Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Optimizing Fuzzy Membership Functions Using Particle Swarm Algorithm

Elijah E. Omizegba and Gbijah E. Adebayo
Electrical and Electronics Engineering Programme
Abubakar Tafawa Balewa University
Bauchi, Bauchi State, Nigeria
omizegbaee@yahoo.co.uk and agbijah@gmail.com

Abstract – The choice and shape of membership functions are known to affect the performance of fuzzy systems; despite their importance however, MFs are generally defined subjectively based on engineering judgment, designer experience or chosen for computational convenience, which does not necessarily give optimal performance when used in modeling or control. In this paper we present a method for optimizing membership functions of a fuzzy system using particle swarm optimization (PSO) algorithm. The method determines the optimal shapes and span of membership function based on a modeling performance measure. To demonstrate its effectiveness, the proposed method was used to optimize the triangular membership functions of the fuzzy model of a nonlinear system; results show that the optimized MFs provided better performance than a fuzzy model for the same system when the MFs were heuristically defined.

***Keywords*—Optimization, Membership Functions, PSO, Fuzzy Systems**

I. Introduction

Fuzzy logic provides a means of processing vague, imprecise or incomplete information, expressed in linguistic terms. This differs from set theory where a value or an object either belongs to a set or is excluded from it. In fuzzy set theory an object can belong to a set fully or partially (i.e. to a certain extent). This vagueness is captured by membership functions, MF. The extent to which an object belong to a set (called the degree of membership or membership value μ) ranges between 0 and 1. It is usual for MFs to be represented graphically; the popular MFs are the triangular, trapezoidal and the sigmoid functions [1].

The MFs together with a rule base form what is known as the knowledge base of a Fuzzy Inferencing System (FIS), the complete arrangement of a FIS is shown in Fig. 1.

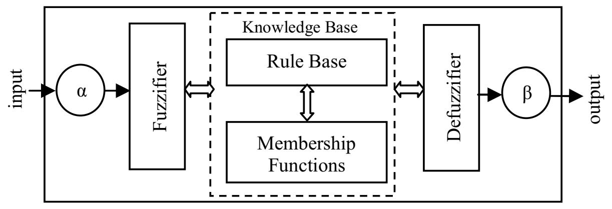


Figure 1. Structure of a Fuzzy Inferencing System

Where α and β are the input and output scaling factors respectively; the fuzzifier converts the crisp input to fuzzy data for processing by the Inference System. After processing the fuzzy output is again converted to a crisp value by the defuzzifier for real world application. Different defuzzification methods are discussed in [2].

By being organized as an inferencing system, it is possible to process information using fuzzy logic; and has found applications in different areas of research and other human endeavors, especially in control, modeling, function estimation and predictive algorithms [3]. Some successful applications of fuzzy set theory can be found in [4]. The results obtained from fuzzy reasoning have been found in many cases to be superior to those of the conventional methods; these successes have been attributed to the ability of the FIS to represent and process imprecise information common to most real-life problems in a practical way [5].

The rest of the paper is organized as follows: Section I presents a brief overview of different fuzzy models, while section II and III gives a brief review of FIS and different methods of determining MFs. Sections IV and V describes the PSO and algorithm for its implementation; The next three sections discusses the problem definition, how MF definition may be encoded into the particles of a PSO and the parameter for determining the progress of the optimization process.. Sections VIII and IX present a numerical problem for verifying the feasibility of the proposed method, and the results of simulations. The last section is the conclusion.

II. Linguistic Variables and Membership Functions

In a FIS, each input variable is divided into overlapping fuzzy partitions such as Large (L), Medium (M), Small (S) and so on, to cover the entire range (or universe-of-discourse) of that input. The different fuzzy partitions are then described by membership functions, and since these partitions overlap, an input value can belong to more than one of the partitions. For the Mamdani type and the New Fuzzy Reasoning Method (NRFM) type fuzzy models, the output is also partitioned in the same way as the input. For Tagaki-Sugeno-Kang (TSK) type fuzzy model, the outputs are represented as linear functions of the input and output variables [1], [6].

From the foregoing it can observed that the fuzzy system is characterized by its MFs [2] and it has been shown by [7] that the type and definitions of the parameters of MFs in a fuzzy set determine the performance of the FIS; For example, increased fuzziness (more partitions) results in more rules and sluggishness of response because of the amount of computations that must be carried out to resolve the rules before an output is generated [8]. Hence the effectiveness of a Fuzzy Inferencing System depends among others factors on the number of partitions and the definitions of the associated MFs.

Despite their importance, there are no empirical methods for determining the shape, the number or the span of membership functions; these are decided subjectively based on engineering judgments, intuition, required application or designer experience; for control purposes for instance, it is common to use triangular MFs, while Gaussian shaped MFs seem to be preferred for function approximators.

III. MF Determination Methods

In [8] an iterative method of gradually reducing the number of MFs in a fuzzy controller based on some performance criteria was proposed, the objective was to reduce the number of rules so as to hasten the speed of response of a flexible spacecraft attitude controller. Also [9] introduced an iterative procedure for altering the spread of MFs for a FIS, even though the method was subjective and time consuming. In other to remove the subjectivity in the definition of MFs, [10] used an aggregation of expert opinions to determine the MFs of a FIS, the final outcome depended largely on each experts' subjective interpretation of the various linguistic terms.

On the other hand, a number of computational methods have been used for the determination and tuning of MFs. One approach by [11] used a clustering algorithm to group a set of raw data into clusters and used an artificial neural network (ANN) to determine the membership functions from the resulting clusters. The method requires a large data set for the algorithm to succeed. Also, [12] used the genetic algorithm (GA) to optimize a set of initially defined MF in a fuzzy system meant to control a helicopter. GA also requires a large data set for a good search, and the encoding of data into the chromosomes of a GA requires good knowledge of the underlying system. In this paper, we use the particle swarm algorithm to optimize the membership function of a NFRM used for modeling a nonlinear plant; the modeling accuracy was compared to that of an expert designed NFRM whose MF were not optimized.

IV. Particle Swarm Optimization

The particle swarm optimization, PSO, algorithm like the genetic algorithm (GA) is inspired by a biological system; while the GA mimics the process of natural evolution, the PSO mimics, for example, a flock of birds migrating in search of a common objective ([13] and [14]). In both algorithms, a population of potential solution candidates (called a *swarm* in PSO, each member is called a *particle*) are used to seek better solution(s) over a number of iteration steps, called generations. The major difference is that while the GA discards the least fit members of the solution candidates after every generation, in PSO each particles is updated based on its own experience and the experience of its neighbors [15].

To guide the search process a performance evaluation parameter called fitness is used; this parameter relates the problem to the search process [13]. Each particle keeps a record of the best position it has found (called the personal best, *pbest*), while the swarm keeps a record of the best solution any particle has attained (called the global best, *gbest* or local best *lbest*). For the case of *gbest*, all particles in the swarm are connected to one another and the search is for a global objective. While for *lbest* a particle is only connected to those particles in its neighborhood; and the search is for some local solutions. The values of *pbest* and *gbest* which are updated after each iteration influences the interaction between particles and ultimately the search process [16].

V. PSO Algorithm

The steps involved in implementing the PSO algorithm can be summarized as:

- Initialize PSO parameters, and N particles
- Determine the fitness of each particle
- Update *pbest* and *gbest*
- Update velocities and positions (particles)
- Terminate on convergence or at end of iteration
- Go to step 2

The N particles together with their velocities may be initialized randomly or based on an expert input; the latter results in a faster convergence. The *pbest* is updated by comparing the current *pbest* of a particle with that of its previous *pbest*, and retaining the better of the two values; while the best of all the *pbests* in any generation is selected as the *gbest*. Velocities and positions are updated using the PSO equations in (1) and (2) respectively.

$$v_n^{i+1} = \omega \cdot v_n^i + c_1 \cdot r_1^i \cdot (\hat{p}_n^i - p_n^i) + c_2 \cdot r_2^i \cdot (\hat{p}_g^i - p_n^i) \qquad (1)$$

$$p_n^{i+1} = v_n^{i+1} + p_n^i \qquad (2)$$

Where ω, c_1 and c_2 are the PSO parameters to be initialized, while r_1 and r_2 are normalized unit random numbers and *i* is the iteration counter. Also $\hat{p}_n$ *pbest* for the n^{th} particle and $\hat{p}_g$ is the *gbest*.

In this implementation, the *gbest* particle is perturbed slightly to ensure that it seeks better solution in its immediate neighborhood using (3) and (4).

$$v_g^{i+1} = \alpha \cdot v_g^i + \beta \cdot r_3^i \qquad (3)$$

$$p_g^{I+1} = v_g^{I+1} + p_g^i \qquad (4)$$

Where r_3 is a uniform random number, α and β are parameters of the PSO.

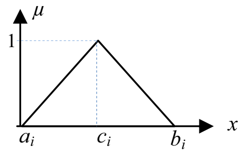


Figure 2. Triangular MF Parameters

VI. Problem Definition

In order to use the PSO to optimize MF, the problem was set up for a triangular MF (The same argument can be extended to other MF types) as follows: Define the span of the MF using a_j, b_j and c_j as in Fig. 2 such that $a_j \geq x_{\min}$, $b_j \leq x_{\max}$. Where the range (universe of discourse) of x is $x_{\max} - x_{\min}$ and c_j is the point of maximum support of the fuzzy set *j*. Such that if $c_j = 0.5(a_j + b_j)$ then c_j is the midpoint.

After encoding into PSO, each particle has $3 \times N$ data elements, where *N* is the number of partitions in the problem space. The task is to improve the performance of the FIS by optimizing a_j, b_j and c_j.

VII. Fitness Measure

The performance of a fuzzy model can be measured by the mean-square-error, *mse*. For two model estimates for example, the one with the smaller *mse* with respect to the experimental model has a better match. The *mse,* which is defined in (5), is used as the fitness measure for the PSO algorithm implementation.

$$mse = \frac{\sum_{k=1}^{q} (y_k - \hat{y}_k)^2}{\sum_{k=1}^{q} y_k^2} \tag{5}$$

Where y_k and $\hat{y}_k$ represents the actual value and the estimated value at data point *k* respectively; *q* is the number of such data points.

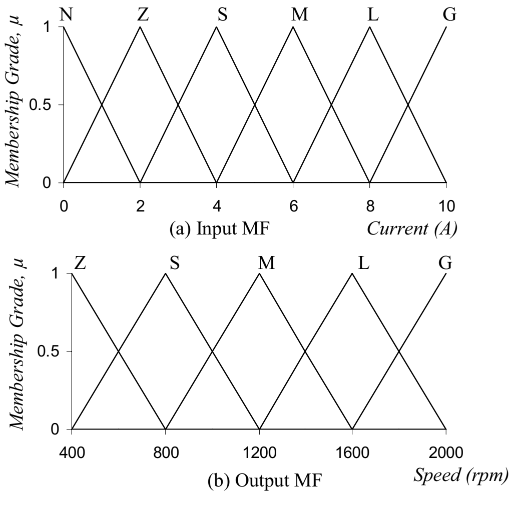


Figure 4. Expert MF for Input and Output

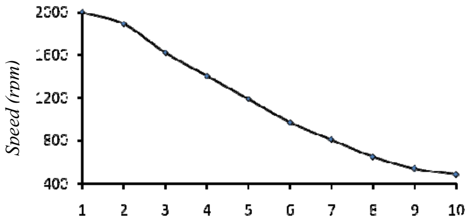


Figure 3. The Experimental Characteristics

TABLE I. Fuzzy Relation Matrix

	Z	**S**	**M**	**L**	**G**
N	0.0	0.0	0.0	0.0	1.0
Z	0.0	0.0	0.0	0.5	0.9
S	0.0	0.0	0.5	0.9	0.3
M	0.5	1.0	0.2	0.0	0.0
L	1.0	0.5	0.0	0.0	0.0
G	1.0	0.2	0.0	0.0	0.0

VIII. Simulation Examples

To illustrate how the PSO may be used in tuning fuzzy membership functions, the new reasoning fuzzy model (NFRM) as designed by [7] for a non-linear system whose characteristics is shown in Fig. 3 was used. The Fuzzy Relation Matrix (FRM) and initial MFs (in the input and output spaces) as designed by [7] are shown in Table I and Fig. 4 respectively.

The task is to optimize the MFs in the input and output spaces such that the resulting fuzzy model is better with the optimized MFs than that resulting from the Expert MF using the same Fuzzy Relation Matrix and the same number of input and output partitions.

IX. Simulation and Results

The PSO parameters were set as in Table II and simulations were carried out as follows:

Case I: Optimize the input MFs only

- (a) With initial expert MF – In this case the MFs of Fig. 6(a) were included as part of the initial particles in the optimization process. This ensures that the performance of the PSO optimized MFs is at least equal to that of the expert designed system.
- (b) Without expert input – In this case, all the initial MFs were randomly defined.

TABLE II. PSO Parameters

Parameter	Value
c_1	0.5
c_2	0.8
ω	0.9
α	0.02
β	0.2
No of Particles	10
iterations	200 - 500
fitness	mse

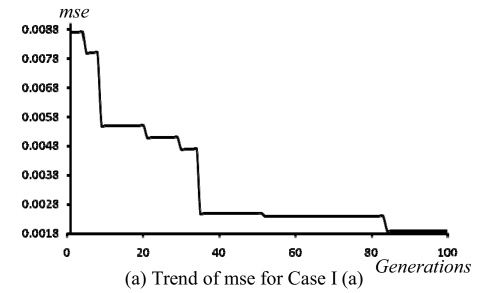
The trend of *mse* over the iteration period for case I(a) is shown in Fig. 5(a), while the fuzzy model for the best initial and final (or optimized) MFs is presented in Fig. 5(b). The best performing initial particle is the expert defined MF included as one of the initial particles. Similar outcome for case I(b) is shown in Fig. 5(c) and 5(d).

Case II: Optimize the output function only

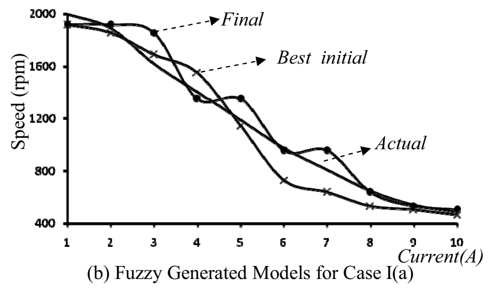
- (a) With an expert input – In this case the output MF of Fig. 6(b) were included as part of the initial particles in the optimization process. Again ensuring that the performance of the PSO optimized MFs is at least equal to that of the expert designed system
- (b) Without an expert input – In this case, all the initial MFs were randomly defined.

Again, for Case II(a) the trend of mse over the iteration period is shown in Fig. 5(e) while the Fuzzy models with the best initial MFs and PSO optimized MFs are shown in Fig 5(f). Similar results for Case II(b) are shown in Figs. 5(g) and 5(h)

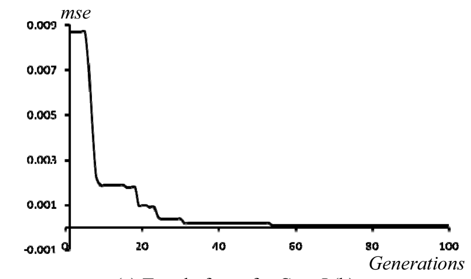
In both cases, an expert input ensured that the *mse* started lower and sped-up the convergence of the learning process as expected. The final MFs generated after the learning process is shown in Fig. 6.



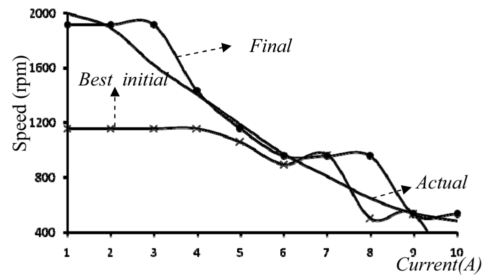
(a) Trend of mse for Case I (a)



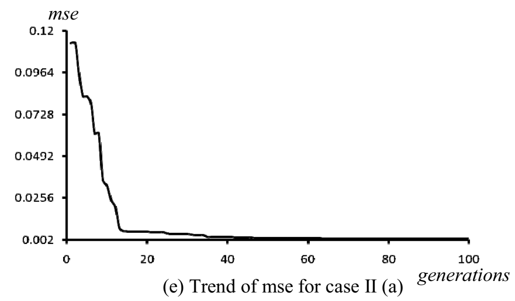
(b) Fuzzy Generated Models for Case I(a)



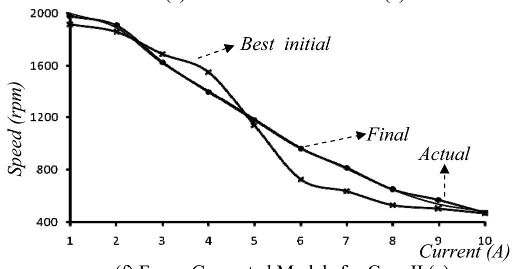
(c) Trend of mse for Case I (b)



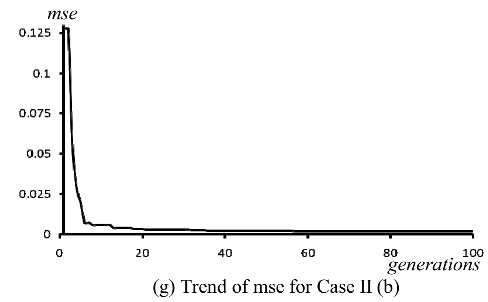
(d) Fuzzy Generated Models for Case I(b)



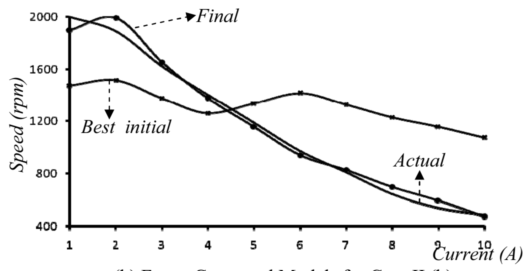
(e) Trend of mse for case II (a)



(f) Fuzzy Generated Models for Case II (a)

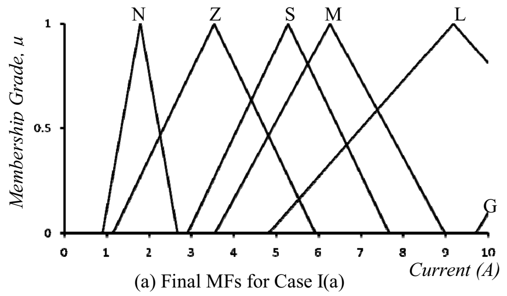


(g) Trend of mse for Case II (b)

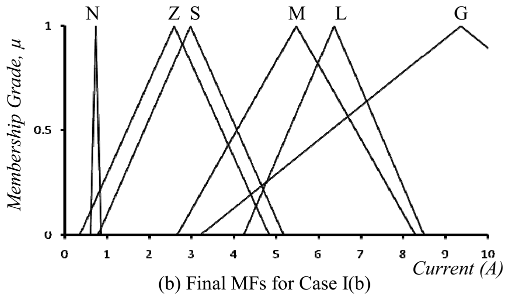


(h) Fuzzy Generated Models for Case II (b)

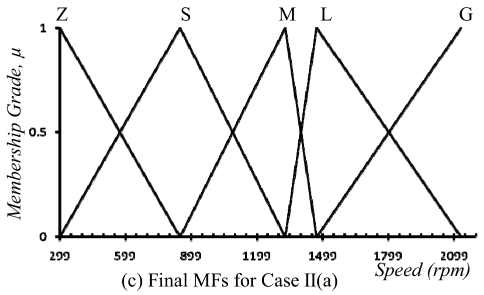
Figure 5: Simulation Results for Cases I and II



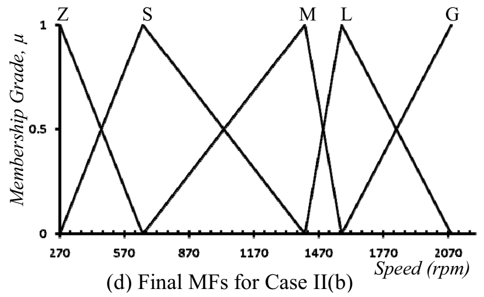
(a) Final MFs for Case I(a)



(b) Final MFs for Case I(b)



(c) Final MFs for Case II(a)



(d) Final MFs for Case II(b)

Figure 6: Simulation Results for Cases I and II

X. Conclusion

It is clear from the results that an expert input helps the PSO search process and reduces the number of iterations required for convergence. Also it is observed that in this case, the output membership functions have more influence on the performance of the fuzzy model, optimizing the output membership functions yielded a result comparable to that of [7] where the GA was used to optimize both the fuzzy relation matrix and input membership functions. The results also show a good promise for pso-fuzzy systems.

References

[1] Babuska R and Verbruggen H. B.: An Overview of Fuzzy modeling for Control. Control Engineering Practice, vol 4 No 11, 1996 pp1593 – 1605.

[2] Lee Chuen Chien: Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I. IEEE Transcations on Systems, Man and Cybernetics. Vol 20 No 2, March/April, 1990.

[3] Sun Hailong and Liu Lixiang: A linear output structure for fuzzy logic controllers. Fuzzy Sets and Systems, No 131, 2002, pp 265 – 270.

[4] www.eng.buffalo.edu/~nagi/papers/fuzzy.pdf accessed January, 2009.

[5] Precup R.-L, Tomescu M. L. and Preid S: Lorenz System Stabilization Using Fuzzy Controllers. International Journal of Computers, Communications and Control, Vol 1 No 3 2007, pp 279 -287.

[6] Tagaki Tomohiro and Michio Sugeno: Fuzzy Identification of Systems and its Application to Modeling and Control. IEEE Transaction on Systems, Man and Cybernetics, vol SMC-15 No 2, February, 1985, pp116 – 132.

[7] Park D., Kendel A. and Langholtz G.: Genetic-Based New Fuzzy Reasoning Models with Application to Fuzzy Control. *IEEE Transactions on Systems, Man and Cybernetics*, vol 29 no 1, January. 1994, pp39 – 46

[8] Elijah E. Omizegba: Rule Optimisation Algorithm for PID type Fuzzy Controller. The Nigerian Journal of Research and Production, Vol. 6 No 1 April, (2005) pp 98 – 108.

[9] Procyk, T. S and Mamdani, E. H.: A Linguistic Self- Organizing Process Controller. Automatica Vol 15, 1979, pp 15 – 30

[10] Watanabe N: Statistical method for estimating Membership Functions. Japanese Journal of Fuzzy Theory and Systems, Vol 5 No 4, 1979.

[11] Tagaki and Yahashi NN-Driven fuzzy Reasoning. International Journal of Approximate Reasoning. Vol 5, 1991, pp 191 – 212.

[12] Meredith D. L., Kar C. L. and Kamur K. K.: The use of Genetic Algorithm in the Design of Fuzzy Logic Controllers. 3rd Workshop on Neural Networks WNN92, 1992, pp 549 – 545.

[13] Paquet U. and Engelbrecht A. P.: Training Support Vector Machines with Particle Swarms. In Proceedings of International Joint Conference on Neural Networks (IJCNN) Conference, 2003, pp 1593 – 1598.

[14] Venu G. G. and Ganesh K. V.: Evolving Digital Circuits Using Particle Swarm. In Proceedings of International Joint Conference on Neural Networks (IJCNN) Conference, 2003, pp 468 – 471.

[15] Montes de Oca and Marco A 'Particle Swarm Optimization 2007' Retrieved 2nd April, 2008 from *http://iridia.ulb.ac.be/~nmontes/CIL/slides.pdf*.

[16] Li Xiaodong: Particle Swarm Optimization: An introduction and its recent developments. *In SEAL 2006*.