Novel Genetic Optimization of Membership Functions of Fuzzy Logic for Speed Control of a Direct Current Motor for Hardware Applications in FPGAs

Yazmin Maldonado, Oscar Castillo, Patricia Melin

Abstract:

This paper proposes a novel method for genetic optimization of triangular and trapezoidal membership functions of fuzzy systems, for hardware applications such as the FPGA (Field Programmable Gate Array). This method consists in taking only certain points of the membership functions, with the purpose of giving more efficiency to the algorithm. The genetic algorithm was tested in a fuzzy controller to regulate engine speed of a direct current (DC) motor, using the Xilinx System Generator (XSG) toolbox of Matlab, which simulate VHDL (Very High Description Language) code.

Keywords: *genetic algorithms, fuzzy, controller, Matlab, Simulink, Xilinx System Generator, VHDL, FPGA.*

1. Introduction

Fuzzy logic controllers are used successfully in many application areas, and these include control, classification, etc. [1],[2],[3],[4],[5],[6],[7]. These systems based on rules incorporate linguistic variables, linguistic terms and fuzzy rules. The acquisition of these rules is not an easy task for the expert and is of vital importance in the operation of the controller.

The process of adjusting these linguistic terms and rules is usually done by trial and error, which implies a difficult task, and for this reason there have been methods proposed to optimize those elements that over time have taken importance, such as genetic algorithms [8],[9],[10].

A Genetic Algorithm (GA) [9],[10] is a stochastic optimization algorithm inspired by the natural theory of evolution. From a principle proposed by Holland [9], GAs have been used successfully to manage a wide variety problems such as control, search, etc. [11].

This paper proposes a novel method for genetic optimization of the triangular and trapezoidal membership functions of a fuzzy logic system for hardware applications such as FPGA (Field Programmable Gate Array). This method involves taking only a small number for points of the membership functions in order to give greater efficiency to the algorithm. The GA has been tested in a fuzzy logic controller to regulate the speed engine direct current (DC) using the Matlab [12] platform and XSG [13] with good results.

This paper is organized as follows: in section 2 we present an introductory explanation of Genetic Algo rithms, Fuzzy Inference Systems and FPGAs, section 3 describes the novel method for genetic optimization of membership functions for FLC in FPGAs, the test and result the novel genetic optimization of membership functions for FLC for speed regulate the motor DC are shown in section 4. Finally, section 5 presents the conclusions.

2. Preliminaries

The Genetic Algorithm is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals to evolve under specified selection rules to a final state that maximizes the “fitness” (i.e. minimizes the cost function) [14].

A GA is inspired by the mechanism of natural selection where stronger individuals are likely the winners in a competitive environment. Here the GA uses a direct analogy of such natural evolution. Through the genetic evolution method, an optimal solution can be found and represented by the final winner of the genetic game [15].

Throughout a genetic evolution, the fitter chromosome has a tendency to yield good quality offspring, which means a better solution to any problem. In a practical GA application, a population pool of chromosomes has to be installed and these can be initially randomly set. The size of this population varies from one problem to another. In each cycle of genetic operations, termed as an evolving process, a subsequent generation is created from the

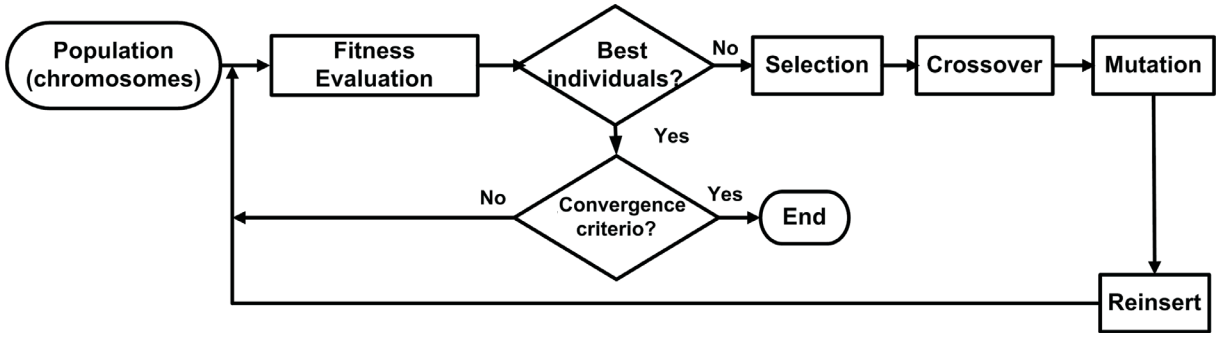

Fig. 1. GA cycle.

chromosomes in the current population. This can only succeed if a group of these chromosomes, generally called "parents is selected *via* a specific selection routine. The genes of the parents are mixed for the production of offspring in the next generation. It is expected that from this process of evolution, the better chromosome will create a larger number of offspring, and thus has a higher chance of surviving in the subsequent generation, emulating the survival of the fittest mechanism in nature. Figure 1 shows the GA cycle.

Of course, the GA is not the best way to solve every problem, GAs have proven to be a good strategy because of its optimal results in several areas of application [3], [16].

The GA has applications in a wide variety of fields to develop solutions to complex problems, including optimization of fuzzy systems, offering them learning and adaptation, they are commonly called genetic fuzzy systems or fuzzy system hybrids.

Fuzzy systems have been used more and more, because they tolerate imprecise information and can be used to model nonlinear functions of arbitrary complexity. A fuzzy system (FIS) consists of three stages: Fuzzification, Inference and Defuzzification [17]. We describe below these stages.

Fuzzification: Is the interpretation of input values (numeric) by the fuzzy system, and the obtained output are fuzzy values.

Definition 1. Let $x \in X$ be a linguistic variable and $T_i(x)$ a fuzzy set associated with a linguistic value T_i. The translation of a numeric value x corresponds to a linguistic value associated with a degree of membership, $x \rightarrow u_{T_i}(x)$, and this is known as Fuzzification. The membership degree $u_{T_i}(x)$ represents a value of membership to a fuzzy set [18].

Inference: Is basically like the brain of the system, here the rules of the form if-then that describe this behavior are used [2]. For example:

$$\text{If } x_1 \text{ is } A_1 \text{ and } \ldots \text{ and } x_n \text{ is } A_n \text{ Then } y \text{ is } B \qquad (1)$$

where $x_1 \ldots x_n$ are the inputs, $A_1 \ldots A_n$, B are linguistic terms and y is the output.

Defuzzification Consists in obtaining a numeric value for the output. This stage basically selects a point that is the most representative of the action to perform [2]. There are several methods to calculate the Defuzzification, such as the Center of Height (COH), Center of Gravity (COG), etc. The COG is shown in Equation 2.

$$y = \frac{\sum_{i=1}^{N} h_i \mu(i)}{\sum_{i=1}^{N} \mu(i)} \qquad (2)$$

where h_1 is the maximum height of the consequent from rule i to rule N [2].

In Figure 2, the fuzzy system information processing is illustrated.

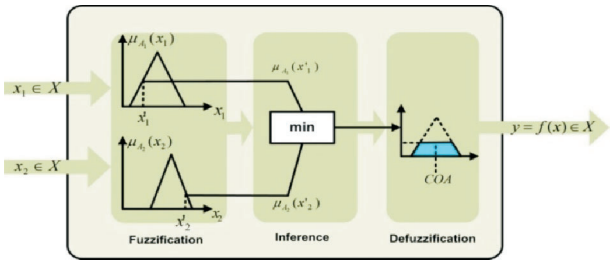

Fig. 2. Fuzzy System.

A fuzzy system can be implemented on a general purpose computer, or by a specific use of microelectronics realization. The first offers a great versatility in terms of ease of development in various high level programs, the second device is performed in high scale integration, such as the ASICs [19].

The ASICs offer great advantages for high performance and price reduction is concerned, however, also have the disadvantage of requiring a high level of production of the same design to actually be affordable and time to get in the market is large relative to that of using a FPGA.

The applications of the FPGAs go beyond the simple implementation of digital logic. The FPGAs can be used to implement specific architectures to accelerate a particular algorithm. Systems based on FPGAs provide better performance than their corresponding implementations in software platforms for general use. A specific architecture for an algorithm can have a yield of 10 to 1000 times higher than an implementation on a DSP (Digital Signal Processor). Applications that require a great number of simple operations are suitable for implementation on FPGA, a processing element can be designed to perform this operation and several instances of it can be played to perform parallel processing [20].

An FPGA is a semiconductor device that contains in its interior components such as gates, multiplexers, etc. These are interconnected with each other, according to a given design. These devices use the VHDL programming language, which is an acronym that represents the combination of VHSIC (Very High Speed Integrated Circuit) and HDL (Hardware Description Language) [13].

Implementing an embedded fuzzy system on an FPGA is not as easy as it seems, since there are few appropriated design tools to achieve this task. Most of the time, the designer needs to construct every part of the inference system from scratch. Fortunately, there is an increasing interest in the development of designing platforms the easily achieve this task, such is the case of the Xfuzzy 2.1 and Xfuzzy 3.0; however, at the present time they cannot provide VHDL code for trapezoidal membership functions for arithmetic calculation. Other implementations of a FIS on an FPGA are reported in [21].

Any hardware implementation of an electronic system requires a complex methodology to test and validate every stage in the design process to guarantee its correct functionality; this is particularly true when the designer decides to use a HDL to make a design.

The FPGAs are good platforms for fast prototyping of digital hardware. FPGAs are very effective in implementing FLSs since they allow fast modeling and hardware verification. FPGAs can be programmed in the system,

without shutting down the system. This functionality allows modification and tuning of rules and-or fuzzifiers to achieve better control performance. In order to accelerate the design of FLS hardware, it is helpful to have a design environment, which allows algorithmic specification of the FLS and eases the automatic synthesis and verification of FLS hardware. The topic of FLS implementation onto FPGAs has been investigated by several researchers. A brief overview of the work done by some researchers is presented next.

The design of an FPGA implementation is done by specifying the logic function to develop, either by a CAD (computer aided design) or through a hardware description language. Having defined the function to perform, the design is transferred to the FPGA. This process program the configurable logic blocks (CLBs) to perform a specific function (there are thousands of configurable logic blocks in the FPGA). The configuration of these blocks and their interconnections flexibility are the reasons why it can get very complex designs. The interconnections enable connecting the CLBs. Finally, it has configuration memory cells (CMC, Configuration Memory Cell) distributed throughout the chip, which store all information necessary for programming programmable elements mentioned. These cells usually consist of configuration RAM and are initialized in the process of loading of the configuration [22]. The programmable elements of an FPGA are:

1. Configurable Logic Blocks (CLBs)
2. In/Out Blocks (IOBs)
3. Programmable Interconnection
   - By fuse technology and be of OTP.
   - By antifusing or by type SRAM cells.

Depending on the manufacturer we can find different solutions. FPGAs currently available on the market, depending on the structure adopted by the logical blocks that are defined, can be classified as belonging to four major families shown, in the Figure 3.

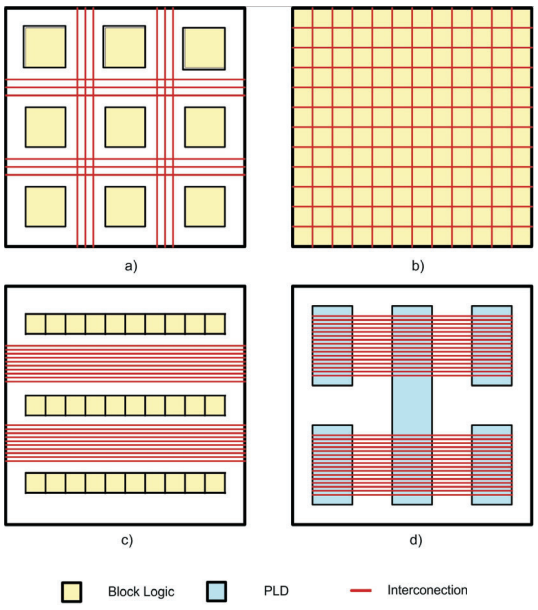

Fig. 3. Block logic a) Symmetrical Array (XILINX), b) Sea of Gates (ORCA), c) Row Based (ACTEL) and d) Hierarchical PLD (ALTERA and XILINX).

The basic elements of an FPGA are:

1. Configurable logic blocks CLBs, and their structure and content is called architecture. There are many types of architectures, which vary largely in complexity (from a simple door to more complex modules or SPLD-like structures). They often include bi-stable or MOS (Metal Oxide Semiconductor) to facilitate the implementation of sequential circuits. Other important modules are the building blocks of I / O (IOB).
2. Interconnection resources, whose structure and content defines the routing architecture.
3. RAM, which is loaded during reset to configure and connect blocks.

Figure 4 shows these elements.

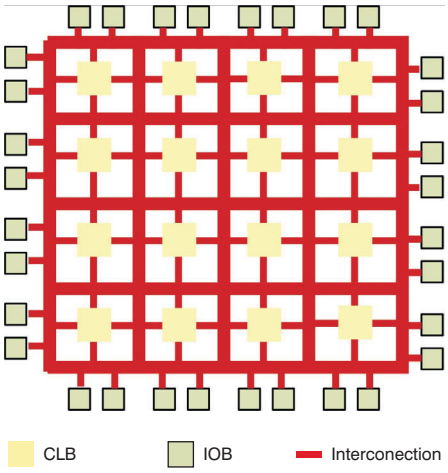

Fig. 4. FPGA Basic Elements.

Figure 5 shows the CHIP Spartan Basic Elements.

3. Novel Genetic Optimization of Membership Functions for Fuzzy Logic Controller in FPGAs

The fuzzy logic controller is coded in VHDL, the FLC for the fuzzification stage, is able to instantly calculate the degree of membership, using a method to calculate the slopes [23][24][25], the inference is working with the max-min [26][27][28] and the defuziffication with the method of heights.

Figure 6 shows the block diagram in XSG of the FLC for the regulation of speed of a DC motor, the system inputs are reset, error $e(t)$, error of change $e'(t)$, and the parameters of each membership function for inputs and output are in total 11. The system only has an output $Y(t)$.

The FLC has two inputs and one output, each input and output contains three membership functions, two trapezoidal one triangular. Figure 7 shows the triangular and trapezoidal membership functions (MF) that are used.

For the optimization of the FLC using GAs, you must define the chromosome that represents the information of the individual, which in this case is related to the universe of discourse and the linguistic terms. Figure 8 shows the chromosome of the GA.

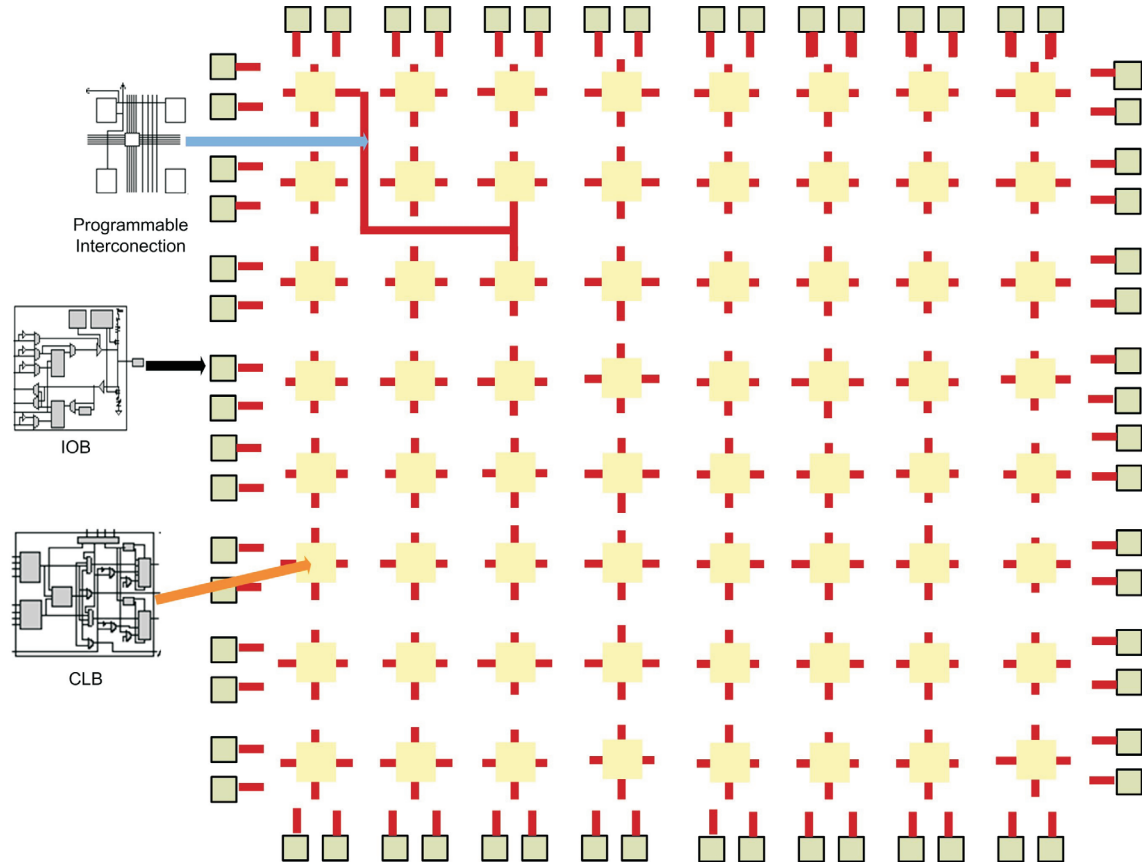

Fig. 5. CHIP Spartan of Xilinx Basic Elements.

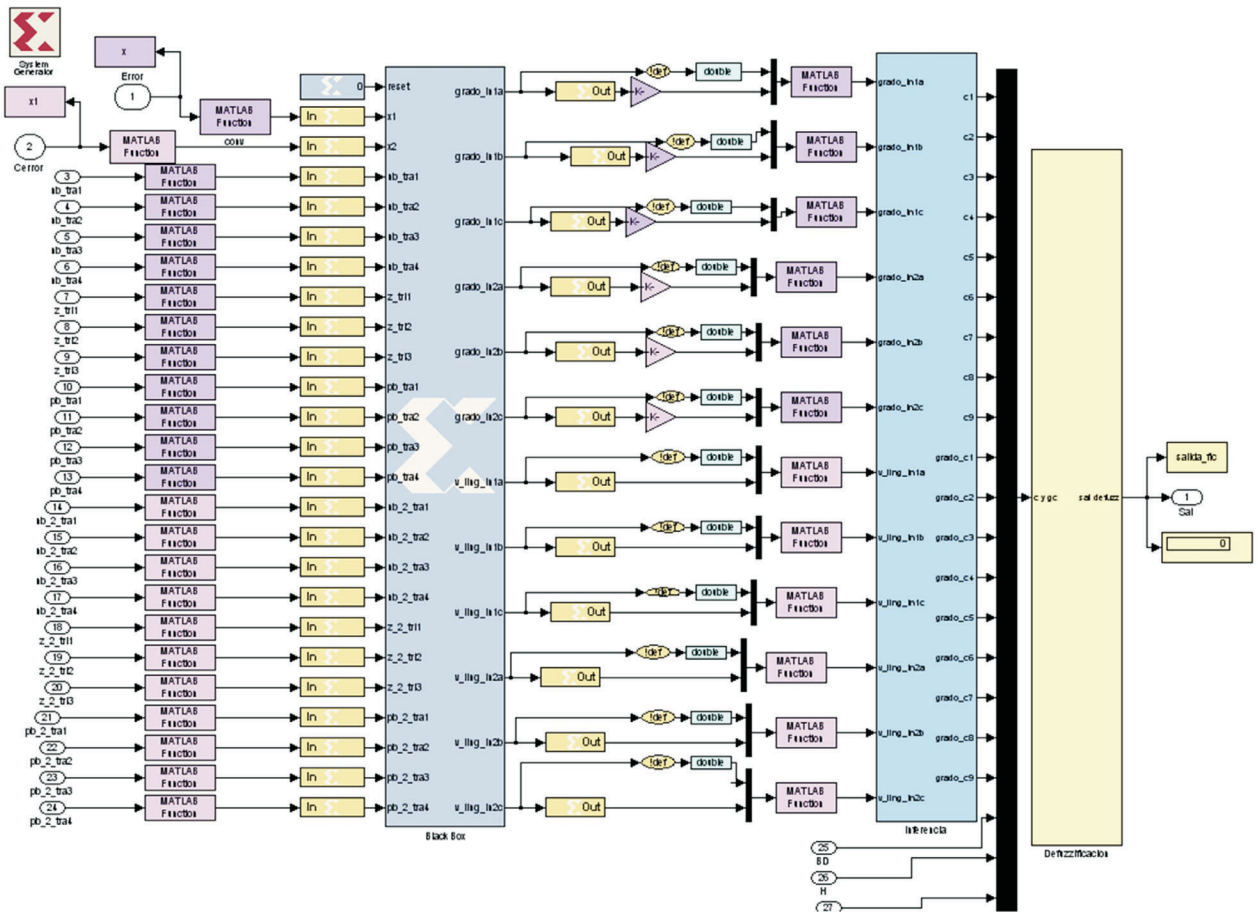

Fig. 6. Block diagram in XSG of FLC.

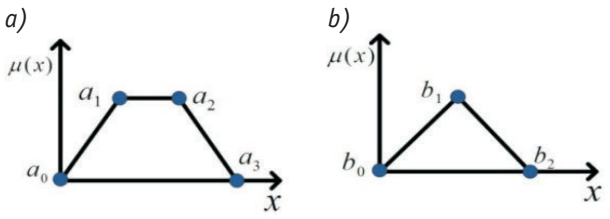

Fig. 7. Parameters of the Membership Functions. a) MF trapezoidal, b) MF triangular.

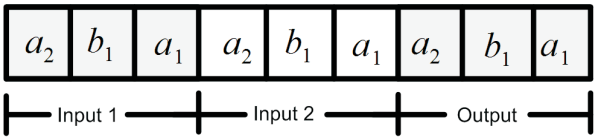

Fig. 8. GA chromosome.

In Table 1, shows the boundary parameters of the chromosome.

Table 1. Boundary parameters of the chromosome.

Parameters	Input 1	Input Input 2	Output
	$0 < a_2 < 128$	$0 < a_2 < 128$	$0 < a_2 < 128$
	$b_1 = 128$	$b_1 = 128$	$b_1 = 128$
	$128 < a_1 < 255$	$128 < a_1 < 255$	$128 < a_1 < 255$

Figure 9 shows the input of the FLC with fixed and variable parameters. Each input and output has a size or 8 bits.

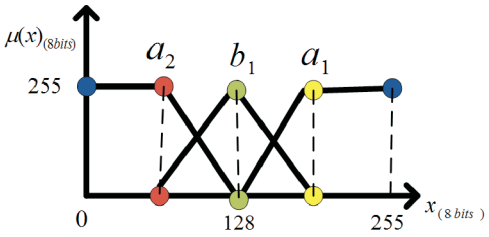

Fig. 9. Points of membership functions input and output.

The blue points are fixed, the red dots are for parameter a_2, the green dots are fixed (b_1) and the yellow dots are for parameter a_1.

Figure 10 shows the range of parameters membership functions.

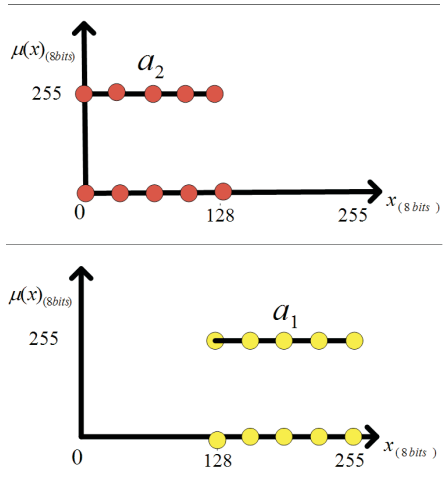

Fig. 10. Range of parameters membership functions.

The GA is of multiobjective type [15], which means that to determine the best individual three evaluations are performed:

a) Minimum overshoot

$$o_1 = \max(y(t)) - r \tag{3}$$

b) Minimum undershoot

$$o_2 = \left|\min(y(t)) - r\right| \tag{4}$$

c) Minimum output steady state error (sse)

$$sse = \sum_{t=201}^{1000} y(t) - r \tag{5}$$

The FLC linguistic terms were optimized with the GA, but the fuzzy rules are not changed. The process of the GA is shown in Figure 11.

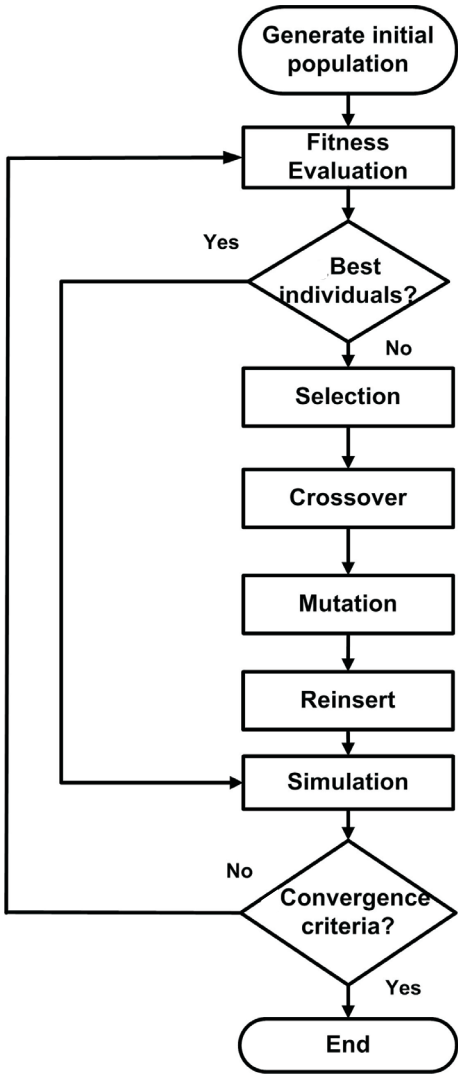

Fig. 11. Optimization GA.

4. Test and Results the Novel Genetic Optimization of Membership Functions for FLC for Speed Regulate the Motor DC

To evaluate the ability of the GA, the FLC was simulated for speed control using a mathematical model of the plant in Matlab-Simulink [12], as shown in Figure 12.

The FLC has the inputs, error $(e(t))$ and change of error $(e'(t))$, and the output is the control signal $(y(t))$. The inputs are calculated as follows:

$$e(t) = r(t) - y(t) \tag{6}$$

$$e'(t) = e(t) - e(t-1) \tag{7}$$

where t is the sampling time.

The reference signal $r(t)$, is given by:

$$r(t) = \begin{cases} 15 & t > 0 \\ 0 & t \le 0 \end{cases} \tag{8}$$

Each input and output of the FIS has three linguistic terms. For the linguistic variable error and change of error, the terms are {NB, Z, PB} in this case NB is Negative Big, Z is Zero and PB is Positive Big. For the linguistic variable control signal the linguistic terms are {BD, H, BI}, in this case BD is Big Decrement, H is Hold and BI is Big Increment.

A series of experiments was performed that are listed in Table 2.

In experiment No. 17 the best FLC was found because this has the lower error value. Below are the FIS characteristics for experiments 14 and 17.

Figure 13 shows how the GA modified the parameters of the membership functions for the input $e(t)$.

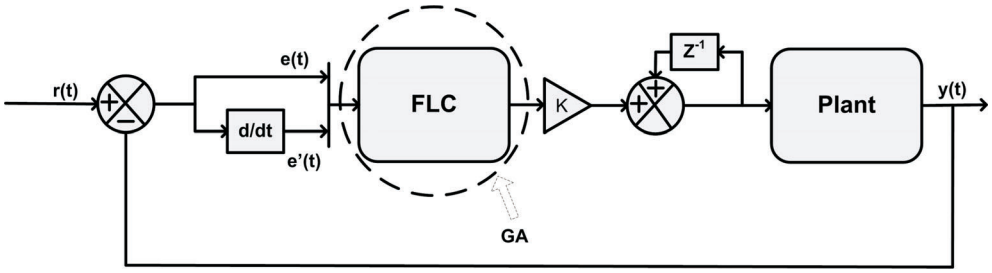

Fig. 12. Model.

Table 2. GA Parameters for different experiments.

No.	Generations	Crossover (XOVSP)	Selection (SUS)	Overshoot-Undershoot	SSE	Error	Time (sec)
1	200	0.8	0.8	0.4000	0.0425	0.0413	1161.761412
2	200	0.8	0.8	0.9104	0.1048	$7.7489e^{-3}$	992.011800
3	200	0.8	0.8	0.4000	0.0425	0.0413	769.952523
4	100	0.8	0.8	0.0148	0.0118	0.0118	859.303842
5	200	0.8	0.8	0.4124	0.0198	0.0021	1030.261653
6	150	0.8	0.8	0.9104	0.1048	0.0077	896.357154
7	150	0.8	0.8	0.1920	0.0110	0.0027	689.695587
8	100	0.8	0.8	0	0.0851	0.0517	754.005340
9	200	0.8	0.8	0.9104	0.1048	0.0077	924.325968
10	200	0.7	0.8	0.9104	0.1048	0.0077	839.401213
11	200	0.7	0.8	0.6600	0.0637	0.0345	810.448215
12	100	07	08	0.9104	0.1048	$7.75e^{-3}$	711.466736
13	100	0.7	0.8	0.0148	0.0118	0.0118	804.630357
14	100	0.7	0.8	0.4124	0.0198	$5.7457e^{-3}$	671.710473
15	100	0.7	0.8	0.9104	0.1048	$7.75e^{-3}$	709.538318
16	50	0.7	0.8	0.0148	0.0118	0.0118	822.235179
17	50	0.7	0.8	0	0.0814	$2.0788e^{-3}$	723.025902
18	300	0.7	0.8	0.9104	0.1048	$7.75e^{-3}$	960.829622
19	300	0.7	0.8	0.0256	0.0205	$2.538e^{-3}$	941.800589
20	300	0.8	0.8	0.0148	0.0118	0.0118	1391.119402
21	200	0.8	0.8	0.9104	0.1048	0.0077	936.360153

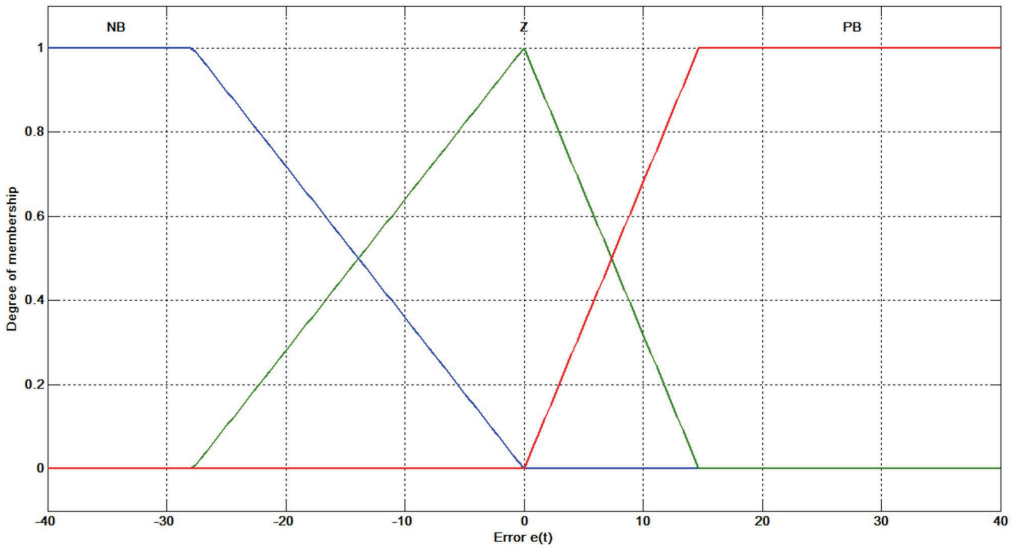

Fig. 13. FIS for the experiment 14, input e(t).

Figure 14 shows the input $e'(t)$ modified by the GA.

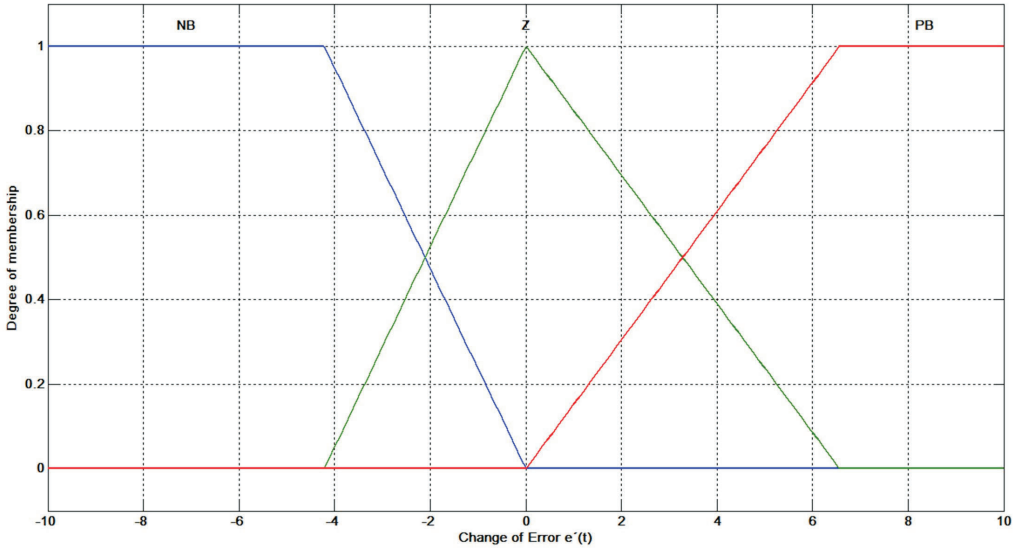

Fig. 14. FIS for the experiment 14, input e'(t).

Figure 15 shows the output $y(t)$ of the FIS.

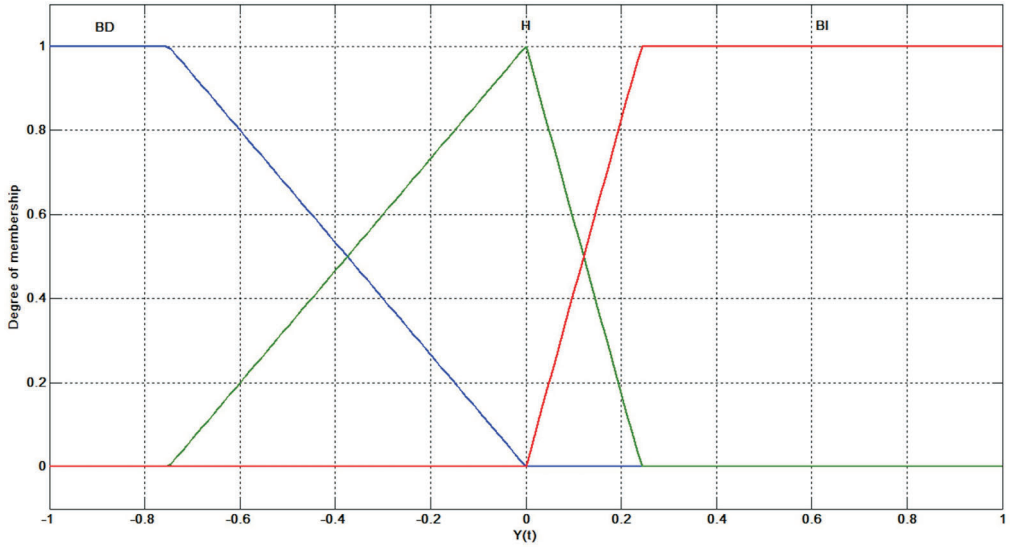

Fig. 15. Output FIS for the experiment 14.

Figure 16 shows the control surface modified by the GA.

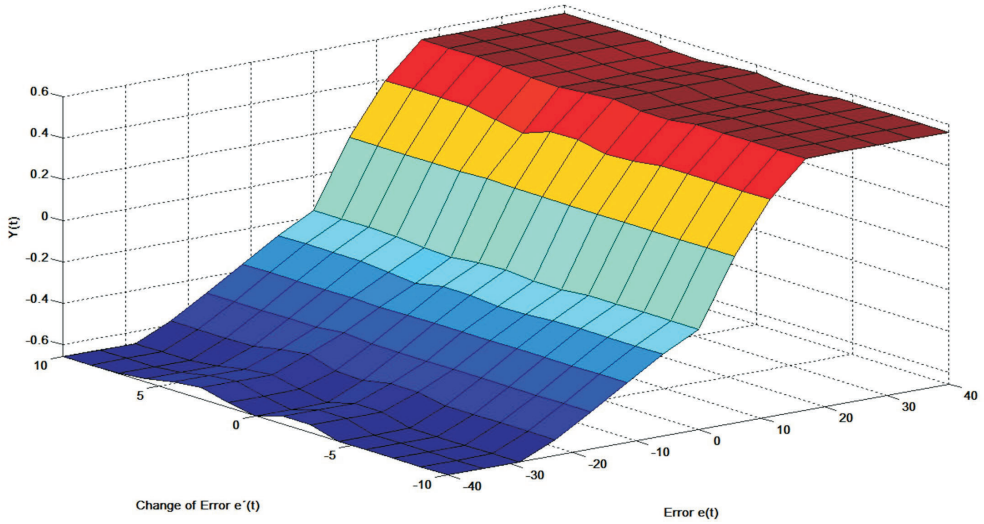

Fig. 16. Control Surface for experiment 14.

Figure 17 shows the output signal of the PD Incremental FLC for experiment 14.

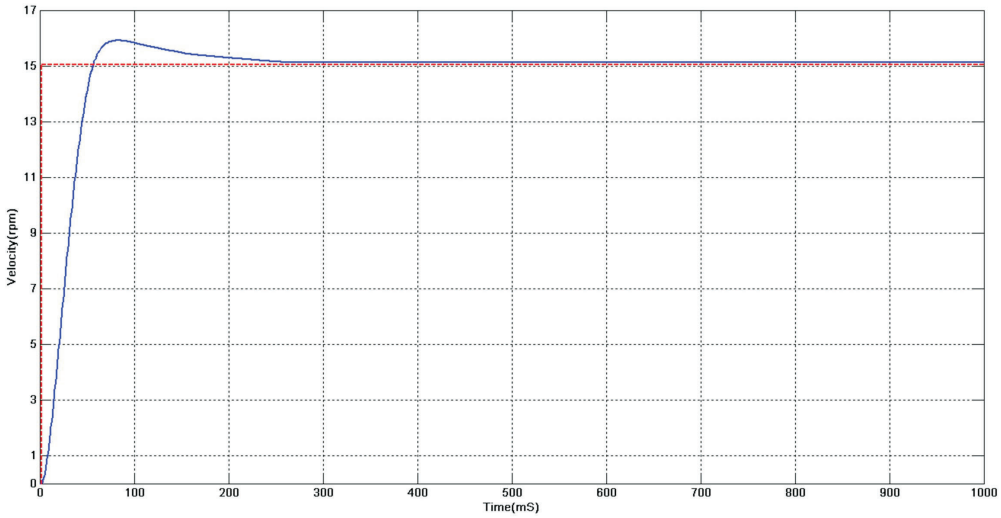

Fig. 17. Velocity of the motor.

Figure 18 shows convergence the GA.

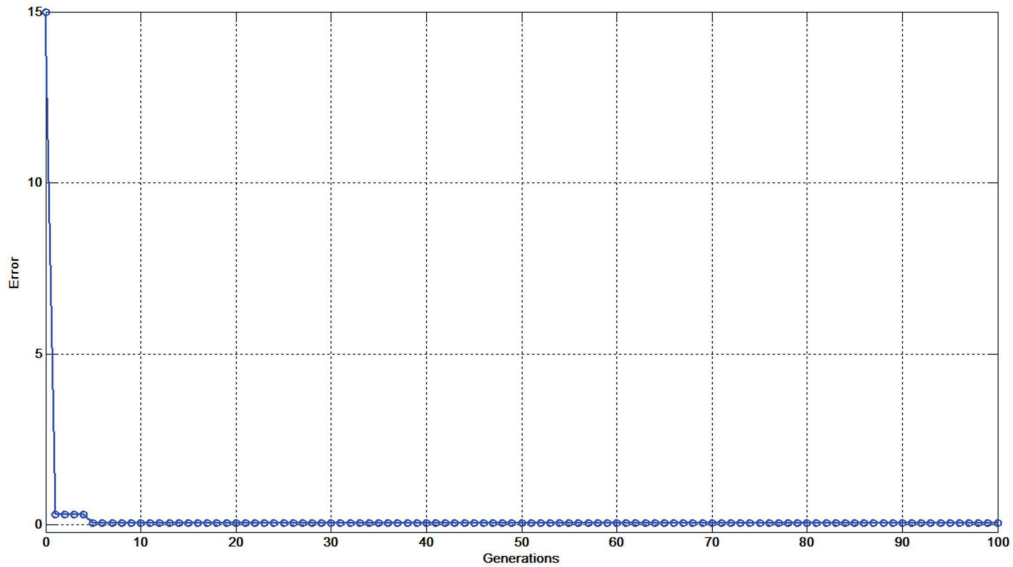

Fig. 18. GA convergence for experiment 14.

Figure 19 shows the best experiment for the input $e(t)$.

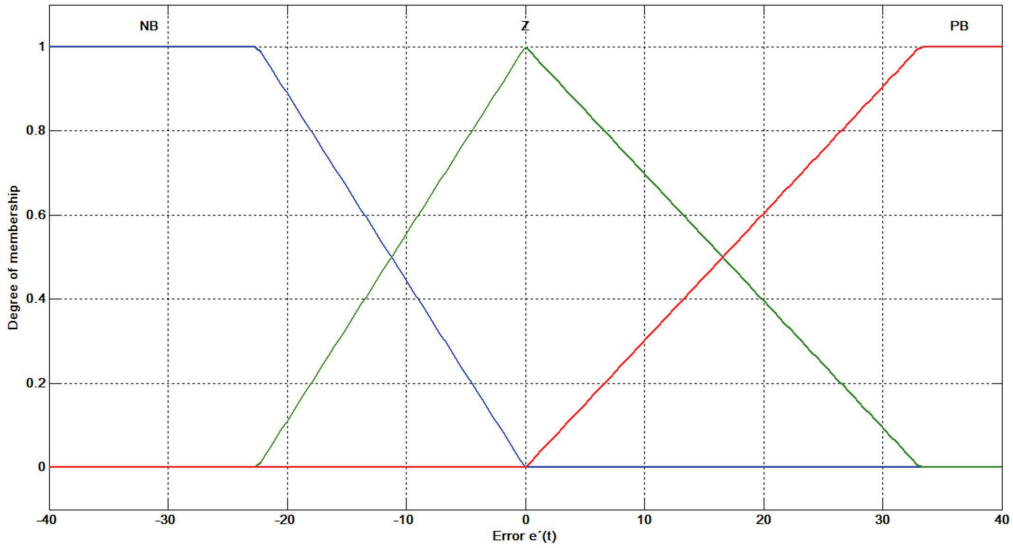

Fig. 19. Best FIS for the input e(t)).

Figure 20 shows the input $e'(t)$ modified by the GA.

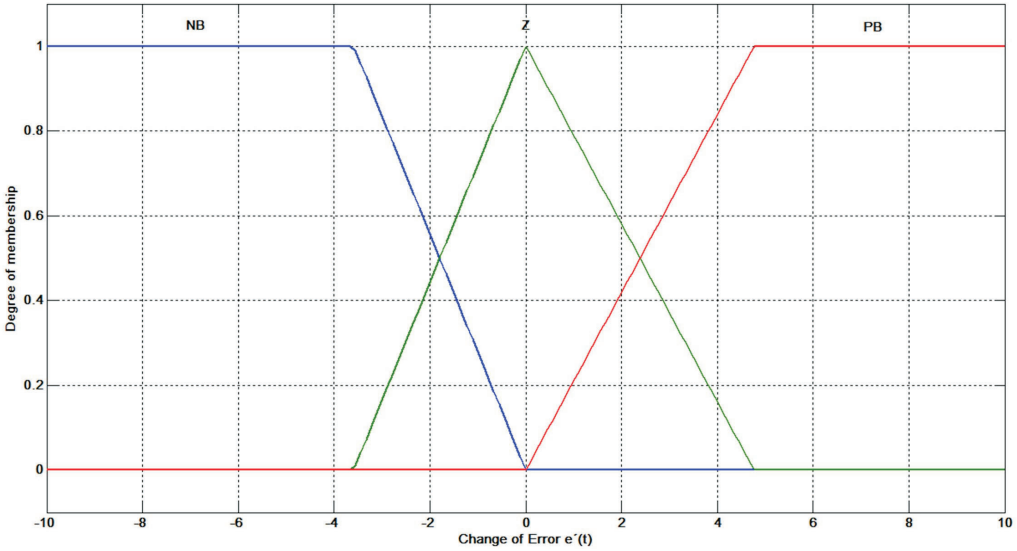

Fig. 20. Best FIS for the input e'(t).

Figure 21 shows the output $y(t)$ of the FIS modified by the GA.

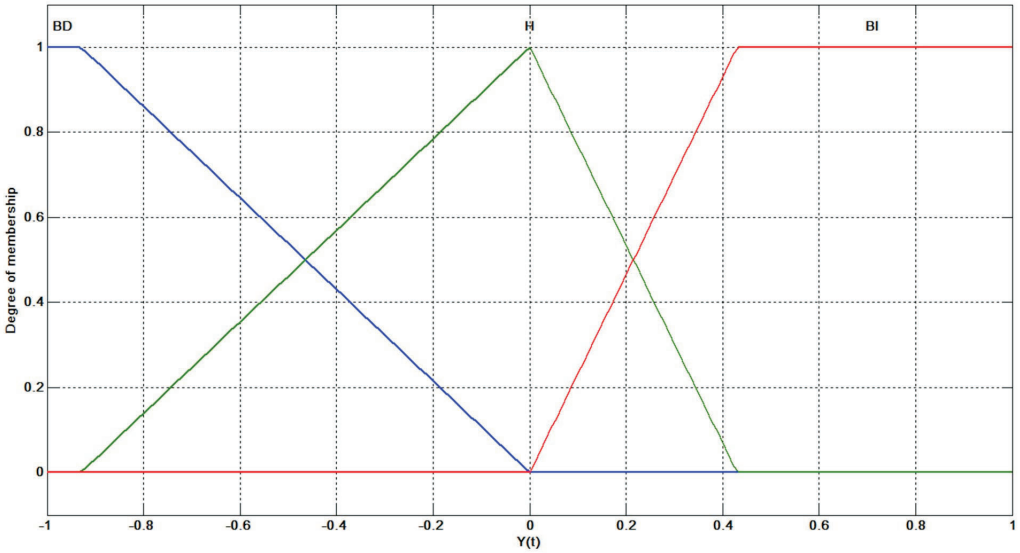

Fig. 21. Best FIS for the output y(t).

Figure 22 shows the control surface modified by the GA.

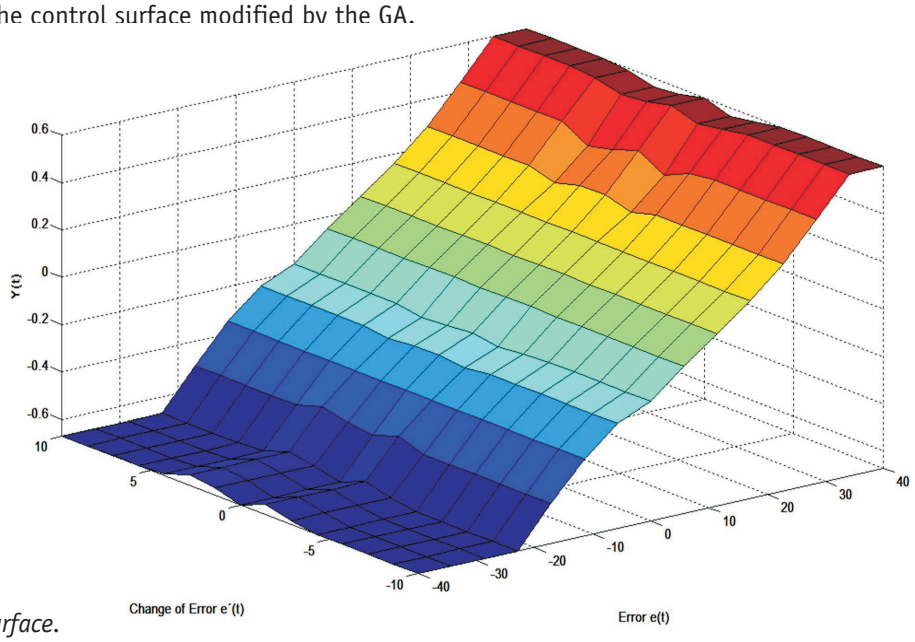

Fig. 22. Control Surface.

Figure 23 shows the output signal close loop of the FLC for experiment 17.

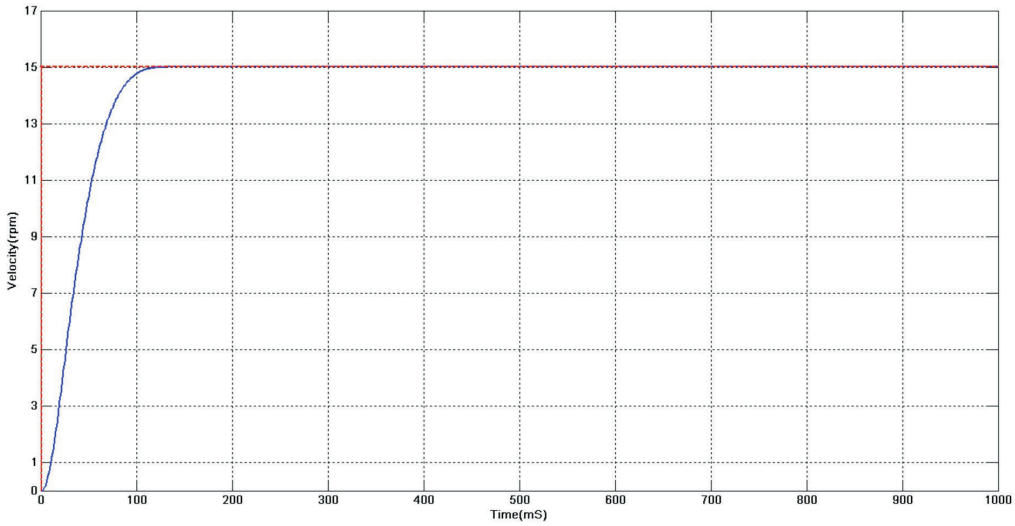

Fig. 23. Velocity

Figure 24 shows convergence the GA for the best experiment.

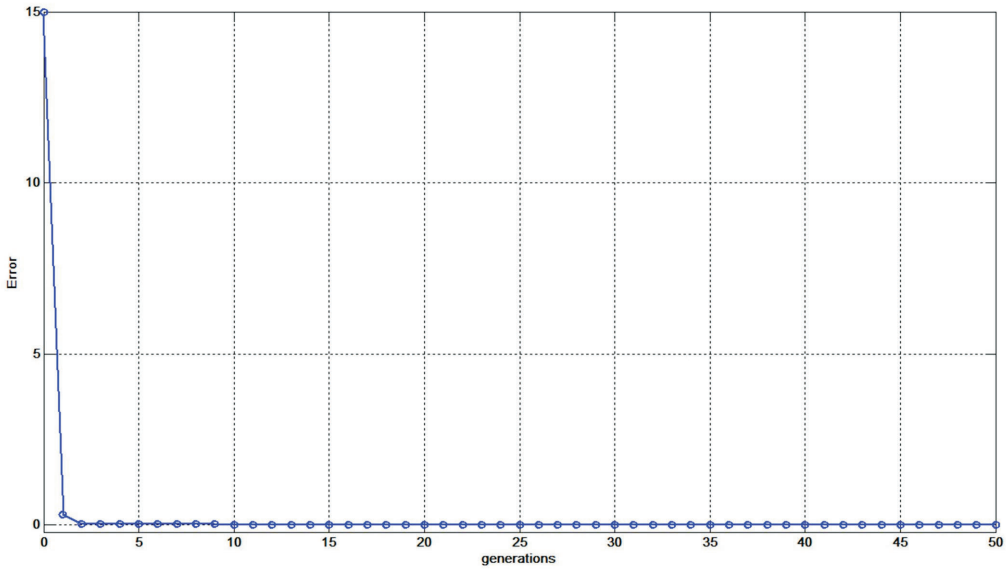

Fig. 24. Shows the GA convergence error for experiment 17.

5. Conclusions

We proposed a novel method for genetic optimization of a fuzzy logic controller in FPGA for the regulation of speed of a DC motor, were the method optimizes three triangular and trapezoidal membership functions for the two inputs and one output of the FLC.

The genetic algorithm optimizes only three of the eleven parameters of the membership functions, the algorithm proved to be very efficient with good results. The objective function of the GA considers three characteristics: overshoot, undershoot and steady state error, so this makes it a multiobjective GA.

Each FIS was simulated in an Incremental PD Fuzzy Con-troller for speed control of the DC motor. The best FLC was obtained in 50 generations with 70% crossover and 80% selection, with a result of zero of overshoot and under-shoot steady state error of 2.0788e-3, in a time of 723.025902 seconds with a speed of 15 rpm. Matlab-Simulink and Xilinx System Generator was used to perform the simulations.

AUTHORS
Yazmin Maldonado, Oscar Castillo*, Patricia Melin - Tijuana Institute of Technology, Tijuana, Mexico. E-mail: ocastillo@tectijuana.mx.
* Corresponding author

References

[1] Cirstea M.N., Khor J.G., McCormick M., *Neural and fuzzy logic control of drives and power system*, Newnes, 2002.

[2] Driankov D., Hellendorn H., Reinfrank M., *An Introduction to Fuzzy Control*, Springer, 1996.

[3] Goldberg D.E., *Genetic Algorithm in Search*, Optimization & Machine Learning, 1989.

[4] Jantzen J., *Tunning of Fuzzy PID Controllers*, 1998, pp. 1-22.

[5] Klir G., Yuan B., *Fuzzy Sets and Fuzzy Logic Theory and Applications*, Prentice Hall, 1995.

[6] McNeilland D., Freiberger P., *Fuzzy Logic: The Revolutionary Computer Technology that is Changing our World*, Simon & Schuster, 1993.

[7] Tsoukalas L., Ohrig R.E., *Fuzzy and Neural Approaches in Engineering*, WileyI-Interscience, 1997.

[8] Tommiska M., Vouri J.,"Implementation of genetic algorithms with programmable logic devices". In: *Proc 2nd Nordic Workshop Genetic Algorithm*, 1996, pp. 71-78.

[9] Holland J. H., *Adaptation in Natural and Artificial Systems*, Cambrige, MA: MIT Press, 1992.

[10] Golberg D.E., *Genetic Algorithms in Sear Optimization and Machine Learning*, Boston, MA: Addison-Wesley, 1989.

[110] Grefenstette J.J., Gopal R., Rosmaita B.J., Van Gucht D., "Genetic Algorithms for the traveling salesman problem". In: *Proc. 1st Int. Conf. Genetic Algorithms*, 1985, pp. 160-168.

[12] Web page of Matlab-Simulink, available in www.mathworks.com, 2010.

[13] Web page of Xilinx system Generator and FPGAs, available in www.xilinx.com, 2010.

[14] Haupt Randy L., Haupt Sue E., *Practial Genetic Algorithms*, Wiley, 2004.

[15] Man K.F., Tang K.S., Kwong S., *Genetic algorithms*, Springer, 2000.

[16] Koza J.R., *Genetic Programming: on the Programming of Computers by means of natural selection*, 1992.

[17] Zadeh L.A., "Fuzzy Sets", Information and Control 8, 1965, pp. 338-353..

[18] Zdenko K., Stjepan B., *Fuzzy Controller Design Theory and Applications*, Taylor and Francis, 2006.

[19] Montiel O., Maldonado Y., Sepulveda R., Castillo O., "Simple Tuned Fuzzy Controller Embedded into an FPGA". In: *IEEE NAFIPS Conference Proceedings*, 2008, pp. 1-6.

[20] Montiel O., Maldonado Y., Sepulveda R., Castillo O., "Development of an Embedded Simple Tuned Fuzzy Controller". In: *IEEE World Congress on computational Intelligence WCCI-2008*, 2008, pp. 555-561.

[21] Velo F.J.M., Baturone L., Solano S.S., Barriga A., "Rapid Design of Fuzzy Systems with XFUZZY", *IEEE International Conference on Fuzzy Systems FUZZ-IEEE-2003*, 2003, pp. 342-347.

[22] Buj Gelonch R.A., Sancho Francisco C., *Procedimiento de Diseño de Circuitos Digitales Mediante FPGAs*, Universidad de Lleida, Escuela politécnica superior Escuela técnica en informática de sistemas, 2007. (in Spanish)

[23] MaldonadoY., Montiel O., Sepulveda R., *Diseño y validación de la etapa de fuzzificacion para sistemas difusos en FPGAs*, ERA-08, 2008, pp. 1-7. (in Spanish)

[24] Maldonado Y., Montiel O., Sepulveda R., Castillo O., "Design and simulation of the fuzzification Stage through the Xilinx System Generator", *Soft Computinf for Hybrid Intelligent systems*, Springer, 2008, pp. 297-305.

[25] Montiel O., Sepulveda R., Maldonado Y., Castillo O., Design and simulation of *The Type-2 Fuzzification Stage: Using Active Membership Functions, Evolutionary Design of Intelligent systems in Modeling, Simulation and Control*, Springer, 2009, pp. 273-293.

[26] Olivas J.A., Sepulveda R., Montiel O., *Validacion y Prueba de una Maquina de Inferencia Difusa Mediante Xilinx System Generator*, ERA-08, 2008, pp. 1-6. (in Spanish)

[27] Olivas J.A., Sepulveda R., Montiel O., Castillo O., *Methodology to Test and Validate a VHDL Inference Engine through the Xilinx System Generator, Soft Computinf for Hybrid Intelligent Systems*, Springer, 2008, pp. 325-331.

[28] Sepulveda R., Oscar Montiel O., Olivas J., Castillo O., *Methodology to Test and Validate a VHDL Inference Engine of a Type-2 FIS, through the Xilinx System Generator, Evolutionary Design of Intelligent systems in Modeling, Simulation and Control*, Springer, 2009, pp. 295- 308.