Evolutionary Optimization of Type-2 Fuzzy Logic Systems Applied to Linear Plants

Ricardo Martinez[1], Oscar Castillo[2], Luis T. Aguilar[3], and Antonio Rodriguez[4]

[1] PhD Student of Computer Science in the Universidad Autónoma de Baja California Tijuana, México
e-mail: mc.ricardo.martinez@hotmail.com

[2] Tijuana Institute of Technology, Tijuana México
e-mail: ocastillo@hafsamx.org

[3] Instituto Politécnico Nacional, Centro de Investigación y Desarrollo de Tecnología Digital, PMB 88, P.O. Box 439016 San Ysidro CA., 92143-9016; Fax: +52(664)6231388
e-mail: luis.aguilar@ieee.org

[4] Universidad Autónoma de Baja California, Tijuana México
e-mail: ardiaz@uabc.mx

Abstract. We describe a different kind of evolutionary methods to optimize a type-2 fuzzy logic controller (FLC) applied to linear plants. The evolutionary method used is a genetic algorithm to find the optimal FLC for the plant control. The plant receives a linear signal of input controlled by an optimized FLC, obtaining as result the control and the stability of the plant. Simulations results were made in Simulink showing the effectiveness of the proposal.

1 Introduction

The evolutionary methods are used for different purposes such as optimization, solution search's processes and other kind of applications. This study is about several evolutionary methods applied to the autonomous linear plants. To this purpose, we use a type-2 fuzzy logic system to develop the optimal controller. Previously we optimize a type-2 fuzzy logic controller for an autonomous mobile robot for trajectory tracking, where the genetic algorithms were used to find the optimal controller obtaining good results applied some kind of perturbation. For the study of the evolutionary methods, we use a transfer function to test the optimal type-2 fuzzy logic controller. One of the evolutionary methods is genetic algorithms that we used to find the parameters of the membership functions, using the genetic operators, mutation and crossover obtaining an Optimal FLC for the plant control.

This paper is organized as follows: Section 2 presents the theoretical basis and problem statement, 2.1 presents an introductory explanation of Type-2 Fuzzy Logic, subsection 2.2 presents the basics of Evolutionary Methods. Section 3 introduces the controller design where a genetic algorithm is used to select the parameters. Robustness properties of the closed-loop system are achieved with a type-1 fuzzy logic control system using a Takagi-Sugeno model where the error and the change of error, are considered the linguistic variables. Section 4 provides a simulation study of the plant using the controller described in Section 3. Finally, Section 5 presents the conclusions.

2 Theoretical Basis and Problem Statement

This section describes the theoretical basis of the paper as well as the problem definition. Some basics about type-2 fuzzy systems and genetic-fuzzy systems are first presented.

2.1 Type-2 Fuzzy Logic Systems

If we have a type-1 membership function, as in Figure 1 (a), and we are blurring it to the left and to the right as illustrated in Figure 1 (b), then, for a specific value x', the membership function (u'), takes on different values, which are not all weighted the same, so we can assign an amplitude distribution to all of those points. Doing this for all $x \in X$, we create a three-dimensional membership function –a type-2 membership function– that characterizes a type-2 fuzzy set [1, 14]. A type-2 fuzzy set $\tilde{A}$, is characterized by the membership function:

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{1}$$

in which $0 \le \mu_{\tilde{A}}(x,u) \le 1$. Another expression for $\tilde{A}$ is,

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\tilde{A}}(x,u)/(x,u) \qquad J_x \subseteq [0,1] \tag{2}$$

where $\int\int$ denote union over all admissible input variables x and u. For discrete universes of discourse $\int$ is replaced by $\sum$ [14]. In fact $J_x \subseteq [0,1]$ represents the primary membership of x, and $\mu_{\tilde{A}}(x,u)$ is a type-1 fuzzy set known as

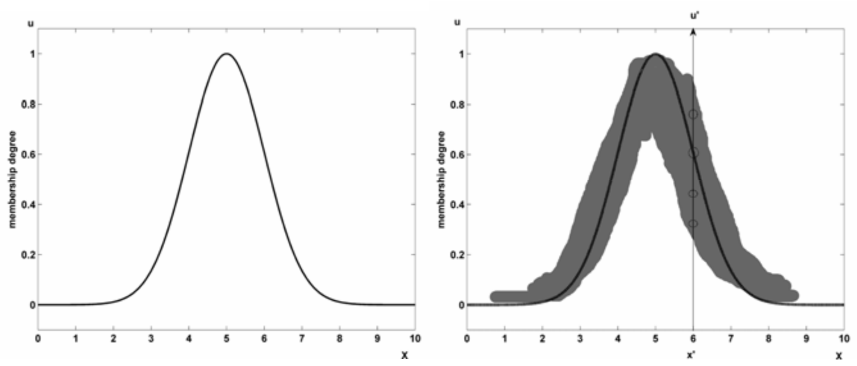

Fig. 1. a) Type-1 membership function and b) Blurred type-1 membership function.

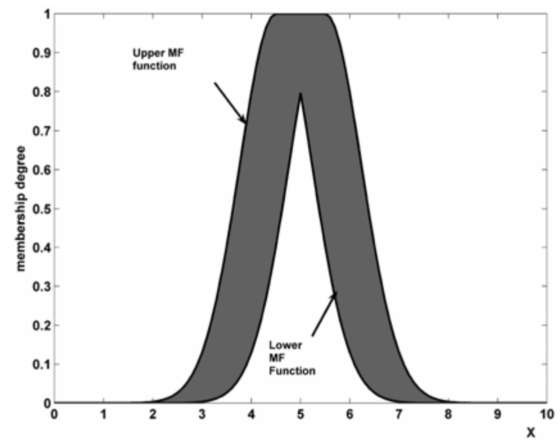

Fig. 2. Interval type-2 membership function.

the secondary set. Hence, a type-2 membership grade can be any subset in [0,1], the primary membership, and corresponding to each primary membership, there is a secondary membership (which can also be in [0,1]) that defines the possibilities for the primary membership [20].

This uncertainty is represented by a region called footprint of uncertainty (FOU). When $\mu_{\tilde{A}}(x,u) = 1, \forall u \in J_x \subseteq [0,1]$ we have an interval type-2 membership function, as shown in Figure 2. The uniform shading for the FOU represents the entire interval type-2 fuzzy set and it can be described in terms of an upper membership function $\bar{\mu}_{\tilde{A}}(x)$ and a lower membership function $\underline{\mu}_{\tilde{A}}(x)$.

A FLS described using at least one type-2 fuzzy set is called a type-2 FLS. Type-1 FLSs are unable to directly handle rule uncertainties, because they use type-1 fuzzy sets that are certain [21]. On the other hand, type-2 FLSs, are very useful in circumstances where it is difficult to determine an exact membership function, and there are measurement uncertainties [23].

It is known that type-2 fuzzy sets enable modeling and minimizing the effects of uncertainties in rule-based FLS. Unfortunately, type-2 fuzzy sets are more difficult to use and understand than type-1 fuzzy sets; hence, their use is not widespread yet. As a justification for the use of type-2 fuzzy sets, in [22] are mentioned at least four sources of uncertainties not considered in type-1 FLSs:

1. The meanings of the words that are used in the antecedents and consequents of rules can be uncertain (words mean different things to different people).
2. Consequents may have histogram of values associated with them, especially when knowledge is extracted from a group of experts who do not all agree.
3. Measurements that activate a type-1 FLS may be noisy and therefore uncertain.
4. The data used to tune the parameters of a type-1 FLS may also be noisy.

All of these uncertainties translate into uncertainties about fuzzy set membership functions. Type-1 fuzzy sets are not able to directly model such uncertainties because their membership functions are totally crisp. On the other hand, type-2 fuzzy sets are able to model such uncertainties because their membership functions are themselves fuzzy. A type-1 fuzzy set is a special case of a type-2 fuzzy set; its secondary membership function is a subset with only one element, unity.

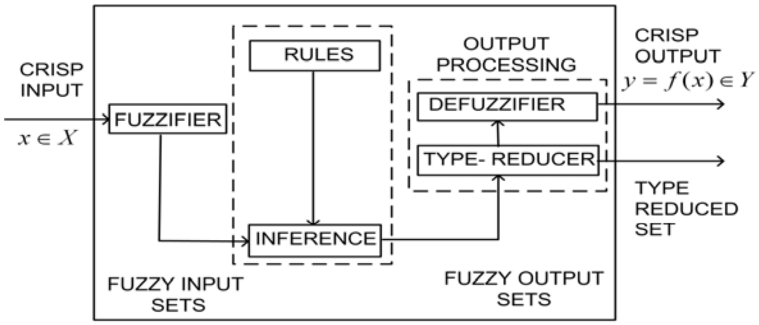

Fig. 3. Type-2 Fuzzy Logic System.

A type-2 FLS is again characterized by IF-THEN rules, but its antecedent or consequent sets are now of type-2. Type-2 FLSs, can be used when the circumstances are too uncertain to determine exact membership grades such as when the training data is corrupted by noise. Similar to a type-1 FLS, a type-2 FLS includes a fuzzifier, a rule base, fuzzy inference engine, and an output processor, as we can see in Figure 3. The output processor includes type-reducer and defuzzifier; it generates a type-1 fuzzy set output (from the type-reducer) or a crisp number (from the defuzzifier) [6,5]. Next we will explain each of the blocks of Figure 3.

2.1.1 Fuzzifier

The fuzzifier maps a crisp point $\mathbf{x}=(x_1,\ldots,x_p)^T \in X_1 x X_2 x \ldots x X_p \equiv \mathbf{X}$ into a type-2 fuzzy set $\tilde{A}_x$ in $\mathbf{X}$ [12], interval type-2 fuzzy sets in this case. We will use type-2 singleton fuzzifier, in a singleton fuzzification, the input fuzzy set has only a single point on nonzero membership [31, 34]. $\tilde{A}_x$ is a type-2 fuzzy singleton if $\mu_{\tilde{A}_x}(x) = 1/1$ for $\mathbf{x}=\mathbf{x}'$ and $\mu_{\tilde{A}_x}(x) = 1/0$ for all other $\mathbf{x} \neq \mathbf{x}'$[23].

2.1.2 Rules

The structure of rules in a type-1 FLS and a type-2 FLS is the same, but in the latter the antecedents and the consequents will be represented by type-2 fuzzy sets. So for a type-2 FLS with p inputs $x_1 \in X_1,\ldots,x_p \in X_p$ and one output $y \in Y$, Multiple Input Single Output (MISO), if we assume there are M rules, the lth rule in the type-2 FLS can be written as follows [23]:

$$R^l\text{: IF } x_1 \text{ is } \tilde{F}_1^{\,l} \text{ and } \cdots \text{and } x_p \text{ is } \tilde{F}_p^{\,l} \text{ , THEN } y \text{ is } \tilde{G}^l \qquad l=1,\dots,M \tag{3}$$

2.1.3 Inference

In the type-2 FLS, the inference engine combines rules and gives a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets. It is necessary to compute the join ⊔, (unions) and the meet Π (intersections), as well as extended sup-star compositions (sup star compositions) of type-2 relations [23].If $\tilde{F}^l{}_1 \times \cdots \times \tilde{F}^l{}_p = \tilde{A}^l$, equation (3) can be re-written as

$$R^l : \tilde{F}^l{}_1 \times \cdots \times \tilde{F}^l{}_p \rightarrow \tilde{G}^l = \tilde{A}^l \rightarrow \tilde{G}^l \quad l=1,\dots,M \tag{4}$$

R^l is described by the membership function $\mu_{R^l}(\mathbf{x}, y) = \mu_{R^l}(x_1,\dots,x_p, y)$, where

$$\mu_{R^l}(\mathbf{x}, y) = \mu_{\tilde{A}^l \rightarrow \tilde{G}^l}(\mathbf{x}, y) \tag{5}$$

can be written as [23]:

$$\mu_{R^l}(\mathbf{x}, y) = \mu_{\tilde{A}^l \rightarrow \tilde{G}^l}(\mathbf{x}, y) = \mu_{\tilde{F}_1^l}(x_1)\,\Pi \cdots \Pi\, \mu_{\tilde{F}_p^{\,l}}(x_p)\,\Pi\, \mu_{\tilde{G}^l}(y) =$$

$$[\Pi_{i=1}^{p}\ \mu_{\tilde{F}^l{}_i}(x_i)\,]\Pi\, \mu_{\tilde{G}^l}(y) \tag{6}$$

In general, the p-dimensional input to R^l is given by the type-2 fuzzy set $\tilde{A}_x$ whose membership function is

$$\mu_{\tilde{A}_x}(\mathbf{x}) = \mu_{\tilde{x}_1}(x_1)\,\Pi \cdots \Pi\, \mu_{\tilde{x}p}(x_p) = \Pi_{i=1}^{p}\ \mu_{\tilde{x}i}(x_i) \tag{7}$$

where $\tilde{X}_i (i = 1,\dots, p)$ are the labels of the fuzzy sets describing the inputs. Each rule R^l determines a type-2 fuzzy set $\tilde{B}^l = \tilde{A}_x \circ R^l$ such that [23]:

$$\mu_{\tilde{B}^l}(y) = \mu_{\tilde{A}_x \circ R^l} = \sqcup_{x \in \mathbf{X}} \left[\mu_{\tilde{A}_x}(\mathbf{x})\, \Pi\, \mu_{R^l}(\mathbf{x}, y)\right] \qquad y \in Y \;\; l=1,\dots,M \tag{8}$$

This equation is the input/output relation in Figure 3 between the type-2 fuzzy set that activates one rule in the inference engine and the type-2 fuzzy set at the output of that engine [23].

In the FLS we used interval type-2 fuzzy sets and meet under product t-norm, so the result of the input and antecedent operations, which are contained in the firing set $\Pi_{i=1}^{p} \mu_{\tilde{F}_{ii}}(x'_i \equiv F^l(\mathbf{x}')$, is an interval type-1 set [23],

$$F^l(\mathbf{x}') = \left[\underline{f}^l(\mathbf{x}'), \bar{f}^l(\mathbf{x}') \right] \equiv \left[\underline{f}^l, \bar{f}^l \right] \tag{9}$$

where

$$\underline{f}^{l}(\mathbf{x}') = \underline{\mu}_{\tilde{F}_1^l}(x_1') * \cdots * \underline{\mu}_{\tilde{F}_p^l}(x_p') \tag{10}$$

and

$$\bar{f}^{l}(\mathbf{x}') = \bar{\mu}_{\tilde{F}_1^l}(x_1') * \cdots * \bar{\mu}_{\tilde{F}_p^l}(x_p') \tag{11}$$

where * is the product operation.

2.1.4 Type Reducer

The type-reducer generates a type-1 fuzzy set output which is then converted in a crisp output through the defuzzifier. This type-1 fuzzy set is also an interval set, for the case of our FLS we used center of sets (cos) type reduction, $Y_{\cos}$ which is expressed as [23]

$$Y_{\cos}(\mathbf{x}) = [y_l, y_r] = \int_{y^1 \in [y_l^1, y_r^1]} \cdots \int_{y^M \in [y_l^M, y_r^M]} \int_{f^1 \in [\underline{f}^1, \bar{f}^1]} \cdots \int_{f^M \in [\underline{f}^M, \bar{f}^M]} 1 \Big/ \frac{\sum_{i=1}^{M} f^i y^i}{\sum_{i=1}^{M} f^i} \tag{12}$$

this interval set is determined by its two end points, y_l and y_r, which corresponds to the centroid of the type-2 interval consequent set $\tilde{G}^i$ [23],

$$C_{\tilde{G}^i} = \int_{\theta_1 \in J_{y1}} \cdots \int_{\theta_N \in J_{yN}} 1 \Big/ \frac{\sum_{i=1}^{N} y_i \theta_i}{\sum_{i=1}^{N} \theta_i} = [y_l^i, y_r^i] \tag{13}$$

before the computation of $Y_{\cos}$ (**x**), we must evaluate equation 13, and its two end points, y_l and y_r. If the values of f_i and y_i that are associated with y_l are denoted f_l^i and y_l^i, respectively, and the values of f_i and y_i that are associated with y_r are denoted f_r^i and y_r^i, respectively, from 12, we have [23]

$$y_l = \frac{\sum_{i=1}^{M} f_l^i y_l^i}{\sum_{i=1}^{M} f_l^i} \tag{14}$$

$$y_r = \frac{\sum_{i=1}^{M} f_r^i y_r^i}{\sum_{i=1}^{M} f_r^i} \tag{15}$$

2.1.5 Defuzzifier

From the type-reducer we obtain an interval set $Y_{\cos}$, to defuzzify it we use the average of y_l and y_r, so the defuzzified output of an interval singleton type-2 FLS is [23]

$$y(\mathbf{x}) = \frac{y_l + y_r}{2} \tag{16}$$

2.2 Evolutionary Methods Applied to Fuzzy Systems

2.2.1 Genetic Fuzzy Systems

Fuzzy systems have been successfully applied to problems in classification [8], modeling [24] control [11], and in a considerable number of applications. In most cases, the key for success was the ability of fuzzy systems to incorporate human expert knowledge. In the 1990s, despite the previous successful history, the lack of learning capabilities characterizing most of the works in the field generated a certain interest for the study of fuzzy systems with added learning capabilities.

Two of the most successful approaches have been the hybridization attempts made in the framework of soft computing, were different techniques, such as neural and evolutionary; provide fuzzy systems with learning capabilities. Neuro-fuzzy systems are one of the most successful and visible directions of that effort. A different approach to achieve hybridization has lead to genetic fuzzy systems (GFSs). A GFS is basically a fuzzy system augmented by a learning process based on a genetic algorithm (GA) [9].

GAs are search algorithms, based on natural genetics, that provide robust search capabilities in complex spaces, and thereby other a valid approach to problems requiring efficient and effective search processes [16,19,20]. Genetic learning processes cover different levels of complexity according to the structural changes produced by the algorithm [10], from the simplest case of parameter optimization to the highest level of complexity of learning the rule set of a rule based system. Parameter optimization has been the approach utilized to adapt a wide range of different fuzzy systems, as in genetic fuzzy clustering or genetic neuro-fuzzy systems.

An analysis of the literature shows that the most prominent types of GFSs are genetic fuzzy rule-based systems (GFRBSs) [9], whose genetic process learns or tunes different components of a fuzzy rule-based system (FRBS). Inside GFRBSs it is possible to distinguish between either parameter optimization or rule generation processes, that is, adaptation and learning.

It is important to distinguish between tuning (alternatively, adaptation) and learning problems:

- Tuning is concerned with optimization of an existing FRBS, whereas learning constitutes an automated design method for fuzzy rule sets that starts from scratch. Tuning processes assume a predefined RB and have the objective to find a set of optimal parameters for the membership and or the scaling functions, DB parameters.
- Learning processes perform a more elaborated search in the space of possible RBs or whole KBs and do not depend on a predefined set of rules.
- They are different kind of genetic fuzzy systems applications, but we focused on genetic tuning for optimization parameters of membership functions [2] [1] [7].

2.2.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling [13].

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) [15]. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike the GA, the PSO has no evolution operators such as crossover and mutation. In the PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles [4].

Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far (The fitness value is also stored). This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest* [17].

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations [27].

In the past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods [3] [18].

Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement [28].

3 Fuzzy Logic Controller Design

In this section we design a fuzzy logic controller (FLC) where the optimal controller was found with first evolutionary method, which in this case is the genetic algorithm.

The FLC a Takagi-Sugeno type of fuzzy systems is used with two inputs a) error, and b) error change, with three membership functions each input, "Negative, Zero and Positive" (Gaussian and triangular), and one output which are constant

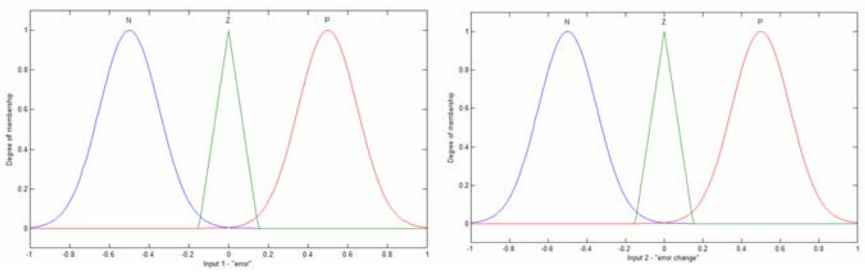

Fig. 4. a) input 1 "error", b) input 2 "error change".

Table 1. Fuzzy Rules of the FLC.

	N	**Z**	**P**
N	N	N	Z
Z	N	Z	P
P	Z	P	P

values. The fuzzy rules "If-Then" type. Figure 5 shows the FLC base for the plant control and Table 1 show the Fuzzy Rules.

Once we obtained the FLC design, we used an evolutionary method (Genetic Algorithm) to find the optimal Controller. The genetic algorithm chromosome has 17 genes of real values and they represent the two inputs, error and error change and one output constant values. Figure 6 show the chromosome representation for the FLC and Table 2 shows the parameters of the membership functions, the minimal and the maximum values in the search range for the genetic algorithm to find the best fuzzy controller system.

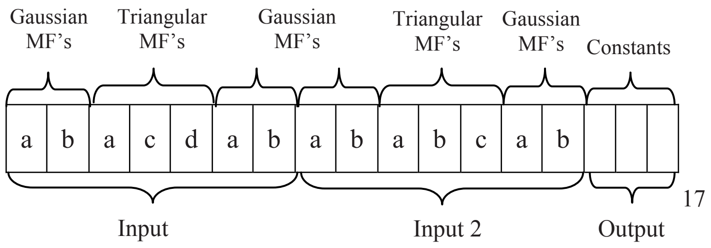

Fig. 5. Chromosome representation for the fuzzy logic controller.

4 Simulations Results

In this section, we evaluate, through computer simulations performed in MATLAB® and SIMULINK®, the designed FLC for two plants.

4.1 Plant 1

Plant 1 is given by the following transfer function:

$$g(s) = \frac{w_n^2}{s^2 + 2\varepsilon\, w_n s + w_n^2} \qquad \varepsilon = 0.5, \quad w_n = 2 \tag{17}$$

Table 2 presents the parameters of the membership functions used in the genetic algorithm for the plant 1.

Table 3 present the main results of the FLC obtained by genetic algorithms showing in the result 3 our best result.

Table 2. Parameters of the membership functions

MF Type	Point	Minimum Value	Maximum Value
Gaussian	a	0.3	0.6
	b	-1	-1
Triangular	a	-0.3	-0.8
	b	0	0
	c	0.3	0.8
Gaussian	a	0.3	0.6
	b	1	1

Table 3. Results of the FLC obtained by genetic algorithms.

No.	Indiv.	Gen.	% Remp.	Cross.	Mut.	GA Time	Average error
1	90	35	0.7	0.6	0.3	00:18:06	0.050870
2	150	80	0.7	0.5	0.2	01:19:13	0.044310
3	**80**	**50**	**0.7**	**0.5**	**0.2**	**00:23:01**	**0.071366**
4	45	60	0.7	0.6	0.3	00:16:03	0.068477
5	75	50	0.7	0.6	0.1	00:21:37	0.068158
6	100	40	0.7	0.6	0.1	00:24:08	0.067052
7	65	35	0.7	0.7	0.2	00:15:35	0.069994
8	200	70	0.7	0.4	0.1	01:30:02	0.072356
9	25	15	0.7	0.8	0.3	00:02:58	0.129872
10	50	45	0.7	0.5	0.2	00:13:25	0.068855
11	90	35	0.7	0.6	0.2	00:19:15	0.065290
12	40	25	0.7	0.7	0.4	00:06:48	0.175755
13	120	454	0.7	0.4	0.1	00:29:51	0.065761

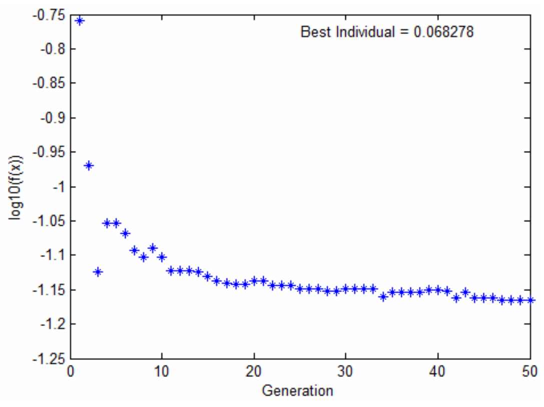

Fig. 6. Evolution of the GA for the FLC optimization.

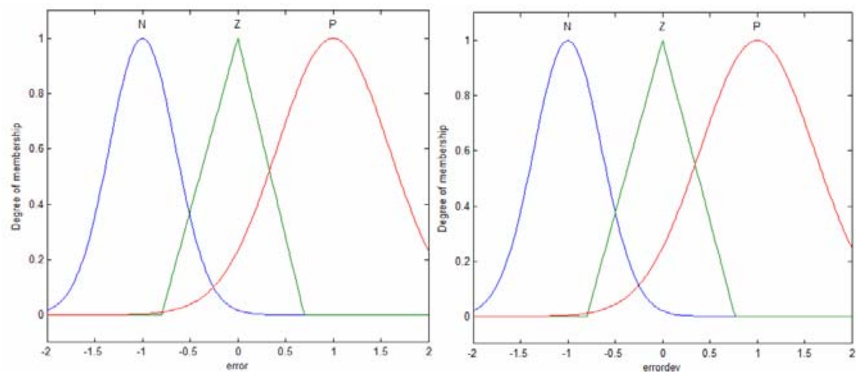

Fig. 7. Membership functions of the FLC obtained by GA.

Figure 6 shows the evolution of the genetic algorithm giving the best FLC for control the plant and figure 7 show the membership functions of the FLC obtained by the GA.

Figure 8 show the control result of the plant 1 using the FLC optimized obtained by GA.

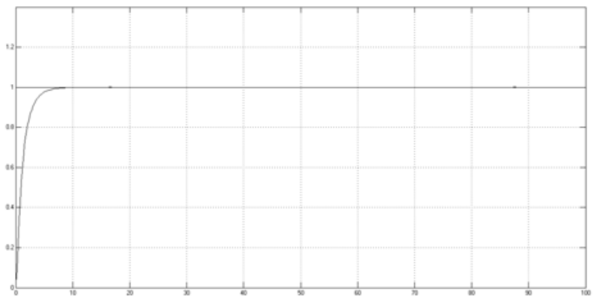

Fig. 8. Simulation result of the control of plant 1.

4.2 Plant 2

Plant 2 is given by the following equation:

$$g(s) = \frac{1}{s^2 + 4} \tag{18}$$

For the experiment with plant 2 we test different range of the membership function finding the best result in the range of the -10 to 10. Table 4 present the main results of the FLC obtained by genetic algorithms.

Table 4. Simulation results of plant 2.

No.	Ind.	Gen.	% Remp.	Cross	Mut.	GA Time	Average error	Param. Range of MF
1	45	20	0.7	0.4	0.1	00:05:10	0.889298	from 0.5 to 1
2	160	75	0.5	0.6	0.1	01:10:43	0.889298	from -1 to 1
3	80	30	0.7	0.6	0.1	00:14:01	0.889302	
4	50	20	0.7	0.5	0.1	00:05:13	0.889784	
5	50	20	0.7	0.5	0.1	00:05:11	0.894638	
6	55	45	0.5	0.6	0.1	00:12:45	0.574186	from -2 to 2
7	95	65	0.5	0.5	0.1	00:32:13	0.569497	
8	150	70	0.5	0.4	0.1	00:53:32	0.569667	
9	55	45	0.5	0.6	0.1	00:12:45	0.473400	from -4 to 4
10	150	70	0.5	0.4	0.1	00:53:52	0.447686	
11	95	65	0.5	0.5	0.1	00:30:50	0.461219	
12	150	100	0.5	0.5	0.1	01:20:24	0.064981	from -8 to 8
13	95	65	0.5	0.5	0.1	00:32:07	0.065394	
14	85	65	0.5	0.5	0.1	00:29:11	0.066594	
15	150	70	0.5	0.4	0.1	00:54:30	0.068396	
16	55	45	0.5	0.6	0.1	00:12:50	0.073958	
17	80	50	0.5	0.6	0.1	00:22:18	0.076842	
18	150	70	0.5	0.4	0.1	00:54:48	0.196999	
19	95	65	0.5	0.5	0.1	00:30:56	0.197443	
20	55	45	0.5	0.6	0.1	00:37:22	0.203600	
21	55	45	0.5	0.6	0.1	00:12:16	0.213285	
22	200	90	0.5	0.4	0.1	01:33:19	0.065268	from -10 to 10
23	**120**	**85**	**0.5**	**0.4**	**0.1**	**00:52:13**	**0.070636**	
24	90	35	0.5	0.5	0.2	00:16:26	0.074058	
25	65	35	0.5	0.4	0.2	00:11:53	0.076711	
26	55	45	0.5	0.6	0.1	00:39:01	0.077597	
27	100	40	0.5	0.3	0.1	00:20:41	0.078355	
28	40	25	0.5	0.7	0.1	00:05:12	0.134507	
29	25	15	0.5	0.8	0.3	00:02:15	0.261787	

Figure 9 shows the evolution of the genetic algorithm and figure 10 shows the membership functions of FLC obtained by GA.

Figure 11 show the control result of the plant 2 using the FLC optimized obtained by GA.

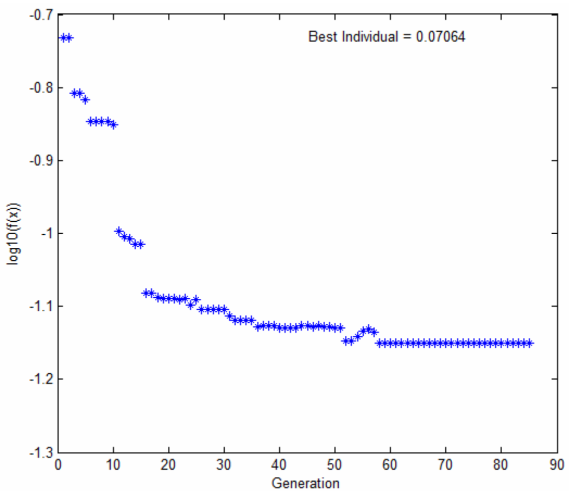

Fig. 9. Evolution of the GA for the FLC optimization.

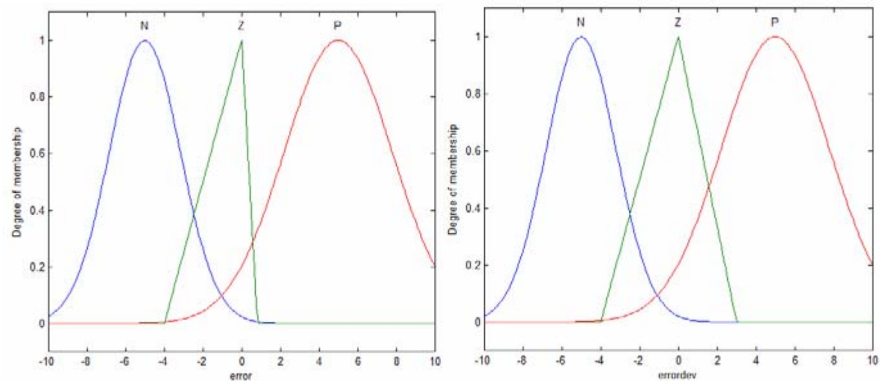

Fig. 10. Membership functions of the FLC obtained with the GA.

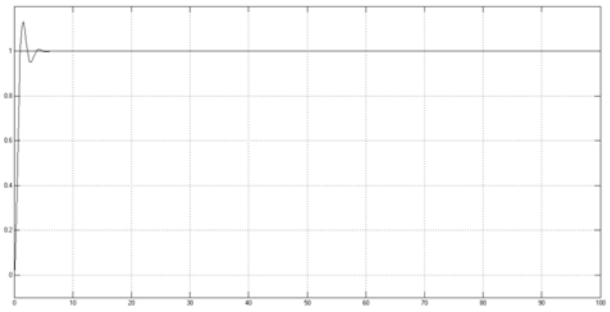

Fig. 11. Simulation result of the control Plant 2.

5 Conclusions

We described the use of evolutionary methods for the design of optimized FLC's; in particular we present results of a genetic algorithm in FLC optimization for linear plants. The results of first plant are satisfactory controlling the plant and getting stability in less than 10 seconds using a fuzzy logic controller with three membership functions and nine fuzzy rules. On the other hand, in second plant we can get stability faster that the first plant close to five seconds, but with a high overshoot and undershoot for the plant. We have achieved satisfactory results with genetic algorithms and the next step is to solve the problem using multiple objective optimizations to obtain better results. Moreover, we will extend the results for nonlinear systems like autonomous mobile robots.

References

1. Alcalá, R., Alcalá-Fdez, J., Herrera, F.: A proposal for the Genetic Lateral Tuning of Linguistic Fuzzy Systems and its Interaction with Rule Selection. IEEE Transactions on Fuzzy Systems 15(4), 616–635 (2007)
2. Alcalá, R., Gacto, M.J., Herrera, F., Alcalá-Fdez, J.: A Multi-objective Genetic Algorithm for Tuning and Rule Selection to Obtain Achúrate and Compact Linguistic Fuzzy Rule-Based Systems. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 15(5), 539–557 (2007)
3. Angeline, P.J.: Using Selection to Improve Particle Swarm Optimization. In: Proceedings 1998 IEEE World Congress on Computational Intelligence, Anchorage, Alaska, pp. 84–89. IEEE, Los Alamitos (1998)
4. Angeline, P.J.: Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)
5. Bentalba, S., El Hajjaji, A., Rachid, A.: Fuzzy Control of a Mobile Robot: A New Approach. In: Proc. IEEE Int. Conf. on Control Applications, Hartford, CT, October 1997, pp. 69–72 (1997)
6. Brockett, R.W.: Asymptotic stability and feedback stabilization. In: Millman, R.S., Sussman, H.J. (eds.) Differential Geometric Control Theory, pp. 181–191. Birkhauser, Boston (1983)
7. Casillas, J., Cordon, O., del Jesús, M.J., Herrera, F.: Genetic Tuning of Fuzzy Rule Deep Structures Preserving Interpretability and its Interaction with Fuzzy Rule Set Reduction. IEEE Transaction on Fuzzy Systems 13(1), 13–29 (2005)
8. Chi, Z., Yan, H., Pham, T.: Fuzzy Algorithms: With Applications to Image Processing and Pattern recognition. World Scientific, Singapore (1996)
9. Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten Years of Genetic Fuzzy Systems: Current Framework and New trends. Fuzzy Sets and Systems 141(1), 5–31 (2004)
10. DeJong, K.: Learning with genetic algorithms: an overview. Mach. Learning 3(3), 121–138 (1988)
11. Driankov, D., Hellendoorn, H., Reinfrank, M.: An Introduction to Fuzzy Control. Springer, Berlin (1993)
12. Duc Do, K., Zhong-Ping, J., Pan, J.: A global output-feedback controller for simultaneous tracking and stabilizations of unicycle-type mobile robots. IEEE Trans. Automat. Contr. 30, 589–594 (2004)

13. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995); Lu, J.-G.: Title of paper with only the first word capitalized. J. Name Stand. Abbrev. (in press)
14. Fierro, R., Lewis, F.L.: Control of a Nonholonomic Mobile Robot Using Neural Networks. IEEE Trans. on Neural Networks 9(4), 589–600 (1998)
15. Fogel, D.B.: An introduction to simulated evolutionary optimization. IEEE transactions on neural networks 5(1), 3–14 (1994)
16. Fukao, T., Nakagawa, H., Adachi, N.: Adaptive Tracking Control of a NonHolonomic Mobile Robot. IEEE Trans. on Robotics and Automation 16(5), 609–615 (2000)
17. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceeding of IEEE conference on Evolutionary Computation, pp. 1671–1676 (2002)
18. Kennedy, J., Mendes: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
19. Lee, T.H., Leung, F.H.F., Tam, P.K.S.: Position Control for Wheeled Mobile Robot Using a Fuzzy Controller, pp. 525–528. IEEE, Los Alamitos (1999)
20. Liang, Q., Mendel, J.M.: Interval Type-2 Fuzzy Logic Systems: Theory and Design. IEEE Trans. on Fuzzy Systems 8(5), 535–550 (2000)
21. Mendel, J., Mouzouris George, C.: Type-2 fuzzy logic systems. IEEE Trans. Fuzzy Systems 7, 643–658 (1999)
22. Mendel, J., Bob John, R.I.: Type-2 Fuzzy Sets Made Simple. IEEE Transactions on Fuzzy Systems 10(2) (April 2002)
23. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and new directions. Prentice Hall, USA (2000)
24. Pedrycz, W. (ed.): Fuzzy Modelling: Paradigms and Practice. Kluwer Academic Press, Dordrecht (1996)
25. Sepulveda, R., Castillo, O., Melin, P., Montiel, O.: An Efficient Computational Method to Implement Type-2 Fuzzy Logic in Control Applications, Analysis and Design of Intelligent Systems Using Soft Computing Techniques. Advances in Soft Computing 41, 45–52 (2007)
26. Takagi, T., Sugeno, M.: Fuzzy Identification of Systems and its application to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics 15(1) (1985)
27. Veeramachaneni, K., Osadciw, L., Yan, W.: Improving Classifier Fusion Using Particle Swarm Optimization. In: IEEE Fusion Conference, Italy (July 2006)
28. Wei, W., Jiatao, S., Zhongzxiu, Y., Zheru: Wavelet-based Illumination Compensation for Face Recognition using Eigenface Method Intelligent Control and Automation. In: WCICA 2006, June 21-23, vol. 2, pp. 10356–10360 (2006)