(A2(i)) THEN 410 ELSE 420
410     min=0
420   IF e2>X0(A2(i)) AND e2<X1(A2(i)) THEN 430 ELSE 440
430     min=(e2-X0(A2(i)))/(X1(A2(i))-X0(A2(i)))
440   IF e2>X1(A2(i)) AND e2<X2(A2(i)) THEN 450 ELSE 460
450     min=1
460   IF e2>X2(A2(i)) AND e2<X3(A2(i)) THEN 470 ELSE 480
470     min=(e2-X3(A2(i)))/(X2(A2(i))-X3(A2(i)))
480   IF e2<X3(A2(i)) THEN 490 ELSE 500
490     min=0
500   IF min<H(i) THEN H(i)=min
510 NEXT i
```

Now, we obtain the inferred fuzzy set from each rule using the consequent and the matching of each rule. The selected fuzzy implication operator produces an output fuzzy set with a specific shape that is defined by a parametrical expression. This expression is graphically represented in Appendix A for the implication operators presented in the preceding section. Usually, the new fuzzy sets obtained from each rule must be stored to perform the defuzzification. In singular situations, it is not necessary to store them because the Defuzzification Interface can act at the same time, as we shall see later.

The inferred fuzzy sets could be described with a variable number of six or eight points in the same way as the fuzzy sets for the antecedents and consequents are described in the Knowledge Base, adding an additional value to each one of them for the height in this case. Thus, the data structure for these fuzzy sets will be an array or list of points (the Consequent Vector). For example, if we use the minimum t-norm as an implication operator, the inferred fuzzy sets will be like the one presented in Figure 8. This graph is the result of applying the minimum implication operator expression (*I(x,y)=Min(x,y)*) over the matching *h* and the consequent fuzzy set of the rule.

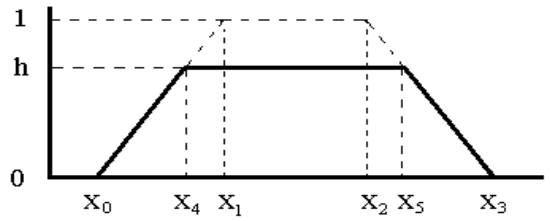

Fig. 8. Minimum implication operator.

In the case of the minimum implication operator and many other ones, the height information agrees with the matching one. The points x_0 and x_3 are the same as in the consequent. The other two, x_4 and x_5, can be calculated as the intersection of the line that passes through x_0 and x_1 and the line that passes through x_2 and x_3 with the horizontal one at height *h*, respectively. Then, the expressions are:

$$x_4 = x_0 + (x_1 - x_0) \cdot h_i$$
$$x_5 = x_3 - (x_3 - x_2) \cdot h_i \qquad (6.3)$$

These expressions could be directly incorporated to the source code for the FLC with their corresponding height, *h*. In the same way, any other implication operator could be used by obtaining the definition points and implementing the expressions describing its computation.

In those cases that rules have more than one consequent variable, as many different problems as existing consequents will be considered.

Now, we are going to make some comments as regards the kind of software implementation of the FLC:

- Exact Method: The graphical representation of the inferred fuzzy sets is required. Then, we obtain the definition points of the inferred fuzzy set. These points will be stored in the data structure for the Inference System mentioned in the previous subsections, that is, a list of points for each inferred fuzzy set proceeding from the rules. For example, if we look at Figure 8, the definition points that will be stored are:

$$(x_0' = x_0 , 0)$$
$$(x_1' = x_4 , h)$$
$$(x_2' = x_5 , h)$$
$$(x_3' = x_3 , 0) \qquad (6.4)$$

  where x_4 and x_5 have to be computed with the expression previously shown, and x_0 and x_3 are the definition points for the consequent of the rule. Thus, the advantage of this method from the point of view of the Inference System is the minimum wasting of memory, and the inference process efficiency. The disadvantage is the requirement of a previous study of the inference operator to determine the graphical representation and point expressions. If we want to add a lot of implication operators, the design process will be very hard and error sensitive.
- Approximate Method: The universe of the consequent is divided into a fixed number of points, establishing a discretization[4,5]. The mathematical expression of the implication operator will be directly incorporated into the code. Thus, the inferred fuzzy set will be the result computed with

the implication operator expression using the discretization points and the matching as inputs. A potential discretization for the Goguen implication operator is shown in Figure 9. The expression of the Goguen implication operator, as shown in previous sections, is:

$$I_{Goguen}(x,y) = \begin{cases} 1, & if\ y \le x \\ y, & otherwise \end{cases} \quad (6.5)$$

then, the discrete function *f* that approximates it is:

$$f_i = I_{Goguen}(h\ ,\ i) \quad (6.6)$$

with *i* taking a finite number of equidistant values between 0 and the high extent of the variable dominion.

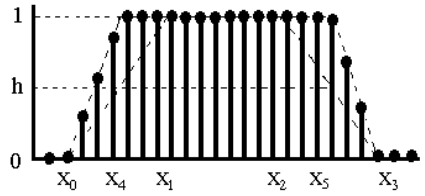

Fig. 9. An approximation to the Goguen implication operator.

The advantage of this method from the point of view of the Inference System is that it does not need a huge prior study of the implication operators. But the disadvantages are the low efficiency, the fact that the accuracy depends on the granularity of the discretization that is inversely proportional to the speed, and the large amount of wasted memory, as we mentioned in the data structure description.

6.3. *Defuzzificación Interface*

The implementation of the Defuzzification Interface strongly depends on the Defuzzification Mode selected.

Mode A:

As we mentioned, firstly, the Defuzzification Interface performs the aggregation of the individual fuzzy sets inferred, B_i', to accomplish the *also* operator, with the aim of obtaining the final output fuzzy set B'. The aggregation operators modeling the connective *also* are usually the minimum or the maximum.

Figures 10 and 11 graphically show the behaviour of the *also* connective operators maximum and minimum respectively when applied to two inferred fuzzy sets.

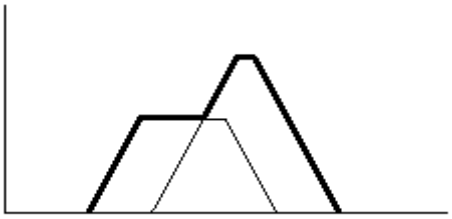

Fig. 10. Aggregation with maximum.

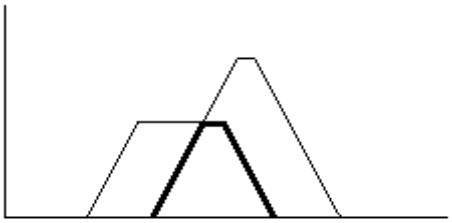

Fig. 11. Aggregation with minimum.

In this case, the Exact Method involves a significant degree of complexity. The aggregation operators maximum and minimum must act with fuzzy sets described by rectilinear segments. The data structure to manage the inferred fuzzy sets is a list of points. A new list must be created in the aggregation procedure. The typical way is to build a function that aggregates fuzzy sets two by two, that is, obtaining a new list of points representing the aggregated fuzzy sets from the two preceding ones. By executing this function as many times as individual fuzzy sets have been inferred less one, the final aggregated fuzzy set will be obtained. The aggregation algorithm will calculate the relative positions of the integrating segments and study their relative position computing the cross points if they exist, and adding them to the output list. Sometimes it is interesting to design a simplification function to study and eliminate the unnecessary points added to the list. This will improve the performance of the said aggregation function.

The aggregation of the inferred fuzzy sets in the Approximate Method when working in Mode A is easy to implement with an algorithm which goes through all the points of the discretization of the inferred fuzzy sets computing the maximum or minimum of them, for the two said aggregation operators. The same operation mode may be used with any other operator modeling the also connective. Figures 11 and 12 graphically show the maximum and minimum aggregation.

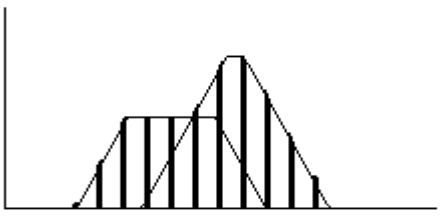

Fig. 11. Approximate aggregation with maximum.

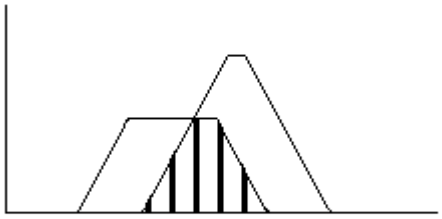

Fig. 12. Approximate aggregation with minimum.

After that, to make the defuzzification of the final fuzzy set *B'*, it could be used:

- The Middle of Maxima (usually called MOM):

$$y_0 = \frac{y_1 + y_2}{2} \qquad (6.7)$$

where $y_1 = Min\ \{z/\ \mu_{B'}(z) = Max\ \ \mu_{B'}(y)\}$ and $y_2 = Max\ \{z/\ \mu_{B'}(z) = Max\ \mu_{B'}(y)\}$

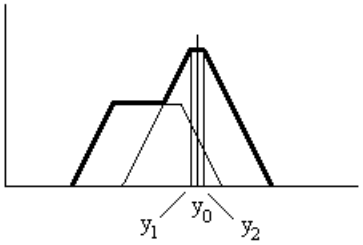

Fig. 13. **D3**: MOM with *also* connective maximum.

Figure 13 graphically shows the previously called **D3** defuzzification method: Middle of Maxima of the fuzzy set *B'*, result of the aggregation of the individual fuzzy sets B'_i with *also* connective maximum.

In the Exact Method, the Middle of Maxima point is easily computed considering the different possibilities:

- The maximum value point is unique: this point will be an extent of a segment and can be found by comparing the extents of the segments that compose the aggregated fuzzy set.
- The maximum value is a countable set of points: in a similar way, this set of points will be the extents of segments. The final result is computed as the average of the lower and upper values (Figure 13).
- The maximum value is an uncountable set of points: that is, an horizontal segment has the maximum value. In this case, if there are no other single points, the extents to compute the middle point are the lower and upper values.

In the Approximate Method the Middle of Maxima is computed going through the discretization and recording the maximum height points. If the maximum height point is not unique, then it must be computed as the average of the lower and upper ones as well.

- or the Center of Gravity, which exact expression is:

$$W_i = \frac{\int_V y \bullet \mu_{B'_i}(y)\, dy}{\int_V \mu_{B'_i}(y)\, dy} \qquad (6.8)$$

Figure 14 graphically shows the previously called **D2** defuzzification method: Center of Gravity of the fuzzy set *B'*, result of the aggregation of the individual fuzzy sets B'_i with *also* connective minimum

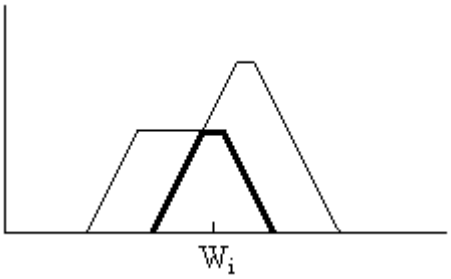

Fig. 14. **D2**: Center of Gravity with *also* connective minimum.

Usually, the resulting shape is not as simple as in Figure 14. In the Exact Method, complex figures must be decomposed into more simpler geometric pieces between every two points. The numerator on the expression of the Center of Gravity is computed as the sum of the different computations developed on these simple geometric pieces. The denominator is calculated in a similar way and finally the division is computed. These are the expressions for the pieces in the two possible cases:

Case 1:

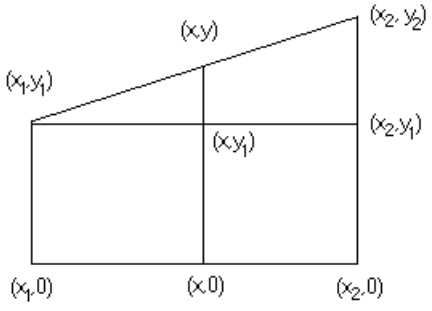

Fig. 15. Case 1.

Numerator ($\int_V u \cdot \mu_A(u)du$):

$$\int_{x_i}^{x_j} x \cdot f(x) = c \cdot \frac{x_2^2 - x_1^2}{2} + \frac{x_2^3 - x_1^3}{3} \qquad (6.9)$$

with a being $\frac{y_2 - y_1}{x_2 - x_1}$, and c being $\frac{x_2 \cdot y_1 - x_1 \cdot y_2}{x_2 - x_1}$

Denominator (areas $\int_V \mu_A(u)du$):

$$\int_{x_i}^{x_j} f(x) = c \cdot (x_2 - x_1) + a \cdot \frac{x_2^2 - x_1^2}{2} \qquad (6.10)$$

Case 2:

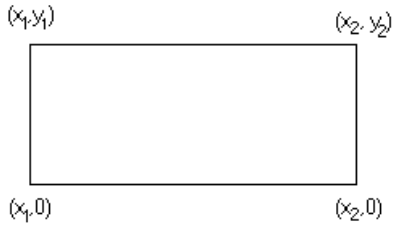

Fig. 16. Case 2.

Numerator ($\int_V u \cdot \mu_A(u)du$):

$$\int_{x_i}^{x_j} x \cdot f(x) = y_1^2 \cdot \frac{x_2^2 - x_1^2}{2} \qquad (6.11)$$

Denominator (areas $\int_V \mu_A(u)du$):

$$\int_{x_i}^{x_j} f(x) = y_1^2 \cdot (x_2 - x_1) \qquad (6.12)$$

In the Approximate Method, the Center of Gravity will be easily computed because the integrals are calculated as sums in discrete cases like this:

$$W_i = \frac{\sum_{j=1..N} y_j \cdot \mu_{B'}(y_j)}{\sum_{j=1..N} \mu_{B'}(y_j)} \qquad (6.13)$$

with N being the number of points of the discretization.

Mode B:

As we mentioned, this mode avoids the sometimes complex computation of the final fuzzy set *B'* by considering the contribution of each rule output individually, obtaining the final control action by taking a calculation (an average, a weighted sum or a selection of one of them) of a concrete crisp characteristic value associated to each of them.

The defuzzification methods in Mode B compute the final value using the following measures:

Importance Degrees:

- Area of the inferred fuzzy set: It is computed as a sum of the areas of simple geometric figures. Area expressions have already been shown in the Exact Method. The Approximate Method accomplish the sum of the product of the function value in every point and the length of the discretization interval.
- Matching, previously computed.
- Height of the inferred fuzzy set, computed in a similar way to that described for the Middle of Maxima for the Exact Method. In the Approximate Method, the maximum value in the Y axis found going through the complete discretization.

Characteristic Values:

- Maximum Value of the inferred fuzzy set: it is the projection of the maximum height point on the X axis.
- Center of Gravity of the inferred fuzzy set: Computed in the same way as described in Defuzzification Mode A.

On the other hand, Mono-operator FLCs may also be implemented using the Exact or Approximate Methods. Then, all the previous descriptions are equally valid. However, Mono-operator FLCs are rarely implemented using the Exact or the Approximate Methods mainly due to the following two reasons: they are designed for a specific application and they run in embedded computers. These issues allow us to compact the algorithms reducing the required resources and force us to use computationally efficient algorithms, so it is possible to comprise the Inference System and the Defuzzification Interface in a very short algorithm. It is desirable to defuzzify each fuzzy set inferred before inferring the following one, because this allows us not to store that fuzzy set, we only store a real value. After that, we do the simple calculation proposed by the method to obtain the final value with N preceding values.

Nevertheless, it is not possible to compact the algorithms in a Mono-operator FLC when working in Mode A. Therefore, practical Mono-operator FLCs for embedded applications must be implemented using Defuzzification Mode B. Moreover, when FLCs built in this way present the best behaviour in many cases. In Section 3.2.3, we showed the conditions for the fuzzy implication operators to obtain the best FLC accuracy[8] without using complex operators with difficult implementations.

As we have said, Mode B defuzzification methods are easy to implement. The different options for the Defuzzification Interface shown in the preceding

sections were based on operations related to the calculation of the Value of Importance and Characteristic Values. When the membership function of the inferred fuzzy set is known, this advantage must be used to directly incorporate the simplified mathematical expressions to compute the crisp output. Defuzzification Mode B especially benefits from this: it computes an easy calculation on known forms. For example, the Maximum Value of a symmetric trapezoidal fuzzy set will be simply $(x_1+x_2)/2$.

Thus, following with the example of how to implement an FLC in Intel MCS BASIC-52 language: let us go on to see the compact form of the Inference System and defuzzification method, if the former uses the minimum implication operator and the later is based on the Maximum Value weighted by the matching, the implementation will be:

$$y_0 = \frac{\sum_i h_i \cdot \left(\frac{x_{1i} + x_{2i}}{2} \right)}{\sum_i h_i} \qquad (6.14)$$

The BASIC-52 code is:

```
520  sum1=0
530  sum2=0
540  FOR i=1 TO 7
550   sum1=sum1+(H(i)*(X1(C(i))+X2(C(i)))/2)
560   sum2=sum2+H(i)
570  NEXT i
580  y=sum1/sum2
```

with *y* being the crisp output.

7. Comparative Study of Approximate and Exact Methods

We are going to compare the behaviour of the two methods presented for implementing Multi-operator FLCs, in order to determine the loss of precision achieved by the sometimes used quick method to construct them.

Three applications have been considered to analyze the behaviour of the fuzzy implication operators selected in the two implementation methods for the Multi-operator FLC: the fuzzy modeling of the simplest functional relation Y=X, and of a three-dimensional surface, and the well known problem of the control of the inverted pendulum. A description of the three applications is to be found in Appendix B.

The Medium Square Error (SE) has been calculated as an FLC performance measure:[7,8]

$$SE\ (S[i,j]) = \frac{\frac{1}{2} \sum_{k=1}^{N} (y_k - S[i,j](x_k))^2}{N} \qquad (7.1)$$

where S[i,j] denotes the FLC for which the Inference System uses the implication operator I_i, and for which the Defuzzification Interface is based on the defuzzificacion method, D_j. This measure employs a set of system evaluation data formed by N arrays of numerical data $Z_k=(x_k, y_k)$, $k=1,...,N$, with x_k being the values of the state variables, and y_k the corresponding values of the associated control variables.

The conjunctive operator used for the experiments was always the minimum t-norm.

In order to see the dependence on the discretization granularity in the universe of the consequent variable when working with the Approximate Method, we have used two partitions with 25 and 100 points.

Tables 1 to 3 in Appendix C show the SE values obtained in the Y=X application (see Appendix B) with the Exact Method, and the Approximate Method with 25 and 100 point in the discretization interval, respectively. Tables 4 to 6 presents the corresponding ones for a three dimensional surface (see Appendix B), and Tables 7 to 9 the results for the inverted pendulum (see Appendix B).

Some implication operators (I_4, I_8, and I_9) present problems (marked with " * " on the Tables 1 to 12) when making inference due to the discontinuities that appear in the inferred membership functions. In those cases, we only used Defuzzification Mode B, which defuzzifies the one-element to that single element exactly. We do not aggregate fuzzy sets of this kind.

Tables 10, 11 and 12 show a summary of the results in Tables 1 to 9.

Next, we present some comments as regards these tables:

As regards the two implementation methods:

Characteristic Values:

- SE values are, in general, very similar in both methods. Good combinations of operators usually show the same good behaviour in both implementations. Therefore, the choice of implementation presents less importance than the operator selection.
- The *Exact Method* always shows the precise results of the fuzzy operators. This is the reason for the lower error generally presented when using it. However, a small number of combinations of fuzzy operators show lower errors when they are implemented with the *Approximate Method.* This fact could be apparently contradictory, but there is no contradiction because the smoothing effect performed by the *Approximate Method* on the membership function of the inferred fuzzy sets is advantageous in a few cases. This improvement in the accuracy for these combinations of fuzzy operators does not correspond to a true good behaviour for them.

Table 10. SE abstract of the Y=X application.

	Exact Method		Approximate Method N=25		Approximate Method N=100	
	I_2	I_{10}	I_2	I_{10}	I_2	I_{10}
D_2	0.0962	0.06604	0.2019	2.9883	0.1079	2.0120
D_5	0.0498	0.0498	0.0623	0.2970	0.0921	0.2788

Table 11. SE abstract of the three dimensional surface application.

	Exact Method		Approximate Method N=25		Approximate Method N=100	
	I_1	I_{10}	I_1	I_{10}	I_1	I_{10}
D_1	0.3494	3.5258	0.3621	4.2172	0.3504	0.3613
D_7	4.7566	0.3481	4.7566	0.9301	4.7543	0.7621

Table 12. SE abstract of the Inverted Pendulum application.

	Exact Method		Approximate Method N=25		Approximate Method N=100	
	I_2	I_5	I_2	I_5	I_2	I_5
D_1	6826.3	20004.4	11315.3	23763.9	6919.3	22701.2
D_5	6425.7	6425.7	9720.6	9720.6	6425.7	6425.7

- Some combinations of fuzzy operators show highly significant differences when they are implemented with the *Exact* or the *Approximate Method*, e.g., I_9 and I_{10}. The source of this disparity may be found by studying the membership functions for the inferred fuzzy sets of both implication operators (shown in Appendix A) in combination with the concrete defuzzifiers used. These two operators present more than one maximum value point or one continuous maximum value point. For example I_{10} has two separate maximum value points. This situation introduces a significant error in the *Approximate Method*, when it defuzzifies with a maximum value defuzzifier because the algorithm only finds one maximum value point. The other maximum value is not considered to be at the same height because the approximation makes one of them bigger than the other one. As we mentioned in the preceding sections with respect to the implementation of defuzzifiers, when more than one maximum value point appears, the result is computed as the average for them. That is, the *Exact Method* places the result of the defuzzification in the centre of the inferred fuzzy set whilst the *Approximate Method* places it at one of the extents, too far away from the other solution. This fact is highly significant in the case of I_{10}, especially when the matching values are lesser than ½.

As regards the behaviour of the fuzzy operators:

- As we showed in preceding works[7,8], T-norms are very robust implication operators regardless of the implementation method, that is, they show good average behaviour in different applications and in combination with different defuzzification methods. In the same way, Implication Functions are not robust implication operators for fuzzy control.

8. Concluding Remarks

In this work we have studied the usual ways to implement practical FLCs.

When we design a specific FLC, we must use good fuzzy operators, and we must select an easy way to implement them. Section 3.2.3 shows the three basic properties to have a robust implication operator, i.e., an operator presenting good average behaviour with different applications and in combination with different defuzzification methods.

Besides, if we need a quick FLC, efficient considerations must be added. All these requirements are complete if we use the minimum or the algebraic product t-norms as the implication operator in combination with the maximum value weighted by the matching defuzzifier. An example code of Mono-operator design for a specific embedded practical FLC with the Intel MCS-52 microcontroller has been shown. It was developed in Intel BASIC-52 and, therefore, it is only recommended for teaching purposes or for applications that do not need the fastest answer.

As regards the Multi-operator FLC implementation methods, we have shown that the results are false in different ways in the *Approximate Method*. In the best cases, it introduces imprecision or produces a low artificial reliability, but in the worst cases, it produces significant variations. To study the pernicious effects of the Approximate Method, the first step is to obtain the geometrical study of the inferred fuzzy sets, losing the primary advantage of this method with respect to the *Exact Method*.

References

1. A. Bardossy and L. Duckstein, *Fuzzy rule-based modeling with application to geophisical, biological and engineering system.* (CRC Press 1995)
2. H. Berenji, Fuzzy Logic Controllers, An Introduction to Fuzzy Logic Applications in Intelligent Systems, in *R. R. Yager and L. A. Zadeh*, eds Kluwer Academic (Boston 1993) 69-96.
3. Z. Cao and A. Kandel, Applicability of some Fuzzy Implication Operators, *Fuzzy Sets and Syst.* **31** (1989) 151-186.
4. Z. Cao, D. Park and A. Kandel, Investigations on the applicability of fuzzy inference, *Fuzzy Sets and Syst.* **49** (1992) 151-169.
5. Z. Cao, A. Kandel and L. Li, Fuzzy Inference and its applicability to control systems, *Fuzzy Sets and Syst.* **48** (1992) 99-111
6. O. Cordón, F. Herrera and A. Peregrín, T-norms versus Implication Functions as Implication Operators in Fuzzy Control, in *Proc. 6th IFSA Congress* (1995) 501-504.
7. O. Cordón, F. Herrera and A. Peregrín, Applicability of the Fuzzy Operators in the Design of Fuzzy Logic Controllers, *Fuzzy Sets and Syst.* **86** (1997) 15-41.
8. O. Cordón, F. Herrera and A. Peregrín, A Study of the Use of Implication Operators Extending the Boolean Conjunction in Fuzzy Control, in *Proc. 7th IFSA Congress* (1997) 243-248.
9. O. Cordón, F. Herrera and A. Peregrín, Searching for Basic Properties Obtaining Robust Implication Operators in Fuzzy Control, *Fuzzy Sets and Syst.* (1998), To appear.
10. D. Driankov, H. Hellendoorn and M. Reinfrank, *An Introduction to Fuzzy Control*. (Springer Verlag, 1993).
11. Ch. Dujet and N. Vincent, Force Implication: A new approach to human reasoning, *Fuzzy Sets and Syst.* **69** (1995) 53-63.
12. M.M. Gupta and J. Qi, Design of Fuzzy Logic Controllers Based on Generalized T-operators, *Fuzzy Sets and Syst.* **40** (1991) 473-489.
13. H. Hellendoorn and C. Thomas Defuzzification in Fuzzy Controllers. *J. of Intelligent and Fuzzy Syst.* **1** (1993) 109-123.
14. K. Hirota, *Industrial Applications of Fuzzy Technology*. (Springer Verlag, 1993).
15. J. Kiszka, M. Kochanska and D. Sliwinska, The influence of Some Fuzzy Implication Operators on the Accuracy of a Fuzzy Model - Parts I and II, *Fuzzy Sets and Syst.* **15** (1985) 111-128, 223-240.
16. C.C. Lee, Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Parts I,II, *IEEE Trans. on Syst., Man and Cybernetics* **20**, 2 (1990) 404-418, 419-435.
17. E. Lembessis and R. Tanscheit, The Influence of Implication Operators and Defuzzification Methods on the Deterministic Output of a Fuzzy Rule-Based Controller, in *Proc. 4th IFSA Congress* (1991) 109-114.
18. C. Liaw and J. Wang, Design and Implementation of a Fuzzy Controller for a High Performance Induction Motor Drive, *IEEE Trans. On Syst. Man and Cybernetics* **21**, 4 (1991) 921-929.
19. E.H. Mamdani, Applications to fuzzy algorithms for simple dynamic plant. in *Proc. IEE* **121**, 12 (1974) 1585-1588.
20. E.H. Mamdani, and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. of Man-Machine Studies* **7** (1975) 1-13.
21. M. Mizumoto, Fuzzy Controls under Various Approximate Reasoning Methods, *Preprints of 2nd IFSA Congress* (1987) 143-146.
22. M. Mizumoto, Pictorial Representations of Fuzzy Connectives, Part I: cases of t-norms, t-conorms and averaging operators, *Fuzzy Sets and Syst.* **31** (1989) 217-242.
23. K. Nishimori, H. Tokutaka, K. Fujimura, S. Hirakawa, S. Hamano, S. Kishida and I. Naganori, Comparison of Fuzzy Reasoning Methods on Driving Control for Right or Left Turning of a Model Car, in *Proc. 5th IFSA Congress* (1993) 242-245.
24. R. Palm, D. Driankov, H. Hellendoorn. *Model Based Fuzzy Control*. (Springer-Verlag 1997).
25. T. Takagi and M. Sugeno. Fuzzy Identification of Systems and Its Applications to Modeling and Control, *IEEE Trans. on Syst., Man, and Cybernetics* **15**, 1 (1985) 116-132.
26. E. Trillas and L. Valverde, On Implication and

Indistiguishability in the Setting of Fuzzy Logic, J. Kacpryzk, R. R. Yager, Eds., *Management Decision Support Systems Using Fuzzy Sets and Possibility Theory*. (Verlag TÜV Rheinland, Köln, 1985) 198-212.

27. E. Trillas, On a Mathematical Model for Indicative Conditionals, in *Proc. 6th IEEE International Conf. on Fuzzy Systems* (1997) 3-10.

28. T. Tsukamoto. An approach to fuzzy reasoning method, M.M. Gupta, R.K. Ragade y R.R. Yager Eds., *Advances in Fuzzy Set Theory and Applications*. (North-Holland, Amsterdam 1979).

29. T. Yamakawa, Stabilization of an Inverted Pendulum by a High-Speed Fuzzy Logic Controller Hardware System, *Fuzzy Sets and Syst.* **32** (1989) 161-180.

30. L.X. Wang and J. Mendel, Generating Fuzzy Rules by Learning from Examples, *IEEE Trans. on Syst., Man, and Cybernetics* **22**, 6 (1992) 1414-1427.

31. L.X. Wang, *Adaptive Fuzzy Systems and Control. Design and Stability Analysis*. (Prentice Hall, 1994).

32. L.A. Zadeh, Fuzzy sets, *Information and Control* **8** (1965) 338-353.

33. L.A. Zadeh Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. on Syst., Man and Cybernetics* **3**, 1 (1973) 28-44

Appendix A: Graphical representation of the membership functions of the inferred fuzzy sets for the Implication Operators presented

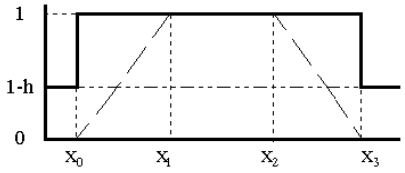

Fig. 17. I_1 (Dubois-Prade)

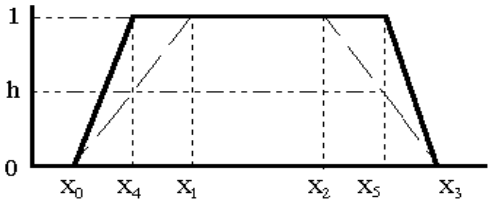

Fig. 18. I_2 (Goguen)

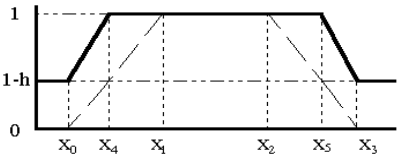

Fig. 19. I_3 (Lukasiewicz)

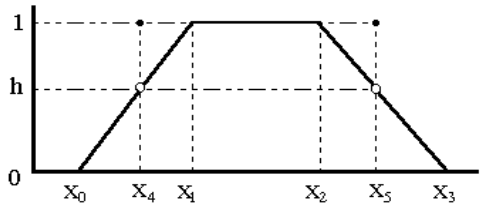

Fig. 20. I_4

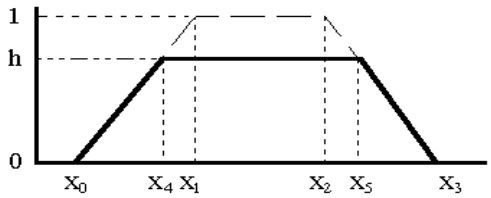

Fig. 21. I_5 (Logical Product)

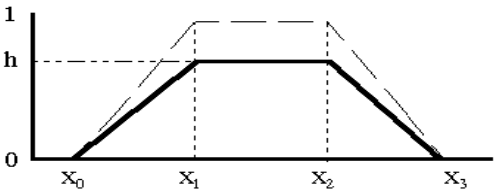

Fig. 22. I_6 (Algebraic Product)

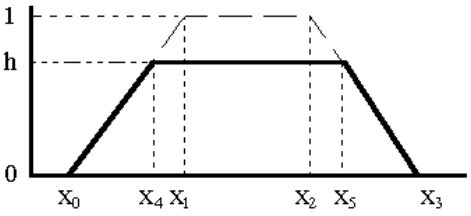

Fig. 23. I_7

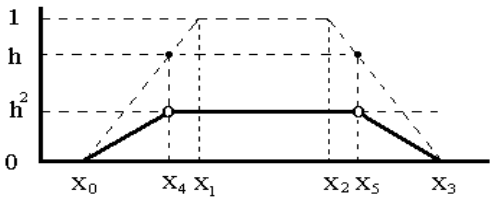

Fig. 24. I_8

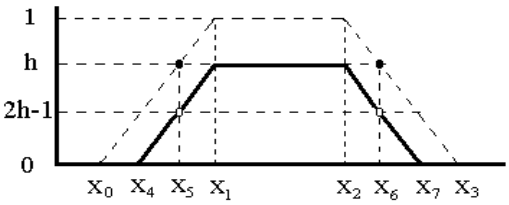

Fig. 25. I_9 when $h \geq ½$

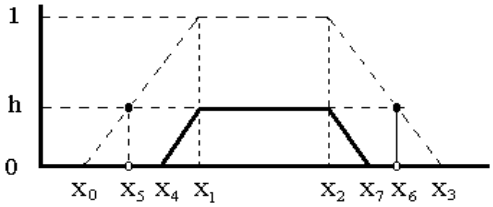

Fig. 26. I_9 when $h \leq ½$

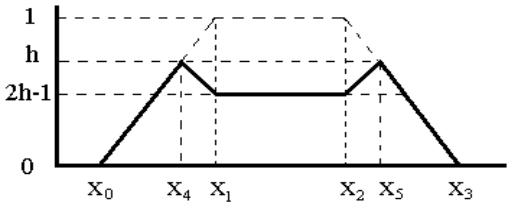

Fig. 27. I_{10} when $h \geq ½$

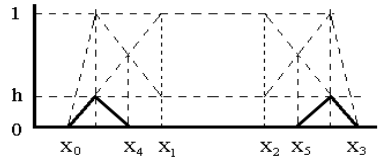

Fig. 28. I_{10} when $h \leq ½$

Appendix B: Applications Description

Three applications have been selected to analyze the behaviour of the Multi-operator FLC implementation methods: the fuzzy modeling of the simplest functional relation Y=X and of a three-dimensional surface, and the widely studied problem of controlling the *Inverted Pendulum.*

The selection of the first application is based on the studies developed by Cao and Kandel[4] which state that the independence between the application considered and the accuracy obtained by the FLC is a very important fact in the comparison of the influence of the fuzzy operators used to design it. Hence, in order to avoid the lack of generality in a fuzzy model, we are going to work with the simplest functional relation Y=X, making a fuzzy model of it in the interval [0,10].

In this case, the five linguistic labels {VS, S, M, L, VL} are used to make a fuzzy partition of the domain of the variables X and Y, where:

VS is very small,
S is small,
M is medium,
L is large,
VL is very large.

The corresponding membership functions presented by Cao and Kandel,[4] are shown in Figure 29:

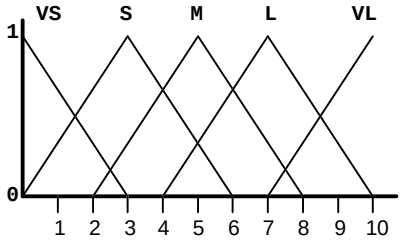

Fig. 29. Fuzzy partition considered
For the modeling of function Y=X

and the Knowledge Base presents the following five control rules:

If X is VS then Y is VS,
If X is S then Y is S,
If X is M then Y is M,
If X is L then Y is L
If X is VL then Y is VL

In this application, the set of evaluation data used to compute the accuracy of the implication operators is composed of 41 data pairs with a frequency of 0.25 in the interval [0,10].

The three-dimensional surface F_1 is shown in Figure 30, along with its mathematical expression.

$$F_1(x_1,x_2) = 10 \bullet \frac{x_1 + x_1 \bullet x_2}{x_1 - 2 \bullet x_1 \bullet x_2 + x_2} \qquad \text{(A.B.1)}$$

$$x_1, x_2 \in [0,1], F_2(x_1,x_2) \in [0,10]$$

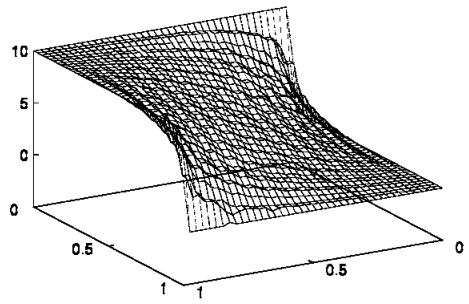

Fig. 30. Graphical representation of function F_1

The domains of the input and output variables of F_1 are fuzzy partitioned by using seven linguistic labels, called {NB, NM, NS, ZR, PS, PM, PB} where:

NB is negative,
NM is negative medium,
NS is negative small,
ZR is zero,
PS is positive small,
PM is positive medium,
PB is positive big.

Figure 31 shows the associated membership functions:

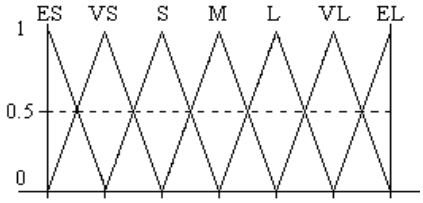

Fig. 31. Fuzzy partition considered
for the modeling of function F_1.

For the experiments developed with function F_1, a Mamdani-type Knowledge Base (KB) of 49 rules has been generated from a training data set by means of the Wang and Mendel generation process.[30] The KB generated is shown in Figure 32. The process considered is characterized by performing the rule generation following an inductive criterion relating to the covering of the data. Therefore, the KB obtained by this method is not dependent on the concrete Inference System used to make inference, which is a major requirement in order to adequately compare the behaviour of the implication operators. The training data set, consisting of 674 examples, has been obtained by generating the input variable values uniformly distributed in the variable

domains and by computing the associated output value using the expression of the function. Subsequently, a test data set, formed by 67 pieces of data, and obtained by generating the state variable values at random and computing the associated output variable value, will be used to measure the accuracy of the implication operators.

Table 13: Rule Base for F_1

		x_2						
		ES	VS	S	M	L	VL	EL
	ES	ES	ES	ES	ES	ES	ES	ES
	VS	EL	M	S	VS	VS	ES	ES
	S	EL	L	M	S	VS	VS	ES
x_1	M	EL	VL	L	M	S	VS	ES
	L	EL	VL	VL	L	M	S	ES
	VL	EL	EL	VL	VL	L	M	ES
	EL	EL	EL	EL	EL	EL	EL	ES

The inverted pendulum system[29] is shown in Figure 32. On the assumption that $|\Theta| \leq 30°$, the behaviour of the pendulum is achieved from the following equation:

$$m\frac{l^2}{3}\omega = \frac{l}{2}(-F + mg\sin\Theta - k\omega) \qquad \text{(A.B.2)}$$

with m being the mass of the pendulum, 2 l its length and $k\omega$ being an approximation of the friction strength.

The system state variables are the pendulum angle, Θ, and the change of angle, ω, whereas the control variable is the force F to apply over its gravity center. The universes of discourse for these variables are the following:

$\omega \in$ [-0.8645,0.8645] rad/s
$\Theta \in$ [-0.5283,0.5283] rad
$F \in$ [-3003.8,3003.8] Nw

In order to carry out our study, we have worked with a simulation model of the system using the parameters m = 5 kg and 2 l = 5 m.

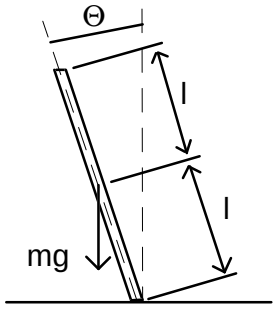

Fig. 30: Inverted pendulum.

The linguistic variables are partitioned by using the seven linguistic labels contained in the following set:[18,29]

{ NB, NM, NS, ZR, PS, PM, PL }

where N is negative, P is positive, B is big, M is medium, S is small and ZR is zero.

The membership functions corresponding to each element in the linguistic set have been obtained following the methodology proposed by Liaw and Wang.[18] The trapezoidal-shaped membership functions shown in Figure 33 are used by scaling the interval [-6,6] to the one corresponding to the specific variable.

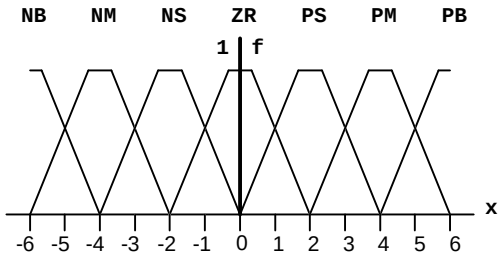

Fig. 33. The domain partition
in the inverted pendulum problem

The Knowledge Base used to control the system is constituted by the seven linguistic control rules shown in Table 14[29]. The set of evaluation data used to compute the SE has 200 data arrays in the form (value of Θ, value of ω, value of F) belonging to the intervals $\Theta \in$ [-0.2569,0.2569] rad, $\omega \in$ [-0.4244,0.4244] rad/s and $F \in$ [-1474.05, 1474.05] Nw in the inverted pendulum problem.

Table 14: Control rule map of F

		Θ						
		NB	NM	NS	ZR	PS	PM	PB
	NB							
	NM							
	NS			NS		ZR		
ω	ZR		NM		ZR		PM	
	PS			ZR		PS		
	PM							
	PB							

Appendix C: Application Results

Table 1. Exact Method for the X=Y application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	0.1372	0.0000	0.0000	*	0.1616	1.2561	0.1616	*	*	0.6585
D_2	1.2859	0.0962	1.6443	*	0.1860	0.9241	0.1860	*	*	0.6604
D_3	4.3750	4.3750	4.3750	*	0.1612	0.8689	0.1612	*	*	0.8197
D_4	4.3750	4.3750	4.3750	*	0.3768	0.8844	0.3768	*	*	0.9278
D_5	0.2220	0.0498	0.0498	0.0498	0.0498	0.0207	0.0498	0.0498	0.0207	0.0498
D_6	0.2622	0.2062	0.2062	0.2062	0.2062	0.2896	0.2062	0.2062	0.2896	0.2062
D_7	2.2874	2.0684	2.0684	2.0684	0.2141	0.1636	0.2141	0.2141	0.1636	0.2141
D_8	3.8089	3.4280	3.9436	1.4868	0.3030	0.2318	0.3030	0.2530	0.1155	0.2085

Table 2. Approximate Method with 25 points for the X=Y application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	0.1995	0.0191	0.0639	*	0.2068	0.6497	0.2068	*	*	0.3134
D_2	1.3792	0.2019	1.6510	*	1.5532	3.8708	1.5532	*	*	2.9883
D_3	4.3738	4.3738	4.3738	*	0.2057	0.3815	0.2057	*	*	0.4582
D_4	4.3738	4.3738	4.3738	*	0.3448	0.2200	0.3448	*	*	0.3103
D_5	0.2868	0.0623	0.0623	0.0670	0.0623	0.0316	0.0623	0.1019	0.6811	0.2970
D_6	0.3936	0.3207	0.3207	0.3910	0.3207	0.3807	0.3207	0.3497	0.5856	0.5179
D_7	2.3035	2.0432	2.0085	1.9196	0.2392	0.1793	0.2392	0.3122	1.3613	0.8358
D_8	3.8044	3.4061	3.9292	2.7448	0.2873	0.2106	0.2873	0.1891	1.0861	0.2242

Table 3. Approximate Method with 100 points for the X=Y application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	0.2054	0.0231	0.0877	*	0.1933	0.6834	0.1933	*	*	0.3223
D_2	1.3884	0.1079	1.7058	*	1.0897	1.1857	1.0897	*	*	2.0120
D_3	4.3738	4.3738	4.3738	*	0.3227	0.4015	0.3227	*	*	0.6272
D_4	4.3738	4.3738	4.3738	*	0.4004	0.2787	0.4004	*	*	0.3360
D_5	0.2907	0.0921	0.0921	0.0619	0.0921	0.0619	0.0921	0.0921	0.7085	0.2788
D_6	0.5875	0.5176	0.5176	0.5955	0.5176	0.5955	0.5176	0.5176	0.7820	0.7369
D_7	2.2578	2.0410	2.0046	1.8770	0.2506	0.1993	0.2506	0.2506	1.3786	0.7277
D_8	3.7940	3.4105	3.9311	2.7063	0.3256	0.2575	0.3256	0.2186	1.1086	0.2483

Table 4. Exact Method for the three dimensional surface application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	0.3494	1.9217	0.2943	*	1.9529	2.0073	1.9529	*	*	3.5258
D_2	3.1681	1.9340	3.5946	*	1.9551	1.9630	1.9551	*	*	3.5251
D_3	5.4630	5.4630	5.4630	*	0.3987	0.4569	0.3987	*	*	0.3987
D_4	5.4630	5.4630	5.4630	*	0.2608	0.2020	0.2608	*	*	0.1996
D_5	0.1338	0.0661	0.0661	0.0661	0.0661	0.0445	0.0661	0.0661	0.0445	0.0661
D_6	0.3875	0.3987	0.3987	0.3987	0.3987	0.4569	0.3987	0.3987	0.4569	0.3987
D_7	4.7566	4.7267	4.7267	4.7267	0.3481	0.3363	0.3481	0.3481	0.3363	0.3481
D_8	5.4067	5.3057	5.4200	5.0598	0.1872	0.1454	0.1872	0.1580	0.1138	0.1770

Table 5. Approximate Method with 25 points for the three dimensional surface application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	0.3621	1.9176	0.3172	*	1.9500	1.9772	1.9500	*	*	4.2172
D_2	3.1239	1.9341	3.5336	*	5.2868	5.4630	5.2868	*	*	5.4630
D_3	5.4630	5.4630	5.4630	*	0.4263	0.4569	0.4263	*	*	0.5919
D_4	5.4630	5.4630	5.4630	*	0.1935	0.1388	0.1935	*	*	0.1853
D_5	0.1338	0.0581	0.0579	0.0445	0.0579	0.0445	0.0579	0.0579	1.4925	0.2420
D_6	0.3875	0.4263	0.4263	0.4569	0.4263	0.4569	0.4263	0.4263	0.4263	0.5919
D_7	4.7566	4.7344	4.7165	4.6791	0.3377	0.3363	0.3377	0.3377	3.1969	0.9301
D_8	5.4059	5.3132	5.4185	5.2561	0.1381	0.0971	0.1381	0.1148	0.3990	0.1656

Table 6. Approximate Method with 100 points for the three dimensional surface application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	0.3504	1.9198	0.2978	*	1.9520	2.0016	1.9520	*	*	3.6713
D_2	3.1679	1.9333	3.5748	*	3.0742	4.3197	3.0742	*	*	4.9558
D_3	5.4630	5.4630	5.4630	*	0.4025	0.4612	0.4025	*	*	0.5357
D_4	5.4630	5.4630	5.4630	*	0.2448	0.1847	0.2448	*	*	0.1961
D_5	0.1286	0.0641	0.0638	0.0468	0.0638	0.0468	0.0638	0.0638	1.5210	0.2199
D_6	0.3860	0.4025	0.4025	0.4612	0.4025	0.4612	0.4025	0.4025	0.4025	0.5357
D_7	4.7543	4.7424	4.7245	4.6828	0.3450	0.3389	0.3450	0.3450	3.2231	0.7621
D_8	5.4069	5.3115	5.4195	5.2546	0.1757	0.1321	0.1757	0.1352	0.3927	0.1749

Table 7. Exact Method for the inverted pendulum application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	8665.6	6826.3	6826.3	*	20004.4	85485.1	20004.4	*	*	6413.2
D_2	29503.2	12851.6	36649.5	*	20004.4	71538.9	20004.4	*	*	6434.5
D_3	183021.0	183021.0	183021.0	*	20289.1	70592.9	20289.1	*	*	20289.1
D_4	183021.0	183021.0	183021.0	*	6580.7	57705.7	6580.7	*	*	8244.4
D_5	6425.7	6425.7	6425.7	6425.7	6425.7	6425.7	6425.7	6425.7	6425.7	6425.7
D_6	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1
D_7	135297.6	135297.6	135297.6	135297.6	20004.4	20004.4	20004.4	20004.4	20004.4	20004.4
D_8	169490.6	169310.0	171837.0	137589.5	6908.7	6425.7	6908.7	6518.0	6916.6	7963.8

Table 8. Approximate Method with 25 points for the inverted pendulum application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	14549.9	11315.3	11172.3	*	23763.9	57908.1	23763.9	*	*	94536.9
D_2	31724.2	14561.4	37666.2	*	336706.1	570763.5	336706.1	*	*	644750.5
D_3	183021.0	183021.0	183021.0	*	24205.3	28121.5	24205.3	*	*	64306.9
D_4	183021.0	183021.0	183021.0	*	6548.5	6481.7	6548.5	*	*	8532.4
D_5	12479.6	9720.6	9720.6	12479.6	9720.6	12479.6	9720.6	9720.6	37448.7	26034.2
D_6	28121.5	24205.3	24205.3	28121.5	24205.3	28121.5	24205.3	24205.3	24205.3	64306.9
D_7	135764.4	135377.5	135578.9	135639.7	24390.5	27601.8	24390.5	21923.3	156800.3	59205.3
D_8	170005.3	169898.1	172329.8	161332.5	6826.7	6425.7	6826.7	7545.6	14306.8	8217.7

Table 9. Approximate Method with 100 points for the inverted pendulum application.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
D_1	8976.6	6919.3	7063.6	*	22701.2	59902.0	22701.2	*	*	105137.4
D_2	29727.3	12910.5	36820.1	*	93856.9	136253.1	93856.9	*	*	328995.4
D_3	183021.0	183021.0	183021.0	*	20289.1	20289.1	20289.1	*	*	42820.9
D_4	183021.0	183021.0	183021.0	*	6580.4	6496.5	6580.4	*	*	8244.4
D_5	6563.8	6425.7	6425.7	6425.7	6425.7	6425.7	6425.7	6425.7	37773.9	22064.4
D_6	20441.0	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1	20289.1	42820.9
D_7	135513.2	135297.6	135502.2	135502.2	22510.8	22510.8	22510.8	22510.8	163340.3	68016.5
D_8	169614.5	169427.7	171938.0	160459.0	6908.1	6425.6	6908.1	7507.5	14306.5	7963.5