Particle Swarm Optimization for Designing an Optimal Fuzzy Logic Controller of a DC Motor

Fevrier Valdez, Patricia Melin and Oscar Castillo

Tijuana Institute of Technology
Tijuana B.C. Mexico
fevrier@tectijuana.mx,ocastillo@tectijuana.mx,pmelin@tectijuana.mx

Abstract— **The speed of the DC motor can be adjusted to a great extent so as to provide easy control and high performance. There are several conventional and numeric types of controllers intended for controlling the DC motor speed and executing various tasks: PID Controller, Fuzzy Logic Controller; or the combination between them: Fuzzy-Swarm, Fuzzy-Neural Networks, Fuzzy-Genetic Algorithm, Fuzzy-Ants Colony. We describe in this paper the use of Particle Swarm Optimization (PSO) for designing an optimal fuzzy logic controller of a DC Motor. In this case, our approach will optimize the membership functions of a fuzzy logic controller (FLC) using PSO and the obtained results were simulated on Simulink of Matlab.**

Keywords-PSO;Fuzzy Logic;Optimization

I. INTRODUCTION

We describe in this paper an approach for designing a fuzzy logic controller of a DC Motor optimized with PSO.

Nowadays, to obtain optimal controllers is a hard task and is complicated to choose the optimal parameters in the control system; therefore, in this case, we design an optimal fuzzy logic controller using an optimization method, in this case PSO [1][2]. The method is called FLC+PSO. Several approaches have been developed in the control area to solve problems of tuning some parameters in some parts of the systems, for example, in [3] it is shown a similar approach using PSO applied to the Preview Control problem, which regularly is obtained using the Algebraic Riccati Equation. The solution obtained is good but it is not optimal and has a scope of improvement. Therefore the authors presented a method of design of a Preview Controller using the Particle Swarm Optimization method. The procedure is based on improving the performance by minimizing the objective function i.e. IAE (Integral of Absolute Error). In [4] it is presented a PSO for reactive power and voltage control (volt/VAr control: VVC) considering voltage security assessment (VSA). VVC can be formulated as a mixed-integer nonlinear optimization problem (MINLP). The proposed method expands the original PSO to handle a MINLP and determines an online VVC strategy with continuous and discrete control variables such as automatic voltage regulator (AVR) operating values of generators, tap positions of on-load tap changer (OLTC) of transformers, and the number of reactive power compensation equipment. However our approach is different because we are using a fuzzy system, where the PSO is within the membership functions to find the best parameters of the membership function and to achieve a good control. The paper is described as follow: In section I the introduction is presented, the theory about the PSO is shown in section II. Also, the fuzzy logic controller used in this paper is presented in section III, the description of the proposed method is mentioned in section IV, in section V the experimental results obtained are shown; and finally the conclusions are presented in section VI.

II. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by the social behavior of bird flocking or fish schooling [5].

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) [6]. 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 PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles [7].

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*.

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 [11] [12].

In the past several years, PSO has been successfully applied in many research and application areas. It has been demonstrated that PSO gets better results in a faster, cheaper way compared with other methods [8].

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.

The pseudo code of the PSO is as follows:

For each particle
Initialize particle
End
Do
For each particle
Calculate fitness value
If the fitness value is better than the best fitness value (pBest) in history
set current value as the new pBest
End
Choose the particle with the best fitness value of all the particles as the gBest
For each particle
Calculate particle velocity
Update particle position
End
While maximum iterations or minimum error criteria is not attained

III. Fuzzy Logic Controller

Fuzzy Logic is a form of logical reasoning that can be incorporated into automation systems typically human reasoning schemes. Fuzzy theory was first proposed and investigated by Prof. Zadeh in 1965 [9]. One of the main features of fuzzy logic is its ability to operate with vague or ambiguous concepts typical of qualitative reasoning, based on a mathematical support quantitative conclusions can be drawn from a set of observations (records) and qualitative rules (based on knowledge). Fuzzy Control is the application of fuzzy inference process automation. A typical fuzzy controller infers the consequent of more or less large simple rules (knowledge base), this process of reasoning can be performed in parallel, yielding the result (consistent) with a simple logical sum.

This parallel processing capability allows even relatively complex controllers to perform the fuzzy inference in a minimal computation time. In addition, due to the characteristics of fuzzy logic, it is often possible to design a well-adjusted to the process controller leveraging previous experience of an operator, thus eliminating the need for complex and laborious technical studies of the control problem. On the other hand, the controller tuning is usually much easier (it is relatively easy to know what rules are affecting the behavior of the regulator in a given situation), and also much safer to be able to operate with a high degree of redundancy (it is possible to define conflicting rules for similar situations without this the system stops working, the redundancy increases the immunity of the system against errors in the knowledge base). It should be noted that the application of fuzzy logic in process control is not at all at odds with the use of other conventional control techniques. On the contrary, fuzzy logic is particularly suitable for the formulation of hybrid controllers, allowing you to convert a variety of control structures.

A. Linguistic variables and rules

There are two input variables (error and change of error), and one variable in the output (voltage) and seven linguistic variables. The best fuzzy controller attributes obtained with PSO are the following:

Name='motordcp'
Type='mamdani'
NumInputs=2
NumOutputs=1
NumRules=15
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'
[Input1]
Name='VelM'
Range=[0 120]
NumMFs=5

The fuzzy inference system generated by the PSO is shown in Fig. 1, in this case, the PSO changed several times the membership functions. The membership functions were triangular because in past work [10] we obtained good results. However, PSO is able to find these functions.

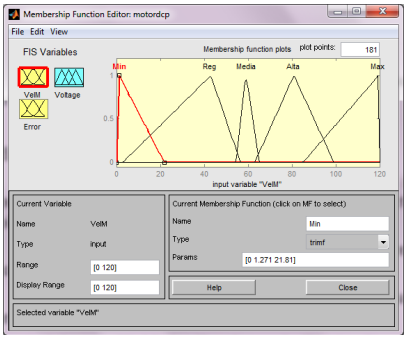

Figure 1. Best FLC obtained by the PSO.

The output in the FLC obtained by the is shown in Fig. 2. We can see, how the second membership function is bigger than anothers 4 membership functions.

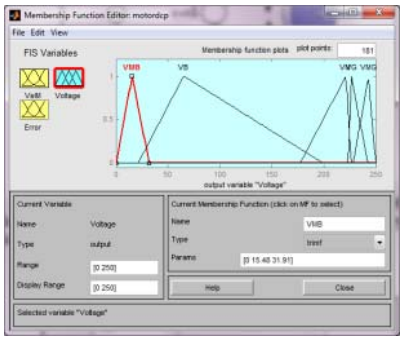

Figure 2. Output of the FLC obtained by the PSO.

The membership functions generated by the PSO are used and rules are design tools that give opportunity to model a control surface and controller properties. It is obvious that

using this attributes one can more precisely fulfill a quality criterion in full operational range. The control surface is shown in Fig. 3

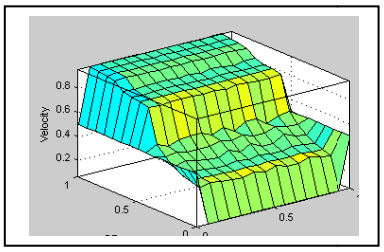

Figure 3. Surface of the fuzzy system.

Fig. 2 shows the best membership functions obtained by the PSO in the performed experiments.

B. Physical Setup

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in Fig. 4. The DC motor is a machine that converts electrical energy into mechanical, causing a rotary motion. A DC machine (generator or motor) is composed mainly of two parts, a stator which gives mechanical support to the unit and has a hole in the center generally cylindrical in shape. In addition to the stator, poles are located, which may be permanent magnets or windings of copper wire on the iron core. The rotor is generally cylindrical in shape, too winding and core, which is reached by two current brushes.

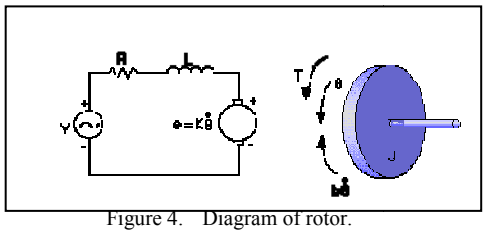

Figure 4. Diagram of rotor.

Fig. 5 shows the Simulink model implemented in Matlab, where the main block is a fuzzy logic controller obtained by the PSO.

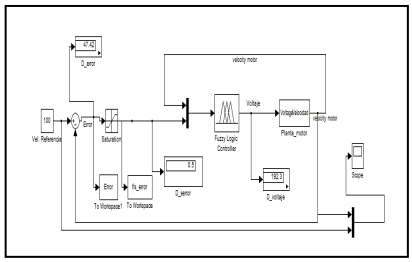

Figure 5. Simulink model of the controller.

IV. Proposed Method

The methodology of this paper involved the development of a PSO algorithm to optimize the parameters of membership functions of a fuzzy logic controller that controls the velocity of a DC motor. This is possible evaluating the ranges of membership functions within the fuzzy logic controller. After getting the optimal parameter the full model was implemented in Matlab with Simulink where the PSO will create the optimal topologies to achieve a possible solution to this problem, allowing each particle vector containing the potential range of optimal membership for the fuzzy system and to obtain a better result in the simulation. As the error achieved by the plant is lower, the cost of the particle will be saved as the minimum cost of such a particle to subsequently save the best overall result from all over the swarm.

A. Mathematical Description

For this example, we will assume the following values for the physical parameters.

moment of inertia of the rotor (J) = 0.01 kg.m^2/s^2
damping ratio of the mechanical system (b) = 0.1 Nms
electromotive force constant (K=Ke=Kt) = 0.01 Nm/Amp
electric resistance (R) = 1 ohm
electric inductance (L) = 0.5 H
input (V): Source Voltage
output (theta): position of shaft

The rotor and shaft are assumed to be rigid.
The motor torque, **T**, is related to the armature current, **i**, by a constant factor $\mathbf{K_t}$. The back electromotive force constant, **e**, is related to the rotational velocity by the following equations:

$$T = K_t i$$
$$e = K_e \dot{\theta} \quad (1)$$

In SI units $\mathbf{K_t}$ (armature constant) is equal to $\mathbf{K_e}$ (motor constant).
From Fig. 4 above we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J\ddot{\theta} + b\ \dot{\theta} = Ki$$
$$L\frac{di}{dt} + Ri = V - K\dot{\theta} \quad (2)$$

Using Laplace Transforms, the above modeling equations can be expressed in terms of s.

$$s(Js + b)\Theta(s) = Ki$$
$$(Ls + R)I(s) = V - Ks\Theta(s) \quad (3)$$

By eliminating I(s) we can get the following open-loop transfer function, where the rotational speed is the output and the voltage is the input.

$$\frac{\theta}{V} = \frac{K}{(Js + b)(Ls + R) + K^2} \quad (4)$$

In the state-space form, the equations above can be expressed by choosing the rotational speed and electric current

as the state variables and the voltage as an input. The output is chosen to be the rotational speed.

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}=\begin{bmatrix}-\frac{b}{J} & \frac{K}{J}\\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}+\begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}V$$
$$\dot{\theta}=[1\quad 0]\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} \tag{5}$$

The flow diagram used in this paper is shown in Fig. 6

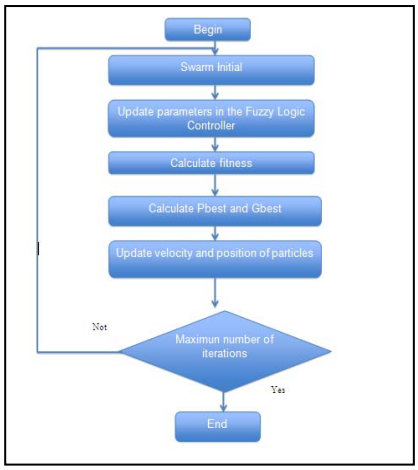

Figure 6. Flow diagram of the PSO.

It is important to mention that a fuzzy system was used with 2 inputs and 2 outputs to obtain in run time the best values of cognitive acceleration (C_1) and social aceleration (C_2). This fuzzy system is describe as follows:

'fuzzypso': In this case we are using a fuzzy system called 'fuzzypso', and the structure of this fuzzy system is as follows:

Number of Inputs: 2
Number of Outputs: 2
Number of membership functions: 3
Type of the membership functions: Triangular
Number of rules: 9
Defuzzification: Centroid
The main function of the fuzzy system called 'fuzzypso' is to adjust the parameters of the PSO. In this case, we are adjusting the following parameters: 'C_1' and 'C_2'; where:
'c_1' = Cognitive Acceleration
'c_2' = Social Acceleration

We are changing these parameters to test the proposed method. In this case, with 'fuzzypso' is possible to adjust in real time the 2 parameters that belong to the PSO. The equation of velocity in PSO is describe as follow:

$$v_{ij}(t+1)=vi_j(t)+c_1r_{1j}(t)[y_{ij}(t)-x_{ij}(t)]+c_2r_{2j}(t)[\hat{y}_j(t)-x_{ij}(t)] \tag{6}$$

In equation 2 it is described the way in which the basic equation of PSO is modified to achieve the goal, and then, convert part of it in fuzzy parameters, we can see the differences between two equations that C_1 and C_2 in equation 2 are those that change, because an essential part of the proposed method lies in those two variables. Traditionally these two variables are constant, in this case, the importance of these two accelerations, is that we decided to obtain these two as follows parameters.

Where $vi_j(t)$ is the velocity of particle i in dimension j = 1,…,n_x at time step t, $xi_j(t)$ is the position of particle i in dimension j at time step t, c_1 and c_2 represents the cognitive and social acceleration. In this case, these values are fuzzy because they are changing dynamically when the FPSO is running, and are defined by expressions 3 and 4, and r_{1j}(t), r_{2j}~U (0, 1) are random values in the range [0, 1].

$$c_1=\frac{\sum_{i=1}^{r_{c1}}\mu_i^{c_1}(c_{1i})}{\sum_{i=1}^{r_{c1}}\mu_i^{c_1}} \tag{7}$$

Where: c_1 = Percentage of cognitive acceleration of the particle *i*.

r_{c1} = Number of rules of the fuzzy system corresponding to c_1.

c_{1i} = Output result for rule *i* corresponding to c_1.

$\mu_i^{c_1}$ = Membership function of rule *i* corresponding to c_1.

$$c_2=\frac{\sum_{i=1}^{r_{c2}}\mu_i^{c_2}(c_{2i})}{\sum_{i=1}^{r_{c2}}\mu_i^{c_2}} \tag{8}$$

Where: c_2 = Percentage of cognitive acceleration of the particle *i*.

r_{c2} = Number of rules of the fuzzy system corresponding to c_2.

c_{2i} = Output result for rule *i* corresponding to c_2.

$\mu_i^{c_2}$ = Membership function of rule *i* corresponding to c_2.

V. Experimental Results

Several tests were made using a computer with 2.4 Ghz speed CPU and 4 GB RAM Memory. The programming

language used was Matlab. Also, to validate our approach and to use the algorithm graphically, a graphic interface was developed. Fig. 7 shows the main screen of this interface. The users easy and fast can use the algorithm. The interface contains 5 edit texts and 3 buttons, the users can setup the parameters on PSO and with the start button the PSO algorithm begin. Finally the simulation can be seen with the Simulation button. The button Show FIS show the structure of the Fuzzy Inference System obtained by the PSO. Also, the best global particle and the optimal parameters are shown.

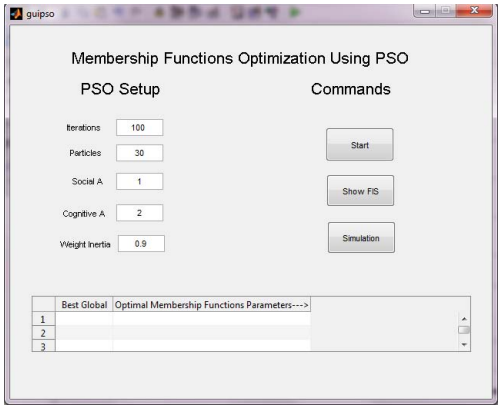

Figure 7. Graphic interface developed for simulation.

Table I shows the experimental results obtained using PSO. We can see in this Table how in the 15 tests the PSO was able to achieved good results, however the best result obtained with the method was in test number 10. Each test was executed 10 times to calculated the Mean Error. Fig. 8 shows a PSO convergence in one of the tests made.

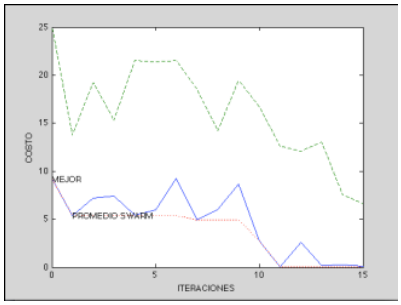

Figure 8. PSO convergence with 15 iterations.

Several simulations were made in simulink of Matlab, Fig. 9 shows one simulation obtained with the FLC+PSO to a constant reference of 117. We can see how the controller was able to achieve good result.

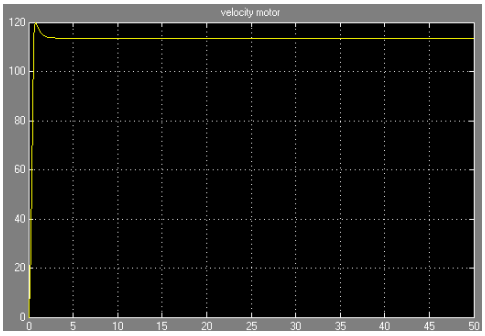

Figure 9. Example of simulation results.

TABLE I. EXPERIMENTAL RESULTS USING PSO

Test	Setup PSO			
	Iterations	Particles	Error Mean	Best Error
1	10	10	1.39	0.0489
2	15	10	1.03	0.0180
3	30	25	3.98	0.9723
4	20	25	2.35	0.0723
5	25	30	1.20	0.0623
6	35	30	2.46	0.2180
7	40	30	1.51	0.9815
8	50	30	2.98	0.9999
9	100	30	1.01	0.0258
10	150	40	0.10	0.0005
11	80	40	2.05	0.9524
12	40	50	1.94	0.7891
13	50	50	1.69	0.9875
14	60	60	1.412	0.0541
15	100	60	1.225	0.5568

Also, several tests were made with a fuzzy system without use the PSO. We developed 15 fuzzy inference systems manually, and the best results obtained manually are shown in Table II. We calculated the Mean Error as was made in Table I. In this case, test number 6 was better than the other tests. However, to achieve good results it was necessary optimizing the membership functions. Maybe, to improve the results obtained manually, more tests should be made but to solve this problem the PSO was used. Also, several configurations were made in the PSO algorithm to achieved good results, for example, PSO was applied with Weight inertia, constriction factor, only cognitive and only social model. Also several topologies were used in the different applied configurations. However, the best PSO that was obtained is the following:

Number of particles: 100.

Cognitive Acceleration (C_1) = 0.5. (Obtained with a fuzzy system)

Social Acceleration (C_2) = 1. (Obtained with a fuzzy system)

Topology of particles = Star.

Weight of inertia (W) = 0.8.

TABLE II. EXPERIMENTAL RESULTS WITHOUT PSO

Test	Results of the FLC without PSO	
	Error Mean	Best Error
1	2.45	2.0214
2	5.75	3.0125
3	6.87	2.9999
4	7.75	3.5226
5	10.52	5.2563
6	1.58	0.9955
7	6.87	2.5625
8	9.85	3.6987
9	2.65	1.9586
10	5.25	1.2542
11	3.21	1.2546
12	2.14	2.0011
13	9.87	2.5123
14	6.35	4.6890
15	2.59	1.1874

A. T- Student Test

To validate the proposed method with PSO a T-Student Test was made between the FLC+PSO and FLC without PSO. Table III shows the obtained results after having applied this test. We can see in Table III how the proposed method FLC+PSO was better than only using the FLC. The T-Value obtained by the T-Student was good. Fig. 10 shows the plot of the T-Student test, where, we can see how the difference between both methods is statistically significant.

TABLE III. TWO-SAMPLE T-TEST FOR FLC+PSO VS FLC

Method	T Student Test			
	Data	Mean	StDev	SE Mean
FLC	15	0.449	0.444	0.11
FLC+PSO	15	2.60	1.288	0.33
Estimate for difference	-2.146			
95% CI for difference	-2.886, -1.405			
T-Value	-6.11			
P-Value	0.000			
DF	17			

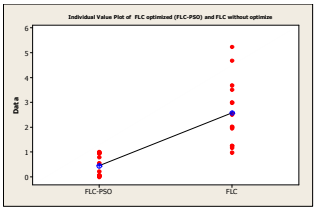

Figure 10. T-Student Test plot of the values.

VI. CONCLUSIONS

After having applied the proposed FLC+PSO method we can conclude in this paper, that the use of optimized Fuzzy Logic Controllers is possible to achieve very good results. In particular with this application we are demonstrating statistically that there is significant difference when the controllers are developed manually or automatically. Therefore, with the results presented in the paper we can recommend the use of optimization methods to find some important parameters, in this case, PSO was only used to design the optimal topology of the membership functions. However, Genetic Algorithms, Ant Colony Optimization and other approaches could also be used to achieve this goal. We decided to use PSO, because it is a very good method to implement with this type of applications and is faster that other similar approaches to achieve the goal. Experimental results show better performances that are achieved with the proposed method, optimization with the proposed method FLC+PSO, when it is compared with the controllers without optimization. Our approach could be compared with other similar methods, for example, in [13] a similar approach is shown, this method optimizes the membership functions, however our proposed method optimizes membership functions and is able to adapt parameters in the PSO in run time, which leads to more robust and better method than the above mentioned approach.

ACKNOWLEDGMENT

We would like to express our gratitude to CONACYT, Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

REFERENCES

[1] H. Hénao, G.A. Capolino, "Méthodologie et application du diagnostic pour les systèmes électriques", Article invité dans Revue de l'Electricité et de l'Electronique (REE), No. 6, (in French), June 2002, p.p. 79 – 86.

[2] S. Raghavan, "Digital Control for Speed and Position of a DC Motor", MS Thesis, Texas A&M University, Kingsville, August 2005.

[3] N. Birla, A. Swarup A., "PSO approach to preview tracking control systems" TENCON 2009 - 2009 IEEE Region 10 Conference, p.p 1-6, Singapore, 23-26 Jan. 2009.

[4] H. Yoshida, K. Kawata, Y. Fukuyama; S. Takayama; Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment" Power Systems, IEEE Transactions, Volume: 15, p.p. 1232 - 1239, Nov 2000.

[5] J. Kennedy, and R. C. Eberhart, "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. p.p. 1942-1948, 1995.

[6] O. Castillo, P. Melin, "Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory" Neural Networks, IEEE Transactions on Volume 13, Issue 6, p.p. 1395 – 1408. Nov. 2002.

[7] F. Valdez, and P. Melin, "Parallel Evolutionary Computing using a cluster for Mathematical Function Optimization", Nafips. San Diego CA, USA, p.p. 598-602. June 2007.

[8] P. Angeline, "Using Selection to Improve Particle Swarm Optimization", In Proceedings 1998 IEEE World Congress on Computational Intelligence, Anchorage, Alaska, IEEE, p.p. 84-89. 1998.

[9] L. Zadeh, "Fuzzy sets" Info. & Ctl., Vol. 8, p.p. 338-353, 1965.

[10] F. Valdez, P. Melin, O. Castillo, "An improved evolutionary method with fuzzy logic for combining Particle Swarm Optimization and Genetic Algorithms". Appl. Soft Comput. 11(2): p.p. 2625-2632, 2011.

[11] J. Mendes and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space". IEEE Transactions on Evolutionary Computation, 6(1):58-73. 2002.

[12] J. Mendes and J. Kennedy, "Population structure and particle swarm performance". Proceedings of IEEE conference on Evolutionary Computation, p.p. 1671-1676. 2002.

[13] B. Allaoua, A. Abderrahmani, B. Gasbaoui , A.Nasri. "The Efficiency of Particle Swarm Optimization Applied on Fuzzy Logic DC Motor Speed Control", Serbian Journal of Electrical Engineering 2008, vol. 5, no. 2, p.p. 247-262. 2008.