SICE Journal of Control, Measurement, and System Integration, Vol. 9, No. 2, pp. 070–077, March 2016

Special Issue on SICE Annual Conference 2015

A Self-Organizing Quasi-Linear ARX RBFN Model for Nonlinear Dynamical Systems Identification

Imam SUTRISNO [*,**], Mohammad ABU JAMI'IN [*,**], Jinglu HU [*], and Mohammad HAMIRUCE MARHABAN [***]

Abstract: The quasi-linear ARX radial basis function network (RBFN) model has shown good approximation ability and usefulness in nonlinear system identification and control. It has an easy-to-use structure, good generalization and strong tolerance to input noise. In this paper, we propose a self-organizing quasi-linear ARX RBFN (QARX-RBFN) model by introducing a self-organizing scheme to the quasi-linear ARX RBFN model. Based on the active firing rate and the mutual information of RBF nodes, the RBF nodes in the quasi-linear ARX RBFN model can be added or removed, so as to automatically optimize the structure of the quasi-linear ARX RBFN model for a given system. This significantly improves the performance of the model. Numerical simulations on both identification and control of nonlinear dynamical system confirm the effectiveness of the proposed self-organizing QARX-RBFN model.

Key Words: nonlinear dynamical system, system identification and control, quasi-linear ARX model, self-organization, radial basis function network.

1. Introduction

Neural networks (NNs), neuro-fuzzy networks (NFNs) and radial basis function networks (RBFNs) have shown a highly sophisticated capacity for nonlinear function approximation. However, from a user's point of view, these models are basically viewed as vehicles for adjusting the fit of the data, and rarely reflect physical considerations in the system [1],[2]. RBFNs are used in nonlinear system modeling and control because of their easy design, good generalization, strong tolerance to input noise, and online learning ability [3],[4]. The capabilities of the final network of RBFNs are determined by the parameters optimization algorithms and the structure size. The numbers of hidden nodes in these RBFNs, however, are often assumed to be constant [5]. Only the parameters of these RBFN models are adjusted; the structures themselves are not. Huang [6] proposed a simple sequential learning algorithm for RBFNs, which is referred to as the RBF growing and pruning algorithm (GAP-RBF). The original design of GAP-RBF was enhanced to produce a more advanced model known as the generalized growing and pruning RBF algorithm (GGAP-RBF). Both GAP-RBF and GGAP-RBF neural networks use a pruning and growing strategy that is based on the "significance" of a neuron and links it to learning accuracy [7].

Recently, a quasi-linear ARX RBFN (QARX-RBFN) model has been proposed, and proved to have both universal approximation ability and easy-to-use linear properties in nonlinear system identification and control [8],[9]. It has an ARX-like linear structure, whose coefficients are nonlinear function of regression vector, rather than constants, and an RBFN is embedded to parameterize these coefficients. The ARX-like linear structure represents physical interpretation of applications explicitly. For example, in nonlinear adaptive control, it is desirable to have a prediction model linear with the input variable for controller design, and the ARX-like linear structure and its linearity with the input variable are definitely helpful for this application [8]–[10].

On the other hand, a quasi-linear ARX RBFN model still has room to improve its prediction accuracy by optimizing its structure, since if too simple structures of quasi-linear ARX RBFN model are selected, the prediction accuracy of the QARX-RBFN model might be inadequate to achieve the desired control performance. Thus, it is highly motivated to develop a self-organizing scheme for the structure optimization of QARX-RBFN model. In this paper, we propose a self-organizing QARX-RBFN model for nonlinear system identification by extending the results appeared in [11]. We concentrate our discussions on the modeling, the parameter estimation and the self-organizing scheme by: 1) reorganizing the paper; 2) redesigning and improving the self-organizing algorithm; 3) adding new numerical simulations to show the effectiveness of the self-organizing algorithm. The proposed self-organizing QARX-RBFN modeling scheme consists of simultaneously model structure optimization and model parameter optimization. It offers two important advantages: First, the RBF nodes in the quasi-linear ARX RBFN model is added or removed, based on the RBF node activity and the connection mutual information (MI), to achieve the appropriate network complexity and maintain overall computational efficiency for identification. Second, the model performance is significantly improved through the structure optimization.

The paper is organized as follows. Section 2 formulates the problem to solve. Section 3 introduces the quasi-linear ARX RBFN models. Section 4 proposes the self-organizing QARX-

[*] Graduate School of Information, Production and Systems, Waseda University, Kitakyushu-shi, Fukuoka 808-0135, Japan
[**] Politeknik Perkapalan Negeri Surabaya, Kampus ITS Sukolilo, Surabaya 60111, Indonesia
[***] CAPER, Faculty of Engineering, Universiti Putra Malaysia, Malaysia
E-mail: jinglu@waseda.jp
(Received September 18, 2015)
(Revised January 6, 2016)

RBFN model. Section 5 carried out numerical simulations to demonstrate the effectiveness of the proposed model. Finally, Section 6 gives conclusions.

2. Problem Formulation

Consider a single input single output nonlinear time-invariant dynamical system with input-output relation described by

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

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

where $y(t)$ is the output at the time t ($t = 1, 2, ...$), $u(t)$ the input, d the known integer time delay, $\varphi(t)$ the regression vector and, n, m are the system orders, and $g(\cdot)$ is a nonlinear function. We assume that at a small region around $\varphi(t) = 0$, $g(\cdot)$ is C^∞ continuous, and the origin is an equilibrium point, that is $g(0) = 0$.

In order to identify the system (1), in this paper we propose a self-organizing quasi-linear ARX RBFN model with an easy-to-use structure. For this purpose, we firstly construct a macro model with an easy-to-use structure, based on domain knowledge; secondly parameterize the macro model using a multi-input and multi-output RBF network; and finally optimize the model structure as well as the model parameters by introducing a self-organization identification algorithm.

3. Quasi-Linear ARX RBFN Model

As described in Ref. [9], a quasi-linear ARX model is constructed in two steps. In the first step, a macro-model is derived based on domain knowledge, which serves as a useful interface to introduce some properties favorable to specific applications, while the system complexity is embedded in the coefficient vector which is a unknown nonlinear function of the regression vector. In the second step, a flexible nonlinear nonparametric model such as an RBF network model is used to parameterize the coefficient vector.

3.1 Macro Models

3.1.1 Regression expression [9]

Under the continuous condition, the unknown nonlinear function $g(\varphi(t))$ can be Taylor-expanded on a small region around $\varphi(t) = 0$, i.e.,

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

where the prime denotes differentiation with respect to $\varphi(t)$. By introducing a coefficient vector Θ_φ, defined by

$$\Theta_\varphi = [a_{1,t} \cdots a_{n,t} \;\; b_{0,t} \cdots b_{m-1,t}]^T \tag{4}$$

to represent $(g'(0) + \frac{1}{2}\varphi^T(t)g''(0) + \cdots)$, we have a regression expression for the system:

$$y(t) = \varphi^T(t)\Theta_\varphi \tag{5}$$

where the coefficient vector Θ_φ is unknown nonlinear function of $\varphi(t)$.

3.1.2 Predictor expression [9]

In order to predict $y(t+d)$ by using the input-output data up to time t, the coefficients $a_{i,t}$ and $b_{j,t}$ should be calculable using the input-output data up to time t. To do so, let us iteratively replace $y(t + l)$ in the expressions of $a_{i,t}$ and $b_{j,t}$ with their predictions:

$$\hat{y}(t + l) = \hat{g}(\hat{\varphi}(t + l)), \quad l = 1, \cdots, d-1 \;\; (d > 1) \tag{6}$$

where $\hat{\varphi}(t+l)$ is $\varphi(t+l)$ whose elements $y(t+k)$, $(k = 1, 2, ..., l-1)$ are replaced with their predictions $\hat{y}(t + k)$. Define the new expressions of the coefficients by:

$$\tilde{a}_{i,t} = \tilde{a}_i(\phi(t)), \quad \tilde{b}_{j,t} = \tilde{b}_j(\phi(t)) \tag{7}$$

where $\phi(t)$ is a vector including all the elements of $\{\varphi(t+1), \varphi(t+2), ..., \varphi(t + d)\}$ except $y(t + k)$, $(k = 1, ..., d - 1)$, defined by

$$\phi(t) = [y(t) \cdots y(t - n + 1) \;\; u(t) \cdots u(t - m - d + 2)]^T \tag{8}$$

In a similar way to the linear case [9], we can derive a predictor expression for the system

$$y(t + d) = \phi^T(t)\Theta_\phi \tag{9}$$

where Θ_ϕ is a coefficient vector defined by

$$\Theta_\phi = [\alpha_{0,t} \cdots \alpha_{n-1,t} \;\; \beta_{0,t} \cdots \beta_{m+d-2,t}]^T \tag{10}$$

where the coefficients $\alpha_{i,t}, \beta_{j,t}$ are unknown nonlinear functions of $\phi(t)$, and their relations with the coefficients $\tilde{a}_{i,t}$, $\tilde{b}_{j,t}$ are referred to Ref. [9] for the details.

3.1.3 d-Difference predictor expression [8]

Sometimes it is better to have a differential expression for controller design. For this purpose, let's consider the coefficients $\alpha_{i,t} (i = 0, \cdots, n-1)$ and $\beta_{j,t} (j = 0, \cdots, m + d - 2)$ as a summation of two parts: the constant part α_i^l and β_j^l and the nonlinear function part on $\phi(t)$ which are denoted by $\alpha_{i,t} - \alpha_i^l$ and $\beta_{j,t} - \beta_j^l$, respectively. By introducing $\Theta_\phi = \theta + \Theta_\phi^n$, the expression of system in the predictor form Eq. (9) can be described by:

$$y(t + d) = \phi^T(t)\theta + \phi^T(t)\Theta_\phi^n \tag{11}$$

where $\theta = [\alpha_0^l \cdots \alpha_{n-1}^l \beta_0^l \cdots \beta_{m+d-2}^l]^T$ is a constant vector, and $\Theta_\phi^n = [(\alpha_{0,t} - \alpha_0^l) \cdots (\alpha_{n-1,t} - \alpha_{n-1}^l)(\beta_{0,t} - \beta_0^l) \cdots (\beta_{m+d-2,t} - \beta_{m+d-2}^l)]^T$ is a coefficient vector that is unknown nonlinear function of $\phi(t)$.

Applying a d-difference operator, defined by $\Delta = 1 - q^d$, to Eq. (11) and introducing $\psi(t) = \Delta\phi(t)$ and $\Psi^T(t)\tilde{\Theta}_\Psi^n = \Delta\phi^T(t)\Theta_\phi^n$, we obtain a d-difference expression of the system:

$$\Delta y(t + d) = \psi^T(t)\theta + \Psi^T(t)\tilde{\Theta}_\Psi^n \tag{12}$$

where $\Psi(t) = [y(t) \cdots y(t - d - n + 1)u(t) \cdots u(t - m - 2d + 2)]^T$ and $\tilde{\Theta}_\Psi^n$ is unknown nonlinear function of $\Psi(t)$.

As in Refs. [8],[10],[12],[13], we introduce the following assumptions for the system, in which $\varsigma(\Psi(t)) = \Psi^T(t)\tilde{\Theta}_\Psi^n$.

Assumption 1: (i) The system under consideration has a global representation Eq. (11); (ii) The linear part parameters θ lie in a compact region $\sum$; (iii) The system has a globally uniformly asymptotically stable zero dynamics; (iv) The nonlinear difference term $\varsigma(\cdot)$ is globally bounded, i.e. $\|\varsigma(\cdot)\| \leq D$ and the bound is known; (v) The system is controllable, in which a reasonable unknown controller may be expressed by $u(t) = \rho(\xi(t))$, where $\xi(t)$ is defined in Subsection 3.1.4.

3.1.4 d-Difference predictor expression linear in $u(t)$ [8]

If we consider a controller design, it is better to have the prediction model linear of $u(t)$. However, the macro model described by Eq. (12) is a general one which is nonlinear in

the variable $u(t)$, because the coefficient vector $\tilde{\Theta}^n_\Psi$ is a nonlinear function of $\Psi(t)$ whose elements contain $u(t)$. To solve this problem, an extra variable $x(t)$ is introduced and an unknown nonlinear function $\rho(\xi(t))$ is used to replace the variable $u(t)$ in $\tilde{\Theta}^n_\Psi$. Obviously, in a control system, the reference signal $y^*(t+d)$ can be used as the extra variable $x(t+d)$. Under Assumption 1(v), the function $\rho(\xi(t))$ exists. Define

$$\begin{aligned}\xi(t) \;=\; [&y(t)\;\cdots\; y(t-n_1)\; x(t+d)\;\cdots\\ &x(t-n_3+d)\; u(t-1)\;\cdots\; u(t-n_2)]^T\end{aligned} \tag{13}$$

including the extra variable $x(t+d)$ as an element. A typical choice for n_1, n_2 and n_3 in $\xi(t)$ is $n_1 = n+d-1, n_2 = m+2d-2$ and $n_3 = 0$. Then we have a d-difference predictor expression linear in $u(t)$, defined by

$$\Delta y(t+d) = \psi^T(t)\theta + \Psi^T(t)\Theta^n_\xi \tag{14}$$

where the coefficient vector $\Theta^n_\xi = \tilde{\Theta}^n_\Psi$ is an unknown nonlinear function of $\xi(t)$, in which the element $u(t)$ in $\Psi(t)$ is replaced by $\rho(\xi(t))$.

Remark 1: Macro models described by Eqs. (5), (9), (12) and (14) provide useful interfaces favorable to certain applications. These models are obtained by applying mathematical transformation. On the other hand, the representation flexibility of the models is realized by the following parameterization of the coefficient vectors using neural network models. Therefore, it is not important if their derivations are mathematically strict.

3.2 Parameterization of Macro Models

In the previous subsection, we derive macro models with various expressions described by Eqs. (5), (9), (12) and (14). In order to obtain the quasi-linear ARX models, the unknown nonlinear coefficient vectors should be parameterized using RBFN modes. By parameterizing the coefficient vector Θ_φ of Eq. (5), we have a quasi-linear ARX model; by parameterizing the coefficient vector Θ_ϕ of Eq. (9) we have a quasi-linear ARX prediction model; by parameterizing the coefficient vector $\tilde{\Theta}^n_\Psi$ of Eq. (12) we have a d-difference quasi-linear ARX prediction model; by parameterizing the coefficient vector Θ^n_ξ of Eq. (14) we have a d-difference prediction model linear in $u(t)$. These different quasi-linear ARX models can be used for different applications. Without losing the generality, we only discuss the parameterizations of Θ^n_ξ in Eq. (14).

The elements of Θ^n_ξ are unknown nonlinear functions of $\xi(t)$, which can be parameterized by RBFN or other neural network models. By using a multi-input and multi-output RBFN model, we have

$$\Theta^n_\xi = \sum_{j=1}^{M} \mathbf{w}_j R_j(\xi(t), \Omega_j) \tag{15}$$

where M is the number of RBF nodes, $\mathbf{w}_j = [\omega_{1j}\; \omega_{2j}\; \cdots\; \omega_{Nj}]^T$ the weight vector ($N = \dim(\Psi(t))$), and $R_j(\xi(t), \Omega_j)$ the RBF node functions defined by:

$$R_j(\xi(t), \Omega_j) = e^{-\lambda_j \|\xi(t) - Z_j\|^2} \qquad j = 1, 2, \cdots, M \tag{16}$$

where $\Omega_j = \{Z_j, \lambda_j\}$ is the parameter sets of the RBF node functions; Z_j is the center vector of a RBF and λ_j are the scaling parameters; $\|\cdot\|$ denotes the vector two-norm. Then we can express the quasi-linear ARX RBFN prediction model for Eq. (14) in a form of:

$$\Delta y(t+d) = \psi^T(t)\theta + \sum_{j=1}^{M} \Psi^T(t)\mathbf{w}_j R_j(\xi(t), \Omega_j) \tag{17}$$

Now, introducing the following notations:

$$\mathbf{W} = [\mathbf{w}_1\; \mathbf{w}_2\; \cdots\; \mathbf{w}_M] \tag{18}$$

$$\aleph(\xi(t)) = [e^{-\lambda_1\|\xi(t)-Z_j\|^2}\; \cdots\; e^{-\lambda_M\|\xi(t)-Z_j\|^2}]^T \tag{19}$$

the quasi-linear ARX RBFN model is further expressed by

$$\begin{aligned}\Delta y(t+d) \;&=\; \psi^T(t)\theta + \Psi^T(t)\mathbf{W}\aleph(\xi(t))\\ &=\; \psi^T(t)\theta + \Xi^T(t)\Theta\end{aligned} \tag{20}$$

where $\Theta = [w_{11} \cdots w_{n1} \cdots w_{1M} \cdots w_{nM}]^T$, $\Xi(t) = \aleph(\xi(t)) \otimes \Psi(t)$, while the symbol $\otimes$ denotes Kronecker production.

The quasi-linear ARX RBFN prediction model described by Eq. (17) is a parameterized model of the system in d-difference form described by Eq. (14). It is linear in the input variable $u(t)$, which is useful for controller design.

4. Self-Organizing QARX-RBFN Model

4.1 Model Parameter Estimation

The model structure is determined with the given number of RBF nodes, M. According to the parameter property, the model parameters as in Eq. (20) are divided into three groups: the linear parameter vector θ of the linear part $\psi^T(t)\theta$, the linear parameter vector Θ and the nonlinear parameter sets Ω_j of the nonlinear part $\Psi^T(t)\mathbf{W}\aleph(\xi(t))$. The nonlinear parameter sets Ω_j are determined off-line. Let us denote the estimates of Ω_j by $\hat{\Omega}_j$. In order to determine the centers and the widths of the RBF, an affinity propagation (AP) clustering method is employed. The center Z_j is the arithmetic mean value of all training data in each cluster. The width λ_j is ϱ (an appropriate positive constant) times the largest distances between all training data in each cluster. The parameter vectors θ and Θ are estimated by using iterative algorithms, respectively [14].

The linear parameter vector θ of the linear part of the model is updated as [15]:

$$\hat{\theta}(k) = \hat{\theta}(k-d) + \frac{a(k)\psi(k-d)e_1(k)}{1 + \psi(k-d)^T\psi(k-d)} \tag{21}$$

where $\hat{\theta}(k)$ is the estimate of θ at step k, which also denotes the parameter of a linear model used to approximate the system in d-difference form, and

$$a(k) = \begin{cases} 1 & \texttt{if } |e_1(k)| > 2D \\ 0 & \texttt{otherwise} \end{cases} \tag{22}$$

where $e_1(k)$ denotes the error of the linear model, defined by

$$e_1(k) = \Delta y(k) - \psi(k-d)^T\hat{\theta}(k-d) \tag{23}$$

The linear parameter Θ of nonlinear part of the quasi-linear ARX RBFN model is updated by a least square (LS) algorithm:

$$\hat{\Theta}(k) = \hat{\Theta}(k-d) + \frac{P(k)\Xi(k-d)e_2(k)}{1 + \Xi(k-d)^T P(k)\Xi(k-d)} \tag{24}$$

where $\hat{\Theta}(k)$ is the estimate of Θ at step k. $\hat{\Theta}(0) = \Theta_\circ$ is assigned with an appropriate initial value. $e_2(k)$ is the error of quasi-linear ARX RBFN model, defined by:

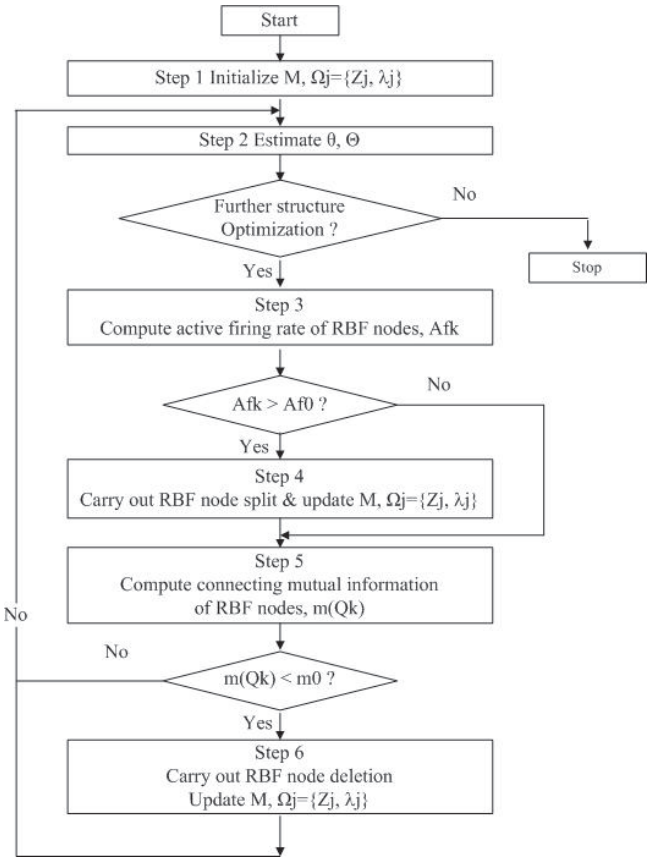

Fig. 1 Flowchart of self-organization of the ARX-RBFN model.

$$e_2(k) = \Delta y(k) - \psi(k-d)^T \hat{\theta}(k-d) - \Xi^T(k-d)\hat{\Theta}(k-d) \tag{25}$$

$$P(k) = \frac{P(k-d) - P^T(k-d)\Xi(k-d)^T \Xi(k-d)P(k-d)}{1 + \Xi(k-d)^T P(k)\Xi(k-d)} \tag{26}$$

No restriction is made on how the parameters $\hat{\Theta}(k)$ are updated except they always lie inside some pre-defined compact region $\hbar$:

$$\hat{\Theta}(k) \in \hbar \quad \forall\, k \tag{27}$$

4.2 A Self-Organization of QARX-RBFN Model

With a given number of RBF nodes to predetermine the model structure, a reasonable model can be obtained by using the algorithm described in Subsection 4.1. The model performance can be further improved by optimizing the model structure. Then, by adding a procedure to optimize the structure, we propose a self-organizing QARX-RBFN (SOQARX-RBFN) model. Figure 1 shows a flowchart of the self-organization of the QARX-RBFN model, which is summarized in the following six steps:

Step 1) Set a proper initial value of the number of RBF node, M, and predetermine the parameter sets of RBF nodes, $\Omega_j = \{Z_j,\ \lambda_j\}\ j = 1, 2, .., M$.

Step 2) Estimate the linear parameter vectors of θ and Θ by using the iterative algorithms described in Subsection 4.1. Check the performance of the quasi-linear ARX RBFN model. If it needs further structure optimization, go to Step 3, otherwise stop.

Step 3) Compute the active firing rate Af_k for each RBF node by

$$Af_k = \frac{1}{N_s}\sum_{t=1}^{N_s} \frac{\rho Q_k(t)}{\sum_{k=1}^{M} Q_k(t)}, \tag{28}$$

where $Q_k(t)$ is the output of the k-th RBF node ($k = 1, 2, \cdots, M$), $\rho \gg 1$ is a positive constant, and N_s is the number of training data. The RBF nodes with active firing rate larger than an activity threshold Af_0 ($0.05 < Af_0 < 0.3$) will split into multiple RBF nodes. Check the value of $Af_k\ \forall k$, if there is $Af_k > Af_0$, go to Step 4), otherwise go to Step 5).

Step 4) Carry out RBF node split for the RBF node k when $Af_k > Af_0$ ($k = 1, 2, ..., M$): the old k-th RBF node will be deleted and N_{new} new RBF nodes will be inserted. The initial parameters of the newly inserted RBF nodes are determined in the following way. The centers $Z_{k,j}$ and the width $\lambda_{k,j}$ of newly inserted RBF nodes are given by

$$Z_{k,j} = \gamma_j Z_k + \delta_j \frac{1}{N_s}\sum_{t=1}^{N_s} \xi(t) \tag{29}$$

$$\lambda_{k,j} = \gamma_j \lambda_k,\ j = 1, 2, \cdots, N_{new} \tag{30}$$

where $0.95 < \gamma_j < 1.05$ and $0 < \delta_j < 0.1$ (For better performance, γ_j should be close to 1, and δ_j should be close to 0.05), Z_k and λ_k are the center and width of the old k-th RBF node, and N_{new} ($N_{new} = 2$ in our simulations) is the number of the newly inserted RBF nodes corresponding to the old k-th RBF node; And the output connection weights of newly inserted RBF nodes are calculated by

$$\omega_{k,j} = r_j \frac{1}{N_s}\sum_{t=1}^{N_s}\left(\frac{\omega_k Q_k(t) - e_2(t)}{Q_{k,j}(t)}\right) \tag{31}$$

$$\sum_{j=1}^{N_{new}} r_j = 1,\ j = 1, 2, \cdots, N_{new}$$

where r_j is the allocating parameters for the new RBF nodes, $Q_k(t)$ is the output value of the old k-th RBF node, $Q_{k,j}(t)$ is the output value of the newly inserted j-th RBF node, $e_2(t)$ is the current prediction error. ω_k is the output weight of the old k-th RBF node, and $\omega_{k,j}$ is the output weight of the j-th new RBF node.

After completing all RBF node splits, update the number of RBF node, M, and the parameter sets of RBF nodes, $\Omega_j = \{Z_j,\ \lambda_j\}\ j = 1, 2, .., M$, and go to Step 5.

Step 5) Compute the mutual information between the k-th RBF node and the corresponding output nodes, $m(Q_k)$ ($k = 1, 2, ..., M$) by

$$m(Q_k) = \sum_{q=1}^{N}\left(\frac{M(Q_k; y_q)}{\min(H(Q_k), H(y_q))}\right) \tag{32}$$

$$\begin{aligned} M(Q_k; y_q) &= H(Q_k) - H(Q_k|y_q) \\ &= H(y_q) - H(y_q|Q_k) \end{aligned} \tag{33}$$

where $N = \dim(\Psi(t))$ is the number of corresponding output nodes for the $k-$th RBF node, $H(Q_k) =$

$\ln(2\pi\varepsilon)^K|COV(Q_k)|/2, COV(Q_k)$ is a standard covariance of Q_k, ε is a mathematical constant here (2.718) and $H(y_q) = \ln(2\pi\varepsilon)|COV(y_q)|/2$. The RBF nodes with mutual information smaller than a threshold $m_0, (0 < m_0 < 0.05)$ will be deleted. Check $m(Q_k), \forall k$, if there is $m(Q_k) < m_0$, go to Step 6). Otherwise, go back to Step 2).

Step 6) Carry out RBF node deletion for the RBF node k when $m(Q_k) < m_0$ $(k = 1, 2, ..., M)$. Delete the k-th RBF node as well as all connections between the node Q_k and the corresponding output nodes, and update the remaining RBFN parameters in the following way. Find the RBF node k' which has the minimal Euclidean distance to the k-th RBF node. The parameters of the RBF node k' are adjusted as follows:

$$Z_{k',after} = Z_{k',before}, \quad \lambda_{k',after} = \lambda_{k',before} \tag{34}$$

$$\omega_{k',after} = \omega_{k',before} + \omega_k \frac{1}{N_s} \sum_{t=1}^{N_s} \left(\frac{Q_k(t)}{Q_{k'}(t)} \right) \tag{35}$$

where $\omega_{k',before}$ and $\omega_{k',after}$ are the connecting weight of the RBF node k' before and after structure adjustment, $Z_{k',before}$ and $Z_{k',after}$ are the centers of the RBF node k' before and after deleting the RBF node k and, $\lambda_{k',before}$ and $\lambda_{k',after}$ are the widths of the RBF node k' before and after deleting the RBF node k.

After completing all RBF node deletions, update the number of RBF node, M, and the parameter sets of RBF nodes, $\Omega_j = \{Z_j,\ \lambda_j\}\ j = 1, 2, .., M$, and go to Step 2.

Remark 2: The convergence of the self-organizing algorithm can be evaluated in two phases. First, when the number of RBF node, M, and the parameter sets of RBF nodes, $\Omega_j = \{Z_j,\ \lambda_j\}\ j = 1, 2, .., M$ are fixed, the parameter vectors θ and Θ are estimated by LS algorithms (21) and (24), which are guaranteed to converge. Secondly, when the number of RBF nodes is changing: 1) in case of new RBF nodes inserted, the parameter set $\{Z_{k,j},\ \lambda_{k,j}\}$ and the weight $\omega_{k,j}$ are calculated by (29)-(31) in such a way that the prediction error $e_2(t)$ to be zero; 2) in case of RBF nodes deleted, the prediction error $e_2(t)$ is adjusted to be no change by (35), which ensures the convergence of the algorithm is not influenced. Due to limitation of space, we will not carry out detailed discussion here, and similar detailed discussions are referred to Ref. [7],[16],[17].

5. Numerical Simulations

5.1 Simulations for System Identification

In order to study the behavior of the proposed control method, a numerical simulation is described in this section. The system is a nonlinear one governed by:

$$y(t) = f[y(t-1), y(t-2), y(t-3), u(t-2), u(t-3)] \tag{36}$$

where:

$$f[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2} \tag{37}$$

The system is exited by 1000 random sequence with the amplitude between -1 and 1. To test the obtained model, a set of 800 input-output data is sampled as test data and the input data are described as:

$$u(t) = \begin{cases} \sin(\frac{2\pi t}{250}) & \text{if } t < 500 \\ 0.8\sin(\frac{2\pi t}{250}) + 0.2\sin(\frac{2\pi t}{250}) & \text{otherwise.} \end{cases} \tag{38}$$

When identifying the system, the self-organizing QARX RBFN prediction model Eq. (17) is used, in which the number of RBF nodes is first given with a suitable value, say $M = 6$ or automatically determined by a clustering algorithm, the model orders $m = 3, n = 2$, the delay $d = 2$, and the bound of the nonlinear difference term of the system is set to $D = 0.05$.

Estimation of nonlinear parameters Ω_j: The nonlinear parameter vectors $\Omega_j = \{Z_j, \lambda_j\}, j = 1, \ldots, M$ are first determined offline. Then an AP clustering algorithm is applied to the dataset for partitioning the input space of $\xi(t) = [y(t) \ldots y(t-n)y^*(t+1)u(t-1) \ldots u(t-m)]^T$. After clustering, six clusters are generated automatically in the input space, so that $M = 6$. The parameter vector Z_j corresponds to the center of each cluster, while λ_j is calculated by multiplying a constant $\varrho = 0.2$ to the largest distance of the data in each cluster.

The results of $\Omega_j = \{Z_j, \lambda_j\}, j = 1, \ldots, 6$ are shown in Table 1. What should be mentioned is that the nonlinear parameters and number of RBF node are fixed during the whole identification procedure for quasi-linear ARX RBFN model without self-organization, but it has been added and removed for the self-organizing QARX-RBFN model.

The simulations are carried out as follows: first, at beginning M is given a suitable value, then we do parameter estimation as described in Subsection 4.1, after that we adjust M to optimize the structure following the algorithm described in Section 4.2. Figure 2 shows the number of RBF node with the initial value $M = 2$, in which the final number of RBF nodes $M = 4$ is shown. Figure 3 shows the number of RBF nodes with the

Table 1 Estimations of parameters Ω_j, $j = 1, \ldots, 6$.

	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6
ξ_1	-0.6904	0.3529	-0.3977	0.1227	0.4506	0.7655
ξ_2	0.6364	0.5067	0.5446	0.5786	0.3103	0.7084
ξ_3	0.7064	0.5483	0.5124	0.7187	0.1876	0.1487
ξ_4	0.7350	0.3166	0.3356	0.7124	0.4563	0.3415
ξ_5	0.0812	0.3556	0.2856	0.2578	0.2034	0.0589
ξ_6	0.6412	0.4780	0.7912	0.3324	0.5849	0.8315
ξ_7	0.7637	0.7355	0.7085	0.8172	0.5081	0.8417
λ	0.0181	0.0272	0.0315	0.0231	0.0281	0.0172

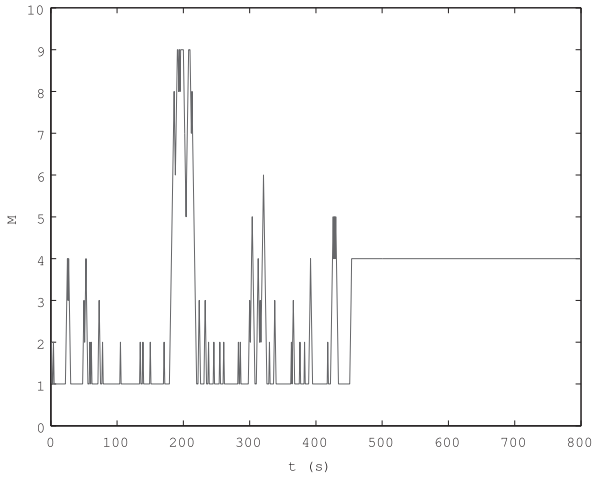

Fig. 2 The number of RBF nodes with initial value $M = 2$.

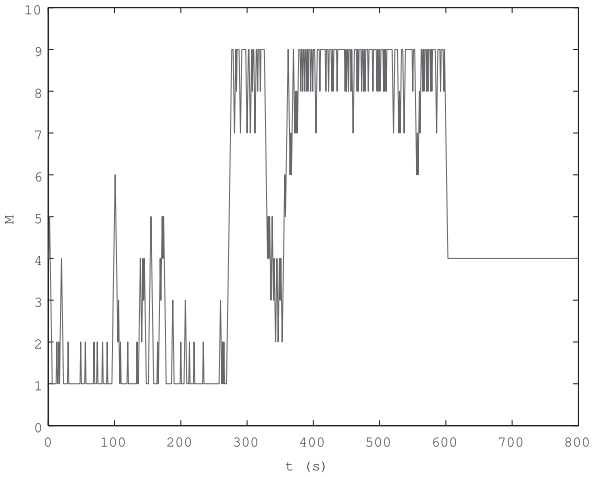

Fig. 3 The number of RBF nodes with initial value $M = 4$.

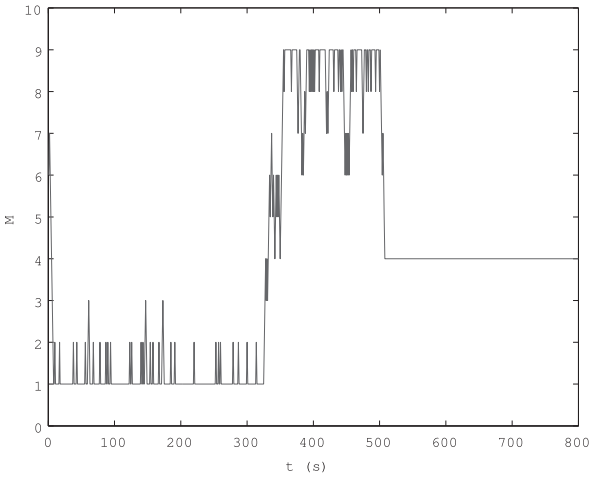

Fig. 4 The number of RBF nodes with initial value $M = 6$.

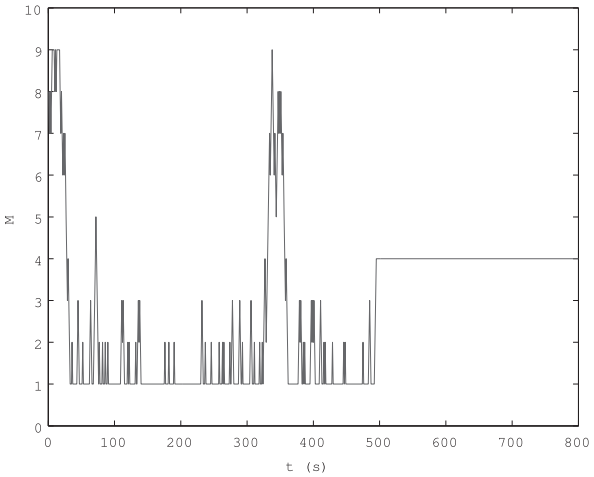

Fig. 5 The number of RBF nodes with initial value $M = 8$.

initial value $M = 4$, in which the final number of RBF nodes $M = 4$ is shown. Figure 4 shows the number of RBF nodes with the initial value $M = 6$, in which the final number of RBF nodes $M = 4$ is shown. Figure 5 shows the number of RBF nodes with the initial value $M = 8$, in which the final number of RBF nodes $M = 4$ is shown. Although the algorithm begins with different initial values of M, it converges to $M = 4$.

A comparison is shown in Table 2, in which M denotes the number of RBF nodes and the accuracy denotes the root mean square error (RMSE) of between the system output and the model output simulaltion on the test data. The result shows that the self-organizing QARX RBFN model gives better performance than the quasi-linear ARX RBFN model, which confirms the effectiveness of the self-organization algorithm.

Table 2 Identification result of QARX-RBFN model for two methods.

Method	M	RMSE
QARX-RBFN without self-organization	6	0.073
QARX-RBFN with self-organization	4	0.015

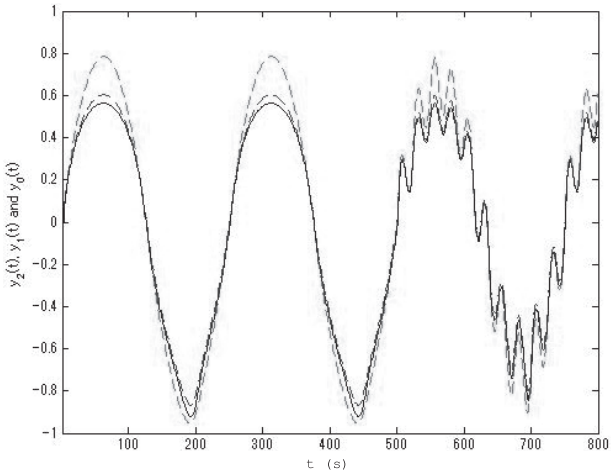

Fig. 6 Simulation of identified models on the test data. The solid line shows the system true output, the dashed line denotes the simulated model output by using QARX-RBFN with self-organization and the dashdot line denotes the simulated model output by using QARX-RBFN without self-organization.

The comparison simulation on the test data for identification result of QARX-RBFN model without and with self-organization is shown in Fig. 6, which gives an RMSE of 0.073 without self-organization and 0.015 with self-organization for $t = 1000$ and the value of $M = 4$. It can be found that the identification result of the QARX-RBFN model with self-organization gives better prediction accuracy than that of the QARX-RBFN model without self-organization.

5.2 Simulations for Control

The desired output in this example is a piecewise function:

$$y^*(t) = \begin{cases} 0.6y^*(t-1) + r(t-1), \\ \qquad t \in [1, 100] \cup [151, 200] \\ 0.7\mathtt{sign}(0.4493y^*(t-1) + 0.57r(t-1)), \\ \qquad t \in [101, 150] \end{cases} \tag{39}$$

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

Two main control methods are implemented for comparison between using fixed RBF nodes, $M = 6$ and optimized $M = 4$.

- Method 1: Adaptive fuzzy switching controller based on SOQARX-RBFN model [8]
- Method 2: An improved Elman neural network (IENN) controller based on SOQARX-RBFN [18]

Since the quasi-linear ARX RBFN prediction model described by Eq. (17) is linear in the input variable $u(t)$, it is easy to design the control system. Please refer Refs. [8] and [18] for the details of control systems based on the quasi-linear ARX RBFN models.

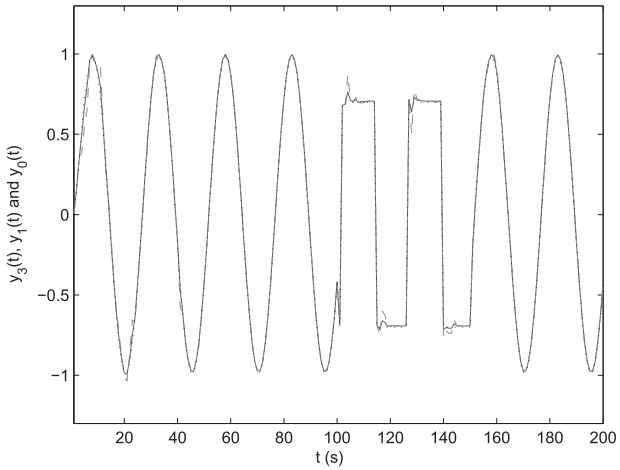

Fig. 7 Control results. Method 1: Adaptive fuzzy switching self-organizing QARX-RBFN control output using optimized $M = 4$ (solid), adaptive fuzzy switching quasi-linear ARX-RBFN control output using fixed $M = 6$ (dashed) and reference (dotted).

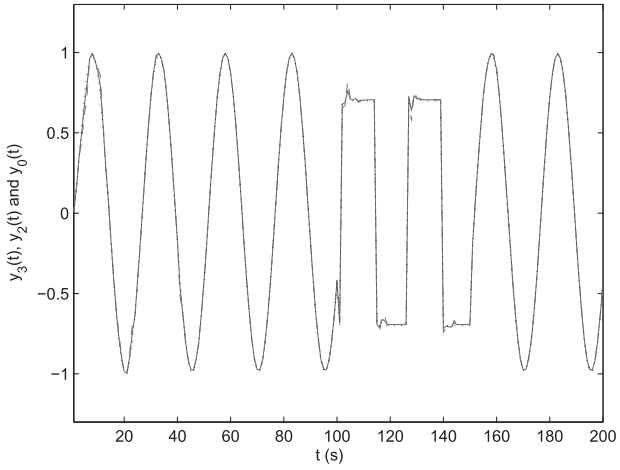

Fig. 8 Control results. Method 2: IENN SOQARX-RBFN control output using optimized $M = 4$ (solid), IENN QARX-RBFN control output using fixed $M = 6$ (dashed) and reference (dotted).

Figure 7 shows the control results, in which the comparisons between Method 1 using fixed RBF nodes $M = 6$ and Method 1 using optimized $M = 4$ are used.

In Fig. 7 the dotted line is the desired output $y_0(t)$, the dashed line is the control output $y_1(t)$ of Method 1 using fixed RBF nodes, $M = 6$ and the solid line is the control output $y_3(t)$ of Method 1 using optimized $M = 4$.

We can easily see that the proposed Method 1 using optimized $M = 4$ has given a good result since $t = 10$, which is better than Method 1 using fixed RBF nodes, $M = 6$. It is also better than Method 1 using fixed RBF nodes, $M = 6$ when $t \in [(10, 100) \cup (110, 200)]$ and the robustness of proposed Method 1 using optimized $M = 4$ is much better than Method 1 using fixed RBF nodes, $M = 6$ since $t = 100$, as illustrated. Therefore, the proposed Method 1 using optimized $M = 4$ has a better control result than Method 1 using fixed RBF nodes, $M = 6$.

Figure 8 shows the control results, in which the comparisons between Method 2 using fixed RBF nodes, $M = 6$ and Method 2 using optimized $M = 4$ are shown. In Fig. 8 the dotted line is the desired output $y_0(t)$, the dashed line is the control output $y_2(t)$ of method 2 using fixed RBF nodes, $M = 6$ and the solid line is the control output $y_3(t)$ of Method 2 using optimized $M = 4$. We can easily see that the proposed method 2 using optimized $M = 4$ has achieved a good result, since $t = 10$, which is better than Method 2 using fix hidden neurons $M = 6$. It is also better than Method 2 using fixed RBF nodes, $M = 6$ when $t \in [(10, 100) \cup (110, 200)]$ and the robustness of the proposed method 2 using optimized $M = 4$ is much better than Method 2 using fixed RBF nodes, $M = 6$ since $t = 100$, as illustrated. Therefore, the proposed method 2 using optimized $M = 4$ has a better control result than Method 2 using fixed RBF nodes, $M = 6$.

Table 3 Comparison result of the errors.

Method	Mean of RMSE	Mean of variances
Method 1 using fix $M = 6$	0.0141	0.057
Method 1 using $M = 4$	0.0103	0.021
Method 2 using fix $M = 6$	0.0117	0.035
Method 2 using $M = 4$	0.0075	0.011

Table 3 shows the results of the two methods. We can see that Method 1 using optimized $M = 4$ gives smaller control error than Method 1 using fixed RBF nodes, $M = 6$, and Method 2 using optimized $M = 4$ improves the control performance further than Method 2 using fixed RBF nodes, $M = 6$. The proposed method 2 using optimized $M = 4$ gives better accuracy than the other methods.

6. Conclusion

This paper has proposed a self-organizing QARX-RBFN model by adding a self-organizing procedure to optimize the structure of the quasi-linear ARX RBFN model. Based on the active firing rate and the connection mutual information of RBF nodes, the RBF nodes in the quasi-linear ARX RBFN model can be added or removed, so that the structure of the quasi-linear ARX RBFN model is optimized for the given system. This significantly improves the performance of the model. Numerical simulations on identification and control have confirmed the effectiveness of the proposed self-organizing QARX-RBFN model.

References

[1] L. Ljung: *System Identification: Theory for The User*, 2nd ed., Prentice-Hall PTR, 1999.

[2] M.A. Jami'in, I. Sutrisno, and J. Hu: Deep searching for parameter estimation of the linear time invariant (LTI) system by using quasi-ARX neural network, *Proceedings of The 2013 International Joint Conference on Neural Networks (IJCNN2013)*, pp. 2758–2762, 2013.

[3] H. Yu, T.T. Xie, S. Paszczynski, and B.M. Wilamowski: Advantages of radial basis function networks for dynamic system design, *IEEE Trans. Ind. Electron*, Vol. 58, No. 12, pp. 5438–5440, 2012.

[4] M.Z. Hou and X.L. Han: Constructive approximation to multivariate function by decay RBF neural network, *IEEE Trans. Neural Netw.*, Vol. 21, No. 9, pp. 1517–1523, 2009.

[5] K. Dalamagkidis, K.P. Valavanis, and L.A. Piegl: Nonlinear model predictive control with neural network optimization for autonomous autorotation of small unmanned helicopters, *IEEE Trans. Control Syst. Technol.*, Vol. 19, No. 4, pp. 818–831, 2011.

[6] G.B. Huang, P. Saratchandran, and N. Sundararajan: An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks, *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, Vol. 34, No. 6,

pp. 2284–2292, 2004.
[7] H.G. Han, Q.L. Chen, and J.F. Qiao: An efficient self-organizing RBF neural network for water quality predicting, *Neural Netw.*, Vol. 24, No. 7, pp. 717–725, 2011.
[8] L. Wang, Y. Cheng, and J. Hu: Stabilizing switching control for nonlinear system based on quasi-ARX model, *IEEJ Trans. Electrical and Electronic Engineering*, Vol. 7, No. 4, pp. 390–396, 2012.
[9] J. Hu, K. Kumamaru, and K. Hirasawa: A quasi-ARMAX approach to the modeling of nonlinear systems, *International Journal of Control*, Vol. 74, No. 18, pp. 1754–1766, 2001.
[10] J. Hu, K. Hirasawa, and K. Kumamara: A method for applying neural networks to control of nonlinear systems, *Neural Information Processing: Research and Development*, pp. 351–369, Springer, 2004.
[11] I. Sutrisno, M.A. Jami'in, J. Hu, and M.H. Marhaban: Self-organizing quasi-linear ARX RBFN modeling for identification and control of nonlinear systems, *Proceedings of The SICE Annual Conference 2015 (SICE2015)*, Hangzhou, China, pp. 788–793, 2015.
[12] I. Sutrisno, M.A. Jami'in, and J. Hu: Modified fuzzy adaptive controller applied to nonlinear systems modeled under quasi-ARX neural network, *Artificial Life Robotics*, Vol. 19, No. 1, pp. 22–26, 2013.
[13] I. Sutrisno, M.A. Jami'in, and J. Hu: Neural predictive controller of nonlinear systems based on quasi-ARX neural network, *Proceedings of the 2012 International Conference on Automation and Computing (ICAC 18th 2012)*, Loughborough, UK, pp. 83–88, 2012.
[14] L. Wang: Study on adaptive control of nonlinear dynamical systems based on quasi-ARX models, *DSpace at Waseda University*, available at https://dspace.wul.waseda.ac.jp/dspace/handle/2065/37540, 2013.
[15] L. Chen and K.S. Narendra: Nonlinear adaptive control using neural networks and multiple models, *Automatica*, Vol. 37, No. 8, pp. 1245–1255, 2001.
[16] H.G. Han and J.F. Qiao: An adaptive computation algorithm for RBF neural network, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 23, No. 2, pp. 342–347, 2012.
[17] M.A. Jami'in, I. Sutrisno, and J. Hu: Lyapunov learning algorithm for quasi-ARX neural network to identification of nonlinear dynamical system, *Proceedings of The 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC2012)*, Seoul, pp. 3141–3146, 2012.
[18] I. Sutrisno, M.A. Jami'in, and J. Hu: An improved Elman neural network controller based on quasi-ARX neural network for nonlinear systems, *IEEJ Trans. Electrical and Electronic Engineering*, Vol. 9, No. 5, pp. 494–501, 2014.

Imam Sutrisno

He received his B.S. and M.S. degrees in Control System of Electrical Engineering from Institute of Technology 10th Nopember Surabaya, Indonesia, in 1998 and 2005, respectively. He is currently a Ph.D. student at Waseda University. His research interests include neural predictive controller, modified Elman neural network, fuzzy switching controller, Lyapunov learning algorithm, adaptive neurofuzzy, system identification and control systems design. He is a student member of IEEE.

Mohammad Abu Jami'in

He received his B.S. degrees in Marine Engineering and M.S. degrees in Control System of Electrical Engineering from Institute of Technology 10th Nopember Surabaya, Indonesia, in 2000 and 2008, respectively. He is currently a Ph.D. student at Waseda University. His research interests include Lyapunov stability, linear system, system identification and control systems design. He is a student member of IEEE.

Jinglu Hu (Member)

He received a M.Sci. degree in Electronic Engineering from Sun Yat-Sen University, Guangzhou, China in 1986, and a 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 a Lecturer at Sun Yat-Sen University. From 1997 to 2003, he worked as a Research Associate at Kyushu University. From 2003 to 2008, he worked as an Associate Professor and since April 2008, he has been a Professor at Graduate School of Information, Production and Systems of Waseda University. 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. He is a member of IEEE, IEEJ and IEICE.

Mohammad Hamiruce Marhaban

He received the B.E. degree in Electrical and Electronic Engineering from Salford University, United Kingdom, in 1998 and the Ph.D. degree in Electronic Engineering from Surrey University, United Kingdom, in 2003. He is an Associate Professor at the Department of Electrical and Electronics Engineering, UPM. His area of interest is intelligent control system and computer vision.