IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
IEEJ Trans 2014; **9**: 494–501
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI:10.1002/tee.21998

Paper

An Improved Elman Neural Network Controller Based on Quasi-ARX Neural Network for Nonlinear Systems

Imam Sutrisno[*,**], Non-member
Mohammad Abu Jami'in[*,**], Non-member
Jinglu Hu[*a], Member

An improved Elman neural network (IENN) controller with particle swarm optimization (PSO) is presented for nonlinear systems. The proposed controller is composed of a quasi-ARX neural network (QARXNN) prediction model and a switching mechanism. The switching mechanism is used to guarantee that the prediction model works well. The primary controller is designed based on IENN using the backpropagation (BP) learning algorithm with PSO. PSO is used to adjust the learning rates in the BP process for improving the learning capability. The adaptive learning rates of the controller are investigated via the Lyapunov stability theorem. The proposed controller performance is verified through numerical simulation. The method is compared with the fuzzy switching and 0/1 switching methods to show its effectiveness in terms of stability, accuracy, and robustness. © 2014 Institute of Electrical Engineers of Japan. Published by John Wiley & Sons, Inc.

Keywords: Elman neural network, particle swarm optimization, quasi-ARX neural network, Lyapunov stability theorem

Received 10 December 2012; Revised 19 September 2013

1. Introduction

Neural predictive control (NPC) of nonlinear dynamical systems has attracted much attention and has developed significantly over the past few decades. Some researchers have successfully applied a neural network (NN) as the model identifier of the NPC system for controlling real processes. Zeng *et al*. [1] have established an NN predictive control scheme for studying the coagulation process of waste water treatment in a paper mill, and Wang *et al*. [2] have presented an adaptive NN model based predictive control for the air–fuel ratio of spark ignition (SI)[vnsn1] engines.

The application of NNs has been shown to be limited to static problems because of its feed-forward network structure. Recurrent NN (RNN), a modified model of NN, can be more suitable for real-time control applications than NN. Huang and Lewis [3] developed and analyzed an RNN predictive feedback control structure for a class of uncertain, continuous-time, nonlinear, and dynamic time-delay system in a canonical form. The improved Elman neural network (IENN) as a kind of RNN was proposed to improve the dynamic characteristics and convergence speed [4, 5]. A computed-torque controller based on an IENN approximation was proposed to deal with unmodeled, bounded disturbances and unstructured, unmodeled dynamics of a robot arm [4]. An IENN was developed to realize failure detection in a hydraulic servo system [5].

However, the major problems for these applications are (i) time-consuming computation and (ii) stability of the whole system. There are three objectives of this research. The first objective is to develop an IENN-PSO (particle swarm optimization) predictive controller based on quasi-ARX neural network (QARXNN) for nonlinear systems where the predictive controller is derived from the well-known generalized predictive performance criterion. This controller takes less time for computation, thereby overcoming a major obstacle of conventional NPCs which have large matrix inversion calculations and numerous matrix multiplications for finding the loner predictive control. The second objective is to guarantee the stability of the whole system with sufficient conditions that are established via Lyapunov stability theory. The third objective is to verify the proposed method by computer simulation results for nonlinear single-input single-output (SISO) systems and nonlinear governed systems. In order to guarantee that the QARXNN prediction model works well, a switching mechanism is established based on system input–output variables and prediction errors.

The rest of the paper is organized as follows: In Section 2, the considered system is given. In Section 3, a QARXNN prediction model is introduced based on an NN. Section 4 describes the IENN controller using PSO for learning rates adjustment, investigated by the Lyapunov stability theorem. Then, numerical simulations are carried out to show the effectiveness of the proposed method in Section 5, by comparison with fuzzy switching, 0/1 switching, and linear control. Finally, Section 6 presents the conclusions.

2. Problem Description

Consider a nonlinear, time-invariant, dynamical system with a single input and single output whose input–output relation is described by

$$y(t) = g(\varphi(t)) + v(t) \tag{1}$$

$$\varphi(t) = [y(t-1), ..., y(t-n), u(t-d), ..., u(t-m-d+1)]^T \tag{2}$$

where $y(t)$ denotes the output at time t (t = 1, 2, ...), $u(t)$ is the input, d is the known integer time delay, $\varphi(t)$ is the regression vector, and $v(t)$ is the system disturbance. $g(.)$ is a nonlinear function.

Now the following assumptions will be made:

[a] Correspondence to: Jinglu Hu. E-mail: jinglu@waseda.jp

[*] Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 808-0135, Japan

[**] Politeknik Perkapalan Negeri Surabaya, Kampus ITS Sukolilo, Surabaya 60111, Indonesia

ASSUMPTION 1.

(i) g(.) is a continuous function and, at a small region around $\varphi(\mathrm{t}) = 0$*, it is* C^{∞} *continuous;*

(ii) there is a reasonable unknown controller which may be expressed by $\mathrm{u(t)} = \tilde{\rho}(\tilde{\xi}(\mathrm{t}))$ *where* $\tilde{\xi}(\mathrm{t}) = [\mathrm{y(t)...y(t-n)u(t-1)...u(t-m)\ y^*(t+1)...y^*(t+1\text{-}l)}]^T$ *(*$\mathrm{y^*(t)}$ *denotes reference output).*

3. Quasi-ARX Neural Network

3.1. Regression form representation *Under Assumption* 1(i), the unknown nonlinear function $g(\varphi(t))$ can be Taylor-expanded in (1) over a small region around $\varphi(t) = 0$ and $y_0 = g(0)$:

$$y(t) = g(0) + g'(0)\varphi(t) + \frac{1}{2}\varphi^T(t)g''(0)\varphi(t) + \cdots + v(t) \quad (3)$$

$$y(t) = y_0 + \varphi^T(t)\theta(\varphi(t)) + v(t) \quad (4)$$

where

$$\theta(\varphi(t)) = (g'(0) + \frac{1}{2}\varphi^T(t)g''(0))^T \quad (5)$$

$$\theta(\varphi(t)) = [a_{1,t} \cdots a_{n,t}\ \ b_{0,t} \cdots b_{m-1,t}]^T. \quad (6)$$

In this case, the coefficients $a_{i,t} = a_i(\varphi(t))$ and $b_{j,t} = b_j(\varphi(t))$ are nonlinear functions of $\varphi(t)$.

With the aim of predicting $y(t)$ by the input–output data up to time $t-d$ in the model predictor, the expressions for the coefficients $a_{i,t}$ and $b_{j,t}$ are

$$a_{i,t} = \tilde{a}_{i,t} = \tilde{a}_i(\phi(t-d)), \quad (7)$$

$$b_{j,t} = \tilde{b}_{j,t} = \tilde{b}_j(\phi(t-d)) \quad (8)$$

where $\phi(t-d) = q^{-d}\phi(t)$, and q^{-1} is the backward shift operator, e.g., $q^{-1}u(t) = u(t-1)$. Then, $\phi(t)$ is a vector

$$\phi(t) = [y(t) \cdots y(t-n+1)\, u(t) \cdots u(t-m-d+2)]^T. \quad (9)$$

Then, two polynomials $A(q^{-1}, \phi(t))$ and $B(q^{-1}, \phi(t))$ based on the coefficients $a_{i,t}$ and $b_{j,t}$ are defined as

$$A(q^{-1}, \phi(t)) = 1 - a_{1,t}q^{-1} - \cdots - a_{n,t}q^{-n}, \quad (10)$$

$$B(q^{-1}, \phi(t)) = b_{0,t} + \cdots + b_{m-1,t}q^{-m+1}. \quad (11)$$

A similar linear ARX model can be developed:

$$A(q^{-1}, \phi(t))y(t) = y_0 + B(q^{-1}, \phi(t))q^{-d}u(t-1), + v(t). \quad (12)$$

For a system described by (12), a d-step predictor is given as in Ref. [6]:

$$\hat{y}(t+d \mid t, \phi(t)) = y_\phi + \alpha(q^{-1}, \phi(t))y(t) + \beta(q^{-1}, \phi(t))u(t) \quad (13)$$

where

$y_\phi = F(q^{-1}, \phi(t))y_0$,
$\alpha(q^{-1}, \phi(t)) = G(q^{-1}, \phi(t))$,
$\alpha(q^{-1}, \phi(t)) = \alpha_{0,t} + \alpha_{1,t}q^{-1} + \cdots + \alpha_{n-1,t}q^{-n+1}$,
$\beta(q^{-1}, \phi(t)) = F(q^{-1}, \phi(t))B(q^{-1}, \phi(t))$,
$\beta(q^{-1}, \phi(t)) = \beta_{0,t} + \beta_{1,t}q^{-1} + \cdots + \beta_{m+d-2,t}q^{-m-d+2}$,

and $G(q^{-1}, \phi(t)), F(q^{-1}, \phi(t))$ are unique polynomials satisfying

$$F(q^{-1}, \phi(t))A(q^{-1}, \phi(t)) = 1 - G(q^{-1}, \phi(t))q^{-d}. \quad (14)$$

Considering the linear ARX prediction model to be linear in the input variable $u(t)$, a controller can be obtained with the parameters directly from the predictor. To resolve this problem, an extra variable $x(t)$ is introduced to replace the variable $u(t)$ in $\phi(t)$ with an unknown nonlinear function $\rho(\xi(t))$, where

$$\xi(t) = [y(t) \cdots y(t-n_1+1)\, x(t+d) \cdots x(t-n_3+d+1)\, u(t-1) \cdots u(t-n_2)]^T \quad (15)$$

including the extra variable $x(t+d)$ as an element. Under Assumption 1(ii), the function $\rho(\xi(t))$ exists. Then we have a predictor represented as

$$\hat{y}(t+d \mid t, \xi(t)) = y_\xi + \alpha(q^{-1}, \xi(t))y(t) + \beta(q^{-1}, \xi(t))u(t) \quad (16)$$

where y_ξ is y_ϕ, where the variable $u(t)$ is replaced by $\tilde{\rho}(.)$.

3.2. Switching mechanism Consider that the predictor has a linear part and a nonlinear part whose coefficients depend on $\xi(t)$. Determine a switching function $\chi(t)$ as introduced by Wang *et al.* [7, 8]; when $\chi(t)$=1, the nonlinear part is chosen with the linear part to predict the nonlinear system. Otherwise, if $\chi(t)$=0, then the nonlinear part is neglected for some reasons. Now we have the new d-step-ahead predictor described by

$$\hat{y}(t+d \mid t, \xi(t)) = y_{\xi,\chi} + \alpha(q^{-1}, \xi(t), \chi(t))y(t) + \beta(q^{-1}, \xi(t), \chi(t))u(t), \quad (17)$$

$$\Psi(t) = [1\ y(t) \cdots y(t-n+1)u(t) \cdots u(t-m-d+2)]^T, \quad (18)$$

$$\Theta(\xi(t), \chi(t)) = [y_{\xi,\chi}\ \alpha_{0,t} \cdots \alpha_{n_y-1,t}\ \beta_{0,t} \cdots \beta_{n_u+d-2,t}]^T. \quad (19)$$

Finally, we get the ARX-like macro-model expression given by

$$\hat{y}(t+d \mid t, \xi(t)) = \Psi^T(t)\Theta(\xi(t), \chi(t)). \quad (20)$$

The QARXNN with switching law is described by

$$\hat{y}(t+d \mid t, \xi(t)) = \Psi^T(t)\aleph(\xi(t), \chi(t), \Omega) \quad (21)$$

where $\aleph(.,.,.)$ is the generalized three-layer NN with the structure using n input nodes, M sigmoid hidden nodes, and $n+1$ linear output nodes. The number of input nodes is $N = \dim(\xi(t)) = n + m$, and the number of output nodes is equal to $\dim(\Psi(t)) = N + 1$. The three-layer NN is defined as

$$\aleph(\xi(t), \chi(t), \Omega) = \theta + \chi(t)W_2\Gamma(W_1 + B) \quad (22)$$

where $\Omega = W_1, W_2, B, \theta$ is the parameters set of the NN. We can divide the above into two classes: the linear part θ, and nonlinear parts W_1, W_2, and B. Now, θ is updated as

$$\hat{\theta}(t) = \hat{\theta}(t-d) + \frac{a(t)\Psi(t-d)e_1(t)}{1 + \Psi(t-d)^T\Psi(t-d)} \quad (23)$$

where $\hat{\theta}(t)$ is the estimate of θ at time instant t. And $a(t) = 1$ if $|e_1(t)| > 2\Delta$; otherwise $a(t) = 0$ where $e_1(t)$ is the error of the linear part and is defined as follows:

$$e_1(t) = y(t-d) - \Psi(t)^T\hat{\theta}(t). \quad (24)$$

The error of the nonlinear part parameters is adjusted by the BP algorithm and defined by

$$e_2(t) = y(t+d) - \Psi(t)^T\hat{\theta}(t) - \Psi(t)^T W_2(t)\Gamma(W_1(t)\xi(t) + B(t)) \quad (25)$$

where $W_1(t), W_2(t)$, and $B(t)$ are the estimates of W_1, W_2, and B at time instant t, respectively. Finally, the switching law is defined

by the following function:

$$J_i(t) = \sum_{l=k}^{t} \frac{a_i(l)(\|e_i(l)\|^2 - 4\Delta^2)}{2(1 + a_i(l)\Psi(l-k)^T\Psi(l-k))} + c \sum_{l=t-N+1}^{t} (1 - a_i(l)\,\|e_i(l)\|^2), \quad i = 1, 2 \tag{26}$$

where N is an integer and $c \geq 0$ is a predefined constant. The expression of the switching law $\chi(t)$ is based on the switching criterion function $\chi(t) = 1$ if $J_1(t) > J_2(t)$; otherwise $\chi(t) = 0$. By comparing $J_1(t)$ and $J_2(t)$, we can decide when the nonlinear part can be neglected. If $J_1(t) > J_2(t)$, the nonlinear part is added; else, only use the linear part to identify. If $J_2(t)$ is larger than $J_1(t)$, the nonlinear parameter should be reset and switch to linear part so the prediction model is working well. The switching mechanism was used for making sure that the prediction model was working well. In the linear case, the system is surely stable but in the nonlinear case the system may be overfitting, so we need to switch to the linear case.

4. Design of Controller

4.1. Controller scheme The purpose of IENN-PSO is to design a control law and an updating law for the primary controller parameters, such that the system output y follows the input reference signal and the closed-loop dynamic performance of the system follows the predictive model. Figure 1 shows the IENN-PSO scheme. A switching mechanism is employed to improve the performance of the QARXNN prediction model.

4.2. Improved Elman neural network controller design An IENN performs better than the common Elman neural network (ENN) because the feedback of the output layer is taken into account, so better learning efficiency can be obtained. In a common ENN, the hidden layer neurons are fed by the outputs of the context neurons and the input neurons. Because a common ENN only employs the hidden context nodes to disperse the message, it has low learning speed and convergence precision. Figure 2 shows the structure of the proposed IENN including the input layer (i layer), the hidden layer (j layer), the context layer (r layer), and the output layer (o layer) with two inputs and one output [9]. The basic function and the signal propagation of each layer are introduced in the following:

Layer 1 (input layer): the node input and the node output are represented as

$$X_i(k) = f_i(\mathrm{net}_i) = \mathrm{net}_i = e_i(k) \tag{27}$$

where $e_i(k)$ and $X_i(k)$ are the input and the output of the input layer, respectively, and k represents the kth iteration.

Layer 2 (hidden layer): the node input and the node output are represented as

$$X_j(k) = S(\mathrm{net}_j) \tag{28}$$

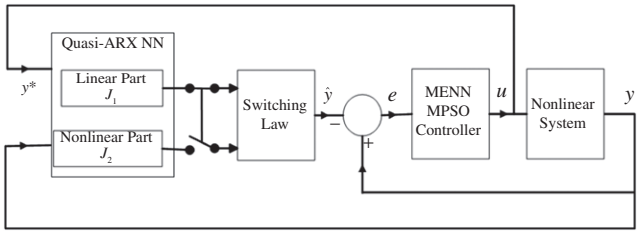

Fig. 1. IENN-PSO structure by neural network

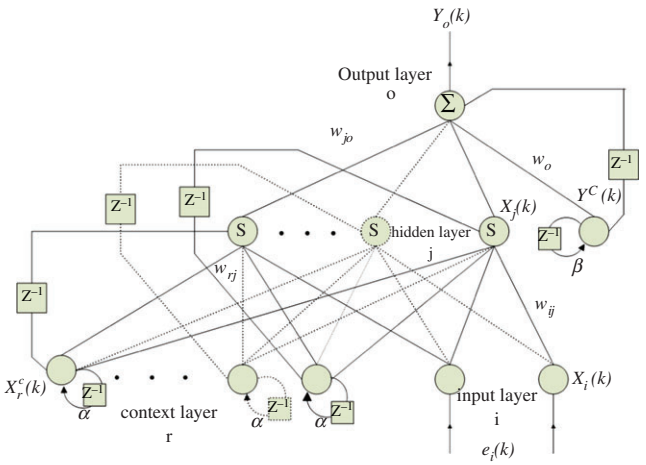

Fig. 2. Structure of the improved Elman neural network

$$\mathrm{net}_j = \sum_i w_{ij} X_i(k) + \sum_r w_{rj} X_r^c(k) \tag{29}$$

where $X_j(k)$ and net_j are the output and the input of the hidden layer, w_{ij} and w_{rj} are the connective weights of input neurons to hidden neurons and context neurons to hidden neurons, respectively, X_r^c is the output of the context layer, and $S(X)$ is sigmoid function, that is, $S(X) = 1/(1 + e^{-X})$.

Layer 3 (context layer): the node input and the node output are represented as

$$X_r^c(k) = \gamma X_r^c(k-1) + X_j(k-1) \tag{30}$$

where $0 \leq \gamma < 1$ is the self-connecting feedback gain.

Layer 4 (output layer): the node input and the node output are represented as

$$Y_o(k) = f(\mathrm{net}_o(k)) = \mathrm{net}_o(k), \tag{31}$$

$$\mathrm{net}_o(k) = \sum_j w_{jo} X_j(k) + w_o Y^c(k), \tag{32}$$

$$Y^c(k) = \zeta Y^c(k-1) + Y_o(k-1) \tag{33}$$

where $Y_o(k)$ is the output of the IENN and also the control effort of the proposed controller, $Y^c(k)$ is the output of the output feedback neuron, $0 \leq \zeta < 1$ is the self-connecting feedback gain, and w_{jo} and w_o are the connective weights of the hidden neurons to the output neurons and the output feedback neuron to the output neuron, respectively.

4.3. Learning algorithm The learning algorithm of the IENN is referred to as the *BP learning rule method*. To describe it using the supervised gradient decent method, first the energy function E is defined as

$$E = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}e \tag{34}$$

where y and $\hat{y}$ represent the output of the system and output of the prediction model, respectively, and e denotes the tracking error. Then the learning algorithm is defined as follows:

Layer 4: the error term to be propagated is given by

$$\delta_o = -\frac{\partial E}{\partial Y_o(k)} = -\frac{\partial E}{\partial e}\frac{\partial e}{\partial Y_o(k)} = -\frac{\partial E}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial Y_o(k)}, \tag{35}$$

$$\begin{aligned}\Delta w_o &= -\eta_1 \frac{\partial E}{\partial w_o} \\ &= -\eta_1 \frac{\partial E}{\partial Y_o(k)} \frac{\partial Y_o(k)}{\partial \mathtt{net}_o(k)} \frac{\partial \mathtt{net}_o(k)}{\partial w_o} \\ &= \eta_1 \delta_o Y^c(k), \end{aligned} \tag{36}$$

$$\begin{aligned}\Delta w_{jo} &= -\eta_2 \frac{\partial E}{\partial w_{jo}} \\ &= -\eta_2 \frac{\partial E}{\partial Y_o(k)} \frac{\partial Y_o(k)}{\partial \mathtt{net}_o(k)} \frac{\partial \mathtt{net}_o(k)}{\partial w_{jo}} \\ &= \eta_2 \delta_o X_j(k). \end{aligned} \tag{37}$$

The connective weights w_o and w_{jo} are updated according to the following equations:

$$w_o(k+1) = w_o(k) + \Delta w_o, \tag{38}$$

$$w_{jo}(k+1) = w_{jo}(k) + \Delta w_{jo} \tag{39}$$

where the factors η_1 and η_2 are the learning rates.

Layer 3: using the chain rule, the update law of w_{rj} is

$$\begin{aligned}\Delta w_{rj} &= -\eta_3 \frac{\partial E}{\partial w_{rj}} \\ &= -\eta_3 \frac{\partial E}{\partial Y_o(k)} \frac{\partial Y_o(k)}{\partial \mathtt{net}_o(k)} \frac{\partial \mathtt{net}_o(k)}{\partial X_j(k)} \frac{\partial X_j(k)}{\partial w_{rj}} \\ &= \eta_3 \delta_o w_{jo} X_j(k)[1 - X_j(k)] X_r^c(k), \end{aligned} \tag{40}$$

$$w_{rj}(k+1) = w_{rj}(k) + \Delta w_{rj}. \tag{41}$$

Layer 2: using the chain rule, the update law of w_{ij} is

$$\begin{aligned}\Delta w_{ij} &= -\eta_4 \frac{\partial E}{\partial w_{ij}} \\ &= -\eta_4 \frac{\partial E}{\partial Y_o(k)} \frac{\partial Y_o(k)}{\partial \mathtt{net}_o(k)} \frac{\partial \mathtt{net}_o(k)}{\partial X_j(k)} \frac{\partial X_j(k)}{\partial w_{ij}} \\ &= \eta_4 \delta_o w_{jo} X_j(k)[1 - X_j(k)] X_i(k), \end{aligned} \tag{42}$$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} \tag{43}$$

where the factors η_3 and η_4 are the learning rates, and $(\eta_1, \eta_2, \eta_3, \eta_4)$ will be optimized by the PSO algorithm.

4.4. Learning rate adjustment using PSO

PSO is an algorithm based on a group of flying birds to simulate the behavior of a swarm as a simplified collective system.

To further improve the learning ability, the PSO algorithm is adopted for tuning the learning rates η_1, η_2, η_3, and, η_4 of the IENN [10-12].

Step 1: initialization

With $R_i^d = [R_i^1, R_i^2, R_i^3, R_i^4]$ for learning rates (η_1, η_2, η_3, and η_4), the population size is set to $P = 15$, and the dimension of particle is set to d=4. The parameters need to be optimized with minimum and maximum ranges.

Step 2: Define location and velocity

The initial location $R_i^d(N)$ and the velocities $V_i^d(N)$ of all particles are randomly generated in the search space. The elements in vector $R_i^d(N)$ are randomly generated by

$$R_i^d \sim U[\eta_{\min}^d, \eta_{\max}^d] \tag{44}$$

where $U[\eta_{\min}^d, \eta_{\max}^d$ designates the result of a uniformly distributed random variable ranging over the known lower and upper bounded values $\eta_{\min}$ and $\eta_{\max}$ of the learning rate.

Step 3: Update the velocity

Every particle in the swarm is updated using (45). The velocity update law is described as

$$\begin{aligned}V_i^d(N+1) = wV_i^d(N) &+ c_1 \cdot r_1 \cdot (pbest_i^d - R_i^d(N)) \\ &+ c_2 \cdot r_2 \cdot (gbest_i^d - R_i^d(N)) \end{aligned} \tag{45}$$

where the pseudo-random sequences $r_1\ U(0,1)$ and $r_2\ U(0,1)$ are used to simulate the stochastic nature of the algorithm. For all dimensions d, let $R_i^d, pbest_i^d$ be the current position and current personal best position.

Step 4: Update position

The new velocity is then added to the current position of the particle to find its next position by

$$R_i^d(N+1) = R_i^d(N) + V_i^d(N+1)\ i = 1, \cdots, P. \tag{46}$$

Step 5: Update *pbests* If the current position of a particle is located within the analysis space and does not intrude the territory of other *gbests*, the objective function of the particle is evaluated. The fitness of each particle is calculated by

$$FIT = \frac{1}{0.1 + \mathtt{abs}(\hat{y} - y)}. \tag{47}$$

By using (46), a gradually increasing fitness function can be obtained.

Step 6: Update *gbests*

The *gbest* is replaced by the best *pbest* among the particles. Each particle R_i^d memorizes its own fitness value and chooses the maximum one that is the best so far, defined as $pbest_i^d$ and the maximum vector in the population $pbest_i^d = [pbest_i^1, pbest_i^2, pbest_i^3, \cdots pbest_p^d]$, is obtained.

Step7: Check convergence

Steps 3–6 are repeated until the best fitness value for the *gbest* is obviously improved or a set count of the generation is reached. Finally, the highest fitness value $gbest_i^d$ is the optimal learning rate $(\eta_1, \eta_2, \eta_3, \eta_4)$ of IENN. The acceleration coefficients c_1 and c_2 can be used to control the moved distance of a particle in a single iteration, typically set to 2.0 for simplicity. The inertia weight w in (48) is used to control the convergence behavior of the PSO. Small values of w result in a more rapid convergence usually on a suboptimal position, while large values may cause divergence. In general, the inertia weight w is set according to

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{\eta_{\max}} \cdot \eta \tag{48}$$

where $\eta_{\max}$ is the maximum number of iterations, and η is the current iteration count. The maximum values of the inertia weights are $w_{\max} = 0.5$ and $w_{\min} = 0.3$, respectively.

4.5. Lyapunov stability analysis for the whole system

The following theorem states that the IENN-PSO controller is also convergent based on Lyapunov stability theory.

Theorem. Let the weights of IENN are updated along with (36), (37), (40), and (42). Then the convergence of the IENN (33) is guaranteed if the learning rate η_i is chosen to satisfy

$$\eta_i = \frac{\beta \left[\sum_{o=1}^{m_y} e_o(k) \frac{\partial \hat{y}_o(k)}{\partial \varpi} \right]^T \left[\sum_{o=1}^{m_y} e_o(k) \frac{\partial \hat{y}_o(k)}{\partial \varpi} \right]}{\left[\sum_{o=1}^{m_y} \frac{\partial \hat{y}_o(k)}{\partial \varpi} \right]^T \left[\sum_{o=1}^{m_y} e_o(k) \frac{\partial \hat{y}_o(k)}{\partial \varpi} \right]} \tag{49}$$

where i=1,2,3,4; the conditions are $0 < \beta < 2$ and $\varpi = \left[w_{11}^o \cdots w_{m_y n_h}^o\ w_{11}^i \cdots w_{n_h n_i}^i\ w_1^j \cdots w_{n_h}^j \right]^T$. IENN has n_i inputs, n_h hidden units and m_y output variables.

Proof. Let a Lyapunov function candidate be chosen as $l(k) = \sum_{o=1}^{m_y} e_o^2(k)$, and let $\delta l(k) = l(k+1) - l(k)$ and $\delta e_o(k) \equiv e_o(k+1) - e_o(k)$. Then

$$\delta l(k) = 2\sum_{o=1}^{m_y} e_o(k)\delta e_o(k) + \sum_{o=1}^{m_y}(\delta e_o(k))^2. \tag{50}$$

By the method in Ref. [13], $\delta e_o(k)$ can be represented by $\delta e_o(k) = [\partial e_o(k)/\partial \varpi]^T \delta\varpi$, while $\delta\varpi = -\eta_i(\partial y_o(k)/\partial\varpi)$, and (50) becomes

$$\begin{aligned}\delta l(k) &= 2\sum_{o=1}^{m_y} e_o(k)\left[\frac{\partial e_o}{\partial \varpi}\right]^T \delta\varpi + \sum_{o=1}^{m_y}\left(\left[\frac{\partial e_o}{\partial\varpi}\right]^T \delta\varpi\right)^2 \\ &= -2\eta_i\left[\sum_{o=1}^{m_y} e_o(k)\frac{\partial \hat{y}_o(k)}{\partial\varpi}\right]^T\left[\sum_{o=1}^{m_y} e_o(k)\frac{\partial\hat{y}_o(k)}{\partial\varpi}\right] \\ &\quad + \eta_i^2\left(\left[\sum_{o=1}^{m_y}\frac{\partial\hat{y}_o(k)}{\partial\varpi}\right]^T\left[\sum_{o=1}^{m_y} e_o(k)\frac{\partial\hat{y}_o(k)}{\partial\varpi}\right]\right)^2. \end{aligned}\tag{51}$$

To ensure that this selected learning rate is inside the stable region, we set the learning rate η_i as in (49) and then have $\delta l(k) < 0$, which shows that the IENN-PSO is convergent. This completes the proof of the theorem.

5. Simulation Result

In order to study the behavior of the proposed control method, a numerical simulation is described in this section.

Example 1:

The nonlinear SISO system to be controlled is described by

$$\begin{aligned} y(t) &= \frac{\exp(-y^2(t-2)) * y(t-1)}{1+u^2(t-3)+y^2(t-2)} \\ &\quad + \frac{\exp(-0.5*(u^2(t-2)+y^2(t-3)))*y(t-2)}{1+u^2(t-2)+y^2(t-1)} \\ &\quad + \frac{\sin(u(t-1)*y(t-3))*y(t-3)}{1+u^2(t-1)+y^2(t-3)} \\ &\quad + \frac{\sin(u(t-1)*y(t-2))*y(t-4)}{1+u^2(t-2)+y^2(t-2)} + u(t-1). \end{aligned}\tag{52}$$

The desired output in this example is a piecewise function:

$$y^*(t) = \begin{cases} 0.4493y^*(t-1) \\ +0.57r(t-1), & t\in[1,100]\cup[151,200] \\ 0.7\mathtt{sign}(0.4493 & \\ y^*(t-1) & \\ +0.57r(t-1)), & t\in[101,150] \end{cases} \tag{53}$$

where $r(t) = 1.2\sin(2\pi t/25)$.

We use the following QARXNN prediction model to identify the system:

$$\begin{aligned} y_1(t+d|t,\xi(t)) &= \Psi(t)^T\theta \\ &\quad + \chi(t)\Psi(t)^T\cdot W_2\Gamma(W_1\xi(t)+B). \end{aligned}\tag{54}$$

In the nonlinear part, the NN has 1 hidden layer and 20 hidden nodes, and the other parameters are set as m=4, n=3, and d=1. First, the QARXNN model can be trained offline by the hierarchical training algorithm as in Ref. [6].

Figure 3 shows the performance when the IENN controller with PSO is used. In Fig. 3, the black dotted line is the desired output, the red solid line denotes proposed method IENN with PSO control output $y(t)$, and the blue dashed line shows the 0 or 1 switching control output $y_0(t)$. Obviously, the control output

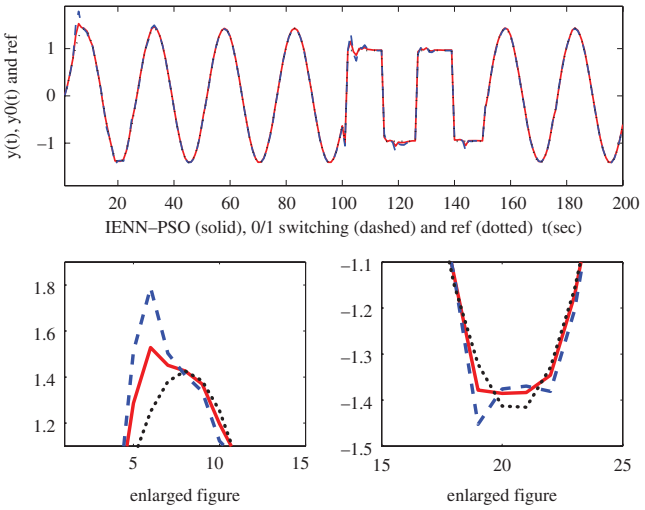

Fig. 3. Comparison result of the proposed method with 0/1 switching method for Example 1

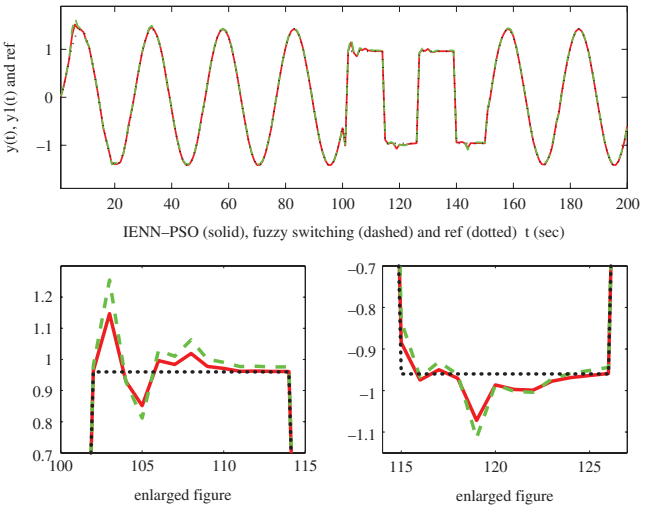

Fig. 4. Comparison result of the proposed method with fuzzy switching method for Example 1

with the proposed method is nearly consistent with the desired output most of the time.

In Fig. 4, the black dotted line is the desired output, the red solid line denotes the proposed method IENN with PSO control output $y(t)$, and the green dashed line shows the fuzzy switching control output $y_1(t)$. We can see that the proposed IENN with PSO control can perform better than fuzzy switching controller and also better than 0/1 switching controller in three points: (i) convergence speed, (ii) stability, and (iii) adaptability.

The switching sequence is presented in Fig. 5, where 1 is the model with nonlinear part and 0 is the model without nonlinear part. Even though the model with nonlinear part can control very well, it degrades sometimes and the model with only the linear part has to work until the nonlinear part can recover. Therefore, the linear part will work all the time, but the nonlinear part will work under the switching sequence. The convergence characteristic of the errors is shown in Fig. 6. A better nonlinear system controller effect can be seen for the proposed algorithm. Table I gives a comparison of the errors of the three methods. The error of the proposed IENN-PSO control system is smaller than that in the other methods.

Obviously, IENN-PSO controller gives better performance than the other controllers. For additional comparison, the convergence

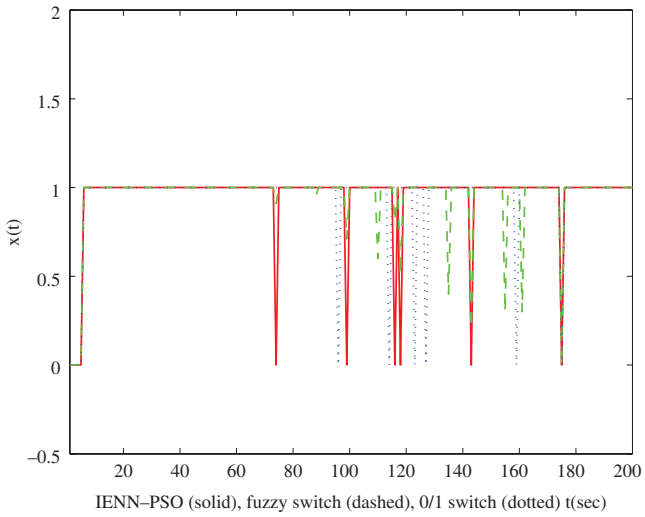

Fig. 5. Switching sequence of the proposed method compared to other methods for Example 1

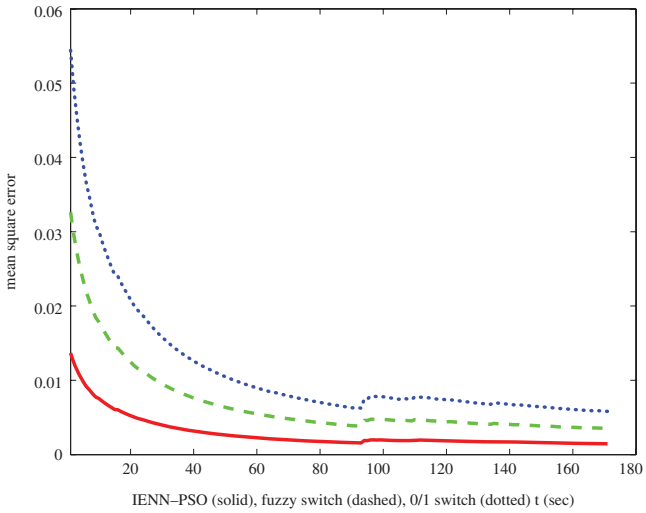

Fig. 6. Convergence characteristics of the errors for Example 1

speeds of the different methods are given in Fig. VI, and classified in Table II. The proposed IENN-PSO control method gives better accuracy and also faster convergence than the other methods.

Example 2:

The system is a nonlinear one governed by

$$y(t) = f[y(t-1), y(t-2), y(t-3), u(t-1), u(t-2)] \quad (55)$$

where

$$f[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2}. \quad (56)$$

The desired output in this example is a piecewise function:

$$y^*(t) = \begin{cases} 0.6y^*(t-1) \\ +r(t-1), & t \in [1, 100] \cup [151, 200] \\ 0.7\mathtt{sign}(0.4493 \\ y^*(t-1) \\ +0.57r(t-1), & t \in [101, 150] \end{cases} \quad (57)$$

where $r(t) = \sin(2\pi t/25)$.

In the nonlinear part, the NN has 1 hidden layer and 20 hidden nodes and the other parameters are set as m=4, n=3, and d=1.

Table I. Comparison result of the errors

Method	Mean of errors	Variance of errors
IENN + PSO control method	0.00015	0.0033
Fuzzy switching control method	0.00033	0.0051
0/1 switching control method	0.0061	0.0365

Table II. Classification result for different methods

Method	Iteration	CPU time	Accuracy
IENN + PSO control method	183	0.56	98.38
Fuzzy switching control method	254	0.98	97.14
0/1 switching control method	273	1.82	95.87

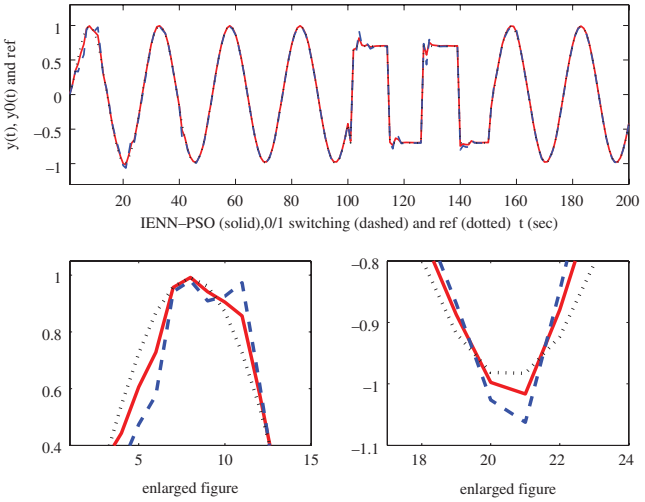

Fig. 7. Comparison result of the proposed method with 0/1 switching method for Example 2

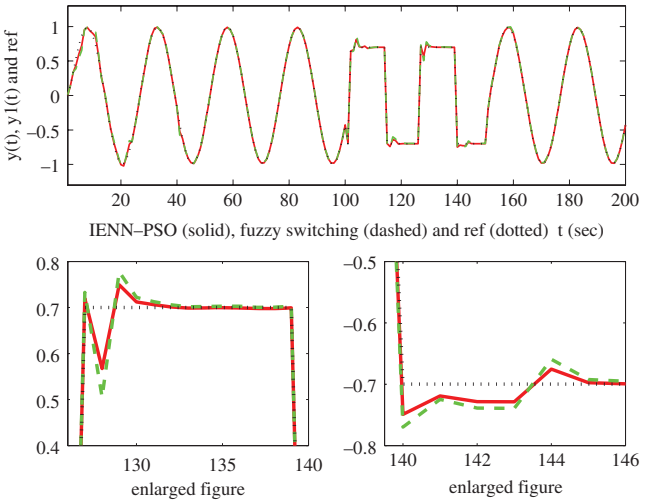

Fig. 8. Comparison result of the proposed method with fuzzy switching method for Example 2

Figure 7 shows the performance when the IENN controller with PSO is used.

Figure 7 gives the results of Example 2 whose marks are the same as in Example 1. In Fig. 7, the output of the proposed IENN with PSO control almost coincides with the desired output. It can also be seen that the 0/1 switching control results have some wobble at the last half-time.

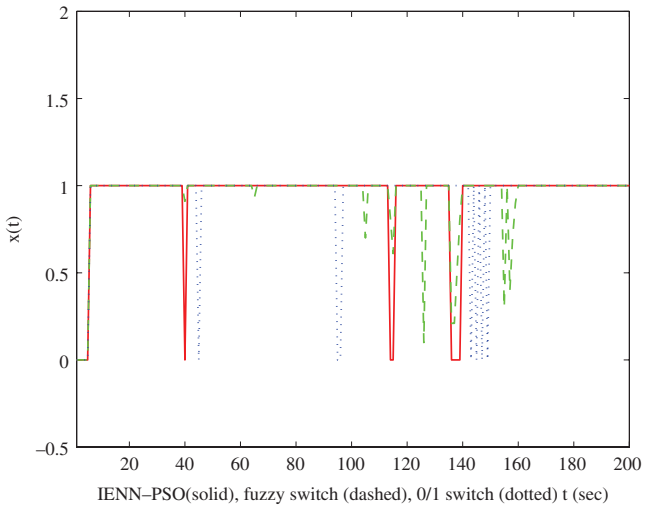

Fig. 9. Switching sequence of the proposed method compared to other methods for Example 2

Table III. Comparison result of the errors

Method	Mean of errors	Variance of errors
IENN + PSO control method	−0.0035	0.0013
Fuzzy switching control method	−0.0044	0.0032
0/1 Switching control method	−0.0051	0.0067

Table IV. Classification results for the different methods

Method	Iteration	CPU time	Accuracy
IENN + PSO control method	153	0.36	97.38
Fuzzy switching control method	194	0.58	95.14
0/1 Switching control method	253	1.02	93.87

In Fig. 8, we can see that the proposed IENN-PSO control method performs better than the fuzzy switching controller. The switching sequence is presented in Fig. 9. A similar conclusion also can be drawn from the convergence characteristic of the errors is shown in Fig. 10. Table III gives the comparison of the errors in the three methods. The error of the proposed IENN-PSO control system is smaller than that of the other methods. Obviously, the IENN-PSO controller gives better performance than other controllers. For additional comparison, the convergence speed of the different methods are given in Fig. 10 and classified in Table IV. The proposed IENN-PSO control method gives better accuracy, and also has a faster convergence rate than the other methods.

6. Conclusion

This study has successfully demonstrated the effectiveness of the proposed IENN-PSO algorithm based on the QARXNN prediction model. First, the principles of the QARXNN prediction model were derived. Then, the network structure and theoretical bases of the proposed IENN-PSO were chosen to adapt the learning rates in the BP process to replace the traditional trial-and-error method. Finally, the control performance of the proposed IENN-PSO based on the QARXNN prediction model was confirmed by some simulation and experimental results. The main contributions of this study are as follows: (i) the successful development of an IENN controller; (ii) the successful adoption of a PSO algorithm to tune the learning rates in the BP process; and (iii) the successful application of the IENN-PSO based on the QARXNN prediction model to control a nonlinear system with robust control performance.

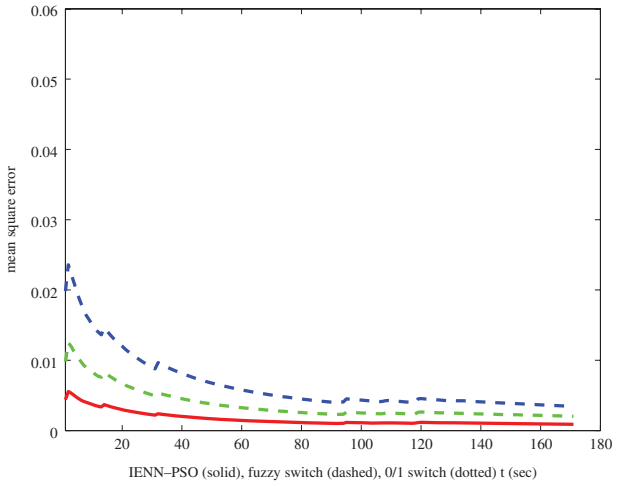

Fig. 10. Convergence characteristics of the errors for Example 2

References

(1) Zeng GM, Qin XS, He L, Huang GH, Liu HL, Lin YP. A neural network predictive control system for paper mill wastewater treatment. *Engineering Application of Artificial Intelligent* 2003; **16**:121–129.
(2) Wang SW, Yu DL, Gomm JB, Page GF, Douglas SS. Adaptive neural network model based predictive control for air-fuel ratio of SI engines. *Engineering Application of Artificial Intelligent* 2006; **19**:189–200.
(3) Huang JQ, Lewis FL. Neural-network predictive for nonlinear dynamics systems with time-delay. *IEEE Transactions on Neural Networks* 2003; **914(2)**:377–389.
(4) Mbede JB, Huang X, Wang M. Robust neuro-fuzzy-sensor-based motion control among dynamic obstacles for robot manipulators. *IEEE Transactions on Fuzzy Systems* 2003; **11(2)**:249–261.
(5) Liu H, Wang S, Ouyang P. Fault diagnosis based on improved Elman neural network for ahydraulic servo system. Proceedings of International Conference on Robotics, Automation and Mechatronics, Bangkok, Thailand, 2006.
(6) Hu J, Hirasawa K. A method for applying neural networks to control of nonlinear systems. In *Neural Information Processing: Research and Development*. Rajapakse JC, Wang L (eds). Springer: Berlin, Germany; 2004; 351–369.
(7) Wang L, Cheng Y, Hu J. A Quasi-ARX neural network with switching mechanism to adaptive control of nonlinear systems. *SICE Journal of Control, Measurement, and System Integration* 2010; **3(4)**: 246–252.
(8) Wang L, Cheng Y, Hu J. Stabilizing switching control for nonlinear system based on Quasi-ARX model. *IEEJ Transactions on Electrical and Electronic Engineering* 2012; **7(4)**:390–396.
(9) Lin F-J, Teng L-T, Chu H. Modified Elman neural network controller with improved particle swarm optimisation for linear synchronous motor drive *IET Electric Power Applications* 2008; **2(3)**:201–214.
(10) Jacob R, Yahya RS. Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation* 2004; **52(2)**: 397–407.
(11) Lin WM, Hong CM. Intelligent approach to maximum power point tracking control strategy for variable-speed wind turbine generation system. *Energy* 2010; **35(6)**:2440–2447.
(12) Lin WM, Hong CM. A new Elman neural network-based control algorithm for adjustable-pitch variabel-speed wind-energy conversion systems. *IEEE Transactions on Power Electronics* 2011; **26(2)**:473–481.
(13) Ku CC, Lee KY. Diagonal recurrent neural networks for dynamical system control. *IEEE Transactions on Neural Networks* 1995; **6(1)**:144–156.

Imam Sutrisno (Non-member) was born in Rembang, Indonesia, on January 16, 1975. He received the B.S. and M.S. degrees in Electrical Engineering (control systems) from the Institute of Technology 10th Nopember Surabaya, Indonesia, in 1998 and 2005, respectively. He is currently pursuing the Ph.D. degree at Waseda University, Japan. His research interests include neural predictive controllers, adaptive neurofuzzy system identification, and control systems design. Mr. Imam is a student member of the IEEE.

Mohammad Abu Jami'in (Non-member) was born in Pasuruan, Indonesia, on May 30, 1975. He received the B.S. and M.S. degrees from the Institute of Technology 10th Nopember Surabaya, Indonesia, in 2000 and 2008, respectively. He is currently pursuing the Ph.D. degree at Waseda University, Japan. His research interests include Lyapunov stability, system identification, and control systems design. Mr. Jami'in is a student member of the IEEE.

Jinglu Hu (Member) was born in Fujian, China. He received the M.Sc. degree in Electronic Engineering from Sun Yat-Sen University, Guangzhou, China, in 1986, and the Ph.D. degree in Computer Science and System Engineering from Kyushu Institute of Technology, Iizuka, Japan, in 1997. From 1986 to 1988, he worked as a Research Associate and from 1988 to 1993 as a Lecturer at Sun Yat-Sen University. From 1997 to 2003, he worked as a Research Associate at Kyushu University, Japan. From 2003 to 2008, he worked as an Associate Professor at the Graduate School of Information, Production and Systems, Waseda University, Japan, and currently is a Professor. His research interests include computational intelligence such as neural networks and genetic algorithms, and their applications to system modeling and identification, bioinformatics, time-series prediction, and so on. Prof. Hu is a member of the IEEE, SICE, and IEICE.