JAVA FUZZY LOGIC TOOLBOX FOR INDUSTRIAL PROCESS CONTROL

Bruno Sielly J. Costa*, Clauber G. Bezerra†, Luiz Affonso H. G. de Oliveira‡

*Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Norte
Campus Caicó
Caicó, RN, Brasil

†Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Norte
Campus Santa Cruz
Santa Cruz, RN, Brasil

‡Universidade Federal do Rio Grande do Norte
Departamento de Engenharia de Computação e Automação
Natal, RN, Brasil

Emails: bruno.costa@ifrn.edu.br, clauber.bezerra@ifrn.edu.br, affonso@dca.ufrn.br

Abstract— This paper describes the design, implementation and application of a fuzzy logic toolbox for industrial process control based on Java language, supporting communication through the OPC industrial protocol. The toolbox is written in Java and is completely independent of any other platforms. It provides easy and functional tools for modelling, building and editing complex fuzzy inference systems and using such logic systems to control a large variety of industrial processes.

Keywords— fuzzy logic, industrial process control, OPC.

1 Introduction

Since the decade of 1970 with the application implemented by Mamdani (1974), the fuzzy logic is being used on control of industrial processes of a diversity of kinds.

One of the main potentialities of fuzzy logic, when compared to other schemes that manipulates inaccurate data, such as neural networks, is that its bases of knowledgement, which are in the format of inference rules, are very easy of examination and understanding. This format of rule also makes easy maintaining and updating the base of knowledgement.

The purpose of this work is to present a complete and powerful environment for manipulating the fuzzy theory, applying its bases and concepts on industrial control tasks, through a set of tools of rich content and high level of integration with real industrial processes.

In the literature, we are able to find a few toolboxes implemented to work with fuzzy logic. Hall and Hathaway (1996) describes a fuzzy logic toolbox integrated to *MathWorks Matlab* platform, with a friendly graphic user interface for building and editing fuzzy inference systems. Chuan et al. (2004) presents a fuzzy logic toolbox based on *Scilab* language, and shows it as a free alternative to the software from MathWorks.

2 Fuzzy Logic

Two of the main aspects of the imperfection of information are the imprecision and the uncertainty. This two characteristics are intrinsically linked and opposite to each other. The most known theories to treat imprecision and uncertainty are the set theory and probabilities theory, respectively.

The fuzzy set theory started to be developed at the decade of 1960 by Zadeh (1965), intending to treat the nebulous aspect of the information. This theory, being less restrictive, may be considered more suitable for treating information provided by human beings than other theories.

The fuzzy logic is an artificial intelligence technique, which incorporates the capacity of a human specialist to modelling the operation of a control system, working similar to a deductive reasoning, controlling industrial processes with nonlinear characteristics, relating plant variables described on the controller. The fuzzy logic allows the treatment of expressions that involves quantities accurately.

To obtain the mathematical formalization of a fuzzy set, is needed to remember that any fuzzy set may be characterized by a membership function, whose definition is given below:

Definition 1: A fuzzy subset A from the universe $\mathbf{U}$ is characterized by a membership function $\mu_A : \mathbf{U} \rightarrow [0, 1]$, where μ_A indicates how pertinent the element x is on A, and also may be interpreted as the degree of compatibility to the associated attribute.

$$A = \{x,\ \mu_A(x) \mid x \in \mathbf{U} \text{ and } \mu_A : \mathbf{U} \rightarrow [0, 1]\}$$

Definition 2: The union between two fuzzy sets A and B may be defined as the fuzzy set F, whose membership function is given by:

$$x \in \mathbf{U} \mid \mu_{A \cup B}(x) = max\{\mu_A(x),\ \mu_B(x)\}$$

Definition 3: The intersection between two fuzzy sets A and B may be defined as the fuzzy set F, whose membership function is given by:

$$x \in \mathbf{U} \mid \mu_{A\cap B}(x) = min\{\mu_A(x),\ \mu_B(x)\}$$

Definition 4: An aggregation is a n-order operation, $A\,[0,1]^n \rightarrow [0,1]$, satisfying:

i) $A(0,0,\ldots,0) = 0$

ii) $A(1,1,\ldots,1) = 1$

iii) $A(x_1,x_2,\ldots,x_n) \geq A(y_1,y_2,\ldots,y_n)$, if $x_i \geq y_i\ \forall i$

Definition 5: A linguistic variable X in the universe $\mathbf{U}$, is a variable that assumes values described by linguistic concepts, represented by fuzzy sets of $\mathbf{U}$.

A fuzzy rule-based system is valid by linguistically presenting very complex relations, or relations not enough well understood to be described by accurate mathematical models. A fuzzy inference system may be illustrated as in figure 1.

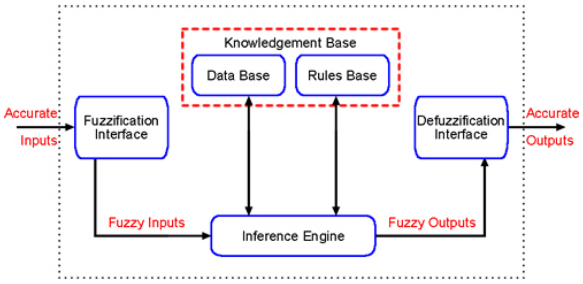

Figure 1: Fuzzy Inference System

The inference may be synthesized in four steps. In a first instance, the degrees of compatibilities of the variables are compared with its respective antecedents. Usually, fuzzy inference systems receive accurate inputs. In that case, it is necessary to use a *fuzzification* interface, relating accurate values to a fuzzy value. The fuzzy value is obtained through membership functions designated to each system input variable.

After the fuzzification step, a mechanism called *inference engine* is responsible to perform system inferences, supported by a knowledgement base, subdivided in data base and rules base. The data base stores the definitions about discretization and normalization of the speech universe, as well the definitions about the membership functions of fuzzy terms. The rules base is composed of *If-Then* structures, previously defined in the inference system. The inference engine is the entity responsible to determine the degree of activation of each rule, and generate a fuzzy value to output.

Based in the degree of activation, is determined the consequence produced by a specific rule. Too often fuzzy inference systems contain more than one rule. Each rule produces a consequence, and the global result from the inference step will depend on the combination of those consequences. This step is called *aggregation*, which has as result a fuzzy set.

The last step of the inference process is the *defuzzification*. The output processor generates a fuzzy set that describes the system output, and is represented by a real number. The Mamdani's fuzzy inference method, which output is a fuzzy set, and the Takagi-Sugeno method, which is computationally less expensive, are the most used inference methods.

The Mamdani's procedure comprehends the identification of the variables domain in a correspondent at the speech universe. The fuzzy output evolves to a non-fuzzy output. There are plenty of methods for defuzzification, each one presenting suitable results for specific situations. The most used defuzzification methods in the literature are *centroid*, *bisector*, *middle of maximum*, *smallest of maximum* and *largest of maximum*.

At the Takagi-Sugeno's method, the consequent of each rule is a function of input variables, and the system output is a real number. There is thus no need for using an output processor. The output level z_i of each rule is weighted by the firing strength w_i of the rule. The final output from the system z is the weighted average of all rule outputs, computed as

$$z = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i}$$

where n is the number of fuzzy rules.

3 Ole for Process Control

OPC is the acronym to Ole for Process Control. OPC is an open communication protocol, based on OLE COM/DCOM technology from Microsoft, which aims to allow the vertical integration among different systems into an organization (Opc-Foundation, 2003).

OPC consists of a server program, usually provided by the Programmable Logic Controller manufacturer, which communicates with PLCs through a proprietary protocol and provides data in OPC standard. The client instead, only needs the OPC client driver installed and configured. The server and the client may be installed on the same machine simultaneously.

The figure 2 illustrates a basic example of a set of applications accessing data over the network, through well defined OPC interfaces.

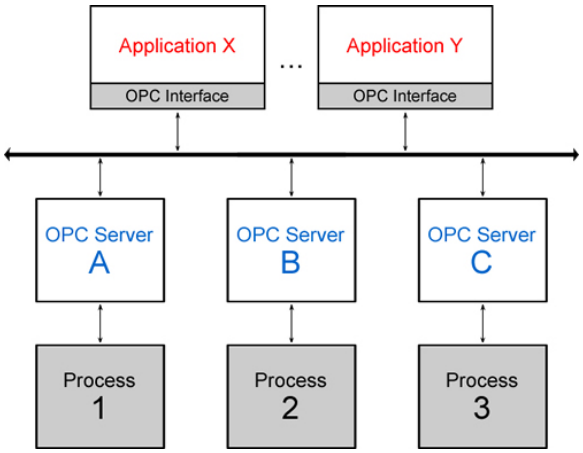

Figure 2: Applications working with many OPC servers

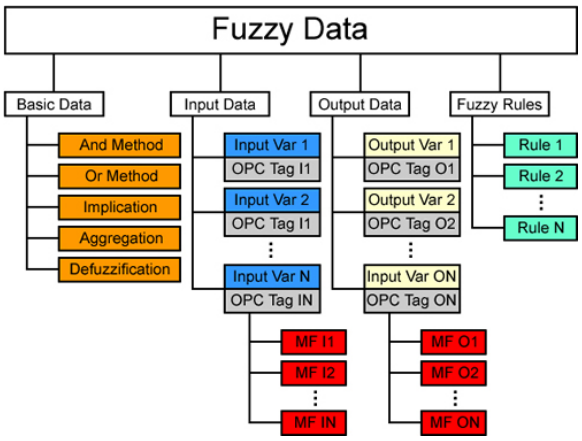

Figure 3: Java Fuzzy Logic Toolbox Data Structure

4 Motivations of this work

MathWorks Matlab Toolbox (MathWorks, 2009) for fuzzy systems presents a good approach to building fuzzy sets and fuzzy rule-based systems, with a powerful graphical user interface, which makes very easy the building/editing process. Nevertheless, this alternative is still not suitable for industrial environments.

The fuzzy logic toolbox from Matlab does not have intrinsically an integration tool for industrial processes and protocols, not being possible to directly read and write data from the plant. Although there are tools for that kind of connection on Matlab platform, this makes the process more difficult to implement and manage. Another crucial factor is that Matlab is a very expensive software, specially for non-academic environments.

The system proposed in this paper, presents itself as a good alternative for industrial fuzzy control, being a proprietary platform-independent software and able to communicate directly with the industrial plant, and linking its input and output variables to the process variables.

5 Details of Implementation

The system was completely developed in Java language. All parameters of configuration for the fuzzy inference system in execution are stored in a structure file, saved with a '.FUZ' extension.

The fuzzy data saved and loaded in memory for execution of the system may be divided in four distinct groups, as shown on figure 3.

Basic Data: Include and/or methods, implication, aggregation and defuzzification methods. Most of the important methods were implemented in the toolbox, as listed below:

- And methods: minimum, product;
- Or methods: maximum, probabilistic or;
- Implication: minimum, product;
- Aggregation: maximum, sum, probabilistic or;
- Defuzzification: centroid, bisector, middle of maximum, largest of maximum, smallest of maximum.

Input and Output Data: Made up of several input/output entries. Each input/output variable may be linked or not to a tag from OPC, and each variable has a set of associated membership functions. Three of the most used types of function were implemented in this work:

- Triangular Function: three parameters are defined to model the function. t_1 represents the first non-zero point, t_2 the middle point and t_3 the last non-zero point of the triangle. Mathematically, the triangular function is defined as follows:

$$fF(x) = \begin{cases} \frac{x-t_1}{t_2-t_1}, & \text{if } t_1 \leq x \leq t_2 \\ \frac{t_3-x}{t_3-t_2}, & \text{if } t_2 \leq x \leq t_3 \\ 0, & \text{otherwise} \end{cases}$$

- Trapezoidal Function: four parameters are defined to model the function. r_1 represents the first non-zero point, r_2 and r_3 the interval where the function pertinence is equals to 1, and r_4 the last non-zero point of the trapeze. Mathematically, the trapezoidal function is defined as follows:

$$fF(x) = \begin{cases} \frac{x-r_1}{r_2-r_1}, & \text{if } r_1 \leq x \leq r_2 \\ 1, & \text{if } r_2 \leq x \leq r_3 \\ \frac{r_4-x}{r_4-r_3}, & \text{if } r_3 \leq x \leq r_4 \\ 0, & \text{otherwise} \end{cases}$$

- Gaussian Function: often referred as Gauss distribution. Three parameters are defined to model the function. g_1 and g_2 represents the function bounds and g_3 the variance, indicating the function width. In this work, the Gaussian function is defined as follows:

$$fF(x) = \frac{2}{5\sqrt{2\pi}} \exp\left(\frac{(-x+0.5g_2+0.5g_1)^2}{2g_3}\right)$$

Fuzzy Rules: If-Then inference rules, associated to input/output variables, logic connectors (and/or/not) and fuzzy values.

6 System Description

The Java Fuzzy Logic Toolbox was completely conceived based on windows and simplified graphic environment, making intuitive the creation of fuzzy inference systems. The developed system has basically five main graphic environments for editing, viewing and running fuzzy inference systems:

- Basic Fuzzy Inference System Editor: manipulates the high-level information about the system, such as number and names of variables, links to OPC tags, basic configuration parameters;
- Membership Function Editor: allows advanced edition of membership functions for input/output variables;
- Fuzzy Inference Rules Editor: tool for automatically creating inference rules based on system variables and fuzzy values associated;
- Rules Viewer: used as a diagnostic, it can show which rules are active, or how individual membership function shapes are influencing the results;
- Control Module: interface between an existent fuzzy inference system and the process. The main screens are shown at figure 4.

Such graphic environments are dynamically linked, and all changes made using one of them are transmitted to the other.

7 System Validation

For validating the Java Fuzzy Toolbox, two approaches were utilized. In a first moment, an existing problem described by Sankar and Kumar (2006) was modelled in both toolboxes: Matlab Toolbox and Java Toolbox. The results were compared by graphical analysis. In a second moment, a control task was implemented on Java Fuzzy Toolbox, and the results are shown in subsections below.

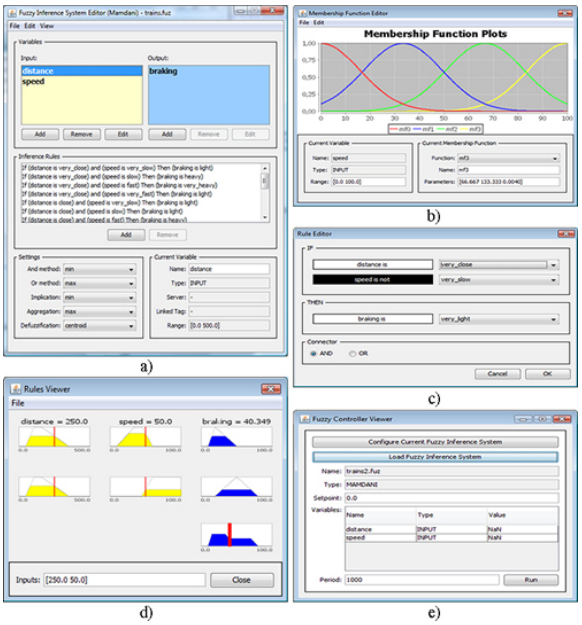

Figure 4: Main Screens of Java Fuzzy Toolbox: a) Basic Inference System Editor b) Membership Function Editor c) Rules Editor d) Rules Viewer e) Control Module

7.1 Matlab Comparison

The example problem focuses on a new approach to braking system in train, by using fuzzy logic. Generally the Indian railways use two persons to operate a train and they employ a manual procedure for stopping the train in each station. The proposed fuzzy logic controller helps in reduction of manpower for the train operation.

- Input Variables - Distance and Speed
- Output Variables - Braking power (or % of braking)
- Membership Functions (Shown at figure 5)
  - Distance: Very Close, Close, Far, Very Far;
  - Speed: Very Slow, Slow, Fast, Very Fast;
  - Braking: Very Light, Light, Heavy, Very Heavy;
- Inference Rules
  1. If distance is **VERY CLOSE** and speed is **VERY SLOW** then braking is **LIGHT**
  2. If distance is **VERY CLOSE** and speed is **SLOW** then braking is **HEAVY**
  3. If distance is **VERY CLOSE** and speed is **FAST** then braking is **VERY HEAVY**
  4. If distance is **VERY CLOSE** and speed is **VERY FAST** then braking is **LIGHT**

5. If distance is **CLOSE** and speed is **VERY SLOW** then braking is **LIGHT**
6. If distance is **CLOSE** and speed is **SLOW** then braking is **LIGHT**
7. If distance is **CLOSE** and speed is **FAST** then braking is **HEAVY**
8. If distance is **CLOSE** and speed is **VERY FAST** then braking is **VEAR HEAVY**
9. If distance is **FAR** and speed is **VERY SLOW** then braking is **LIGHT**
10. If distance is **FAR** and speed is **SLOW** then braking is **VERY LIGHT**
11. If distance is **FAR** and speed is **FAST** then braking is **LIGHT**
12. If distance is **FAR** and speed is **VERY FAST** then braking is **HEAVY**
13. If distance is **VERY FAR** and speed is **VERY SLOW** then braking is **VERY LIGHT**
14. If distance is **VERY FAR** and speed is **SLOW** then braking is **VERY LIGHT**
15. If distance is **VERY FAR** and speed is **FAST** then braking is **LIGHT**
16. If distance is **VERY FAR** and speed is **VERY FAST** then braking is **LIGHT**

The other chosen parameters for the fuzzy system were: **And Method** = *minimum*, **Implication** = *minimum*, **Aggregation** = *maximum* and **Defuzzification** = *centroid*. Those parameters were used in all implementations ahead.

To comparing results, two sets of one-hundred randomic real numbers were generated and applied to each input variable for both systems. The graphics below represents the output forms, described as follows: the figure 6 illustrates the outputs (Braking Power) for a Mamdani's inference system, with the red series representing the Java Toolbox output and the blue series representing the Matlab Toolbox output; the figure 7 illustrates the outputs for a Takagi-Sugeno's inference system, with the red series representing the Java Toolbox output and the blue series representing the Matlab Toolbox output; the figure 8 shows the positive percent error resulting from Mamdani's system implemented on Java toolbox compared to Matlab Toolbox; the figure 9 shows the positive percent error resulting from Takagi-Sugeno's system implemented on Java toolbox compared to Matlab Toolbox.

7.2 Control Task

At this subsection, a real control problem was used to validate the Java Toolbox.

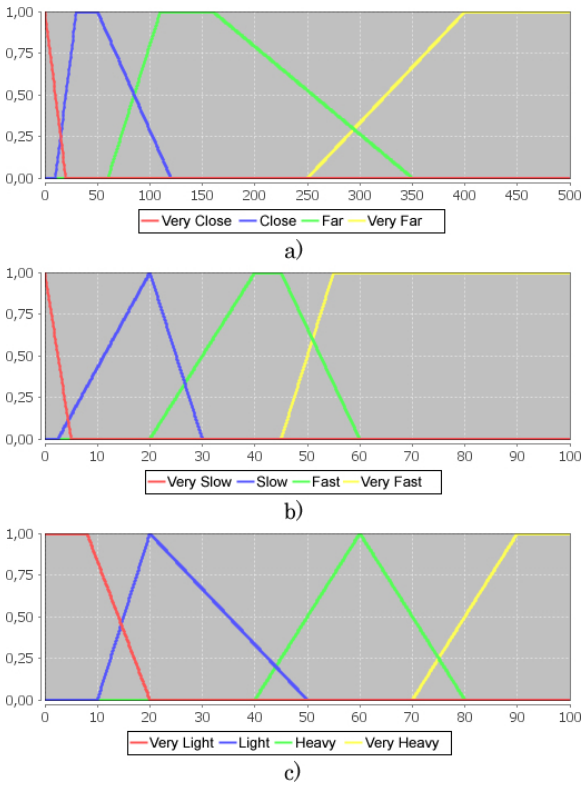

Figure 5: Membership Functions for Braking System in Trains: a) Distance b) Speed c) Braking Power

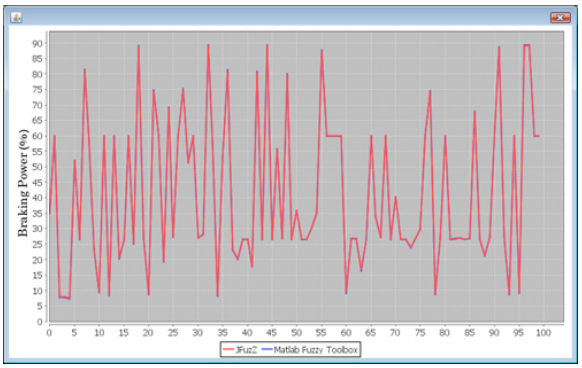

Figure 6: Outputs for Mamdani's Inference System

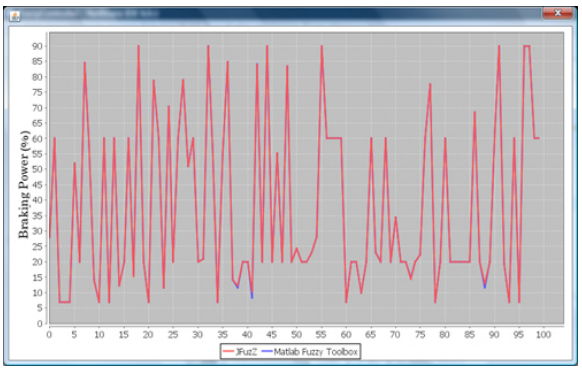

Figure 7: Outputs for Takagi-Sugeno's Inference System

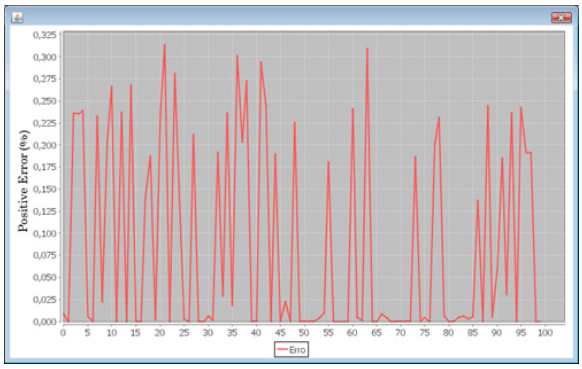

Figure 8: Error for Mamdani's system

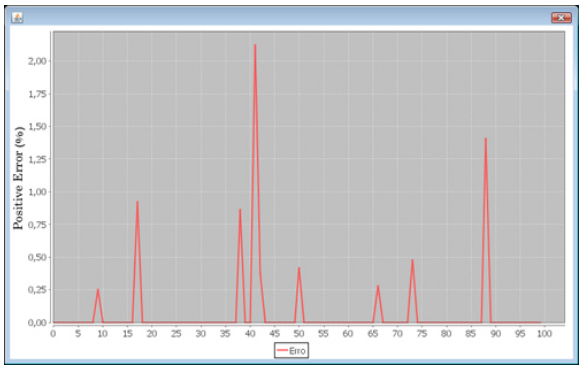

Figure 9: Error for Takagi-Sugeno's system

A common control problem in petrochemical process industries is the control of liquid levels in storage tanks and reaction vessels. In minor proportions, we can solve this problem using an experiment plant, referenced by Quanser Coupled Water Tanks (Quanser, 2004).

The "Two Tank Module" consists of a pump with a water basin. The pump thrusts water vertically to two quick connect, normally closed orifices "Out1" and "Out2". Two tanks mounted on the front plate are configured such that flow from the first tank flows into the second tank and outflow from the second tank flows into the main water basin. The case study for the Java Fuzzy Toolbox is design an intelligent control system to regulate the water level in the second tank, with the set-point equals to 10 centimeters. For this problem, three situations were implemented for results comparison.

In the first situation, the control problem was treated with a PI controller, implemented on a PLC, model ZAP 900 from HI Tecnologia (HI-Tecnologia, 2009). In that case, the proportional gain KP was fixed in 10 and the integral time Ti was fixed in 0.5.

In a second moment, a direct fuzzy control was applied to the plant. The model of the direct fuzzy controller is described below, and the membership functions are shown on figure 10.

- Input Variables - Error, Error Variation
- Output Variable - Output Voltage
- Membership Functions
  - Error: Negative High, Negative Low, Zero, Positive Low, Positive High;
  - Error Variation: Negative, Zero, Positive;
  - Output Voltage: Zero, Low, High;
- Inference Rules
  1. If error is **NEGATIVE HIGH** and errorvariation is **NEGATIVE** then outputvoltage is **HIGH**
  2. If error is **NEGATIVE HIGH** and errorvariation is **ZERO** then outputvoltage is **HIGH**
  3. If error is **NEGATIVE HIGH** and errorvariation is **POSITIVE** then outputvoltage is **LOW**
  4. If error is **NEGATIVE LOW** and errorvariation is **NEGATIVE** then outputvoltage is **LOW**
  5. If error is **NEGATIVE LOW** and errorvariation is **ZERO** then outputvoltage is **LOW**
  6. If error is **NEGATIVE LOW** and errorvariation is **POSITIVE** then outputvoltage is **ZERO**
  7. If error is **ZERO** and errorvariation is **NEGATIVE** then outputvoltage is **LOW**
  8. If error is **ZERO** and errorvariation is **ZERO** then outputvoltage is **ZERO**
  9. If error is **ZERO** and errorvariation is **POSITIVE** then outputvoltage is **ZERO**
  10. If error is **POSITIVE LOW** then outputvoltage is **ZERO**
  11. If error is **POSITIVE HIGH** then outputvoltage is **ZERO**

The last experiment consisted in a simple adaptive PID controller. In this work, there is a Intelligent Control block (inserted in Supervision layer) that sets the proportional gain KP according to the error value. The block diagram of this approach is shown on figure 11 and the architecture of the control system is shown at figure 12.

The last fuzzy controller is described as follows, and the membership functions for the input and output variables are shown on figure 13.

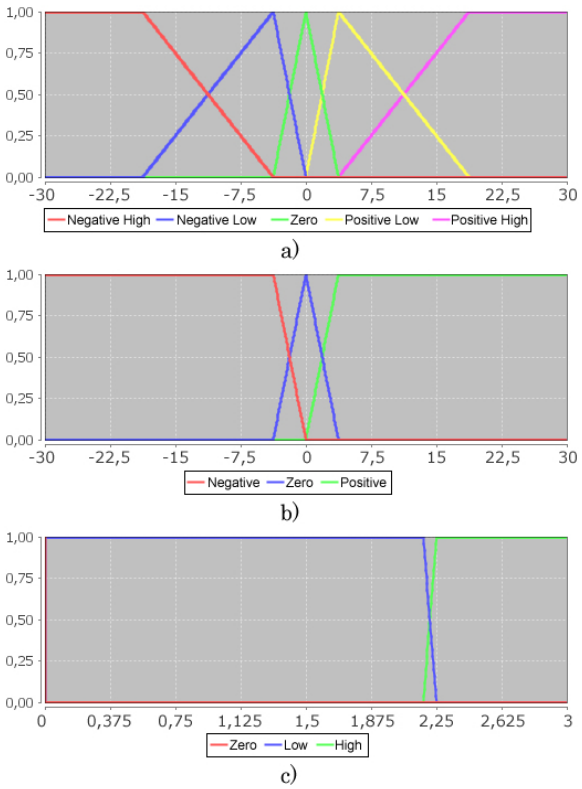

Figure 10: Membership Functions for Direct Fuzzy Control: a) Error b) Error Variation c) Control Signal

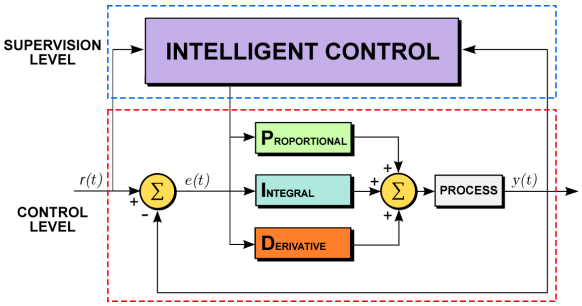

Figure 11: Block Diagram of the Adaptive Control Approach

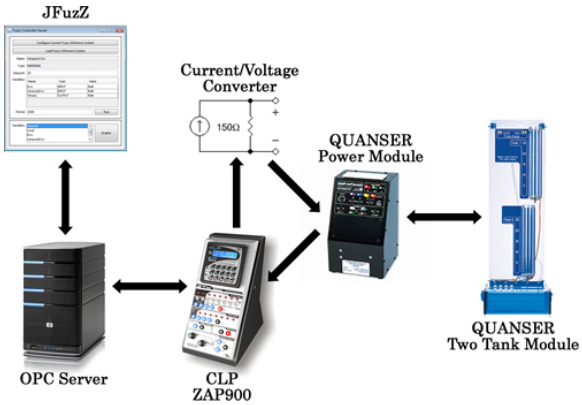

Figure 12: Control System's Architecture

- Input Variable - Error
- Output Variable - KP
- Membership Functions
  - Error: Small, Big, Very Big;
  - KP: Low, High, Very High;
- Inference Rules
  1. If error is **SMALL** then kp is **LOW**
  2. If error is **BIG** then kp is **HIGH**
  3. If error is **VERY BIG** then kp is **VERY HIGH**

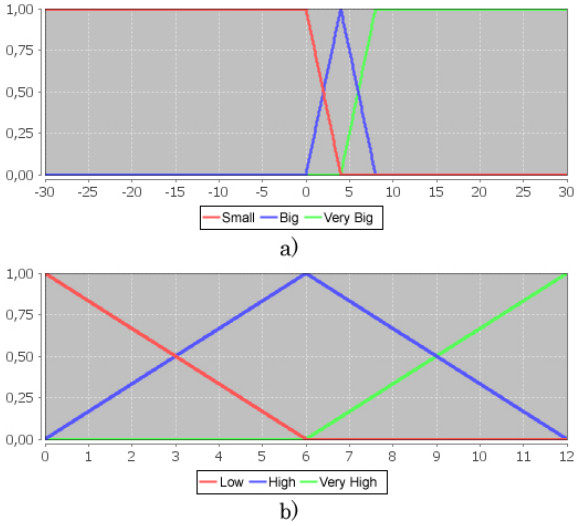

Figure 13: Membership Functions for Fuzzy PID Control: a) Error b) KP

For results comparison, the figure 14 illustrates the tank level (cm) behaviour for the three situations above, the figure 15 measures the positive error resulting from the three control approaches and the table 1 shows the detailed resulting values.

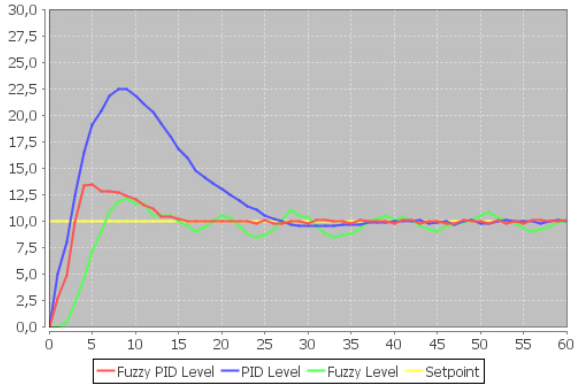

Figure 14: Tank Level (cm) Behaviour for Three Different Control Approaches

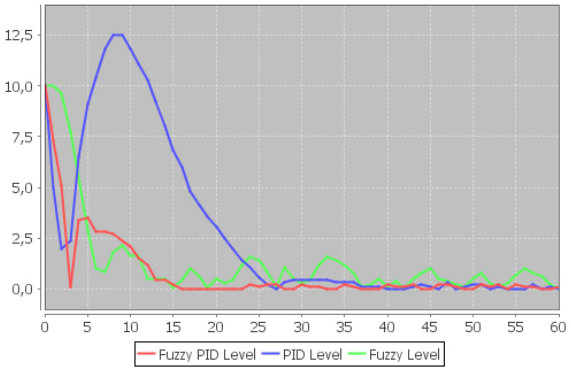

Figure 15: Positive Error from Three Different Control Approaches

Table 1: Detailed Result Values

Controller	Overshoot (%)	Settling Time at 2% (s)	Rise Time (s)
PID	125.0	27.0	2.5
Fuzzy	24.0	∞	3.0
Fuzzy PID	34.0	15.0	6.0

8 Conclusion

The Java Fuzzy Logic Toolbox is a useful software for constructing fuzzy logic systems and applying them on control tasks for industrial processes. Another feature of the toolbox is that it is a proprietary platform-independent, based on Java, with a powerful functions set and friendly interface. The results shown that the implemented system generated high accurate outputs, compared with Matlab Fuzzy Toolbox. The Java Toolbox was also efficient solving the tank level control problem, specially if cascaded to a PI controller. The OPC protocol compatibility increments the field of applicability and the space of usefulness of this tool, allowing direct communication with common industrial processes.

Acknowledgements

Thanks to Computing Engineering and Automation Laboratory and Petroleum Automation Laboratory, both in Federal University of Rio Grande do Norte for providing the necessary structure to developing this research.

References

Chuan, F., Zengqi, S. and Ling, S. (2004). Design and implementation of scilab fuzzy logic tooibox, *IEEE International Symposium on Computer Aided Control Systems Design*.

Gharieb, W. and Nagib, G. (2001). Fuzzy intervention in pid controller design, *IEEE International Symposium On Industrial Electronics*.

Hall, L. and Hathaway, R. (1996). Fuzzy logic toolbox - software review, *IEEE Transactions on Fuzzy Systems*, Vol. 4.

HI-Tecnologia (2009). http://www.hitecnologia.com.br.

Mamdani, E. (1974). Application of fuzzy algorithms for simple dynamic plant, *Proceedings of Institute of Electrical Engineering*.

MathWorks (2009). Matlab and simulink for technical computing. Available on http://www.mathworks.com.

Opc-Foundation (2003). Opc data access custom interface specification 3.0. Available on http://www.opcfoundation.org.

Pedrycz, W. (1993). *Fuzzy Control and Fuzzy Systems*, 2nd edition edn, Research Studies Press.

Quanser (2004). *Coupled tanks user manual*.

Ramakrishna, G. and Rao, N. D. (1999). Implementation of a fuzzy logic scheme for q/v control in distribution systems, *IEEE Power Engineering Society 1999 Winter Meeting* .

Sankar, G. and Kumar, S. (2006). Fuzzy logic based automatic braking system un trains, *India International Conference on Power Electronics*.

X. Fang, T. Shen, X. W. and Zhou, Z. (2008). Application and research of fuzzy pid in tank systems, *Fourth International Conference on Natural Computation*.

Zadeh, L. (1965). Fuzzy sets, *Information and Control* **8**: 338–353.