Adaptive fuzzy APSO based inverse tracking-controller for DC motors

Karim H. Youssef, Manal A. Wahba, Hasan A. Yousef and Omar A. Sebakhy
Electrical Eng. Dept. Faculty of Eng., Alexandria University, Alexandria, Egypt

This paper introduces the use of the Adaptive Particle Swarm Optimization (APSO) for adapting the weights of Fuzzy Neural Networks (FNN). The fuzzy network is used for the identification of the dynamics of a DC motor with nonlinear load torque. Then the speed of the motor is controlled using an inverse controller to follow a required sinusoidal speed trajectory. The parameters of the DC motor are assumed unknown as well as the nonlinear load torque characteristics. In the first stage a nonlinear fuzzy neural network FNN is used to approximate the motor voltage as a function of the motor speed samples. In the second stage, the above mentioned approximator is used to calculate the control signal (the motor voltage) as a function of the speed samples and the required reference trajectory. Unlike the conventional back-propagation technique, the adaptation of the weights of the FNN approximator is done on-line (at each iteration) using adaptive particle swarm optimization based on the least squares error minimization with random initial condition without any off-line pre-training. The adaptive particle swarm algorithm is used to track the changes in the nonlinear load torque.

يقدم هذا البحث استخدام الانتقاءالأمثل السربى المتلائم فى تعديل اوزان الشبكات العصبية الغيمية المتلائمة واستخدام ذلك فى تعريف النموذج الرياضى لمحرك تيار مستمر ذى حمل غير خطى. ومن ثم يتم التحكم فى سرعة المحرك باستخدام متحكم عكسى لتتبع سرعة مرجعية جيبية وذلك مع اعتبار أن ثوابت المحرك والحمل غير معروفة .فى المرحلة الأولى تستخدم شبكة عصبية غيمية متلائمة لتعريف جهد العضو الدوار كدالة من سرعة المحرك فى ثلاث قراءات متعاقبة وفى المرحلة الثانية تستخدم هذه الشبكة المعرفة كمتحكم عكسى لحساب الجهد المطلوب لكى تتبع سرعة المحرك السرعة المرجعية المطلوبة و يتم تعديل أوزان الشبكة الغيمية تكراريا أثناء عملية التحكم باستخدام الانتقاء الأمثل السربى المعتمد على مبدأ أقل مربعات الخطأ مع أخذ الاوزان الابتدائية عشوائيا بدون أى معلومات مسبقة أو تدريب مسبق، ويكون الانتقاء الأمثل السربى متلائما لملاحقة التغير فى الحمل غير الخطى.

Keywords: Particle Swarm Optimization (PSO), Fuzzy Neural Networks (FNN), Least Squares (LS), System identification, Inverse Control (IC).

1. Introduction

Particle Swarm Optimization (PSO) is a population- based optimization technique first introduced by Kennedy and Eberhart [1] in 1995. The PSO is considered the competitor to the Genetic Algorithms (GA) and there is a lot of research comparing between them [2]. The motivation for the development of the PSO was based on the simulation of animal social behaviors such as bird flocking fish, schooling, etc. The population in PSO is called a swarm and the individuals, referred to as particles, are candidate solutions to the optimization problem at hand in the multidimensional search space. Each dimension of this space represents a parameter of the problem to be optimized. The algorithm does not require the objective function to be differentiable and continuous and it can be applied to nonlinear and non-continuous optimization problems. PSO has been applied to some power system problems such as, state estimation, optimal power flow and reactive power compensation [3:7], and has been shown to perform well. It has been used also in PID tuning [8,9], adaptive control [10:11] and nonlinear observers [12]. In the particle swarm during each iteration, each particle accelerates in the direction of its own personal best solution found so far, as well as in the direction of the global best position discovered so far by any of the particles in the swarm. This means that if a particle discovers a promising new solution, all the other particles will move closer to it, exploring the region more thoroughly in the process. In this paper, the PSO is used for training an adaptive inverse nonlinear controller for DC motors. The nonlinear identification is done using a fuzzy neural network approximator. The paper is organized as follows. In section (2), the dynamics of the DC motor with nonlinear load is investigated. The inverse controller for speed tracking is illustrated in section (3) and the construction of the FNN is discussed in section (4). The detailed PSO algorithm is given in section (5) and finally, simulation results are given in section (6) to validate the effectiveness of the controller.

2 . The DC motor dynamics

The DC motor dynamics are given in the following two equations:

$$K\omega(t) = -R_a i_a(t) - L_a \frac{di_a(t)}{dt} + \upsilon(t) . \qquad (1)$$

$$Ki_a(t) = J\frac{d\omega(t)}{dt} + D\omega(t) + T_L(t) , \qquad (2)$$

where,

$\omega(t)$ is the rotor speed in rad/sec.,
$\upsilon(t)$ is the motor terminal voltage in volt,
$i_a(t)$ is the armature current in A,
$T_L(t)$ is the load torque in Nm,
J is the rotor inertia in N.m.sec^2,
K is the motor torque constant and voltage constant in NmA^{-1},
D is the damping constant in N.m.sec.,
R_a is the armature resistance in Ω, and
L_a is the armature inductance in H.

The nonlinear load torque can be expressed as:

$$T_L(t) = g(\omega(t)) . \qquad (3)$$

For our case let us assume that the nonlinear load torque is expressed as:

$$T_L(t) = \mu\omega^2(t)\,\text{sign}\,(\omega(t)) , \qquad (4)$$

where μ is constant. This load torque is common characteristic of most propeller-driven and fan type loads. The discrete time model is derived by combining eqs. (1, 2 and 4) and then replacing the speed differentials and the torque differentials with backward finite difference [13]. The resulting discrete model is:

$$\omega_{k+1} = \alpha\omega_k + \beta\omega_{k-1} + \gamma\omega_k^2\,\text{sign}\,\omega_k + \delta\omega_{k-1}^2\,\text{sign}\,\omega_k + \xi\upsilon_k , \qquad (5)$$

where $\alpha, \beta,$ and ξ are constant values depending on the motor parameters J, K, D, R_a, L_a and the sampling period T, while γ and δ in addition to being functions of the above parameters are also functions of μ.

3. The inverse controller

Inverse controllers have been used in many control and drive applications [11]. Fig. 1 shows the basic concept of identification and control of the DC motor using a Fuzzy Neural Networks (FNN). The motor is first identified using a combination of its input/ output variables using a FNN. The weights from the trained FNN identifier are then used in the controller to calculate the terminal voltage which will asymptotically drive the motor speed ω_k towards the specified reference trajectory r_k .

Eq. (5) can be inversed and manipulated to the form:

$$v_k = f(\omega_{k+1}, \omega_k, \omega_{k-1}) , \qquad (6)$$

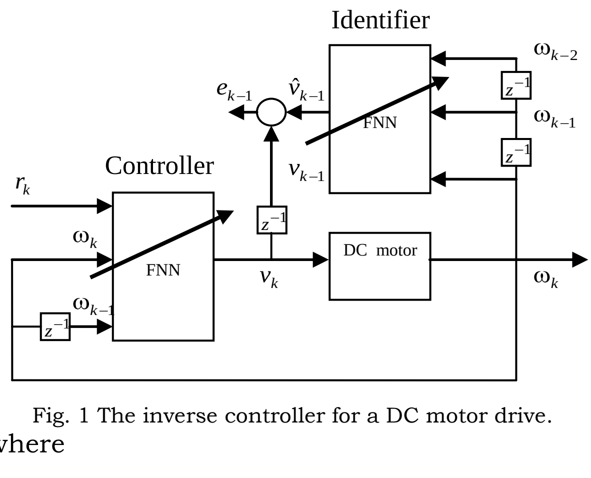

Fig. 1 The inverse controller for a DC motor drive.

where

$$f(\omega_{k+1},\omega_k,\omega_{k-1}) = [\omega_{k+1} - \alpha\omega_k - \beta\omega_{k-1} - \gamma\omega_k^2 sign\omega_k - \delta\omega_{k-1}^2 sign\omega_k]/\xi, \quad (7)$$

and it is assumed unknown. For on-line training, v_k is the controller output and it is not available in the identification stage so we can use the previous sample for the identification process and eq. (6) can be written as:

$$v_{k-1} = f(\omega_k,\omega_{k-1},\omega_{k-2}), \quad (8)$$

and in the control stage the control voltage is calculated using the desired reference trajectory r_k as follows:

$$v_k = f(r_k,\omega_k,\omega_{k-1}). \quad (9)$$

An inverse controller for DC motors was proposed in [13] using neural networks but the training was done off-line with the conventional back-propagation technique and the neural network was so complicated.

4. Fuzzy neural networks

In general, using Takagi-Sugueno fuzzy model, and using center of average defuzzification method, any nonlinear function of states $f(u_1,\cdots u_n)$ can be expressed as:

$$\hat{f}(u_1,\cdots u_n) = \frac{\sum_{l=1}^{M} \theta_l (\prod_{i=1}^{n} \mu_{il}(u_i))}{\sum_{l=1}^{M} (\prod_{i=1}^{n} \mu_{il}(u_i))}, \quad (10)$$

Where

n is the number of states,

$\mu_{il}(u_i)$ is the membership degree of the i^{th} state u_i to the its corresponding membership function in the l^{th} rule,

M is the number of rules and $M = \prod_{i=1}^{n} m_i$ where m_i is the number of membership functions assigned to each state u_i, and

θ_l is the singleton output of the l^{th} rule.

Eq. (10) can be rewritten as:

$$\hat{f}(u_1,\cdots u_n) = \sum_{l=1}^{M} \theta_l \, \xi_l(u), \quad (11)$$

where

$$\xi_l(u) = \frac{\prod_{i=1}^{n} \mu_{il}(u_i)}{\sum_{l=1}^{M}(\prod_{i=1}^{n} \mu_{il}(u_i))}$$

is the Fuzzy Basis Function (FBF). In a matrix form

$$\hat{f}(u_1,\cdots u_n) = \theta\xi(u), \quad (12)$$

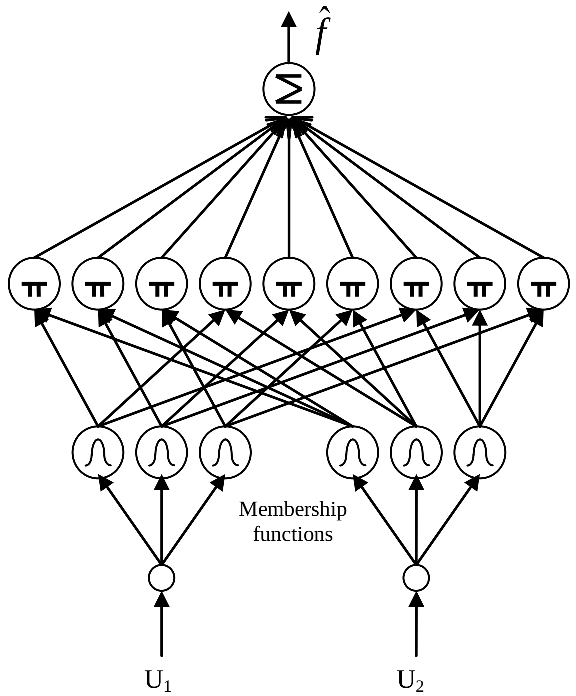

Fig. 2. Basic structure of the Fuzzy Neural Network (FNN).

where

$$\theta = [\theta_1 \cdots \theta_M], \quad \xi^T(u) = [\xi_1(u) \cdots \xi_M(u)]$$

The identification process is done by estimating the parameter vector θ. In our case the input vector $u = [\omega_k \quad \omega_{k-1} \quad \omega_{k-2}]^T$ in the identification stage and $u = [r_k \quad \omega_k \quad \omega_{k-1}]^T$ in the control stage.

5. The PSO algorithm

The PSO has been used for training neural networks [14], FNN [15, 16] and parameter estimation [17:20]. For fuzzy neural networks, the dimension of the problem *DIM* is the number of weights. i.e. the number of fuzzy rules and can be calculated as follows: *DIM* = $(N_m)^{Ni}$ where N_m is the number of membership functions of each input and N_i is the number of inputs.

Each particle in the population constitutes a possible solution of the weight vector θ. In this case, the objective function that has to be minimized at each iteration k should express the least square error among a moving window of N samples as follows:

$$J_i(k) = \sum_{j=k-N}^{k} \left(X_i^T \xi(\omega_j, \omega_{j-1}, \omega_{j-2}) - \upsilon_{j-1} \right)^2 . \quad (13)$$

The basic PSO algorithm is implemented in the following steps:

Step.1. Generation of initial population: Initial positions $X_i = [x_{i1} \cdots x_{iDIM}]^T$ and initial velocities $V_i = [v_{i1} \cdots v_{iD}]^T, \quad i = 1:N$ are generated randomly within the search space. The initial fitness value of each particle is evaluated using the fitness function $J_i(o)$ and the local best of each particle is set to its current fitness value $\text{pbest}_i = J_i(o)$ and the initial local pest position of each particle is set to its initial position $P_i = X_i$.The global best value gbest is set to the best value of pbest and the corresponding particle position is taken as the position of the global best position P_g.

Step. 2. Modification of each particle velocity and position: The velocity and position are updated using the following equations:

$$V_i = K[V_i + c1 * rand(\,) * (P_i - X_i) + c2 * rand(\,) * (P_g - X_i)] \cdot \quad (14)$$

$$X_i = X_i + V_i , \quad (15)$$

where

$K = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}$	is the Clerc's constriction factor [21] and $\phi = c_1 + c_2 > 4$,
c_1 and c_2	are acceleration constants, and
r and( )	is a uniformly distributed random number between 0 and 1

Ref. [21] shows that setting $\phi = 4.1$ gives the best performance so that the inertia weight is kept constant at 0.729 and both acceleration coefficients are kept constant at 1.494.

Step.3. Evaluation of objective function and updating local best memory: The objective function value of each particle $J_i(k)$ is calculated. The best fitness value of each particle is also updated as follows:

$$pbest_i(k) = \sum_{j=k-N}^{k} \left(P_i^T \xi(\omega_j, \omega_{j-1}, \omega_{j-2}) - \upsilon_{j-1} \right)^2 . \quad (16)$$

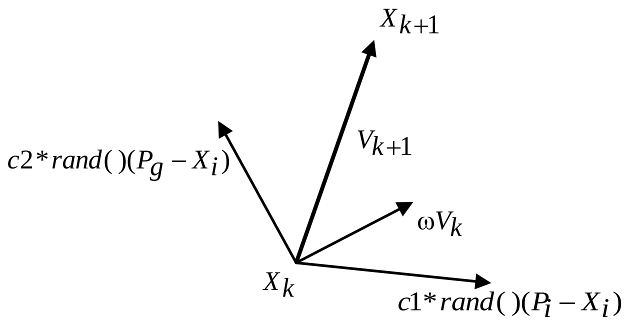

Fig. 3. The motion of one particle in one iteration.

If the value of $J_i(k)$ is better than the current $pbest_i$ of the particle, the $pbest_i$ value is replaced by the current value of the objective function and the local best position of the particle P_i is set to its current position X_i. Mathematically,

$$\begin{aligned}
&If\ J_i(k) < pbest_i \\
&\qquad pbest_i = J_i(k) \\
&\qquad\quad P_i = X_i \\
&else \\
&\qquad pbest_i = pbest_i \\
&\qquad\quad P_i = P_i .
\end{aligned} \tag{17}$$

Step.4. Updating global best memory: The global best value *gbest* is set to the best value of *pbest*. The corresponding local best position P_i is taken as the position of the global best position P_g.

Step. 5. Calculating the control voltage. In each iteration, the control voltage υ_k is calculated using the global best minimum position (the solution with minimum least square error that has been obtained so far in the whole population) as follows:

$$\upsilon_k = P_g^T \xi\left(r_j, \omega_j, \omega_{j-1}\right). \tag{18}$$

Step.6. Starting the next iteration by going to step 2.

6. Simulation

The parameters of a 1hp, 220 V, 550 rpm Dc motor are as follows:

$J = 0.068\,kgm^2$ $\quad K = 3.475\ NmA^{-1}$

$R_a = 7.56\ \Omega$ $\quad L_a = 0.055\ H$

$D = 0.03475\ Nms$ $\quad \mu = 1.95\ Nms^2$

$T = 40ms$

So, the discrete model of the motor is:

$$\begin{aligned}
\omega_{k+1} &= 1.2354\omega_k - 0.4864\omega_{k-1} + 0.0707\upsilon_k \\
&\quad - 0.369 sign(\omega_k)\omega_k^2 + 0.0545 sign(\omega_{k-1})\omega_{k-1}^2 .
\end{aligned} \tag{19}$$

Three membership functions are assigned for each input as shown in fig. 4 and they are given by:

$$\begin{aligned}
\mu_1(u_i) &= \frac{1}{1+e^{0.4(u_i+5)}} \\
\mu_2(u_i) &= e^{-\left(\frac{u_i}{5}\right)^2} \\
\mu_3(u_i) &= \frac{1}{1+e^{-0.4(u_i-5)}} ,
\end{aligned} \tag{20}$$

and the dimension of the search space is therefore 27.

The parameters of the PSO are chosen as follows:

Population size = 300 particle

Inertia weight = 0.729 and the acceleration constants = 1.429 (According to the clerk's construction factor).

Since the motor loading conditions are changing, the PSO algorithm needs to be adaptive. A lot of work has been done in the literature in the field of Adaptive Particle Swarm Optimization (APSO) techniques [22:27]. The chosen adaptation technique in this paper is re-randomizing the positions and velocities of 10 particles randomly chosen in each iteration. The moving window of data testing is chosen to be 50 samples.

The motor speed is required to follow a reference trajectory given by:

$$r_k = \sin(0.5\pi kT) + 0.5\sin(2/7\pi kT) . \tag{21}$$

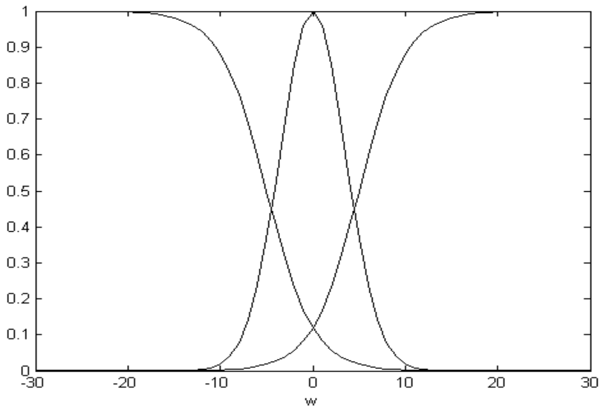

Fig. 4. The membership functions used for each input speed.

The nonlinear load torque is assumed to start at t = 9 sec. (after 225 iterations). Figs. 5, 6 show the reference and actual speed at two different runs (with two different random initial population). The figures show that the actual speed follows the reference trajectory which means that the identification and control processes are both successful and that the identifier and the controller adapted themselves after the introduction of the nonlinear load torque.

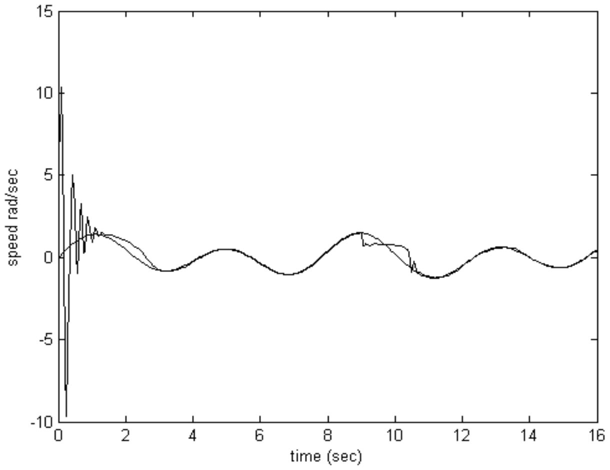

Fig. 5. Response of the motor during starting and application of nonlinear load in the first run.

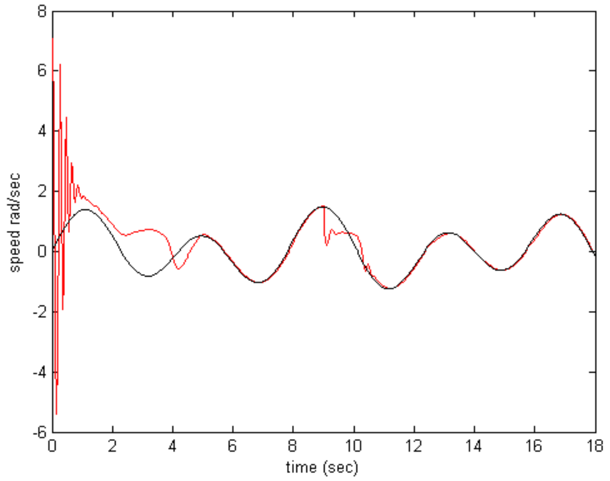

Fig .6. Response of the motor during starting and application of nonlinear load in the second run.

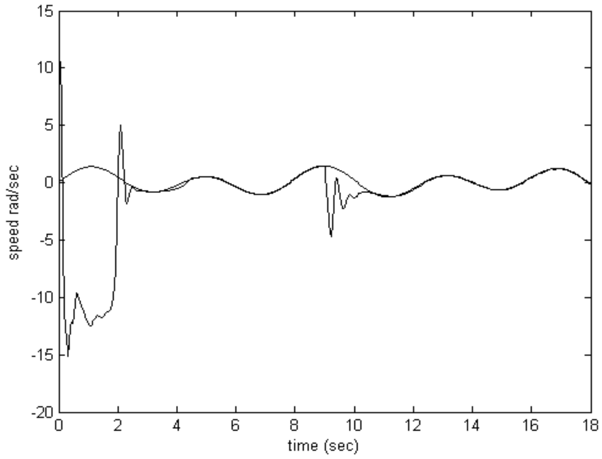

Fig. 7. Response of the motor during starting and application of constant load.

The same motor is now required to follow the same reference trajectory under a constant load torque of 5.5 Nm suddenly applied at t = 9 sec. (after 225 iterations) as shown in fig.7.

7. Conclusions

The particle swarm optimization algorithm has been used for the training of an adaptive inverse controller for a DC motor. The load torque was a nonlinear function of the speed. The nonlinear identification and control was done using a fuzzy neural network approximator and the training of the weights was done using particle swarm optimization. The motor speed has effectively tracked the required trajectory even with the sudden change in motor dynamics and that was due to the use of the adaptive particle swarm algorithm. The algorithm can also used to track the changes in motor parameters and can also be modified to be used in induction motor controllers. Unlike the conventional back-propagation technique, the training is done on-line using the adaptive particle swarm optimization.

References

[1] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization", Place Proc. IEEE Int. Conf. Neural Networks, pp. 1942-1948 (1995).

[2] D.W. Boeringer and D.H. Werner, "Particle Swarm Optimization Versus Genetic Algorithms for Phased Array Synthesis", IEEE Transactions on Antennas and Propagation, Vol. 52 (3), pp. 771-779 (2004).

[3] S. Naka, T. Genji, T. Yura and Y. Fukuyama", A Hybrid Particle Swarm Optimization for Distribution State Estimation," IEEE Transactions on Power Systems, Vol. 18 (1), pp. 60-68 (2003).

[4] S. Naka, Y. Fukuyama, T. Genji, and T. Yura, "Practical Distribution State Estimation Using Hybrid Particle Swarm Optimization," Proc. of IEEE Power Engineering Society Winter Meeting January 28 - February 1st, Columbus, Ohio, USA, pp. 1-6 (2001).

[5] B. Zhao, C.X. Guo and Y.J. Cao, "A Multiagent-Based Particle Swarm Optimization Approach for Optimal Reactive Power Dispatch," IEEE Transaction on Power Systems, Vol. 20 (2), pp. 1070-1078 (2005).

[6] J.S. Heo, K.Y. Lee and R. Garduno, "Multiobjective Control of Power Plants Using Particle Swarm Optimization Techniques", IEEE Transactions on Energy Conversion, Vol. 21 (2), pp. 552-561 (2006).

[7] C. Lu and C. Juang, "Evolutionary Fuzzy Control of Flexible AC Transmission System", IEE Proc.-Gener. Transm. Distrib., Vol. 152 (4), pp. 441-448 (2005).

[8] Z. Jun, F. Xin, Y. Huayong and Z. Jianmin, "A Particle Swarm Optimization Approach for PID Parameters in Hydraulic Servo Control System", Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21-23, Dalian, China, pp.7725-7729 (2006).

[9] P. Han, Y. Huang, Z. Jia, D. Wang and Yong-Ling Li, "Mixed H2 and H-infinity Optimal PID Control for Superheated Steam Temperature System Based on PSO Optimization", Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August, pp.960-964 (2005).

[10] A. Conradie, R. Miikkulainen and C. Aldrich , "Adaptive Control Utilizing Neural Swarming", Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA (2002).

[11] Y. Wang , K. Wang , J. Qu and Y. Yang, "Adaptive Inverse Control Based On Particle Swarm Optimization Algorithm," Proceedings of the IEEE International Conference on Mechatronics and Automation Niagara Falls, Canada, pp. 2169-2172 (2005).

[12] J. Ke, Q. Li, and J. Qian, "Particle Swarm Optimization Based Nonlinear Observer", Proceedings of the 5th World Congress on Intelligent Control 2004 and Automation, June 15-19, Hangzhou, P.R. China, pp. 1580-1583 (2004).

[13] S. Weerasooriya and M.A. El-Sharkawi, "Identification and Control of a DC Motor Using Back-Propagation Neural Networks", IEEE Transactions on Energy Conversion, Vol. 6 (4), December, pp. 663-669 (1991).

[14] U. Natarajan, V.M. Periasamy and R. Saravanan, "Application of Particle Swarm Optimization in Artificial Neural Network for the Prediction of Tool Life," Int. J. Adv. Manuf. Technol, pp. 871–876 (2007).

[15] D.H. Kim and J.H. Cho, "Optimal Learning of Fuzzy Neural Network Using Particle Swarm Optimization Algorithm," ICCAS2005 June 2-5, KINTEX, Gyeonggi-Do, Korea.

[16] H. Feng, "Hybrid Stages Particle Swarm Optimization Learning Fuzzy Modeling Systems Design", Tamkang Journal of Science and Engineering, Vol. 9 (2), pp. 167–176 (2006).

[17] L. Yi-jian, Z. Jian and W. Shu, "Parameter Estimation Of Cutting Tool Temperature Nonlinear Model Using PSO algorithm", Journal of Zhejiang University Science, pp. 1026-1029 (2005).

[18] T.C. Meng, T. Ray and P. Dhar , "Supplementary Material on Parameter Estimation using Swarm Algorithm", Preprint Submitted to Elsevier Science 19 March pp. 1-11 (2004).

[19] R.K. Ursem and P. Vadstrup, "Parameter Identification of Induction Motors using Stochastic Optimization Algorithms,"

Preprint submitted to Applied Soft Computing 13th August, pp. 1-26 (2003).
[20] C. Huang, C. Huang, and M. Wang, "A Particle Swarm Optimization to Identifying the ARMAX Model for Short-Term Load Forecasting", IEEE Transactions on Power Systems, Vol. 20 (2), pp. 1126-1133 (2005).
[21] A. J. Carlisle, "Applying the Particle Swarm Optimizer to Non-Stationary Environments," PhD Dissertation, Auburn University, December, pp. 1-145 (2002).
[22] T. Cai, F. Pan and J. Chen, "Adaptive Particle Swarm Optimization Algorithm", Proceedings of the 5th World Congress on Intelligent Control and Automation. June 15-19, Hang Zhou. P.R. China pp. 2245-2247 (2004).
[23] X. Cui, C.T. Hardin, R.K. Ragade, T.E. Potok1 and A.S. Elmaghraby, "Tracking non-Stationary Optimal Solution by Particle Swarm Optimizer", Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and FirstACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05).
[24] A. Carlisle and G. Dozier, "Tracking Changing Extrema with Adaptive Particle Swarm Optimizer", Proceedings of WAC 2002, Orlando, Florida, June 9-13 (2002).
[25] W. Zhang, Y. Liu and M. Clerc, "An Adaptive PSO Algorithm for Reactive Power Optimization", Proceedings of the 6th International Conference on advances in Power System Control, Operation and Management, APSCOM 2003, Hong Kong, November (2003).
[26] N. Iwasaki and K. Yasuda, "Adaptive Particle Swarm Optimization Using Velocity Feedback", International Journal of Innovative Computing, Information and Control ICIC International 2005, Vol. 1 (3), pp. 369-380 (2005).
[27] X. Xie, W. Zhang and Z. Yang, "Adaptive Particle Swarm Optimization on Individual Level", Proceedings of the international conference on signal processing (ICCP) Beijing, China, pp. 1215-1218 (2002).

Received June 25, 2007
Accepted July 18, 2007