Journal of Engineering Sciences, Assiut University, Vol. 34, No. 2, pp. 557-576, March 2006

MULTIVARIABLE DYNAMIC SYSTEMS MODELING AND CONTROL USING A NEW PARTICLE SWARM ALGORITHM-LOCAL MODEL NETWORK

Nabila M. El-Rabaie and Tarek A. Mahmoud
Faculty of Electronic Engineering, Menouf, 32952, Egypt
Rabaiee2020@yahoo.com ; Tarek_momeen@yahoo.com

(Received December 28, 2005 Accepted February 28, 2006)

ABSTRACT - *This paper introduces the Particle Swarm Algorithm (PSA)-based Local Model network (LMN) for modeling and controlling dynamic systems. Structurally, the proposed PSA-LMN merges the fuzzy set theory and wavelets in a unified form. Learning this network comprises two phases, structure learning phase and parameters learning phase. The former is performed using the Adaptive Resonance Theory (ART) algorithm while the latter is performed using the PSA. The PSA is employed to optimize parameters of the fuzzy sets, the wavelets and the free weights of the proposed LMN. Two simulation nonlinear plants are used to test the soundness of the proposed network; one is a single input single output nonlinear plant and the other is multi-variable medical plant. The latter is employed to test the proposed network in control purposes compared with Genetic Algorithm (GA)-based LMN. Better results were obtained using the proposed PSA-based LMN.*

KEY WORDS: *Fuzzy neural networks, Wavelets, Particle swarm optimization.*

1. INTODUCTION

The Evolutionary Algorithms (EAs) such as the GA [1], the Genetic Programming (GP) [2], and the Multi-objective Evolutionary Programming (MOP) [3] have been proposed to optimize neural networks and avoid drawbacks of using gradient descent methods. GAs are stochastic search procedures [1]. They are more likely to find the global solution of a given problem and they use only a simple scalar performance measure that does not require any derivative information unlike gradient descent algorithms [4]. GAs have been successfully applied in learning fuzzy systems [5], neural networks [6], fuzzy neural networks [7,8] and classification algorithms [9].

Recently, a new evolutionary computing method, the particle swarm algorithm (PSA), is proposed [10]. Similar to the GA, the PSA is initialized with a population of random solutions. It was developed based on the analogy of the swarm theory of bird flocking and fish schooling. Each individual in the PSA is called particle and it is assigned with a randomized velocity according its own and its companion flying

experiences. Compared with the GA, the PSA has some attractive characteristics that can be summarized as follows [11]. It has a memory. That means that knowledge of good solutions is retained by all particles whereas in GA, previous knowledge of the problem is destroyed once the population changes. It has constructive cooperation between particles that share information between them.

Successful applications of the PSA to several optimization problems, such as power systems optimization [12-14], function minimization [15], adaptive control utilizing neural networks [16], systems identification [17], and recurrent neural network designs [11], have demonstrated its potential.

The proposed PSA – based LMN consists of a set of Takagi-Sugeno-Kang (TSK) fuzzy rule fertilized by a wavelet function. Each wavelet determines the contribution of each TSK fuzzy model. These sub-models are merged to provide the final output of the network. The optimization problem of this network comprises two phases, structure phase and parameter phase. The former finds the optimum number of the sub-models, the fuzzy set nodes and the wavelet nodes of the network using the fuzzy Adaptive resonance Theory (ART) [18] in order to track a specified process. The latter optimizes the free network parameters of the network, which are the means and width of a fuzzy set and a wavelet function and the consequent of the fuzzy rule. This optimization is performed using both the PSA and the GA for comparison reasons.
The contribution of this paper can be summarized as follows:

- *Development the PSA-based LMN*
- *Optimizing the parameters of the LMN using PSA and GA.*
- *Development an internal model control scheme using the optimized PSA-based LMN*
- *Employing this developed scheme for controlling a multivariable medical system.*

This paper is organized as follows. Section 2 briefly reviews the PSA. Section 3 describes the proposed PSA-based LMN. Section 4 develops the internal model control scheme using the proposed PSA- based LMN. Simulation results are depicted through sections 3 and 4 respectively. Section 5 concludes the topics discussed in this paper.

2. PARTICLE SWARM ALGORITHM

The PSA was first proposed by Kennedy and Eberhart [Kennedy and Eberhart, 1995]. It is an adaptive algorithm that depends on the observations of the social behavior of animals such as bird flocking, fish schools, and swarm theory. In this section, the basic concepts of the PSA are briefly reviewed.

2.1. The Basic Concepts Of The PSA

Similar to the EAs, in the PSA a set of solutions to the problem under consideration is used to probe the search space. In this algorithm, the term particle is used for each individual and the name swarm is used for the population. Each particle of the swarm has an adaptable velocity (position change) according to which it moves in the search space and it has a memory to remember the best position of the search space that has

ever visited. Thus its movement is an aggregated acceleration towards its best previously visited position and towards the best individual in the swarm [19].

Suppose that the search space is *D*-dimensional, then the i[th] particle of the swarm can be represented by a *D*-dimensional vector, P_i

$$P_i = [p_{i1} \quad p_{i1} \quad . \quad . \quad . \quad p_{iD}]^T$$

The velocity of this particle can be represented by another *D*-dimensional vector, V_i

$$V_i = [v_{i1} \quad v_{i1} \quad . \quad . \quad . \quad v_{iD}]^T$$

The best previously visited position of the i[th] particle is denoted as PL_i defined below:

$$PL_i = [pl_{i1} \quad pl_{i1} \quad . \quad . \quad . \quad pl_{iD}]^T$$

Defining *g* as the index of the best particle in the swarm (i.e., the g^{th} particle is the best). At each time step *t*, the original PSA is described by:

$$V_{id}(t+1) = V_{id}(t) + c_1 r_1 (PL_{id}(t) - P_{id}(t)) + c_2 r_2 (PL_{gd}(t) - P_{id}(t)) \qquad (1)$$

$$P_{id}(t+1) = V_{id}(t+1) + P_{id}(t) \qquad (2)$$

where d=1,2, . . . ,D ; i=1,2, . . . , N , and N is the size of the swarm; c_1 and c_2 are positive acceleration constants, and r_1 and r_2 are uniformly distributed random numbers in the range [0,1]. Equation (1) describes how the velocity is dynamically updated and equation (2) adapts the particle position. Equation (1) consists of three parts. The first part is the momentum part to assure that the velocity can't be changed abruptly. The second part is the "cognitive" part which represents private thinking of itself-learning from its own position experience. The third part is the "social "part which represents the collaboration among particles - learning from group position experience [20]. In equation (1), there was no actual mechanism for controlling the velocity of a particle, it was necessary to impose a maximum value $V_{\max}$. If the velocity violates this limit it has to be set at this value. This parameter proved to be crucial, because large values could result a good indication for past moving of particles, while small values could result in insufficient exploration of the search space [19].

The PSA is simple, easy to implement and has not heavy computation demand. The original procedure for implementing the PSA is summarized below:

1. Initialize a swarm of particles with random positions and velocities on *D* dimensions in the problem space.
2. Evaluate the desired optimization fitness for each particle.
3. Compare particle's fitness evaluation with its *PL*. If current value is better than its *PL* then set *PL* equal to the current position P_i in *D*-dimensional space.
4. Identify the particle in the swarm with the best success so far, and assign its index to the variable *g*.
5. Change the velocity and position of the particle according to (1) and (2).
6. Go to step 2 until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

2.2. The Parameters Of The PSA

As mentioned above, the velocity changes of the PSA consist of three parts, the "social" part, the "cognitive" part, and the momentum part. The balance among these parts determines the balance of the global and the local search ability, and the performance of the PSA that has a set of free parameters.
The first free parameter added into the original PSA is the inertia weight [21]. The velocity adaptation equation of the PSA with inertia weight is given by:

$$V_{id}(t+1) = \Pi V_{id}(t) + c_1 r_1 (PL_{id}(t) - P_{id}(t)) + c_2 r_2 (PL_{gd}(t) - P_{id}(t)) \tag{3}$$

where Π is the inertia weight.

The inertia weight is introduced to regulate between the global and local search abilities. A large inertia weight facilities global search (i.e., searching new area), while a small one facilities local search (i.e., fine- tuning the current search area). A suitable value for the inertia weight provides balance between the global and the local search abilities and consequently results in reduction of the number of iterations required to reach the optimum solution. The experimental results indicated that it is better to initially set to a large value, in order to promote global exploration of the search space, and gradually decreases to get more refined solutions [21]. The following relation is usually utilized to perform the gradually decreasing of the inertia weight [13].

$$\Pi = \frac{\pi_{\max} - \pi_{\min}}{t_{\max}} t \tag{4}$$

where $\pi_{\max}$ is the initial weight, $\pi_{\min}$ is the final weight, and $t_{\max}$ is the maximum iteration number.

Another free parameter called constriction coefficient is introduced with the hope that it insure a PSA to converge [15]. The method of incorporating this factor in a PSA defines in (5).

$$V_{id}(t+1) = k[V_{id}(t) + c_1 r_1 (PL_{id}(t) - P_{id}(t)) + c_2 r_2 (PL_{gd}(t) - P_{id}(t))] \tag{5}$$

where k is a constriction coefficient and is a function of c_1 and c_2 as follows:

$$k = \frac{2}{\left| 2 - \varphi \sqrt{\varphi^2 - 4\varphi} \right|} \tag{6}$$

where $\varphi = c_1 + c_2, \qquad \varphi > 4$

The PSA with the constriction factor can be considered as a special case of the PSA with inertia weight while the three parameters are connected thought (6). The constriction factor k controls the magnitude of the velocities, in a similar way to the V_{max} parameter. A better approach to use as a rule of thumb is to utilize the PSA with constriction factor while limiting V_{max} to $P_{max,}$ the dynamic range of each dimension [20].

3. THE PROPOSED PSA-BASED LMN

In this section, the optimization problem of the proposed PSA-based LMN will be investigated. First, the LMN will be briefly described. Second, the optimization problem using the PSA will be detailed. Finally, the performance of the optimized network using the PSA compared with GA in the modeling scheme will be demonstrated.

3.1. The Proposed Local Model Network

Figure 1 depicts the structure of the proposed LMN. It consists of five layers that are described as follows:

Layer -1: A node at this layer just transmits the input values to the next layer.

Layer -2: This layer consists of two groups, universes of discourse of the input fuzzy variables and their wavelets. The former cover the universes of discourse of the input variables by a set of triangular-shaped function $\mu_{A_j^i}(x_i)$. That is:

$$\mu_{A_j^i}(x_i) = 1 - \frac{2\left|x_i - c_{ij}\right|}{\delta_{ij}} \tag{7}$$

where A_j^i is the j^{th} fuzzy set of the i^{th} input variable x_i, and c_{ij}, and δ_{ij} are the center and width of this fuzzy set respectively.

The latter is a wavelet function generated by dilating and translating the mother wavelet function $h(x) = (1 - x^2)\exp(-\frac{x^2}{2})$. That is :

$$\Phi_j(X) = \prod_{k=1}^{n} h(Z_{jk}) \tag{8}$$

where $Z_{jk} = \frac{x_k - m_{jk}}{d_{jk}}$, n is the number of their inputs, m and d are the translation and dilation parameters respectively.

Layer -3: The firing strength can be obtained using Larsen's product at this layer [22] as follows:

$$\omega^i = \prod_{j=1}^{n} \mu_{A_j^i}(x_i) \tag{9}$$

where, its normalized value is:

$$\varpi^i = \frac{\omega^i}{\sum_{i=1}^{q} \omega^i} \tag{10}$$

Layer-4: A node at this layer is a sub-model that merges the normalized firing strength of a TSK fuzzy rule with a wavelet. That is:

$$y_i^4 = \Phi_i(X)\varpi^i \tag{11}$$

Layer -5: Based on the approximate Center Of Area (COA) defuzzification method, the crisp output y_m can be deduced. That is:

$$y_m = \sum_{i=1}^{q} \varpi^i f_i(\mathrm{X})\Phi_i(X) \tag{12}$$

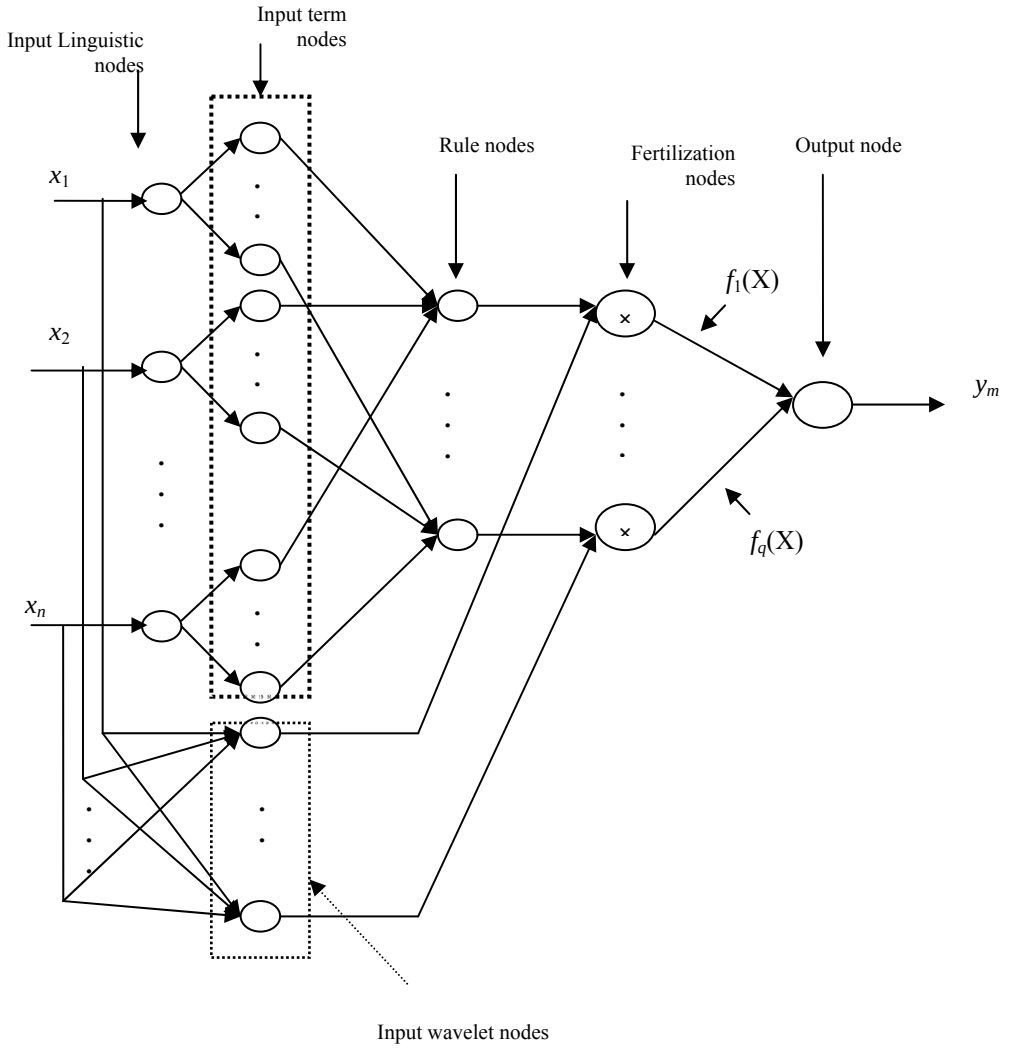

Fig. 1 The structure of the proposed LMN.

The network described above performs the following rule:

R^i : IF x_1 is A_1^i and … and x_n is A_n^i THEN $y^i = f_i(X) * \Phi_i(X)$ (13)

where,
$f_i(X)$ is a linear function of the TSK model. That is:

$$f_i(X) = w_1^i * x_1 + w_2^i * x_2 + . \quad . \quad . + w_1^i * x_n \tag{14}$$

and, $\Phi_i(X)$ is the wavelet function defined in (8).
Reforming (8), results an LMN defined below:

$$f_i(X) = w_{u1}^j * u(k-1) + . \quad . \quad . + w_{ur}^j * u(k-r) + w_{y1}^j * y(k-1) + \quad . \quad . \quad . + w_{ys}^j * y(k-s) \tag{15}$$

Substituting (15) in (12), results:

$$y_m(k) = b_1 * u(k-1) + . \quad . \quad . + b_r * u(k-r) + a_1 * y(k-1) + \quad . \quad . \quad . + a_s * y(k-s) \tag{16}$$

where, $b_i = \sum_{j=1}^{q} w_{ui}^j * \Phi_j(X)\varpi^j$, i=1,2,…r, $a_h = \sum_{j=1}^{q} w_{yh}^j * \Phi_j(X)\varpi^j$, h=1,2,…,s (17)

q is the number of rules generated, r and s are the orders of the plant input and output respectively. Equation (16) represents the proposed LMN with the input and output vectors defined below:
$\underline{X}$ = [u (k-1) u (k-2) . . . u (k-r) y (k-1) y (k-2) . . ., .y (k- s)] ^{T}
$\underline{Y}$=[y (k)] ^{T}

The optimization problem of the proposed LMN can be classified into two phases, the structure phase and the parameter phase. The former should determine the optimal number of the following seeds

- Fuzzy clusters of each fuzzy variable.
- Wavelet nodes
- Fuzzy rules

While the latter optimizes their free parameters.

The clustering algorithms techniques can be employed to perform the structure optimization of the proposed LMN. Among of these algorithms, the fuzzy ART algorithm [18] is employed in this paper to perform this optimization stage. Basically, the fuzzy ART was introduced to finding the parameters of an input membership function (the center "c_{ji}," and the width "δ_{ji}"). This is equivalent to forming the proper fuzzy hyperboxes clusters in the input space, which defines the number of fuzzy rules and wavelet nodes. Forming these clusters requires the initial values of their centers and widths. Determining the parameters of a fuzzy cluster (a fuzzy set), includes the parameters of the corresponding wavelet function. That is

$$m_{ji} = \alpha_1 * c_{ji}, \qquad d_{ji} = \alpha_2 * \delta_{ji} \tag{18}$$

where j =1,2, … q (number of wavelet functions), i=1,2, . . . n (number of input variables), c_{ji}, δ_{ji} are the center and the width of the jth fuzzy set of the ith input , and α_1, α_2 are scaling factors.

In the parameter optimization stage, it should determine the optimum values of the following parameters.

- The fuzzy set parameters.
- The wavelet parameters.
- The rule consequent weights

Basically this optimization stage is a non-linear optimization problem. The main objective of this paper is to investigate this optimization problem using the PSA and to compare it with the GA as follows.

3.2. Optimizing The Parameters Of The LMN Using The PSA

The parameter optimization of the local model network using the PSA comprises two major operators: evaluation and adaptation. Before describing these operators, the issues of coding and initialization are presented. The coding concerns with the way that the network parameters are represented by a set of particles, whereas initialization is the proper assignment of optimization process before entering the evaluation process. The overall optimization process of the proposed LMN by the PSA is described below:

- Coding: A floating point coding scheme is employed in this paper. For the ARX-LMN, suppose the number of the generated sub-models (fuzzy rules) is q and the number of the input nodes is n , and the total number of parameters (D) to be optimized can be set as follows

  D = the number of fuzzy sets nodes ($2*n*q$) + the number of wavelet nodes ($2*n*q$) + the number of rule nodes ($n*q$) = $5*n*q$.

  Thus, the parameters of these nodes can be coded into a particle as follows.

$$P = \begin{bmatrix} F_p & W_p & R_p \end{bmatrix}^T \tag{19}$$

  where, F_p, Wp, and R_p are the vectors that represent the parameters of the fuzzy set nodes (the center and the variance), the parameters of the wavelet nodes, and the parameters of the rule nodes (rule consequent weights). Those are:

$$F_p = \begin{bmatrix} c_{11} \ldots c_{q1} \; c_{12} \ldots c_{q2} \ldots c_{1n} \ldots c_{qn} \; \delta_{11} \ldots \delta_{q1} \; \delta_{12} \ldots \delta_{q2} \ldots \delta_{1n} \ldots \delta_{qn} \end{bmatrix}^T$$

$$W_p = \begin{bmatrix} m_{11} \ldots m_{q1} \; m_{12} \ldots m_{q2} \ldots m_{1n} \ldots m_{qn} \; d_{11} \ldots d_{q1} \; d_{12} \ldots d_{q2} \ldots d_{1n} \ldots d_{qn} \end{bmatrix}^T$$

$$R_p = \begin{bmatrix} w_{11} \ldots w_{q1} \; w_{12} \ldots w_{q2} \ldots w_{1n} \ldots w_{qn} \end{bmatrix}^T$$

- Initialization: In this stage, a population of particles with random positions and velocities was initialized. The position of each particle was assigned to the values of the parameters to be optimized and the velocity was assigned to the change of these parameters. As described above, each particle contains three groups of parameters; the initialization of each group was achieved as follows.

  The first and the second groups, which concern with the fuzzy sets parameters and the wavelet parameters was initialized by using its initial values generated by the fuzzy ART. Those are:

$$
\begin{aligned}
c_{ji}^{h} &= rnd * c_{ji}(0) \\
\delta_{ji}^{h} &= rnd * \delta_{ji}(0) \\
m_{ji}^{h} &= rnd * m_{ji}(0) \\
d_{ji}^{h} &= rnd * d_{ji}(0)
\end{aligned}
\tag{20}
$$

where j=1,2,. . ., q, i =1,2, . . ., n, h=1,2, . . ., N (the size of population), $c_{ji}(0), \delta_{ji}(0)$ are the center and the variance of the j[th] fuzzy set of the i[th] input variable, which were generated by the fuzzy ART, $m_{ji}(0)$, $d_{ji}(0)$ are the translation of the j[th] wavelet function of the i[th] input , which were given by (8), and rnd is a uniformly distributed random number in [0,1].

The third group, which concerns the rule consequence weights, is randomly assigned within the network output domain. For the purpose of generality, the network output domain is normalized be in the range [-1, 1], then the rule consequent weights are randomly initialized within [-1, 1].

- Evaluation: For each particle, the fitness value is calculated. In this paper, the fitness function that measures the quality of each particle is given by the reciprocal of the Root Mean Square(RMS) errors as follows:

$$
\begin{aligned}
RMS &= \sqrt{\frac{1}{N_S}\sum_{k=1}^{N_S}(y_d(k) - y(k))^2} \\
f_i(P_i) &= \frac{1}{RMS}
\end{aligned}
\tag{21}
$$

where i = 1,2,. . . N, y_d (k), y(k) are the desired and the actual outputs of the ARX-LMN at sample k , and N_S is the number of samples.

- Adaptation: In this stage, the best position for each particle PL and the best position swarm P_g are calculated. The original PSA algorithm and the PSA algorithm with the inertia weight are employed in this paper to adapt the position and the velocity of each particle.

3.3. Modeling Simulations

In this sub-section, nonlinear single input single output and multivariable control systems are employed to assure the soundness of the PSA-based LMN in the modeling purposes.

Nonlinear System: A complex system is employed to assure the capabilities of the optimized proposed LMN using the PSA in the dynamic systems modeling. The discrete time difference equation of the system [23] is:

$$
y(k+1) = \frac{\{y(k)y(k-1)y(k-2)u(k-1) * [y(k-2)-1] + u(k)\}}{1 + y^2(k-1) + y^2(k-2)} \tag{22}
$$

The training signal u(k) is defined below:

$$
u(k) = 0.8\sin(2\Pi k / 50) \tag{23}
$$

The input and the output vectors of the network are:

$$X = [y(k) \quad y(k-1) \quad u(k)]^T$$

$$Y_d = [y(k+1)]^T$$

In this example, the PSA with the inertia weight defined in (3), is employed and the parameters $\pi_{\max}$, $\pi_{\min}$, c_1 and c_2 are set to 0.99, 0.4, 2 , and 2 respectively. Initially, 40 individuals are randomly generated in a population, i.e., the size of swarm = 40 and the evolution is processed for 100 generation. To demonstrate the optimized result, two different test signals $u_1(k)$ and $u_2(k)$, are used to assure the modeling capabilities of the optimized network. Those are [23]:

$$u_1(k) = \begin{cases} \sin(2\Pi k / 250) & k \le 500 \\ 0.8\sin(2\Pi k / 250) + 0.2\sin(2\Pi k / 25) & k > 500 \end{cases} \tag{24}$$

and

$$u_2(k) = 0.8\sin(2\Pi k / 200) + 0.2\sin(2\Pi k / 25) \tag{25}$$

The outputs of the optimized network and the process using the two test signals defined in (24) and (25) are depicted in **Fig. 2** and **Fig. 3**. It is clear that the modeling capabilities of the optimized network are outstanding.

For comparison reasons, the network is optimized by the GA to the same example. In the GA, the population size and the initial individuals are the same as those used in the PSA. The parents for crossover are selected from the whole population and the roulette wheel selection method is used. The crossover probability and the mutation probability are set to 0.85 and 0.05 respectively. **Figure 4** depicts the best RMS errors obtained using the PSA and the GA versus a set of generation. Best results are obtained using the PSA compared with the GA. Moreover, computationally the former is smaller than the latter. This is due to the computations needed in GA e.g. mutation and crossover are eliminated in the proposed PSA. A notable feature using the PSA over the GA is that each particle of the swarm has adaptable velocity according to the information acquired from the particles in the search space.

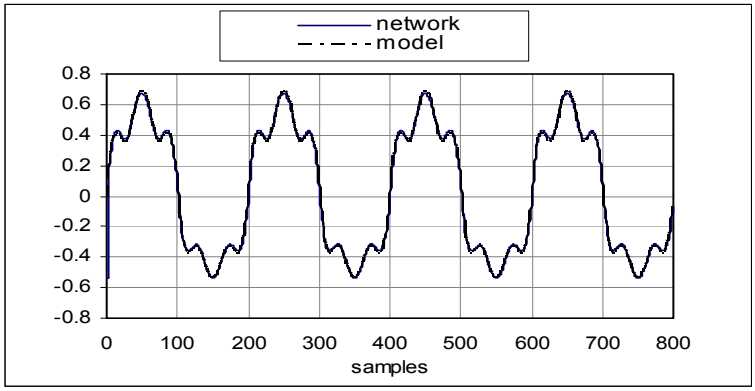

Fig. 2: The outputs of the system and the proposed network using the test signal u_1.

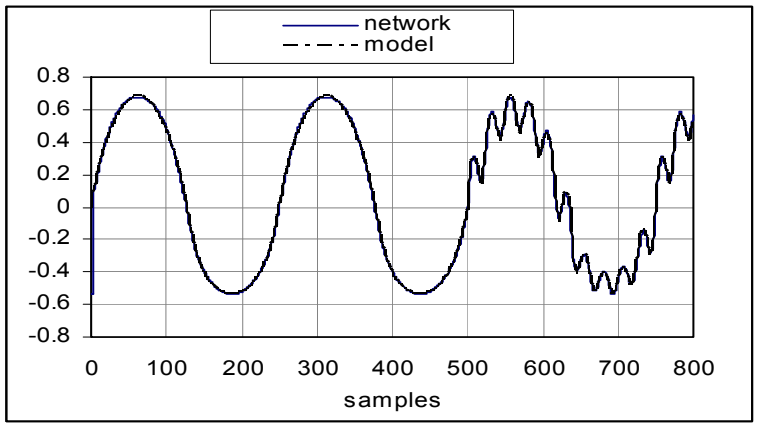

Fig. 3: The outputs of the system and the proposed network using the test signal u_2.

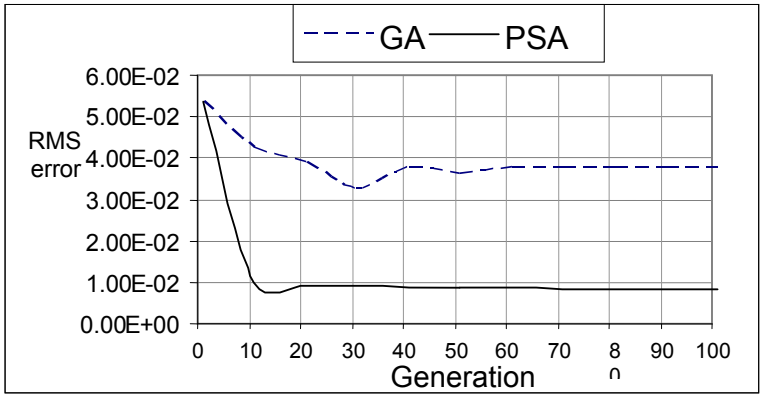

Fig. 4: The best RMS error obtained using the PSA and the GA respectively (nonlinear system).

Multivariable Blood Control System Model

Clinically, it is required to regulate simultaneously the cardiac output (CO) and the mean arterial pressure (MAP) of a patient in hospital intensive care using various drugs. Two typical drugs used are dopamine (DOP), which is an inotropic drug, and sodium nitroprusside (SNP), which is a vasoactive drug. For the purpose of the simulation study Linkens and Nie adopt the same model used in [24, 25] which is given by [26]:

$$\begin{bmatrix} \Delta CO \\ \Delta MAP \end{bmatrix} = \begin{bmatrix} 1.0 & -24.76 \\ 0.6636 & 76.38 \end{bmatrix} \begin{bmatrix} \dfrac{K_{11}\, e^{-\tau_1 s}}{sT_1+1} & \dfrac{K_{12}\, e^{-\tau_2 s}}{sT_1+1} \\ \dfrac{K_{21}\, e^{-\tau_2 s}}{sT_2+1} & \dfrac{K_{22}\, e^{-\tau_2 s}}{sT_2+1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{26}$$

where ΔCO (ml/s) is the change in cardiac output to u_1 and u_2; ΔMAP (mmHg) is the change in mean arterial pressure due to u_1 and u_2 ; u_1 ($\mu g / Kg / \min$) is the infusion rate of dopamine; u_2 (ml/h) is the infusion rate of sodium nitroprusside ; K_{11}, K_{12}, K_{21}, and K_{22} are steady state-gains with nominal values of 8.44, 5.275, -0.09 and -0.15 respectively ; τ_1 and τ_2 represent two time delays with nominal values of τ_1= 60 s and τ_2= 30 s; and T_1 and T_2 are time constants with nominal values of 84.1 s and 58.75 s respectively. In modeling this multivariable system using the ARX-LM network, the input and the output vectors are:

$$X = [\Delta CO(k-1) \quad \Delta MAB(k-1) \quad u_1(k-d_1) \quad u_2(k-d_2)]^T$$

$$Y = [\Delta CO(k) \quad \Delta MAB(k)]^T$$

where $\Delta CO(k)$, $\Delta MAP(k)$ are the changes of the cardiac output and the blood pressure at the k^{th} time step respectively, d_1 and d_2 are the discreet delay times.

The inputs training sequence consists of square pulses with random amplitudes in the range [0, 2.5 ml/h]. The PSA with the inertia weight defined in (3) is also employed and the parameters $\pi_{\max}$, $\pi_{\min}$, c_1 and c_2 are set to 0.99, 0.4, 2 , and 2 respectively. Initially, 20 individuals are randomly generated in a population, i.e., the size of swarm = 20 and the evolution is processed for 500 generation. The outputs of the optimized network and the process and the inputs training sequence for 200 samples are depicted in **Fig. 5** and **Fig. 6** respectively. It is clear that the modeling capabilities of the optimized network are very effective.

Also, for the comparison reasons, the network is optimized by the GA to the same example. The GA used has the same parameters as those used in PSA. Also, the parents for crossover are selected from the whole population and the roulette wheel selection method is used. The crossover probability and the mutation probability are set to 0.85 and 0.05 respectively. **Figure 7** depicts the best RMS error for the PSA and the GA versus a set of generations. The figure shows the superiority of the PSA over the GA in the network optimization for the second example.

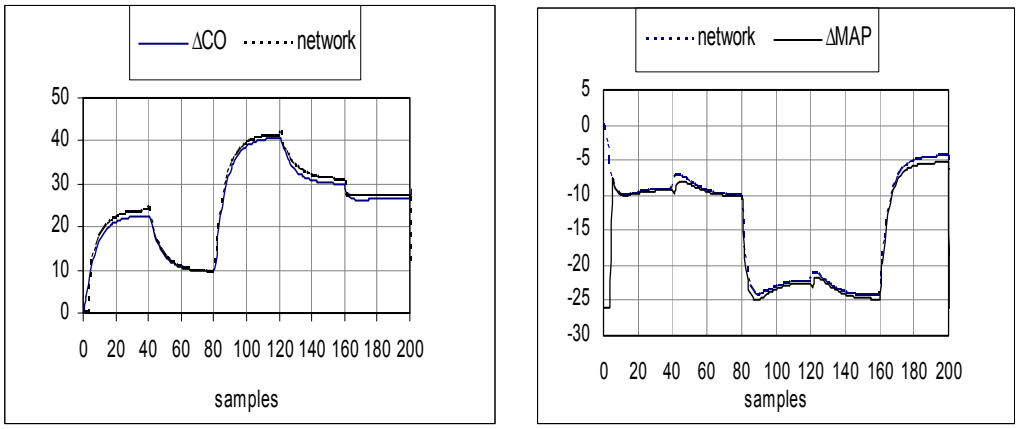

Fig. 5: The change of cardiac and the blood pressure outputs and the network outputs.

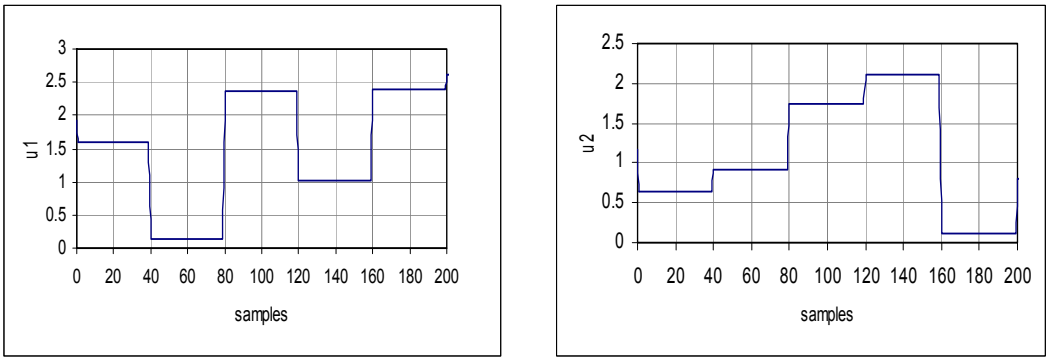

Fig. 6: The training pulses.

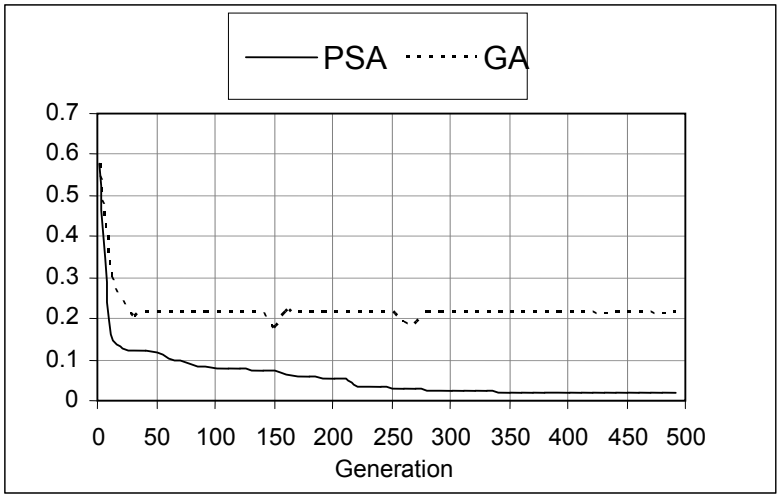

Fig. 7: The RMS error obtained using the PSA and the GA respectively. (Multivariable blood pressure control system)

Outstanding results obtained in modeling simulations, encourages us to extend the proposed PSA-based LMN in control systems. An internal model control (IMC) scheme based on the proposed PSA-based LMN is discussed in section 4.

4. DEVELOPMENT AN IMC USING THE PSA- based LMN

The IMC strategy can theoretically provide perfect control, such that the output of the plant equals to the reference input signal. This is achieved by obtaining a perfect model of the plant and then inverting this model to produce a controller. The basic structure of the IMC depicted in **Fig. 8**, involves two models, the internal model of the plant, and its inverse (controller). An IMC system possesses the dual stability, the perfect control and zero offset properties [27]. The basic structure depicted in **Fig. 8** is not applicable for controlling the dynamic systems that have non-causal, non-minimum phase, and

pure time delay dynamics. Having theses dynamics, leads to unstable and non-causal inverse system model. Isabelle and Leon introduced a modified procedure for the IMC scheme that can be used perfectly with the dynamics [28]. In this scheme, the controller is implemented as the inverse of the model deprived from its delay to eliminate the effect of the delay in the inverse of the model as depicted in **Fig. 9**. The controller is then obtained by cascading this inverse model with rallying model, which ensures the robustness of the stability of the control scheme. A change in the desired regulation dynamic behavior or an improvement of the stability can be obtained by simply tuning of the model. In this procedure, both the delay deprived model and its inverse is structured by using a feed-forward neural network.

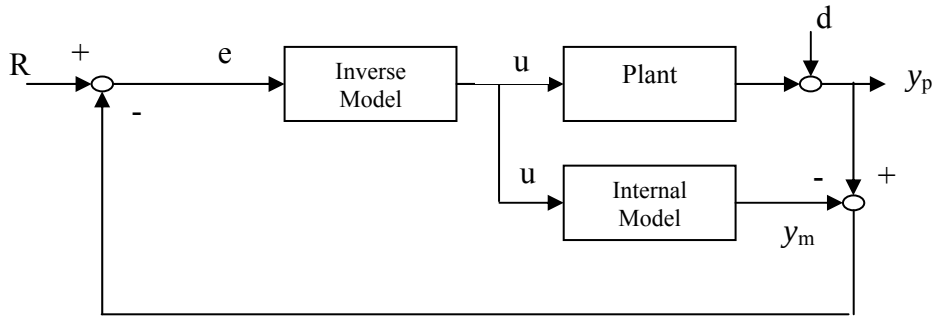

Fig. 8: A typical structure of IMC.

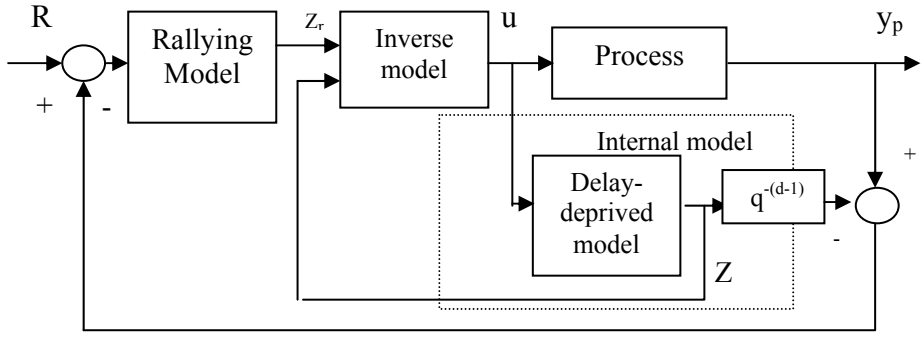

Fig. 9: The modified IMC scheme.

This paper adopts this IMC scheme by using the PSA- based LMN as in the following subsection.

4.1. Design The PSA –Based LMN- Based IMC

Consider the system to be controlled is described by the following discrete-time nonlinear model:

$$y_p(k) = \lambda \ \ (y_p\{_{k-n}^{k-1}, u\{_{k-d-m+1}^{k-d}) \tag{27}$$

where $y_p(k)$ is the process output at instant k, $y_p\{_{k-n}^{k-1}$ denotes the delayed output $\{\{y_p(k-1) \ \ldots \ y_p(k-n)\}$ and similarly $u\{_{k--d-m+1}^{k-d}$ denote the set of m past control inputs $\{u(k-d) \ \ldots \ u(k-d-m+1)\}, d \geq 1$ is the delay, λ is the unknown nonlinear function. Using the optimized ARX-LMN, the given system is defined below:

$$y_m(k) = b_1 * u(k-d) + \ldots + b_{d+m} * u(k-d-m-1) + a_1 * y_p(k-1) + \ldots + a_s * y_p(k-n) \quad (28)$$

where $y_m(k)$ is the output of the model at a sample k. The model parameters a_i and b_i can obtained using the optimized parameters using the PSA which are the fuzzy set, the wavelet, and the rule consequent parameters as described in (17). Using these parameters, the output of the delay deprived model Z(k) can be calculated using the following ARX model.

$$Z(k+1) = b_1 u(k-d) + \ldots + b_{d+m} u(k-d-m+1) + a_1 Z(k) + \ldots + a_n Z(k-n) \quad (29)$$

where: $Z(k) = y_m(k+d)$

Once the delay deprived model stage is constructed, the only problem to be solved is how to develop the inverse model to be used as a controller. The parameters of this inverse model (the controller) are estimated at each time step. That means that the concept of developing an inverse model is deprived and replaced by a parameter estimation scheme. That simplifies the structure of the IMC scheme and makes the enhanced ARX-LMN-based IMC scheme promising for modeling and controlling real process.

Consider the delay deprived model described by the proposed PSA- based LMN as in 29, where the parameters a_i and b_i of the proposed network depends on the operating point. Based on this model formulation, the following exact inverse control law can be postulated:

$$\begin{aligned} u(k-d) = \frac{1}{b_1}(Z(k+1) - a_1 Z(k) - a_2 Z(k-2) - \ldots \\ - a_n Z(k-n) - b_2 u(k-d-1) \ldots - b_{m+d} u(k-d-m+1) \end{aligned} \quad (30)$$

This is possible if and only if the operating point does not depend on u (k). So, a novel approach is suggested to solve this problem by reforming (30) as follows:

$$\begin{aligned} u(k-d) = a'_o Z(k+1) + a'_1 Z(k) + a'_2 Z(k-2) - \ldots \\ + a'_n Z(k-n) + b'_1 u(k-d-1) \ldots + b'_{m+d-1} u(k-d-m+1) \end{aligned} \quad (31)$$

Using the same input-output data used in the learning the internal model, the parameters a'_i and b'_j can be estimated by using the Recursive Lease Square (RLS) method described as follows. Suppose the weight and the input/output vectors are:

$$\beta' = [Z(k+1) \quad Z(k) \quad . \quad . \quad . \quad Z(k-n) \quad u(k-d-1) \quad . \quad . \quad . \quad u(k-d-m-1)]^T$$

$$\theta' = [a'_0 \quad a'_1 \quad . \quad . \quad . \quad a'_n \quad b'_1 \quad b'_2 \quad . \quad . \quad . \quad b'_{d+m-1}]^T$$

The control law can be written as:

$$u(k) = \beta' * \theta' \tag{32}$$

The linear parameters θ' are recursively estimated by:

$$\theta'(k) = \theta'(k-1) + P'(k-1)\beta'(k)e'(k)$$
$$P'(k) = P'(k-1) - \frac{P'(k-1)\beta'(k)\beta'^T(k)P'(k-1)}{1+\beta'^T(k)P'(k-1)\beta'(k)} \tag{33}$$

where $e'(k)$ is the error between the desired and the actual outputs of the inverse of the delay deprived model respectively. That is:

$$e'(k) = (u_d(k) - u\ (k)) \tag{34}$$

where $u_d(k)$ is the desired output of the inverse of the delay –deprived model network which has been used in the internal model learning, and u(k) is the actual output of the controller given by 32. Once the delay deprived model and its inverse are structured off line, the on line structure depicted in **Fig. 2** is applied. In the on line phase, the output of the inverse of the delay-deprived model can easily be calculated as follows:

$$\begin{aligned} u(k-d) = {} & a'_o Z_r(k+1) + a'_1 Z(k) + a'_2 Z(k-2) - . \quad . \quad . \\ & + a'_n Z(k-n) + b'_1 u(k-d-1) . \quad . \quad . + b'_{m+d-1} u(k-d-m+1) \end{aligned} \tag{35}$$

where $Z_r(k+1)$ is the output of the rallying model that can be tuned to improve the system tracking and stability. Controlling the multivariable blood pressure control system described in section III, assures the soundness of the modified IMC scheme.

4.2. Control Simulation Results

In this subsection, the simulation results of the IMC based on the proposed PSA- based LMN for the multivariable blood pressure control system are described. The sampling time is 30 second, and the set- points for CO and MAP was set to be 20 ml/s and -10 mmHg changing from nominal values of 100 ml/s and 120 mmHg respectively [29]. To investigate the effectiveness of the proposed scheme, four cases of the system parameters are performed; these cases are listed as follows [Linkens and Nie, 1995]:

- The first case: The plant parameters (K_{11}, K_{12}, K_{21}, K_{22}, τ_1, τ_2, T_1 and T_2) are set to the nominal values.
- The second case: The plant parameters (K_{21}, K_{22}, τ_1, τ_2, T_1 and T_2) are set to the nominal values and the two parameters K_{11}, K_{12} were abrupt changed by 10% from their nominal values.

- The third case: The plant parameters (K_{11}, K_{12}, τ_1, τ_2, T_1 and T_2) are set to the nominal values and the two parameters K_{21}, K_{22} were abrupt changed by 10% from their nominal values.
- The fourth case: The plant parameters (K_{11}, K_{12}, K_{21}, K_{22}, τ_1 and τ_2) are set to the nominal values and the two parameters T_1 and T_2 were abrupt changed by 10% from their nominal values.

Figure 10 shows the response of the process using the proposed predictive scheme when the fourth case of the system parameters is presented. These dynamics are acceptable according to the results published in [29].

Also, to show the effectiveness and efficiency of PSA in the control problem, the GA is applied to the same control problem. The PSA-based LMN-based IMC is compared with the GA- based LMN-based IMC in the sense of the RMS error defined in 36. The RMS error obtained are depicted in **Table. 1**.

$$RMS = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\Delta CO_d - \Delta CO)^2 + (\Delta MAP_d - \Delta MAP)^2} \tag{36}$$

where N is the number of samples, $\Delta CO_{\,\mathrm{d}}$, $\Delta MAP_{\,\mathrm{d}}$ are the desired changes of ΔCO and ΔMAP respectively. The table shows the superiority of the PSA over the GA in the control scheme.

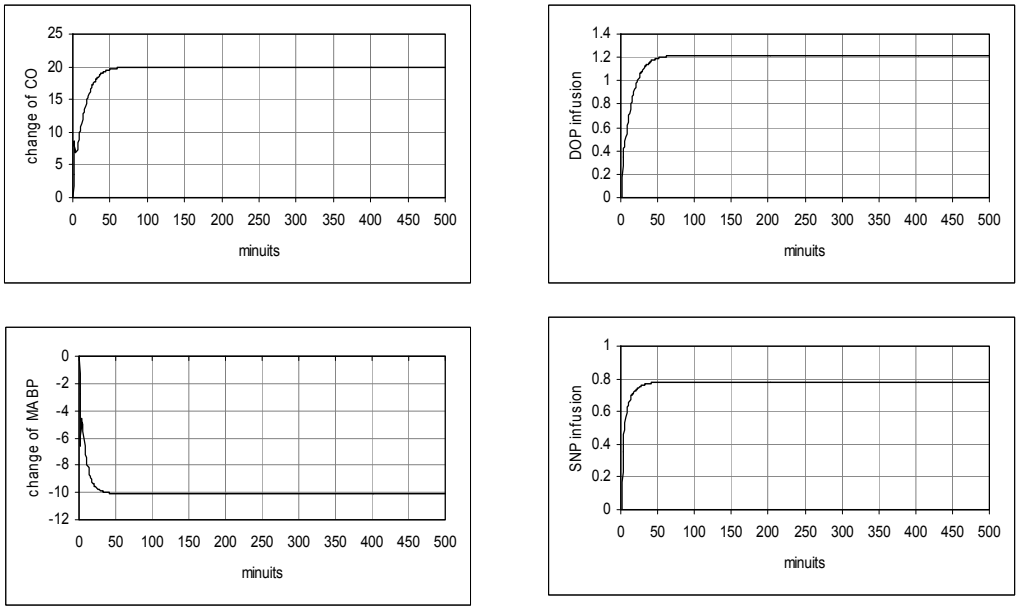

Fig. 10: The change for both cardiac and blood pressure outputs using the proposed PSA- based LMN –based IMC (case four).

Table. 1

System cases	RMS errors	
	The GA- LMN-based IMC	The PSA- LMN-based IMC
Case one	1.951	0.326
Case two	3.164	1.178
Case three	2.231	1.145
Case four	1.953	0.324

5. CONCOLUSIONS

This paper introduced the PSA-based LMN for modeling and controlling dynamical systems. Structurally, it merges the fuzzy set theory and the wavelets in a unified form. The fuzzy ART algorithm was employed in structure learning of the proposed network, while the PSA was used to optimize the parameters of the network. The main notable feature of using the PSA compared with the GA is that its particles observe each other in the search space and adapt themselves based on the information received from those particles closed to the best solution. The philosophy of the IMC scheme was adopted in this paper using the proposed PSA-LMN. Better results have been achieved using the proposed PSA-based LMN in modeling and controlling nonlinear dynamic systems compared with the GA-based LMN.

REFERENCES

[1] D. Goldberg, *Genetic algorithm in search, optimization, and machine learning*. Addison Wesely, Reading, MA, 1989.

[2] Z. Michalewicz, *Genetic algorithm + data structure= evolution program*. Springer, Berlin, 1994.

[3] G. Jesues, R. Ignacio and O. Julio, "Multi-objective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation ," *IEEE Trans. Neural Network*, vol. 14, no. 6, pp. 1478-1495, 2003.

[4] J. M. Zurada, *Introduction to artificial neural systems*. West Publishing Company, New York, USA, 1992.

[5] P. Daihee, K. Abraham, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. System. Man. Cybern.*, vol. 24, no. 1, pp. 39-47, 1994.

[6] P. Vassilios, "A hybrid neural-genetic multimodal parameter estimation algorithm," *IEEE Trans. Neural Network*, vol. 9, no. 5, pp. 862-876, 1998.

[7] R. Marco, " FuGeNeSys- a fuzzy genetic neural system for fuzzy modeling," *IEEE. Trans. Fuzzy System*, vol. 6, no. 3, pp 373-388, 1998.

[8] G. L. Yih, "Nonlinear system modeling using GA-based B-spline membership fuzzy –neural networks," *2nd International Conference on Autonomous Robots and Agents*, December 13-15, Palmerston North, New Zealand, 2004.

[9] U. Sheng and Z. Fangming, "An incremental approach to genetic-algorithms-based classification," *IEEE Trans. System Man. Cybern.,* vol. 35, no. 2, pp. 227-239, 2005.

[10] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *In Proceeding IEEE International Conference Neural Networks*, Perth, Australia, November, pp. 1942-1948, 1995.

[11] F. J. Chia, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. System Man. Cybern.,* vol. 34, no. 2, pp. 997-1006, 2004.

[12] Y. Hirotaka, K. Kenichi, F. Yoshikazu and N. Yosuke, "A particle swarm optimization for reactive power and voltage control considering voltage stability," *IEEE International Conference on Intelligent System Applications to Power Systems (ISAP'99)* , Rao de Janeiro, April 4-8, 1999.

[13] Y. Hirotaka, K. Kenichi, F. Yoshikazu, N. Yosuke and T. Shinichi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Trans. on Power System,* vol. 15, no. 4, pp. 1232-1239, 2001.

[14] N. K. Ioannis, M. A. El-Sharkawi, R. J. Marks, S. Luciano, Moulin and A. P. Alves da Silva, "Dynamic security border identification using enhanced particle swarm optimization," *IEEE Trans. on Power System,* vol. 17, no. 3, pp. 723-729, 2002.

[15] M. Clerc and J. Kennedy, "The particle swarm- explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.

[16] V. E. C. Alex, M. Risto and A. Christiaan, "adaptive control utilizing neural swarming," *Proceedings of genetic and Evolutionary Computation Conference*, New York, USA, 2002.

[17] S. V. Mark and F. Xin, "ARMA model selection using particle swarm optimization and AIC criteria," *IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002.

[18] G. A. Carpenter, S. Grossberg and D. B. Rosen , "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance theory," *Neural Networks*, vol. 4 , pp. 759-771, 1991.

[19] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing* Vol .1, pp. 235-306, 2002.

[20] Y. Shi, "Particle swarm optimization," *IEEE Neural Network Society*, 2004.

[21] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," *Proceeding of the Annual Conference on Evolutionary Computation*, March, 1998.

[22] P. M. Larsen, "Industrial applications of fuzzy logic control," *Int. J. Man Mach. Studies*, vol. 22, pp. 3-10, 1980.

[23] W. Daniel, C. Ho, Ping-An Zhang and Xu. Jinhua, "Fuzzy wavelet networks for function learning," *IEEE. Trans. Fuzzy System,* vol. 9, no. 1, 2001.

[24] D. A. Linkens and J. Nie, "A unified real time approximate reasoning approach for use in intelligent control application to multivariable blood pressure control," *Int. J. Control*, vol. 56, pp. 334-363, 1992-a.

[25] D. A. Linkens and J. Nie, "A unified real time approximate reasoning approach for use in intelligent control application to multivariable blood pressure control," *Int. J. Control*, vol. 56, pp. 365-398, 1992-b.
[26] D. A. Linkens and J. Nie, "FCMAC: a fuzzified cerebellar model articulation controller with self-organizing capacity," *Automatica*. v*ol.* 30, no. 4, pp. 655-664, 1994.
[27] M. Morari and E. Zafiriou. *Robust process control*. Englewood Cliffs, NJ; Prentice-Hall, 1989.
[28] R. Isabelle, P. Leon, "Nonlinear internal model control using neural network: application to processes with delay design issues," *IEEE Trans. Neural Network*, vol. 11, pp. 80 -90, 2000.
[29] D. A. Linkens and J. Nie, *Fuzzy-neural control: principles, algorithms, and applications.* Prentice Hall International (UK) ltd, 1995**.**

النمذجة و التحكم للأنظمة الديناميكية متعددة المتغيرات باستخدام شبكة محلية معتمدة على خوارزم التزاحم الجزئي

د. نبيلة محمود الربيعى م. طارق احمد محمود
قسم هندسة الالكترونيات الصناعية والتحكم - كلية الهندسة الالكترونية - منوف

يقدم هذا البحث شبكة عصبية محلية جديدة معتمدة على خوارزم التزاحم الجزئي وذلك لنمذجة والتحكم في الأنظمة متعددة المتغيرات. تدمج هذه الشبكة مابين المنطق العيمى و الدوال المويجية بطريقة موحدة. تعليم هذه الشبكة بتضمن مرحلتين هما مرحلة تعليم بناء الشبكة و مرحلة تعليم المتغيرات. في المرحلة الأولى يتم توظيف خوارزم (Fuzzy ART) لبناء الشبكة المحلية بينما في المرحلة الثانية يتم استخدام خوارزم التزاحم الجزئي (Particle Swarm Algorithm) لتحديد القيم المثلى لمتغيرات الشبكة و هي متغيرات المجموعات الغيمية و الدوال الميجية. تم استخدام نظامين لا خطين لقياس مدى كفاءة الشبكة المقدمة في النمذجة. النظام الأول هو نظام وحيد الدخل و الخرج بينما النظام الثاني هو نظام طبي متعدد الدخل و الخرج. لقياس كفاءة الشبكة المقترحة في التحكم متعدد المتغيرات حيث تم تطوير نظام تحكم من النوع
((Internal Model Control Scheme (IMC) باستخدام الشبكة العصبية الجديدة. النظام الطبي وهو نظام ضبط كل من ضغط الدم و خرج القلب باستخدام نوعين من العقارات و هما نترات الصوديم و الدوبامين تم استخدامه لقياس كفاءة الشبكة المقترحة في التحكم متعدد المتغيرات . و قد لوحظ من خلال النتائج المحاكية أن الشبكة العصبية المحلية المقترحة المعتمدة على خوارزم التزاحم الجزئي تعطى نتائج أفضل من الشبكة العصبية المحلية المقترحة المعتمدة على الخوارزم الوراثى التقليدي في النمذجة و التحكم. ويمكن تلخيص إسهامات هذا البحث كالآتي :

١. بناء شبكة عصبية محلية جديدة معتمدة على خوارزم التزاحم الجزئي وذلك لنمذجة والتحكم في الأنظمة متعددة المتغيرات.
٢. تطوير نظام تحكم من النوع Internal Model Control Scheme باستخدام الشبكة العصبية الجديدة.
٣. تطبيق نظم التحكم المقترح في كل من ضغط الدم و خرج القلب باستخدام نوعين من العقارات و هما نترات الصوديم و الدوبامين.