Optimization of Neuro-Fuzzy Modeling Using Genetic Algorithm

ATTIA, Abdel-Fattah[1] & HORÁČEK, Petr[2]

[1] Ing., ✉ Department of Control Engineering, FEE, CTU, Technická 2, 166 27 Praha 6

attiaa1@control.felk.cvut.cz

[2] Doc. Ing. CSc., ✉ Center for Applied Cybernetics, FEE, CTU, Technická 2, 166 27 Praha 6

horacek@control.felk.cvut.cz

Abstract: *Many industrial engineering design problems result in complex optimization tasks, which are difficult to solve using conventional optimization techniques. The proposed paper deals with special global optimization method widely applicable in scientific problems. The main point of this research work is to optimize the membership functions parameters of the Fuzzy Logic Neural Network (FLNN) in Genetic Algorithms (GAs) chromosome based on additional information of the domain. This additional information may be indirect definition of the search ranges for every membership shape forming parameter based on 2nd order fuzzy sets. This approach uses Linear Adapted Genetic Algorithm (LAGA) for the optimization of the FLNN parameters. So, the main attribute of the proposed approach is to adapt genetic algorithm using LAGA, and then adopt FLNN model configuration while the optimization process is running using LAGA. In this paper the derivation of 2nd order method is performed for membership function of gaussian shape are assumed for Neuro-fuzzy approach. The explanation of the optimization method is presented in details on two examples.*

Keywords: *Genetic algorithms, Fuzzy Logic Neural Network, and 2nd order fuzzy sets*

1. Introduction

Genetic algorithms (GAs) are search algorithms that simulate the process of natural selection and survival of the fittest. GAs attempt to find a good solution to some problem (e.g., find the maximum of a function) by random generating a collection of potential solutions to the problem and then manipulating those solutions to the problem and then manipulation those solutions using genetic operators. Each solution is assigned a scalar fitness value, which is a numerical assessment of how well it solves the problem. The key idea is to select for reproduction the solutions with higher fitness and apply the genetic operators to generate new solutions. Through crossover and mutation operations new solutions are hopefully generated out of the current set of potential solutions. The process continues until some termination condition is met. Further discussion on GAs can be obtained in (Goldberg 1989), (Winter 1996), and (Gen 2000). When GA is implemented as a learning procedure, the FLNN parameters are coded as a string referred to as a chromosome. Under instances in the population of chromosomes, the genetic operator like crossover probability rate (P_c), and mutation probability rate (P_m) are performed. The fitness is proportional to the whole system error. After a number of generations starting from random or heuristically determined population, the later converges to optimal or close to optimal solution. Being applied to Neuro-Fuzzy systems the initial population of network parameters is formed on the basis of the fuzzy rules obtained from experts. Further genetic learning minimizes the difference

between required and actual network response. The reminder of this paper is organized as follows: Section 2 shows LAGA approach. Section 3 illustrates the structure of Fuzzy Logic Neural Network model, and the derivation of 2nd order method for determining membership functions (*MFs*) parameters boundary search is performed for *MFs* of gaussian shape are assumed in section 4. Section 5 contains application examples. Conclusion is presented in section 6.

2. Linear Adapted Genetic Algorithm (LAGA)

The crossover probability rate P_c, and mutation probability rate P_m are concluded in GA operation to provide faster convergence when compared to constant probability rates. P_c is set up high at the beginning of the generation and decreases linearly during the generations as in (1) (Attia and Horáček 2001). As known from Standared Genetic Algorithm (SGA) at the beginning of generation the randomized initial GA population diverse, it means that promising solutions are scattered through the search space. So, P_c is high in the initial generation, but over the generations these solutions will born even better solutions. It means the population converges to smaller subset of the search space, and P_c value will decrease according to the formula (1), where the large values for P_c (0.5-1.0), and small values of P_m (0.0-0.005) (Goldberg 1989), and (Srinivas 1994).

$$P_c(x) = -x/(2M)+1 \tag{1}$$

where x is the number of generations, and M is the maximum number of generations. As it is known, mutation is not needed at the beginning of generation where the members of the population are very distinct, and the value of P_m is increased linearly also as a function of number of generations to exploit the improved solution in the established region of the current best solution and that is clear in equation (2).

$$P_m(x) = 0.005(M-1)x - 0.005/(M-1) \tag{2}$$

3. Proposed Fuzzy Logic Neural Network (FLNN)

The FLNN model is built using a multilayer fuzzy logic neural network shown in figure 1 proposed by Lin (Lin 1991), and has some modification by (Kolínský 2000), is particular implementation of a fuzzy system equipped with fuzzification and defuzzification interfaces. This network represents linguistic fuzzy system with general rule-base structure. The following example demonstrates this structure:

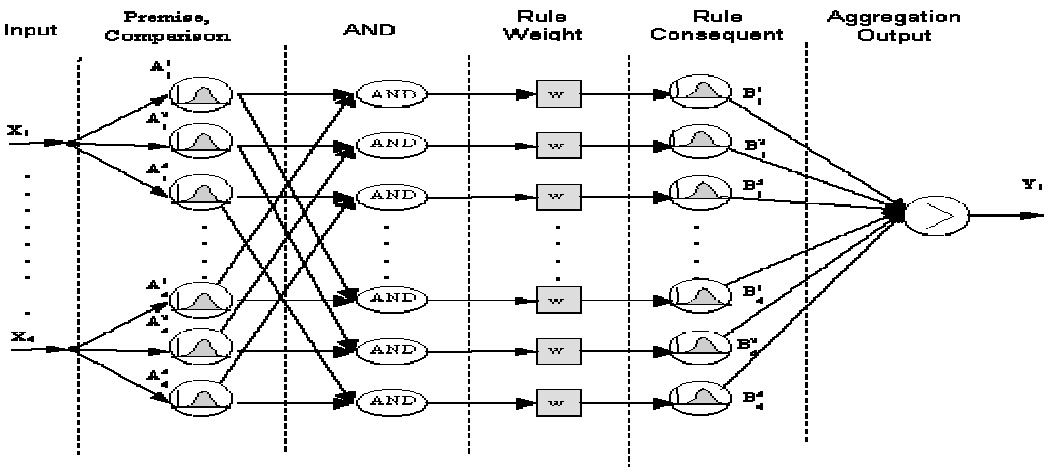


Figure 1 Fuzzy Logic Neural Network Topology.

FLNN consists of several layers, which are described, in the next part (Horáček 1995):

Layer 1: Actual values of the input variables are stored in this layer. Generally fuzzy sets are considered as the input values (crisp numbers are special cases of fuzzy sets). The fuzzy sets are in the parametric form or in the look-up table form.
Layer 2: Rule premises (input reference fuzzy sets) are stored here. Actual input value is compared with the rule premise using degree of overlapping:

$$D_T(X,A) = \sup_x T(X(x), A(x)) = \mathrm{hgt}(X \cap_T A) \tag{3}$$

where T is selected t-norm and *hgt* is height of intersection of X and A with respect to t-norm T. In the special case of crisp input $X=x^*$ the $D_T(X, A)$ is simply.

$$D_T(X,A) = A(x^*) \tag{4}$$

For some parametric fuzzy sets and some t-norms for $D_T(X, A)$ can be derived analytical expression. For the other cases it must be computed numerically.
Layer 3: Every neuron in this layer performs fuzzy conjunction using selected t-norm.

$$y = T(u_1, u_2, ..., u_n) \tag{5}$$

Common parameter of the layer is a type of the t-norm and its parameters.
Layer 4: Every neuron represents a rule weight w. Output of the neuron is a overall degree of rule activation *act* and is computed as follows:

$$y = act = w \cdot u \tag{6}$$

where parameter w has to lie in the interval [0,1].
Layer 5: There are only rule consequents (output reference fuzzy sets) stored in this layer. The fuzzy sets are usually in the parametric form. Input of the neuron is overall degree of rule activation *act*. This value is attached to the reference fuzzy set and together they are fed to the next aggregation layer.
Layer 6: Output of the network is computed here using selected aggregation (inference) algorithm. There is a corresponding fuzzy set B_i with its activation degree act_i in the i-th input of every neuron. When we use Mamdani inference algorithm then the output fuzzy set is computed as follows:

$$Y(y) = \max_{i=1}^{n} T(act_i, B_i(y)) \tag{7}$$

where n is the number of inputs to the neuron.
When we use fuzzy arithmetic based inference algorithm then the output fuzzy set is computed as follows:

$$Y_i = \sum_{i=1}^{n} act_i \cdot B_i \Big/ \sum_{i=1}^{n} act_i \tag{8}$$

Usually only crisp output value y is needed. Then we use some defuzzification method to get the crisp value. The most often used method is a centroid average defuzzification:

$$y = \sum_{i=1}^{n} act_i \cdot y_i \Big/ \sum_{i=1}^{n} act_i \cdot y_i \tag{9}$$

where y_i is a centroid of fuzzy set B_i.
FLNN works in the following manner (Horáček 1995), (Kolínský 2000). In the forward regime as input values (crisp values, fuzzy sets) are first compared with all premises of the rules (input reference fuzzy sets). Outputs of the AND-neuron are then combined with rule-weight (preference between rules) to obtain degree of rule activation. In the last layer these degrees are aggregated with corresponding consequents of the rules (output reference fuzzy sets) according to inference algorithm. Output of the FLNN can be fuzzy set or crisp value (after defuzzification).

Determining MF parameters boundary search.

In case of FLNN these membership functions MF_j (x_i) of input parameters x_i and output parameters are usually approximated by Gaussians. A Gaussian shape is formed by two parameters: mathematical expectation c and standard deviation σ as in formula (10):

$$MF_j(x_i) = G_j(x_i, c_j, \sigma_j) = e^{\left(-\frac{(x_i - c_j)^2}{\sigma_j^2}\right)} \quad (10)$$

To define 2nd order fuzzy set from given MF (x), as it is shown in Figure 2. 2nd order fuzzy set is the area between d^+ and d^-, where d^+, and d^- are the upper and the lower crisp boundaries of the 2nd order fuzzy sets respectively (Ascold 1996). The following expressions to determine the crisp boundaries of it are in (11), and (12):

$$d_j^+(x_i) = min\ (1, MF_j(x_i) + \delta) \quad (11)$$

$$d_j^-(x_i) = max\ (0, MF_j(x_i) - \delta) \quad (12)$$

The formulas (11) and (12) are based on the assumptions that the height of the slice of the 2nd order fuzzy region, bounded by d^+ and d^- and at point x, is equal to the value of 2δ at this point, where $\delta \in [0, 0.3679]$ and these boundaries are equidistant in respect to MF (x). To obtain the ranges for the MFs shape forming parameters, it should be assumed, that these 2nd order fuzzy sets are MF search spaces. Therefore, all MFs with acceptable parameters should be inside of these areas. In general case the intervals of acceptable values for every MFs shape forming parameter (e.g. $c = [c_1, c_2]$, and $\Delta\sigma = [\sigma_1, \sigma_2]$ for Gaussians) may be determined by the means of solution of formulas (10), (11), and (12). Practically, it may be done more roughly, covering some larger search space. For instance, c_1 and c_2 for Gaussians may be found as the maximal and the minimal roots of the equation $d^+ = 1$, that may be easily calculated. This equation is based on the assumption that a fuzzy notion represented by the Gaussian must have a point where it is absolutely true. The σ_1, and σ_2 can be easily found from the following four equations:

$$G(c \pm \sigma, c, \sigma) + \delta = G(c \pm \sigma, c, \sigma_2); \text{ and } \quad G(c \pm \sigma, c, \sigma) - \delta = G(c \pm \sigma, c, \sigma_1) \quad (13)$$

where we choose minimal σ_1 and maximal σ_2 from the roots. These equations are based on the assumption that the acceptable Gaussians with $[\sigma_1, \sigma_2]$ should cross the 2nd order fuzzy region slices at points $x = c \pm \sigma$.

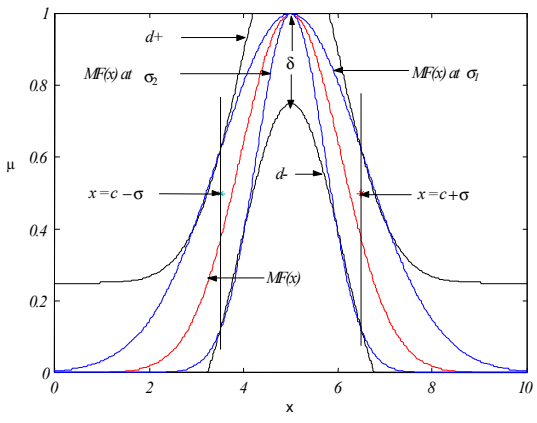


Figure 2 - The upper and lower boundaries of the 2nd order fuzzy set

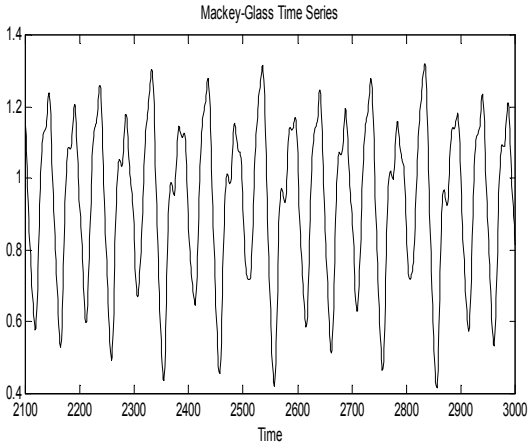


Figure 3 - Mackey-Glass time series

4. Applications

- **The First Numerical Example**

This section discusses how the proposed FLNN is formulated using LAGA approach where all the parameters of the FLNN are initially randomized, and optimized using LAGA. After getting the intervals of acceptable values for every *MFs* shape forming parameter ($c = [c_1, c_2]$, and $\Delta\sigma = [\sigma_1, \sigma_2]$ for Gaussians) using 2^{nd} order fuzzy sets for all membership functions. Let us explain the method applying on a chaotic system example to be clear. The system used in this example is defined by chaotic Mackey-Glass differential delay equation (Jang 1997):

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \qquad (14)$$

The prediction of future values of this time series is a benchmark problem which has been considered by a number of connectionist researchers. Fourth-order Runge-Kutta method was used to find the numerical solution of the equation. The time step used in the method is 0.1, initial condition $x(0) = 1.2$, $\tau = 17$.

- **Coding of FLNN parameters:**

Figure 1 shows four inputs (x_1, x_2, x_3, x_4) and one output (Y_1). Each of the input fuzzy variables is quantified into three Gaussian membership functions, and every membership function has two parameters: mathematical expectation c and standard deviation σ, therefore resulting 24 parameters at input. Using Wang technique for generating rules from the given data (Wang 1992), we have 25 rules, it means we have 25 weights, and 25 centeroids represented by singletons. Thus a total of 74 parameters ($3_{membership\_functions}$ x $2_{parameres}$ x $4_{variables}$ + $25_{wightes}$ + $25_{centroids}$) are needed to be optimized using LAGA. The coded parameters of FLNN are arranged as shown in the following table (1) to form the chromosome of the population.

Table 1 The coded parameters of FLNN.

Chromosome	Sub-chromosome of inputs	Sub-chromosome of rule weights	Sub-chromosome of rule consequents
	x_1, x_2, x_3 , x_4	$w_1, w_2, \ldots\ldots, w_{25}$	$b_1, b_2, \ldots\ldots\ldots, b_{25}$
Parameters 74	c_1, σ_1, c_2, σ_2, c_3, σ_3, c_4, σ_4 3*2*4	$w_1, w_2, \ldots\ldots\ldots, w_{25}$ 25	$b_1, b_2, \ldots\ldots\ldots\ldots, b_{25}$ 25
Gene	1000000000…0110100011 c_1, ……. , σ_4	0100111111… w_1, ……., w_{25}	0111001010 … b_1 … ……, b_{25}

- **Optimization by LAGA**

To describe the LAGA optimization process, consider the block diagram shown in figure 4. At the beginning of the process, the initial population comprises a set of chromosomes. Every chromosome has 74 genes, and every gene has 10 bits. So the chromosome length is 740 bits. The population consists of 200 chromosomes, which are all randomized initially.
After each of the chromosome evaluated and associated with a fitness, the current population undergoes the reproduction process to create the next generation of population, the "roulette wheel' selection scheme is used to determine the member of the new generation population. Then the mating pool is formed, and crossover are applied, and followed by mutation operation due to LAGA approach. Finally, after these three operations, the overall fitness of the population is improved. The procedure is repeated until the termination condition is

reached. The termination condition is a maximum allowable generations or a certain value of MSE required to be reached. The choice of the fitness function is the normalized error function, which is known by root mean square error (RMSE) for the whole model of FLNN as:

$$RMSE = \sqrt{MSE/var(t)}\,, \tag{15}$$

$$MSE = \frac{1}{N*ny}\sum_{j=1}^{N}\sum_{i=1}^{ny}\left(y_i(x_j) - t_{ij}\right)^2\,, \tag{16}$$

$$var(t) = \frac{1}{N*ny}\sum_{j=1}^{N}\sum_{i}^{ny}\left(t_{ij} - \bar{t}_i\right)^2 \tag{17}$$

where, MSE is the mean square error, var (t) is the variance of targets, ny is the number of outputs, N is the number of patterns (size of training set), y_i (x_j) is the i^{th} output of fuzzy system for input vector x_j, x is the input variables, t is the estimated value or (the matrix of targets of size (ny*N)), and $\bar{t}$ is the mean of target t.

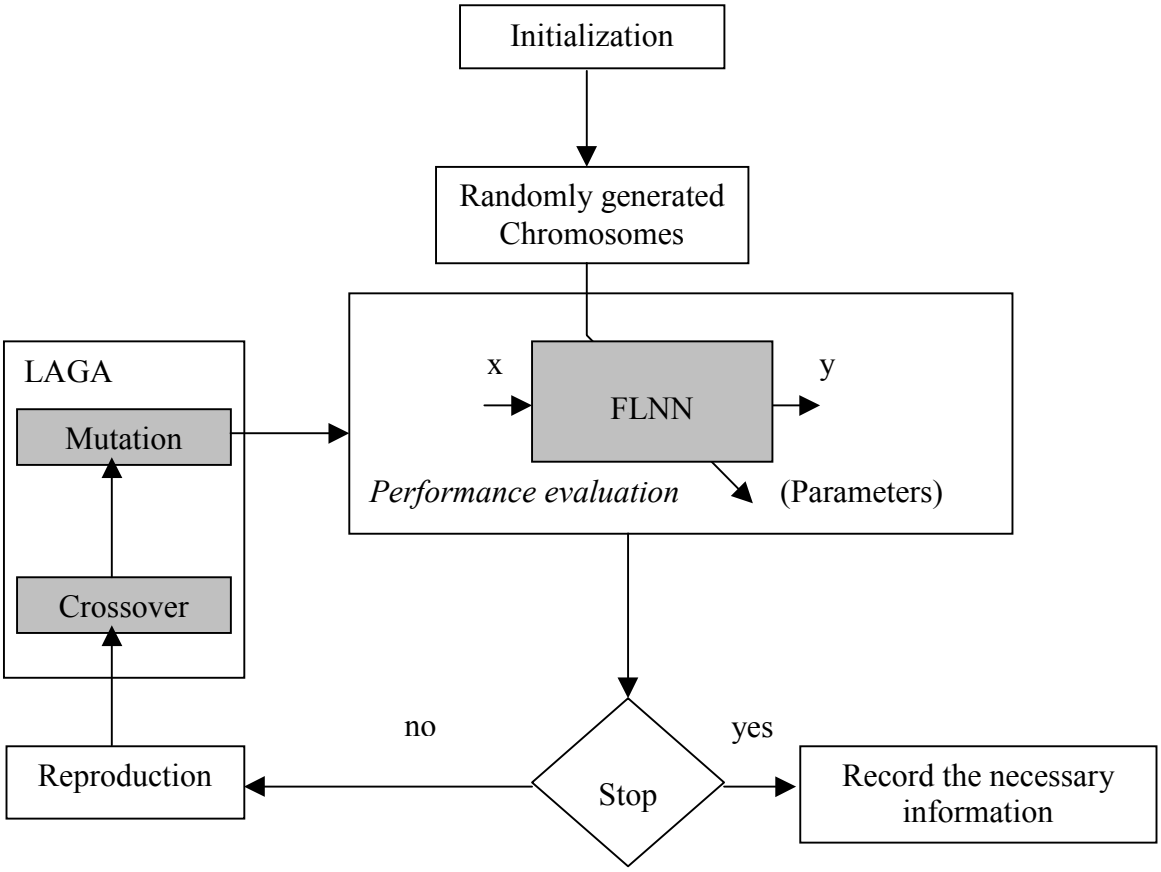
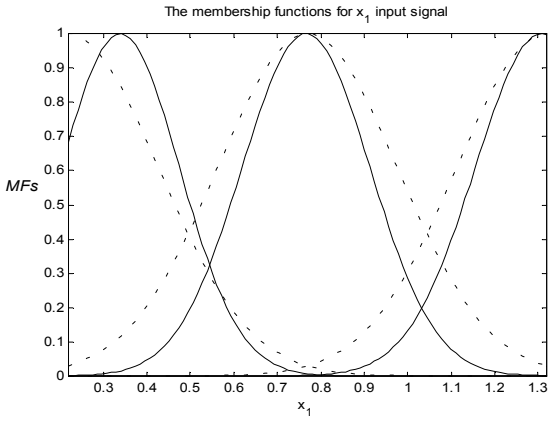


Figure 4 - Block diagram for LAGA optimization process

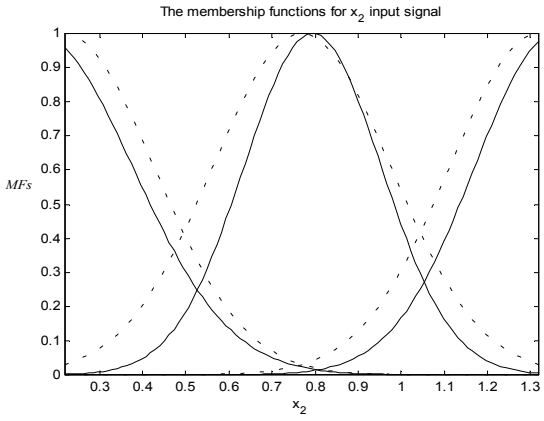
5. Simulation Results

From the Mackey- Glass time series x (t), we extracted 3000 input output pairs. The first 1000 data points are used to build the fuzzy model, while the remaining 2000 data points have been used to identify a FLNN model. The number of MFs assigned to each input of the FLNN was set three. Figure 5 depicts the membership functions for each input variable before and after training using LAGA. After 420 generations, and 90 min computation time on MATLAB compiler, and PC 400MHz with memory 64 MB. It is seen from figure 6, the FLNN model

has a good match with the actual model with a MSE “0.0012”, and RMSE is “0.14” as shown in figure 7.

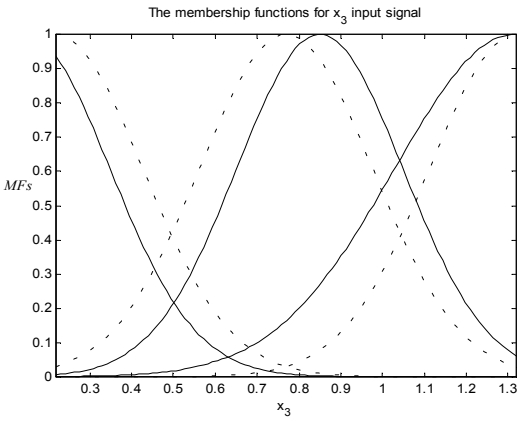


(a)

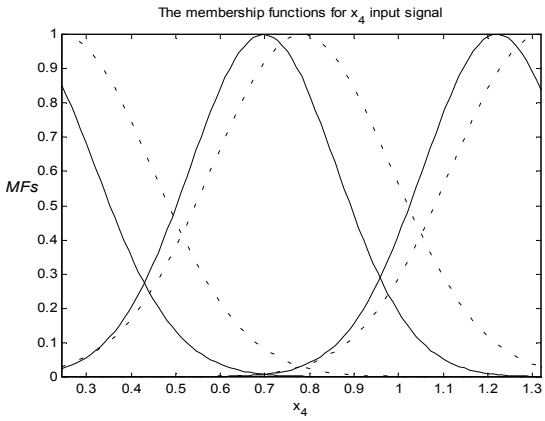


(b)

Figure 5 (a, b) - Membership functions in chaotic system time series prediction:
Dashed line: The normalized MFs before learning.
Solid line: The optimized MFs after using LAGA



(c)



(d)

Figure 5 (c, d) - Membership functions in chaotic system time series prediction:
Dashed line: The normalized MFs before learning.
Solid line: The optimized MFs after using LAGA

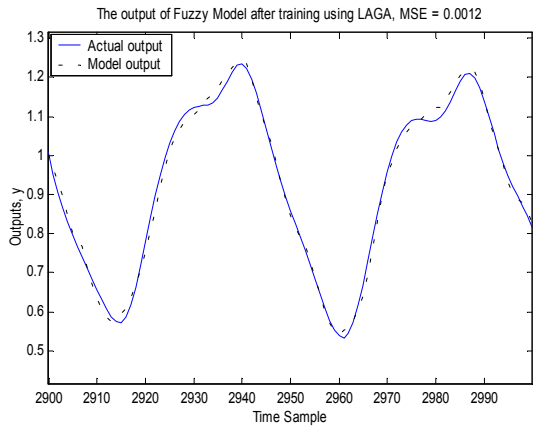


Figure 6 - Testing of the fuzzy model versus the actual model after training.

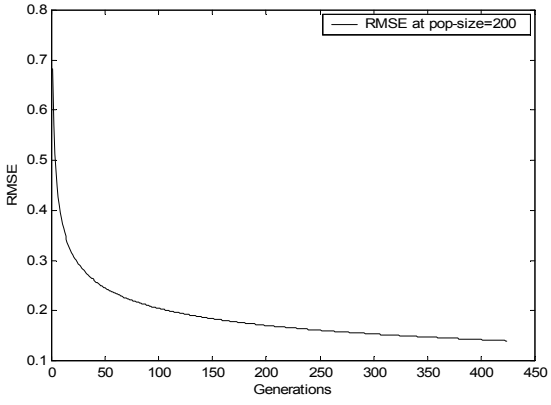


Figure 7- LAGA convergence rate at population size 200.

- **The Second Numerical Example**

This example is taken from (Narendra 1990)], (Wang 1994) in which plant to be identified is governed by the difference equation (18):

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)] \tag{18}$$

where the unknown function has the form $g(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u)$.
The plant is modeled using FLNN described in section 3. The model has three input variables u (k), y(k), and y(k-1) and a single output y(k+1). Each of the input fuzzy variables is quantified into five Gaussian membership functions, and every membership function has two parameters: mathematical expectation c and standard deviation σ, therefore resulting 30 parameters at input of FLNN model. Using Wang technique for generating rules from the given data [Wang 1992], we have 20 rules, it means we have 20 weights, and 20 centeroids represented by singletons. Thus a total of 70 parameters ($5_{\text{membership\_functions}}$ x $2_{\text{parameres}}$ x $3_{\text{variables}}$ + 20_{wightes} + $20_{\text{centroids}}$) are needed to be optimized using LAGA.

The learning procedure of LAGA is applied also as mentioned in the first numerical example. And also figure 4 shows the block diagram for LAGA optimization process for optimizing the FLNN model parameters of second numerical example. The first 250 data points are used to build the fuzzy model at $u(k) = \sin(2\pi k / 250)$, while the remaining 450 data points have been used to identify a FLNN model. Figure 8 shows the FLNN model has a good match with the actual model with a MSE "0.0473", and RMSE is "0.0607 " as shown in figure 9. Figure 10 depicts the membership functions for each input variable before and after training using LAGA. Figure 11 shows the outputs model and the plant for the input $u(k) = 0.5\sin(2\pi k / 250)$ for $1 \le k \le 250$ and $501 \le k \le 700$ and $u(k) = 0.5\sin(2\pi k / 250) + 0.5\sin(2\pi k / 25)$ for $251 \le k \le 500$. We see from figure 11 that the trained identification FLNN model approximates the plant quit well.

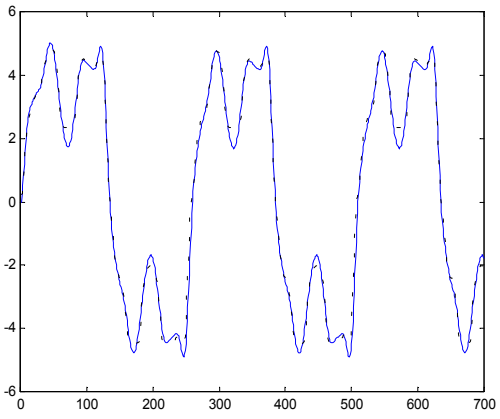


Figure 8 - Outputs of the plant (solid line), and FLNN model (dashed line) for $u(k) = \sin(2\pi k / 250)$.

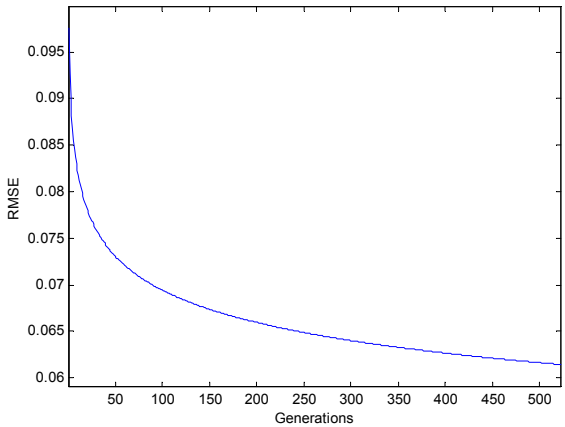
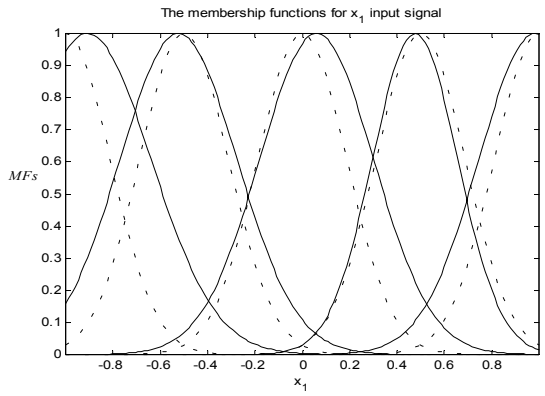
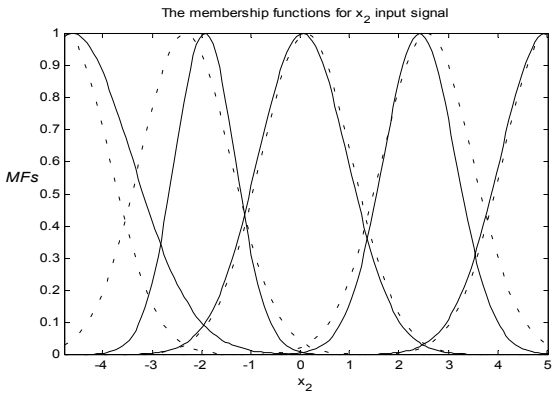


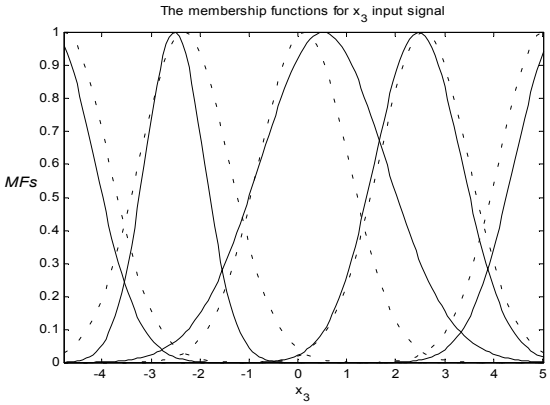
Figure 9 - LAGA convergence rate at population size 200.



(a)



(b)



(c)

Figure 10 - (a, b, and c) are input MFs:
Dashed line: The normalized MFs before learning.
Solid line: The optimized MFs after using LAGA

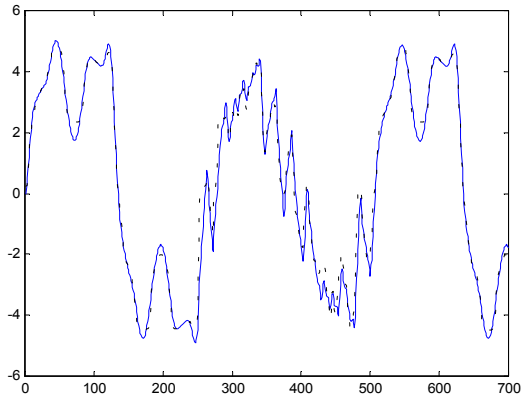


Figure 11 - Outputs of the plant (solid line), and FLNN model (dashed line) for the input $u(k) = \sin(2\pi k / 250)$ for and $1 \leq k \leq 250$ and $501 \leq k \leq 700$,and $u(k) = 0.5\sin(2\pi k / 250) + 0.5\sin(2\pi k / 25)$ for $251 \leq k \leq 500$.

6. Conclusion

In this paper, we have introduced an approach for modifying crossover and mutation probability rates based on generation index. The crossover probability rate decrease, and mutation probability rate increases linearly with the generation index. This paper describes a method for determining boundary in the parameters search space of membership functions depending on 2nd order fuzzy sets. It gives more information about optimization of fuzzy and neuro-fuzzy systems using GAs. The simulation results of the application examples indicate the effectiveness of the proposed approach of LAGA to be promising learning algorithm.

7. Acknowledgement

This work was partially supported by the Ministry of Education of the Czech Republic under Project LN00B096.

8. References

ASCOLD N MELIKHOV, V. MIAGKIKH, and ALEXANDER T. (1996). *Optimization of Fuzzy and Neuro-fuzzy systems by means of Adaptive Genetic search.* Available on URL: (http://web.cps.msu.edu/~miagkikh/web).

ATTIA A., and HORÁČEK P. (2001). *Modification of Genetic Algorithms For Optimization Problem Solving*. Workshop 2001, Architecture Faculty, ČVUT, Feb. 5 – 7, 2001.

GOLDBERG, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine learning*. Addison -Wesley.

GEN, M. AND R. CHENG (2000). *Genetic Algorithms and Engineering Optimizations.* John Wiley & Sons.

HORÁČEK, P. (1995) *Fuzzy Modeling and Control, In: H. Adelsberger*, J. Lažanasky, V. Mařík (eds.): *Information Management in computer Integrated Manufacturing*. Lecture Notes in Computer Science No. 973, Springer-Verlag Berlin, pp. 257-288.

FARAG W., VICTOR H. (1998). *A Genetic-Based Neuro-Fuzzy Approach for modeling and Control of Dynamical Systems.* IEEE Trans. On Neural Networks. Vol. 9, No. 5. September 1998.

JANG, S. R., SUN C. T., MIZUTANI E. (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence.* Prentice Hall, Inc.

KOLÍNSKÝ, J. (2000) *Identifikace parametrů fuzzy-logických neuronových sítí.* Diplomová práce, Katedra řídicí techniky**,** Fakulta elektrotechnická**,** ČVUT. 2000.

LIN, C. T., LEE, C.S.G. (1991). *Neural –Network- Based Fuzzy Logic Control And Design Decision System.* IEEE Trans. on Computers, vol. 40, no. 12, 1991, pp 1320-1336.

NARENDRA K. S, K. PARTHASARATHY (1990). *Identification and control of dynamical systems using neural networks.* IEEE Trans. Neural Networks*,* Vol. 1, March 1990.

SRINIVAS M., AND L. M. PATNAIK (1994). *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms.* IEEE Trans. System. Man. and Cybernetics. Vol. 24, No. 4, April 1994.

SRINIVAS M., AND L. M. PATNAIK (1996). *Genetic Search: Analysis Using Fittness Moments*. IEEE Trans. on Knowledge and Data Engineering. Vol. 8, No. 1, February 1996.

WANG LI-XIN (1994). *Adaptive Fuzzy Systems and Control, design and stability analysis*. 1994 by PTR Prentice Hall, ISBN 0-13-099631-9.

WANG LI-XIN, JERRY MENDEL (1992). *Generating Fuzzy Rules by learning from Examples.* IEEE Trans. Systems, Man, and Cybernetics, Vol. 22, No. 6, November/December 1992

WINTER G., J. PÉRIANX, MGALÁN, P. Cuesta (1996). *Genetic Algorithms in Engineering and Computer Science.* ISBN 0471 95859 X, John Wiley & Sons, 1996.

WRIGHT A. (1990). *Genetic algorithms for real parameter optimization.* In the First workshop on the foundations of Genetic Algorithms and Classifier Systems. Pages 205-218, Indiana University, Bloomington, July 1990.