IJCTA, 9(2-A), 2016, pp. 691-701
© International Science Press

Optimization of TSK Fuzzy Model Using New Improved PSO

Lamine Brikh[1], Ouahib Guenounou[1], Fatah Yahiaoui[1], Mohand-akli Kacimi[1] and Ahmed Ouaret[1]

ABSTRACT

In this paper, we propose an optimization approach of Takagi-Sugeno-Kang fuzzy system using new improved version of Particle Swarm Optimization. The novel version is based on two decoupled adaptive parameters including in the velocity update function. A neuro-fuzzy structure is created in the form of a network which contains five layers to exploit a various parameters of membership functions and rules conclusion. The new improved PSO have been used in order to assess these parameters and choose the best of all the swarm particles (best model). Two systems nonlinear are used to show the accuracy of the obtained model approach and compare it with other version of PSO.

Keywords: Optimization, fuzzy system, neuro-fuzzy structure, membership functions, particle swarm optimization, Takagi-Sugeno-Kang fuzzy system, etc.

1. INTRODUCTION

Inability of the mathematical tools to solve the problems of modeling of nonlinear systems, that the majority of classical optimization methods is to reduce the order of the problem where the reduce order may be unstable even though the original high order system is stable [1], appeals to intelligent methods. Thanks to that, it can always exploit all available information, where embed the intuition and experience of a human operator. Fuzzy modeling is a promising approach to model complex nonlinear systems [2-3] and it is similar to neural network [4], because they can approximate any continuous function [5-6]. Among the fuzzy modeling techniques that exists is that of Takagi-Sugeno-Kang, the model of TSK fuzzy system [7] is one of the most used models; thanks to these good results in various fields of application. This model is based on the IF-THEN rules with fuzzy antecedents and a mathematical function to the model output:

$$R^i : \; if \; x_1 \; is \; A_1^i \; and \; \ldots \; and \; x_j \; is \; A_j^i \; then \; y^i = a_0^i + a_1^i x_1^i + a_2^i x_2^i + \ldots + a_r^i x_r^i . for \; i = 1 : k \tag{1}$$

where k is the number of rules; $x_j (j = 1 : r)$ is the j^{th} input; y^i: is the output of the fuzzy rule R^i; A_1^i, A_2^i, ..., A_r^i: are fuzzy sets that are characterized by membership functions $A_j^i(x_j)$ and a_j^i are a real parameters. The global output of the model is computed as a follows:

$$y = \frac{\sum_{i=1}^{k} w_i y^i}{\sum_{i=1}^{k} w_i} \tag{2}$$

where w_i is the firing strength of rule R^i defined by

$$w_i = A_1^i(x_1) * A_1^i(x_1) * \ldots * A_1^i(x_1) \tag{3}$$

* Laboratoire de Technologie Industrielle et de l'Information, Faculté de Technologie Université de Bejaia, 06000 Bejaia, Algeria

From Eq. (1), it is clear that the TSK model tries to decompose the input space into fuzzy region, and then approximates the system in every region by simple linear equation. The main advantage of this model is its representative power and its ability in describing a highly non-linear system using input/output data. However, the TSK model includes too many parameters that most of them are located in the consequent rules and may lead to a nonlinear programming problem [8].

The optimization of this type of model requires more adaptive optimization methods [9], where we using a Particle Swarm Optimization that it do not require mathematical description of the optimization problem and are capable to locate the global optimum in a difficult environment. The flexibility of PSO able to adjust all parameters of fuzzy rules [10], and the PSO can improve the convergence and diversity using a fuzzy logic [11]. A new version of the improved PSO with two adaptive parameters (where the first parameter is used to control the speed of convergence and the other is to reduce the possibility of stocking in a local minimum) is used to optimize the parameters of membership function and fuzzy rules. Some recent control methods are described in [19-25].

This paper is organized as follows: in Section 2, we illustrate the structure of the fuzzy model. In Section 3, a brief overview on some versions of the Particle Swarm Optimization cited in literature is given. Section 4 is a description of the learning algorithm. In Section 5, we present the simulation results of the new improved version of PSO and compared with other versions using two classicalidentification problem, where is divided in stage of training data and testing stage. In the last section, we conclude the work.

2. STRUCTURE OF THE FUZZY MODEL

The model structure is presented as a multi-layer neural network [8] in order to facilitate the calculation of various parameters and the error between the output and the desired model.

In this section, we given a description of the TSK fuzzy model, the structure is divided as five layers.

First layer:

No function is realized with the neurons in this layer. Each neuron transmits the input signal to the second layer.

$$Q_i^1 = u_i^1, u_i^1 = x_i \tag{4}$$

Second layer:

The neurons of this layer correspond to the membership functions (fuzzy sets). The output of each neuron provides the degree of membership of an input variable to the fuzzy set corresponds to that neuron. It is given by:

$$Q_i^2 = \exp\left(\frac{u_i^2 - c_i^2}{\sigma_i^2}\right) for\ i = 1, 2, ..., n_1, n_2, n_3 \tag{5}$$

$$u_i^2 = \begin{cases} Q_1^1\ for\ i = 1,\ 2,\ ...,\ n_1 \\ Q_2^1\ for\ i = 1+n_1,\ 2+n_1,\ ...,\ n_1+n_2 \\ Q_3^1\ for\ i = 1+n_1+n_2,\ 2+n_1+n_2,\ ...,\ n_1+n_2+n_3 \end{cases} \tag{6}$$

where $n_1 = n_2 = \cdots = n_r$ represents the number of fuzzy sets associated with the model variables; c_i^2 and σ_i^2 are the parameters of the Gaussian function of the i^{th} neuron (the settings to adjust).

Third layer:

Each neuron of this layer corresponds to a fuzzy rule in the rule base unit. Its inputs come from all nodes in layer 2 which participate in the premises part of that rule. The output of each node represents the firing strength of rule and is calculated via the product operation:

$$Q_i^3 = \prod_i u_i^3, u_i^3 = Q_i^2,\ i = 1,\ 2,\ ...,\ k \tag{7}$$

where k is the number of fuzzy rules, equal to $n_1 * n_2 * ... * n_r$. The link weights in this layer are also set to unity.

Fourth layer:

Each node in this layer performs a linear summation of the input variables as shown by the following equation:

$$Q_i^4 = a_0^i + \sum_{j=1}^{r} a_j^i x_j \tag{8}$$

in which a_j^i (j = 1, 2, ..., r and i = 1, 2, ..., k) are the parameters to be adjusted. Links from this layer to output layer are equal equation to unity.

Fifth layer:

The node of this layer is the overall output of the model. The following equation is used for computing the output:

$$Q_i^5 = \frac{\sum_{j=1}^{k} Q_i^3 * Q_i^4}{\sum_{j=1}^{k} Q_i^3} \tag{9}$$

Several methods inspired by nature had been proposed for the optimization and handling a neural network, fuzzy systems, such as, genetic algorithms [12] and the particle swarm optimization [13].

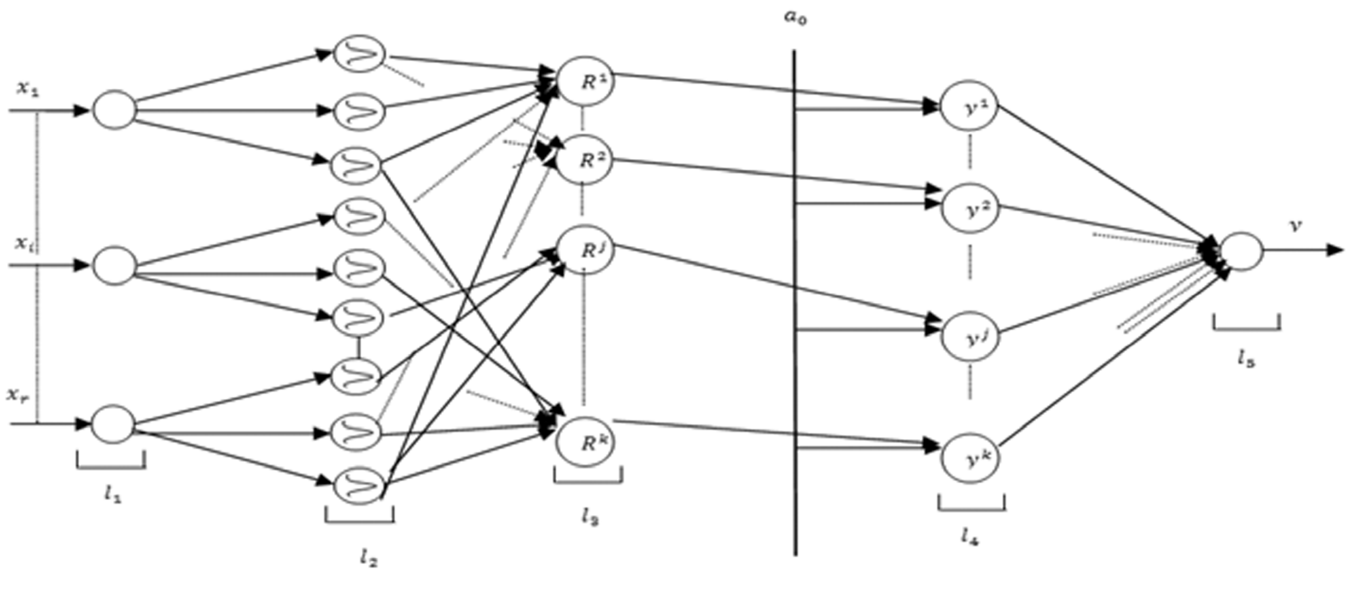

Figure 1: Structure of the fuzzy model

3. PARTICLE SWARM OPTIMIZATION

During the past decade, Particle swarm optimization PSO [11] had been used to improve the performance and minimize models of many systems nonlinear [14-16], because it can find the global optimum and explore search spaces in maximum.

Typically, all the population's members survive from the beginning of a trial until the end. Their interactions result in iterative improvement of the quality of problem solutions over time. The particle swarm is a population-based stochastic.

As its name suggests, the PSO is an optimization algorithm that is based on a set of particle (model). After the random distribution of particles, each one is moving towards a solution in the search space (local optimum $local_{par}$), the entire swarm goes ahead towards the best solution of all (the global optimum $global_{par}$). Each particle's position is evaluated as a possible solution. In a swarm with N particles, the position of the i^{th} particle is defined as a vector as it is given in Eq. (9).

$$Par_i = [c_1, c_2, ..., c_n, \sigma_1, \sigma_2, ..., \sigma_n, a_0, a_1, a_2, ..., a_r] \tag{10}$$

where $c_1, c_2, ..., c_n, \sigma_1, \sigma_2, ..., \sigma_n$ are the centers and the variances of membership functions respectively (settings antecedents) and $a_0, a_1, a_2, ..., a_r$ the conclusions settings. n and r are the number of membership function and the number of input respectively.

3.1. Standard PSO

A standard PSO was proposed by Eberhart [13], defined by the equations (11) and (12) in order to exploit these advantages over other methods (it requires few parameters, it converges faster), in every iteration the velocity and the location of particles are updated.

$$vel_i(k+1) = C\Big(vel_i(k) + C_1 * r_1\big(local_{par} - Par_i(k)\big) + C_2 * r_2\big(global_{par} - Par_i(k)\big)\Big) \tag{11}$$

where C is constriction factor, C_1 and C_2 are the acceleration coefficient; r_1 and r_2 are random numbers uniformly distributed in [0 1].

$$Par_i(k+1) = Par_i(k) + vel_i(k+1) \tag{12}$$

4. IMPROVED PSO ALGORITHM

After, several researchers have proposed improved versions to make it more efficient, M. A. Cavuslu, C. Karakuza and F. Karakaya [14] proposed a version improved that adding an external term to the updated function equation's velocity (13) reduces the possibility of stocking in local minimum.

$$vel_i(k+1) = C\Big(vel_i(k) + C_1 * r_1\big(local_{par} - Par_i(k)\big) + C_2 * r_2\big(global_{par} - Par_i(k)\big)\Big) + C_3 * \lambda(k) \tag{13}$$

where λ is called additive learning constant, C_3 is a normally distributed random number vector.

Another version was improved by D. P. Kanungo, B. Naik, J. Nayak, S. Badoo and H. S. Behera [17], in PSO, the inertia weight used to balance the global and local search ability. A large inertia weight facilitates a global search, while a small inertia weight facilitates a local search. By changing the inertia weight dynamically, the search ability is dynamically adjusted to introduce new parameter called inertia weight adaptive (14) to control both the local and global search behavior.

$$vel_i(k+1) = C\Big(W * vel_i(k) + C_1 * r_1\big(local_{par} - Par_i(k)\big) + C_2 * r_2\big(global_{par} - Par_i(k)\big)\Big) \tag{14}$$

5. NEW IMPROVED PSO ALGORITHM

The version of this paper is a combination of [10-16], in order to use the advantages of each one, it exploits the parameter of the adaptive inertia to control the speed of convergence and the external term at the same time to reduce the possibility of stocking in a local minimum in the velocity update function given by Eq. (15)

$$vel_i(k+1) = C\Big(W * vel_i(k) + C_1 * r_1\big(local_{par} - Par_i(k)\big) + C_2 * r_2\big(global_{par} - Par_i(k)\big)\Big) + C_3 * \lambda(k) \quad (15)$$

6. LEARNING ALGORITHM

In this section, the toolbox GetFis in MATLAB is used to create the structure of neuro-fuzzy to build the particle parameters “(7)” in the first stage, as a matrix of N columns (number of individuals) and $r * n + (r + 1) * k$ rows (the individual length) is initialized randomly.

The main square error criterion from the output signal in the last layer is used in order to calculate the fitness given by the equation:

$$E = \frac{\sum_j^M \big(yd_j(k+1) - yr_j(k+1)\big)^2}{M} \quad (16)$$

where j represents a simple index, $yd(k + 1)$ is the desired output and $yd(k + 1)$ is the real output of the fuzzy model TSK. This error will be useful to perform the model.

After the adjustment of parameters, these particles are evaluated by the fitness function (16) which is the adaptation criterion, we compute the new position of each particle by (12). PSO uses the operator *min* to extract the minimum value in the vector cost E and calculate its average. At first, we consider all the particles as local particles and their cost as local minima. Retrieving the minimum value of the cost vector is used to locate the best particle in the matrix Par_i (the global minimum).

```
Initialization
$C_1 = C_2; C_3; C; r_1; r_2$
$\lambda << 1 / ( max( eig( X*X' ) )$
min_cost = min (min (E)); mean_cost = mean (E)
$global_{min}$ = min_cost; $local_{par} = Par_i$; $local_{par} = E$
K = 0
while k<maxit
    k = k + 1
    w = ( maxit-k)/maxit $\rightarrow$"15" "12" "16"
    $local_{cost} = local_{cost}$*not (better$_{cost}$) + E*better$_{cost}$
    [temp , t] = min( $local_{cost}$)
    if temp<$global_{cost}$
        $global_{par} < Par_i(t , :)$
        index = t
        $global_{cost}$ = temp
    end
    [k $global_{par}$ $global_{cost}$]
    min_cost(k+1) = min(E)
    $global_{min}(k+1) = global_{cost}$
end
```

7. SIMULATION AND DISCUSSION

In this section, we apply the new improved PSO to shown the performance of the proposed approach, a two classical identification problem given by Narendra and Parthasarathy [16] and [12] were applied to evaluate the velocity update function for each version.

In this case, the two plant are described by the difference equation

$$y(k) = \frac{y(k-1)\,y(k-2)\big(y(k-1)+2.5\big)}{1+y^2(k-1)+y^2(k-2)} + u(k-1) \tag{16}$$

$$y(k) = \frac{24+y(k-1)}{30}\,y(k-1) - 0.8\frac{u^2(k-1)}{1+u^2(k-1)}\,y(k-2) + 0.5u(k-1) \tag{17}$$

These plants have three inputs *u*(*k-1*), *y*(*k-1*) and *y*(*k-2*) and single output *y*(*k*). Each version was implemented in Matlab with the same parameters presented in table II:

Table 1
Parameters of simulation

Parameters		*Values*
Maxitr		1500
MF		3
Popsize		100
C_1	2.1	
C_2	2.1	
C_3	0.1	
C		0.76

This section is divided into two stages as a follow:

7.1. Stage of training data

The input data training of 300 points is generated by a sequence of pulses of random amplitude in range [-1 1] and a random period between [1 4] for the first plant. However, the input data training of 200 points

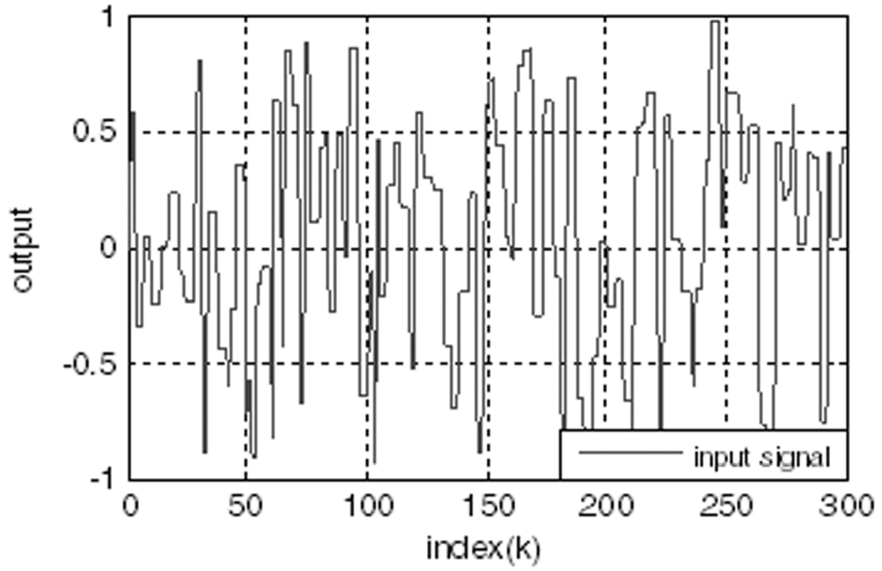

Figure 2: The input signal of the first plant.

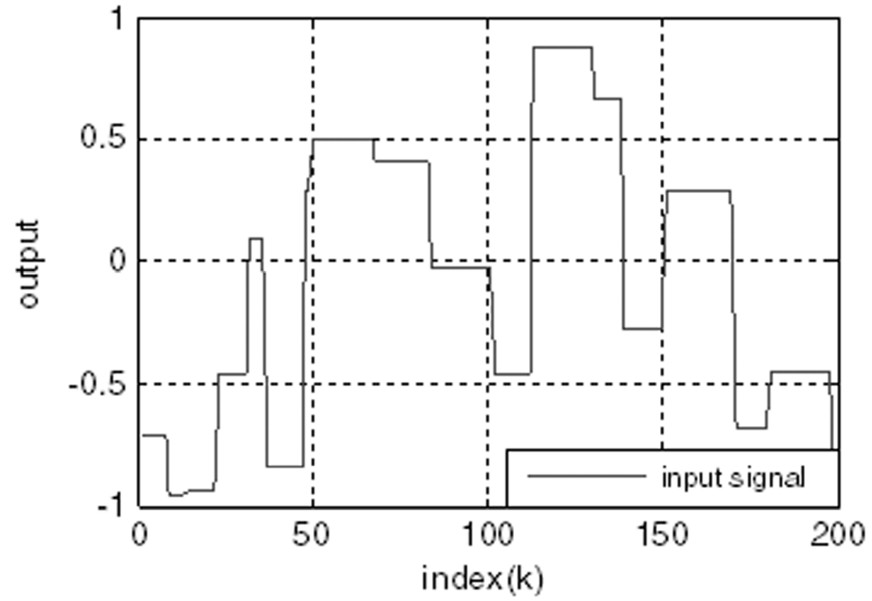

Figure 3: The input signal of the second plant.

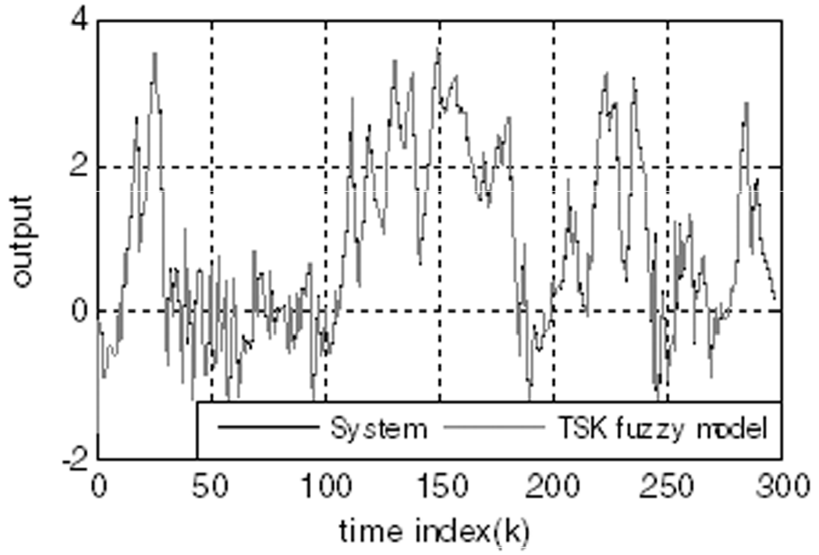

Figure 4: The output signal of the system and fuzzy model of the first plant

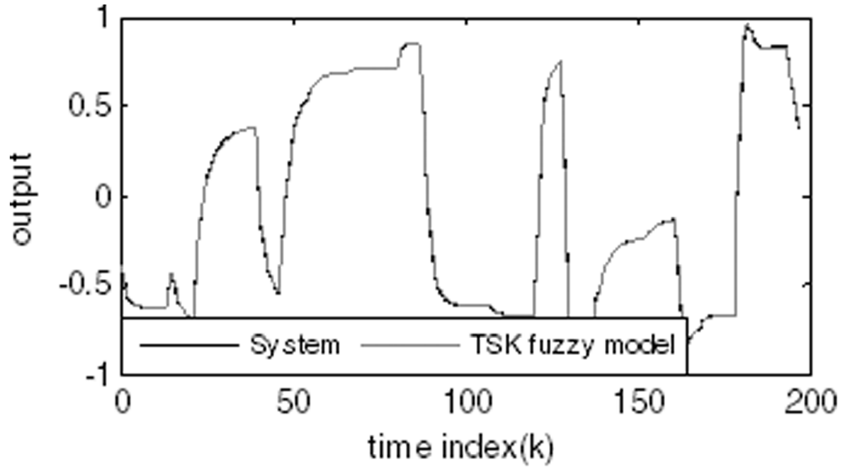

Figure 5: The output signal of the system and fuzzy model of the second plant

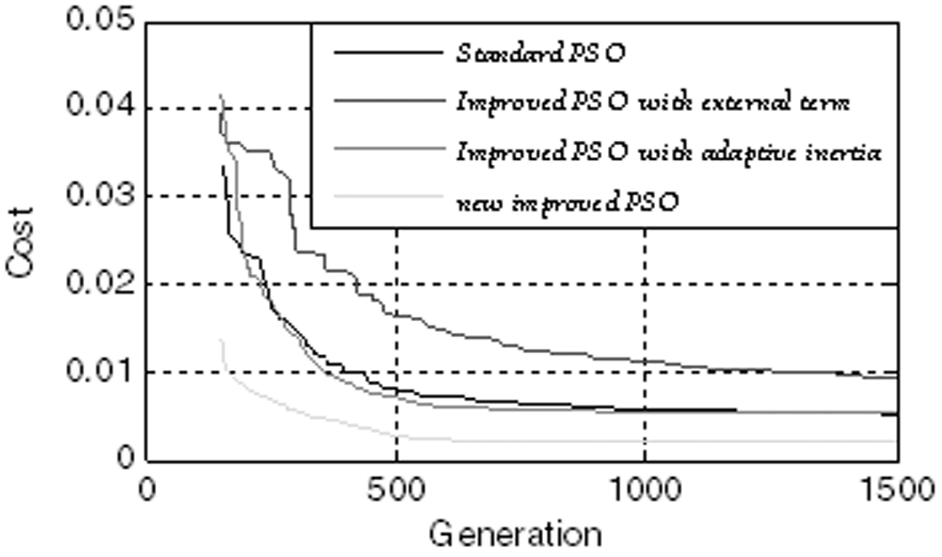

Figure 6: The evaluation of each version of PSO of the first plant

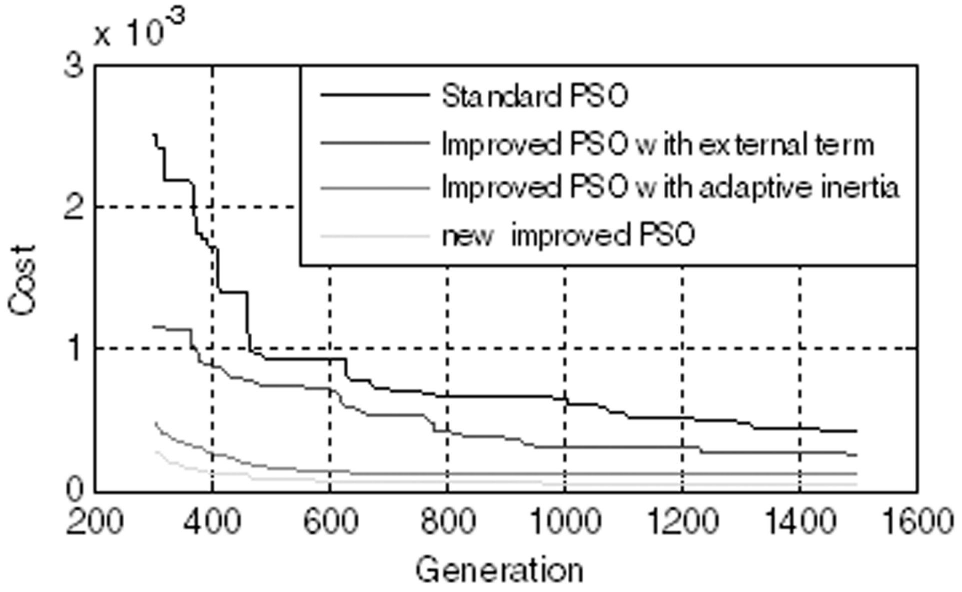

Figure 7: The evaluation of each version of PSO of the second plant

is generated by a sequence of pulses of random amplitude in range [-1 1] and a random period between [1 20] for the second plant, see Figure 2 and 3.

Figure 6 and 7 shows the evolution of the objective function (MSE_Train, MSE_Test are mean square error training and mean square error test) for each version of PSO.However, simulation results are drawn from the three hundredth iterations to show the difference between each curve.

In the first problem the convergence of standard PSO is very slow (after 1470 generation) with a large MSE_train = 0.0053 the improved PSO with external term minimizes the MSE_train = 0.0092 but with a slow convergence (after 1488 generation), the improved PSO with adaptive inertia corrects the problem of convergence (after 1079 generation), but with a significant MSE_train = $0.1000*10^{-4}$. However, in the first problem the convergence of standard PSO is very slow (after 1329 generation) with a large MSE_train = $0.0.4266*10^{-4}$, the improved PSO with external term minimizes the MSE_train = $0.2578*10^{-4}$ but with a slow convergence (after 928 generation), the improved PSO with adaptive inertia corrects the problem of convergence (after 516 generation), but with a significant MSE_train = $0.1000*10^{-4}$. So, this approach is a correction for both problems recognized in previous versions where the MSE_train=0.0020 after only 914 generation and MSE_train=$0.0530*10^{-4}$ in after only 370 for the first and second problem respectively. Hence, overall results of simulation in this stage, we see the new improved PSO able to optimize the TSK fuzzy with any complex nonlinear system.

7.2. Testing stage

When the learning process is finished, a sinusoidal input signal $u_k = \sin(2\pi k/25)$ was applied to both plants to test our contribution, the output of these system and that of the model are shown for each plant in Figure 8 and 9. Here, the table I give the mean square error of simulation result's in this stage

Table 2
The result of simulation

	Standard PSO	*Improved PSO with external term*	*Improved PSO with adaptive inertia*	*New improved PSO*
First Plant	0.0042	0.0078	0.0091	0.0019
Second Plant	0.0037	0.0019	$4.96*10^{-3}$	$4.51*10^{-4}$

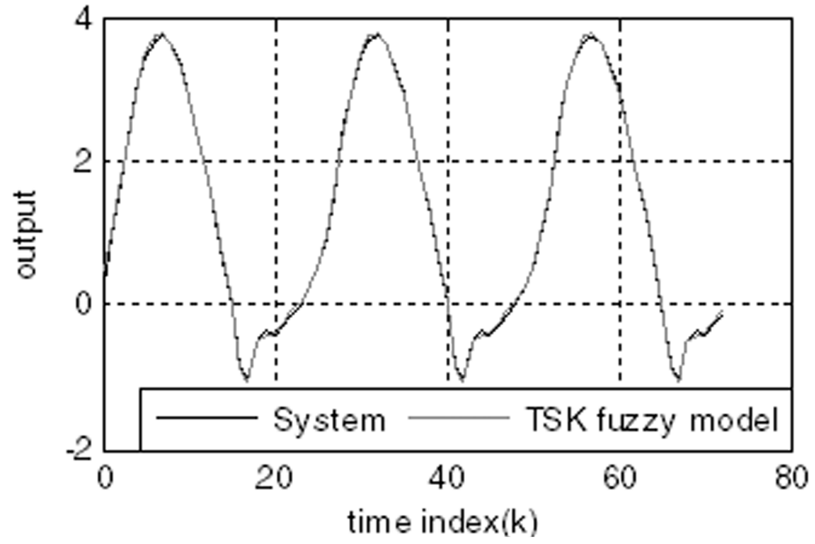

Figure 8: The output of TSK fuzzy model and system of the first plant.

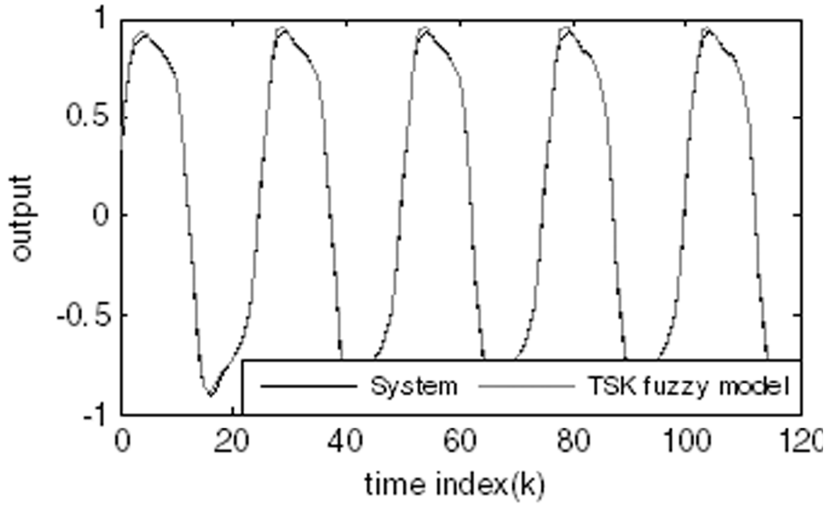

Figure 9: The output of TSK fuzzy model and system of the first plant

According to the values mean square error given in table II, it is noted that the contribution of this paper can reduce the error from 1/2 to 1/4 compared to other contributions for the first plant, whereas, for the second plant the minimum order of reducing the error is in order 1/10.

8. CONCLUSION

This paper presents an optimized Neuro-fuzzy approach by the new improved PSOfor modeling dynamic nonlinear systems, which a proposed version of PSO including two adaptive parameters, the inertia factor decreases from one generation to another to accelerate the convergence and an external term add to reduce the possibility to stocking in the minimum local.

The results of training and testing simulation shown the efficiently to our proposed new improved PSO and the method can be applied to any complex system which can exploit all parameters of this systems.

REFERENCES

[1] J.S. Yadav, N.P. Patidar, J. Singhai, S. Panda and C. Ardil, “A combined conventional and differential evolution method for model order reduction,”*International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, **5** (9), 1284-1291, 2011.

[2] L. Wang and R. Langari, “Complex systems modeling via fuzzy logic”, *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, **26**(1), 100-106, 1996.

[3] C. Gokceoglu and K. Zorlu, “A fuzzy model to predict the uniaxial compressive strength and the modulus of elasticity of a problematic rock”, *Engineering Applications of Artificial Intelligence*, **17** (1), 61-72, 2004.

[4] H.X. Li and C.L.P. Chen, “The equivalence between fuzzy logic systems and feedforward neural networks”, *IEEE Transactions on Neural Networks*, **11**(2), 356-365, 2000.

[5] L.X. Wang and J.M. Mendel, “Back-propagation fuzzy system as nonlinear dynamic system identifiers,” *Proc. IEEE International Conference on Fuzzy Systems*, San Diego, USA, **FUZZY-1992**, 1409-1418, 1992.

[6] R.M. Tong, “The evaluation of fuzzy models derived from experimental data,” *Fuzzy Sets and Systems*, **4** (1), 1-12, 1980.

[7] M. Sugeno and K. Tanaka, “Successive identification of fuzzy model and its application to prediction of a complex system”, *Fuzzy Sets and Systems,* **42** (3), 315-334, 1991.

[8] J. S. R. Jang, “ANFIS: adaptive-networks-based fuzzy inference system,” *IEEE Transactions on Systems, Man, and Cybernetics*, **23** (3), 665-685, 1993.

[9] Y. Shi and R, C, Eberhart, “Fuzzy adaptative particle swarm optimization”, *Proc. 2001 Congress on Evolutionary Computation*, Seoul, **CEC-2001**, 101-106, 2001.

[10] C.J. Lin and S.J. Hong, “The design of neuro-fuzzy networks using particle swarm optimization and recursive singular value decomposition,”*Neurocomputing*, **71**, 297-310, 2007.

[11] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria and M. Valdez,”Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic,” *Expert Systems with Applications*, **40** (8), 3196-3206, 2013.

[12] D.E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison-Wesly, Boston, MA, 1989.

[13] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Proc. IEEE International Conference on Neural Networks*, Perth, Australia, **ICNN-1995**, 1942-1948, 1995.

[14] M.A. Cavuslu, C. Karakuza and F. Karakaya, “Neural identification of dynamic systems on FPGA with improved PSO learning”, *Applied Soft Computing*, **12** (9), 2707-2718, 2012.

[15] T. Takagi and S. Sugeno, “Fuzzy identification of systems and its applications to modeling and control”,*IEEE Transactions on Systems, Man and Cybernetics*, **15** (1), 116-132, 1985.

[16] S.P. Ghoshal, “Optimizations of PID gains by particle swarm optimizations in fuzzy based automatic generation control”, *Electric Power Systems Research*, **72**, 203–212, 2004.

[17] D. P. Kanungo, B. Naik, J. Nayak, S. Badoo and H. S. Behera, “An improved PSO based back propagation learning-MLP (IPSO-BP-MLP) for classification,”*Smart Innovation, Systems and Technologies*, **31**, 333-344, 2014.

[18] K.S. Narendra and K. Parthasarthy, “Identification and control of dynamical systems using neural networks,” *IEEE Transactions on Neural Networks*, **1** (1), 4-27, 1990.

[19] S. Vaidyanathan, “A novel 3-D conservative chaotic system with sinusoidal nonlinearity and its adaptive control”, *International Journal of Control Theory and Applications*, **9** (1), 115-132, 2016.

[20] S. Vaidyanathan and S. Pakiriswamy, “A five-term 3-D novel conservative chaotic system and its generalized projective synchronization via adaptive control method”, *International Journal of Control Theory and Applications*, **9** (1), 61-78, 2016.

[21] S. Vaidyanathan, K. Madhavan and B.A. Idowu, “Backstepping control design for the adaptive stabilization and synchronization of the Pandey jerk chaotic system with unknown parameters”, International Journal of Control Theory and Applications, **9** (1), 299-319, 2016.

[22] A. Sambas, S. Vaidyanathan, M. Mamat, W.S.M. Sanjaya and R.P. Prastio, “Design, analysis of the Genesio-Tesi chaotic system and its electronic experimental implementation”, *International Journal of Control Theory and Applications,* **9** (1), 141-149, 2016.

[23] S. Vaidyanathan and A. Boulkroune, “A novel hyperchaotic system with two quadratic nonlinearities, its analysis and synchronization via integral sliding mode control,” *InternationalJournal of Control Theory and Applications*, **9**(1), 321-337, 2016.

[24] S. Sampath, S. Vaidyanathan and V.T. Pham, “A novel 4-D hyperchaotic system with three quadratic nonlinearities, its adaptive control and circuit simulation,” *International Journal of Control Theory and Applications*, **9** (1), 339-356, 2016.

[25] S. Vaidyanathan and S. Sampath, “Anti-synchronization of identical chaotic systems via novel sliding control method with application to Vaidyanathan-Madhavan chaotic system,” *International Journal of Control Theory and Applications*, **9** (1), 85-100, 2016.