Title	Model-reference adaptive control based on neurofuzzy networks
Author(s)	Liu, XJ; LaraRosano, F; Chan, CW
Citation	Ieee Transactions On Systems, Man And Cybernetics Part C: Applications And Reviews, 2004, v. 34 n. 3, p. 302-309
Issued Date	2004
URL	http://hdl.handle.net/10722/43059
Rights	Creative Commons: Attribution 3.0 Hong Kong License

Model-Reference Adaptive Control Based on Neurofuzzy Networks

Xiang-Jie Liu, Felipe Lara-Rosano, and C. W. Chan

***Abstract*—Model reference adaptive control (MRAC) is a popular approach to control linear systems, as it is relatively simple to implement. However, the performance of the linear MRAC deteriorates rapidly when the system becomes nonlinear. In this paper, a nonlinear MRAC based on neurofuzzy networks is derived. Neurofuzzy networks are chosen not only because they can approximate nonlinear functions with arbitrary accuracy, but also they are compact in their supports, and the weights of the network can be readily updated on-line. The implementation of the neurofuzzy network-based MRAC is discussed, and the local stability of the system controlled by the proposed controller is established. The performance of the neurofuzzy network-based MRAC is illustrated by examples involving both linear and nonlinear systems.**

***Index Terms*—Model-reference adaptive control, neurofuzzy networks, nonlinear controller.**

I. Introduction

THE MODEL reference adaptive control (MRAC) is popular in the area of self-tuning control [1]. In the MRAC, the error between the reference model output and the real plant output is used to adjust its parameters in order to control the plant to follow the desired output from the reference model. It has been demonstrated to be effective in controlling linear plants, and hence has found numerous applications in process control. In general, it performs better than conventional fixed-parameter PID controller, as it can better adapt to changes in the parameters of the plant, and therefore, has attracted numerous attentions in control engineering. The linear MRAC performs well when it is working around the operating point, where the plant can be approximated by a linear model. However, as most industry processes are highly nonlinear, nonminimum, and with various types of uncertainties and load disturbances [2], the performance of the linear MRAC may deteriorate, and suitable nonlinear control may have to be used.

Since neural networks can approximate any nonlinear functions with arbitrary accuracy, they have been applied to develop adaptive control of nonlinear systems [3]. Indirect adaptive control using neural networks is presented in [4]. In [5], the design of a conventional MRAC has been extended to direct neural control for a class of nonlinear system with structural uncertainty. Direct multilayer perceptron (MLP) model reference adaptive control has been presented in [6] by comparison with Lyapunov adaptive control. Also, fuzzy logic has been incorporated into MRAC with fuzzy basis function being used to represent the parameter information [7]. A robust adaptive law to adaptively compensate for the modeling error introduced by fuzzy approximation was proposed in [8].

Both fuzzy logic and neural networks belong to the class of "model-free" modeling and control approach, and both of them have their own advantages [9]. Fuzzy rules can be expressed readily the expert knowledge into linguistic form, while neural networks possess good learning ability, and can approximate nonlinear functions with arbitrary accuracy. Neurofuzzy networks are derived from fuzzy logic and neural network, and therefore inherit the property of both fuzzy logic and neural network [10], giving controllers developed based on neurofuzzy networks the ability to incorporate expert knowledge into the controller, and to be trained from experimental data. Therefore, neurofuzzy networks provide a very useful tool for developing nonlinear adaptive controllers. Further, the neurofuzzy network-based nonlinear controller has the advantage over conventional self-tuning controllers in that once it is trained for certain operating points, no re-training for those operating points is required. Also, the transition from one local model to another is smooth. In contrast, conventional self-tuning controllers have to be retuned every time when the operating point is changed, as previously trained parameters cannot be stored in the controller. This is also the main reason for their inferior performance in controlling nonlinear systems. Some of the earlier work included fuzzy neural networks for nonlinear systems modeling [11], generalized minimum variance (GMV) controller based on the neurofuzzy networks using a simplified recursive least squares method [12] derived from the local change property of neurofuzzy networks [13].

There are several implementation of neurofuzzy network [10], such as the cerebella model articulation controller (CMAC), associate memory network (AMN), and the radial basis function (RBF) network. The AMN is one of the earliest attempts to use neural networks to implement the desired mapping for fuzzy systems [9], and has been successfully applied to problems like backing up a truck-and trailer [14] and target tracking [15], where distinctive features such as modularity, robustness, and adaptability have been demonstrated. B-spline networks are a class of lattice AMN that uses univariate basis functions as the membership function of input fuzzy variable in the fuzzy linguistic statements. The multivariate fuzzy sets are formed from the product operator representing the fuzzy conjunction, which enables the networks to be interpreted as

Manuscript received July 6, 2001; revised July 21, 2003. This work was supported in part by the National Nature Science Foundation of China under Grant 69804003. This paper was recommended by Associate Editor J. Lee.

X.-J. Liu and F. Lara-Rosano are with Centro de Ciencias Aplicadas y Desarrollo Tecnologico, Universidad Nacional Autonoma de Mexico, Mexico City, D.F. 04510, Mexico (e-mail: liu@aleph.cinstrum.unam.mx; lararf@servidor.unam.mx).

C. W. Chan is with Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, China (e-mail: mechan@hkucc.hku.hk).

Digital Object Identifier 10.1109/TSMCC.2003.819702

a set of fuzzy rules. Therefore, by embodying both the qualitative and the quantitative approaches, these networks enable heuristic information to be incorporated into and inferred from the network, and allow fuzzy learning rules to be derived.

The objective of this paper is to derive a nonlinear MRAC based on neurofuzzy networks. The proposed controller can be interpreted as a direct self-tuning control. The online update of the weights of the neurofuzzy network is derived, and the local stability of the closed-loop system using the proposed controller is established. Two simulation examples involving a linear steam-boiler system and a nonlinear system are presented to illustrate the implementation and the performance of the proposed neurofuzzy network-based MRAC.

II. Structure of MRAC

There are two main approaches to derive the MRAC: the MIT approach and the Lyapunov approach. The basic structure of MRAC is depicted in Fig. 1, where the state feedback gain vector is denoted by $d(t) = [d_0(t), d_1(t-1), \ldots]$, and the set point weighting vector by $c(t) = [c_0(t), c_1(t-1), \ldots]$.

In the MRAC, the design criterion is that the output of the plant follows as close as possible to the output of a reference model. In general, the control $\mathrm{u(t)}$ designed using this approach can be expressed in the following form:

$$u(t) = C(z^{-1})r(t) + D(z^{-1})y(t) \tag{1}$$

where $\mathrm{r(t)}$ is the reference input, $\mathrm{y(t)}$ is either the output or the state of the system, and $\mathrm{C(z^{-1})}$ and $\mathrm{D(z^{-1})}$ are rational functions in $\mathrm{z^{-1}}$, the backward shift operator. Rewriting (1) gives

$$u(t) = \theta^T(t)\omega(t) \tag{2}$$

where $\omega \cong [r(t), r(t-1), \ldots, y(t), y(t-1), \ldots]$ is the input vector and $\theta^T \cong [c_o, c_1, \ldots, d_0, d_1, \ldots]$ is the weight vector. The gradient method can be used to update the weight θ. Consider the following cost function:

$$J = \frac{1}{2}e^2 = \frac{1}{2}(y - y_m)^2. \tag{3}$$

The weight θ is adjusted proportional to the negative gradient of J

$$\dot{\theta} = -g\frac{\partial J}{\partial \theta} = -ge\frac{\partial e}{\partial \theta} \tag{4}$$

where g is a positive constant. Equation (4) can also be interpreted as the self-tuning law for updating θ on-line, and g is the learning rate. If there are no *a priori* knowledge of the system, then the initial value $\theta(0)$ is chosen randomly. However, if some *a priori* knowledge is available, $\theta(0)$ can be suitably chosen based on this information to ensure a fast convergence of θ.

The MRAC is now well developed and has found applications in many fields, especially when the system is linear and time-invariant. For nonlinear systems, linear MRAC can only be applied if the operating range is small, as it is unsuitable for highly nonlinear systems. Popular approaches in controlling highly nonlinear systems include the gain scheduling [16] and the Takagi–Sugeno fuzzy controller [17]. In these approaches, a series of local linear controllers are used. Another recent approach is to implement nonlinear controllers using neurofuzzy networks, which can be interpreted as a network consisting of a series of linear local models designed for different operating points of the nonlinear system [10], [12]. The implementation of this class of controller using local linear MRAC is discussed below.

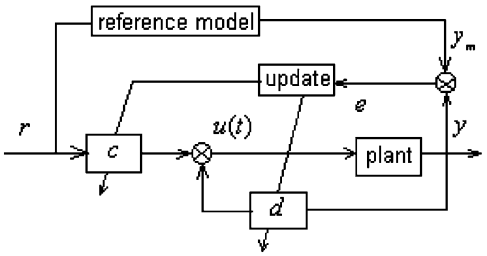


Fig. 1. Structure of MRAC systems.

III. Neurofuzzy Network Implementation of MRAC on Nonlinear Plant

A. Family of Nonlinear Model

Consider the following input–output model for a class of discrete nonlinear dynamical systems [4]:

$$\begin{aligned} y(k+1) &= a_0 y(k) + \cdots + a_{n_y-1}y(k-n_y+1) \\ &\quad + f[u(k), u(k-1), \ldots, u(k-m+1)] + \xi(k) \end{aligned} \tag{5}$$

where $u(k)$ and $y(k)$ are, respectively, the control and the output, $m$and n are the known orders of the system, $\xi(k)$ is a sequence of independent identically distributed random variables with zero mean and a variance of σ^2, and $f[\,\cdot\,]$ is a smooth nonlinear function such that a Taylor series expansion exists. It is assumed that

$$\frac{\partial f}{\partial u(k+1)} = 0; \quad \frac{\partial f}{\partial u(k)} \neq 0. \tag{6}$$

From (6), the Jacobian of $f[\,\cdot\,]$ exists. Rewriting (5) gives

$$A(z^{-1})y(k) = z^{-1}f[u(k), u(k-1), \ldots, u(k-m+1)] + \xi(k). \tag{7}$$

The local linearized model of (5) at the selected operating point $O(t)$ is given as follows, which is used to design the linear MRAC:

$$\bar{A}(z^{-1})y(k) = z^{-1}\bar{B}(z^{-1})u(k) + \xi(k) \tag{8}$$

where $\bar{A}(z^{-1})$ and $\bar{B}(z^{-1})$ are polynomials in $\mathrm{z^{-1}}$, the backward shift operator, with the respective orders n_y' and n_u', and the coefficients of $\bar{A}(z^{-1})$ and $\bar{B}(z^{-1})$ are a function of the operating point $O(t)$ [18]. Let

$$\left.\begin{aligned} x_1(k) &= y(k-n_y+1) \\ &\vdots \\ x_{n_y-1}(k) &= y(k-1) \\ x_{n_y}(k) &= y(k) \end{aligned}\right\}. \tag{9}$$

Equation (5) can be rewritten in state-space form as follows:

$$\begin{aligned} X(k+1) &= AX(k) + Bf[u(k), u(k-1), \ldots, u(k-m+1)] \\ y(k) &= CX(k) \end{aligned} \tag{10}$$

where $X = [x_1\ x_2\ \cdots\ x_{n_y}]^T \in R^n$ is the state vector, and

$$A = \begin{bmatrix} 0 & 1 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & & \cdots \\ 0 & \cdots & \cdots & 0 & 1 \\ a_0 & a_1 & \cdots & \cdots & a_{n_y-1} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 1 \end{bmatrix}$$

$$C = [0 \quad 0 \quad \cdots \quad \cdots \quad 1].$$

B. Implementation of MRAC Using Neurofuzzy Networks

The block diagram for implementing the MRAC using a neurofuzzy network is shown in Fig. 2. A schematic diagram of the neurofuzzy network is shown in Fig. 3. The input of the network consists of the state and reference input $[y(k)\cdots y(k-n_y+1), r(k)\cdots r(k-n_r+1)]$, which is fuzzified using univariate B-spline basis functions. The output of the network is a linear combination of the weights and the fuzzified input. The design of the B-spline neurofuzzy network involves specifying the order of the univariate basis functions v, the range of the input and output variables, and the number of inner knots w, or simply the number of basis functions to be used in the fuzzification process. Similar to neural networks, the weights of the neurofuzzy networks can be trained from experimental data using, for example, the gradient-descent method.

The advantage of using B-spline functions is that the output of the B-spline function can be computed by a recurrence relationship for given v and w. In addition, they have the following desirable properties [10].

1) The B-spline functions are defined on a bounded support and the output of the basis function is compact and positive over its supports, i.e., $\mu_l^j(x) = 0, x \notin [\lambda_{j-l}, \lambda_j]$ and $\mu_l^j(x) > 0, x \in (\lambda_{j-l}, \lambda_j)$. Therefore, the B-spline neurofuzzy network, stores information locally and can be trained from local data, and also only a small number of basis functions are involved in computing the output of the network.
2) The basis functions form a partition of unity, giving that the sum of the output of the basis functions is always one, i. e., $\sum_j \mu_l^j(x) \equiv 1, x \in [x_{\min}, x_{\max}]$.
3) The basis functions are members of the continuity class $C^{k-2}(x_{\min}, x_{\max})$, such that $\mu_l^j(x)$ and its derivatives up to the $(k-2)^{\text{nd}}$ order exist on $(x_{\min}, x_{\max})$.
4) The upper and lower bounds of the output of the neurofuzzy network are finite.

As the output of B-spline neurofuzzy network can be considered as a weighted sum of the outputs of several linear self-tuning controllers designed at several selected operating points, it can also be interpreted conceptually as a nonlinear self-tuning controller. As the fuzzy sets in the neurofuzzy networks are distributed over the neighborhood regions, the control obtained from the neurofuzzy controller is generally smooth.

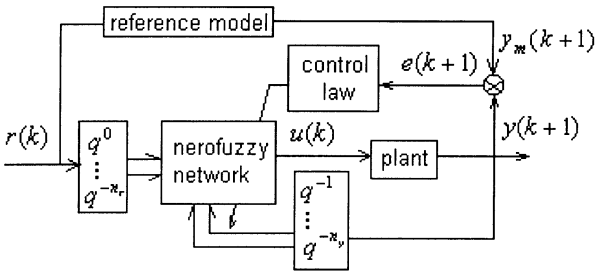


Fig. 2. MRAC implemented by neurofuzzy networks.

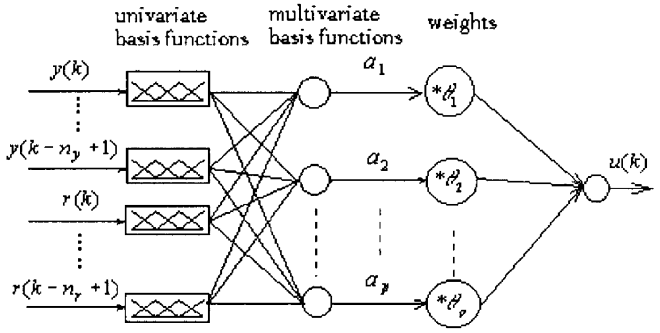


Fig. 3. B-spline neurofuzzy network.

The neurofuzzy network shown in Fig. 3 is derived from a set of fuzzy rules [10], an example of which is given as follows:

Ri: IF $y(k)$ is negative large, ..., and $\mathrm{y}(\mathrm{k}-\mathrm{n_y}+1)$ is negative medium, and $r(k)$ is positive medium, ..., and $\mathrm{r}(\mathrm{k}-\mathrm{n_r}+1)$ is positive medium
THEN $\mathrm{u}(\mathrm{k})$ is positive medium;

The output of the network $\mathrm{u}(\mathrm{k})$ is the sum of the inference from these rules, given as

$$u(t) = a^T(x)\theta \tag{11}$$

where $\mathrm{x}(\mathrm{k})$ is the input vector

$$\mathrm{x}(k) = [y(k), \ldots, y(k-n_y+1), r(k), \ldots, r(k-n_r+1)] \tag{12}$$

and where $\theta = [\theta_1 \cdots \theta_2 \cdots \theta_p]^T$ is the weight vector of the network and p is the total number of weights in the network. For a given order of the basis functions v_i and the number of inner knots w_i, p is given by

$$p = \prod_{i=1}^{n} (w_i + v_i). \tag{13}$$

The multivariate basis function is the transformed input vector obtained by the tensor products of the output of the univariate B-spline basis functions $\mu_{A_l^i}(x_l(k))$, i.e.,

$$a_i(x) = \prod_{l=1}^{n} \mu_{A_l^i}(x_l(k)), \quad \text{for } i = 1, 2, \ldots, p \tag{14}$$

where $n = n_y + n_r$ is the dimension of the input vector $\mathrm{x}(\mathrm{k})$. From (14), the properties of the univariate B-spline basis functions also apply to the multivariate B-spline basis functions.

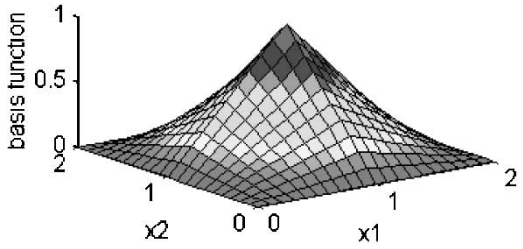


Fig. 4. Multivariate B-spline basis function.

Since these functions are defined on a hyperrectangle, the output of the network is also bounded, and the output is positive inside this domain, but is zero otherwise. An example of a multivariate basis function, which is formed from two second-order univariate basis function is shown in Fig. 4.

From (11), the control $\mathrm{u(k)}$ obtained from the neurofuzzy network can also be expressed as

$$u(k) = \sum_{i=1}^{p} \theta_i a_i(x) = \sum_{i=1}^{p} \theta_i \prod_{l=1}^{n} \mu_{A_k^i}(x_l(k)). \tag{15}$$

C. Training of the Neurofuzzy Networks

The training of the neurofuzzy network involves adjusting the importance of the relevance fuzzy rule, or the weights of the network. For the cost function given by (3), the weights in the network can be updated using (4), given as follows:

$$\begin{aligned} \theta_i(k+1) &= \theta_i(k) - g\frac{\partial J}{\partial \theta_i} \\ &= \theta_i(k) - ge\frac{\partial e}{\partial \theta_i}, \quad \text{for } i = 1, 2, \ldots, p \end{aligned} \tag{16}$$

where

$$\frac{\partial e}{\partial \theta_i} = \frac{\partial y}{\partial \theta_i} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial \theta_i} = \frac{\partial y}{\partial u} a_i(x). \tag{17}$$

From the compactness property of B-spline basis functions, the input space of the neurofuzzy network is separated into q regions [12], where q is given by

$$q = (w_y + 1)^{n_y}(w_r + 1)^{n_r} \tag{18}$$

where w_y and w_r are, respectively, the number of inner knots for $y(k)$ and $r(k)$. Only one region is activated each time by the input $x(k)$, giving the elements of the fuzzified input associated with the region that has been activated are nonzero, whilst those in the other regions are zero. The number of nonzero elements in $a(x(k))$ is given by [12]

$$p' = v_y^{n_y} v_r^{n_r} \tag{19}$$

where v_y and v_r are, respectively, the order of the basis functions for $y(k)$ and $r(k)$. As an illustration of this property, consider a network with two input variables, as shown in Fig. 5. Seven linguistic variables are used in the first input x_1, and six in the second variable x_2. These linguistic variables are denoted respectively by "positive large (PL)", "positive medium (PM)", "positive small (PS)", "zero (O)", "negative small (NS)", "negative medium (NM)", "negative large (NL)". Second-order B-spline basis functions are chosen as the membership functions of these linguistic variables. In this example, $v_1 = v_2 = 2, w_1 = 5$ and $w_2 = 4$. From (13), the number of weights in the network is 42, whilst from (18), the number of regions q is 30. As indicated by the shaded region in Fig. 5, four fuzzy rules are activated each time. The antecedent of this four rules could be expressed as follows:

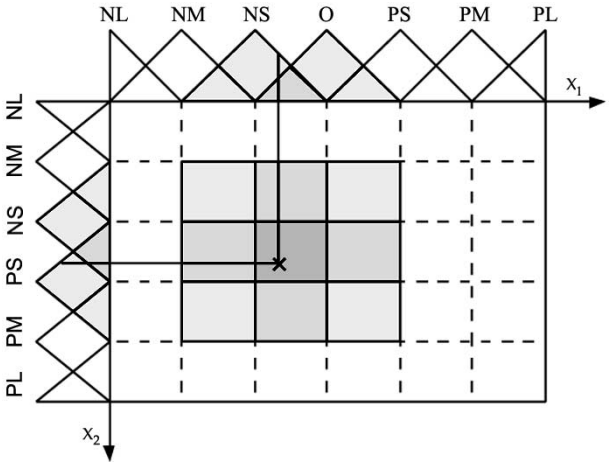


Fig. 5. Local change property of neurofuzzy networks.

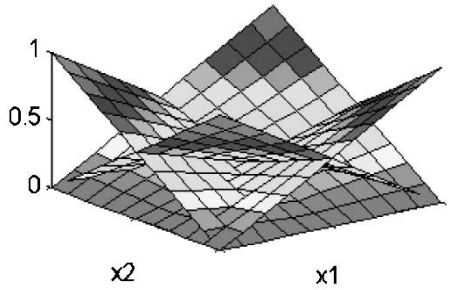


Fig. 6. Multivatiate membership function.

- if x_1 is "negative small" and x_2 is "negative small";
- if x_1 is "negative small" and x_2 is "positive small";
- if x_1 is "zero" and x_2 is "negative small";
- if x_1 is "zero" and x_2 is "positive small."

The multivariate membership functions are shown in Fig. 6.

IV. Properties of the Neurofuzzy Controller

The proposed neurofuzzy controller can be decomposed into two parts: static nonlinear parts, which is a topology conserving map, and an adaptive linear part, which is learned in the same manner as neural networks. Much work has already been published in the literature on the analysis of fuzzy controllers. In this section, a brief description of the relationship between fuzzy control and traditional control is presented.

Fuzzy controllers are developed initially as a two-dimensional (2-D) controller with error (difference between the set point and the output) and the change in error as the input. This controller structure is devised from the way an experienced operator controls the plant. His control strategy is formulated as the following step-respond characteristic: "if the plant output is far away and moving away from the set point, then increase the control value to bring the output back to the set point".

In general, a 2-D fuzzy controller with linear control rules is equivalent to the sum of a global 2-D multilevel relay and a local nonlinear PI controller [19]. Similarly, a three-dimensional fuzzy controller with linear control rules is the sum of a global three-dimensional multilevel relay and a local nonlinear PID controller [20]. A fuzzy controller with normally distribution membership function is essentially equivalent to one with a linear and a nonlinear controller [21], where the nonlinear part is asymptotically linear, if the number of fuzzy sets is large enough. As discussed in previous sections, neurofuzzy controllers can be considered as nonlinear self-tuning controllers. If the number of linear fuzzy rules is sufficiently large, it can be shown that in the limit the defuzzified output can be approximated by a linear function of the controller input [22]. This result can also be extended to fuzzy controllers with multi-input [23].

Due to the nonlinearity present both within the plant and the neurofuzzy controller, global stability of the closed-loop system is difficult to obtain. However, under certain conditions, the local stability of the closed-loop system can be established. The state-space (10) can be rewritten as

$$\begin{aligned} X(k+1) = {} & AX(k) + Bf \\ & \left[\sum_{i=1}^{p}\theta_i \prod_{l=1}^{n}\mu_{A_l^i}(x_l(k)), \sum_{i=1}^{p}{}^{(-1)}\theta_i \right. \\ & \times \prod_{l=1}^{n}\mu_{A_l^i}(x_l(k-1)), \ldots, \sum_{i=1}^{p}{}^{-(m-1)}\theta_i \\ & \left. \times \prod_{l=1}^{n}\mu_{A_l^i}(x_l(k-m+1))\right] \end{aligned} \tag{20}$$

where the weights θ_i obtained in the previous updating steps are denoted by the superscripts $(-1), \ldots, -(m-1)$ to its left. Let the set point, $r = 0$, and $\bar{X} = 0$ be the equilibrium point of the system. Now choose a Lyapunov function $V\{X(k)\}$ defined on the compact set S as:

$$V\{X(k)\} = \frac{1}{2}E^T E \tag{21}$$

where $E = X - \bar{X} = X$.

Then

$$V\{X(k)\} = \frac{1}{2}X^T X = \frac{1}{2}\{y(k-n+1)^2 + \cdots + y(k)^2\} \tag{22}$$

and

$$V\{X(k+1)\} = \frac{1}{2}\{y(k-n+2)^2 + \cdots + y(k+1)^2\} \geq 0. \tag{23}$$

For small enough $\Delta\theta_i$, i.e., $|\Delta\theta_i| \leq d, \forall d \geq 0$, the change in the Lyapunov function $V\{X(k)\}$ is given by

$$\begin{aligned} & V\{X(k+1)\} - V\{X(k)\} \\ & \quad \cong \sum_i \frac{\partial V\{X(k+1)\}}{\partial \theta_i}\Delta\theta_i \\ & \quad = y(k+1)\sum_i \frac{\partial y(k+1)}{\partial u(k)}\frac{\partial u(k)}{\partial \theta_i}\Delta\theta_i. \end{aligned} \tag{24}$$

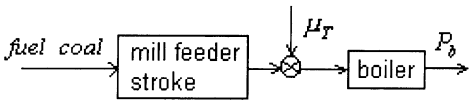


Fig. 7. Controlling steam pressure by fuel consumption and/or steam flow.

Substituting (16) and (17) into (24) gives

$$\begin{aligned} & V\{X(k+1)\} - V\{X(k)\} \\ & \quad \cong y(k+1)\sum_i \frac{\partial y(k+1)}{\partial u(k)}\frac{\partial u(k)}{\partial \theta_i}(-g)e(k+1)\frac{\partial e(k+1)}{\partial \theta_i} \\ & \quad = (-g)y(k+1)^2\sum_i \frac{\partial y(k+1)}{\partial u(k)}\frac{\partial u(k)}{\partial \theta_i}\frac{\partial y(k+1)}{\partial u(k)}\frac{\partial u(k)}{\partial \theta_i} \\ & \quad = (-g)y(k+1)^2\sum_i \left(\frac{\partial y(k+1)}{\partial u(k)}\right)^2\left(\frac{\partial u(k)}{\partial \theta_i}\right)^2 \\ & \quad = (-g)y(k+1)^2\sum_i \left(\frac{\partial y(k+1)}{\partial u(k)}\right)^2 (a_i(x))^2 \leq 0 \end{aligned} \tag{25}$$

where $(\partial y(k+1))/(\partial u(k)) = (\partial f)/(\partial u(k))$ is the Jacobian of the nonlinear function $f[\cdot]$ in the nonlinear model (5), which is assumed to exist. Consequently, the state converges to the equilibrium point under neurofuzzy control law (15) if the learning rate g satisfies the condition $g \leq \lambda$ to ensure $\Delta\theta_i$ to be small enough, though the convergence may be slow, and hence the local stability of the system around the equilibrium point is established.

V. Case Studies

Two examples are presented to illustrate the implementation and the performance of the neurofuzzy network-based MRAC. Comparison with linear MRAC is also made. In example 1, the local linear model of the combustion process of a 300-MW steam-boiler power-generation system under a changing load condition is considered, and in example 2, a noisy nonlinear system is used.

A. Example 1: Local Linear System of a Steam-Boiler Power-Generation System

The fuel combustion system is an important part of a fossil fuel power generation plant to produce high pressure steam that drives the steam-turbine generator. The main objective of the boiler control system is to maintain the outlet steam pressure at a desired level under varying load conditions or subject to other types of disturbances. The steam pressure is controlled by adjusting the amount of coal to be delivered to the furnace of the boiler and load is controlled by adjusting the steam valve that controls the steam flow going into the steam turbine, as shown in Fig. 7. For an increase in the load, the control valve μ_T will be opened immediately to increase the steam flow to the steam turbine, and thus increases the output power of the steam turbine. The steam pressure will be decreased. The transient process of the steam pressure P_b is shown in Fig. 8.

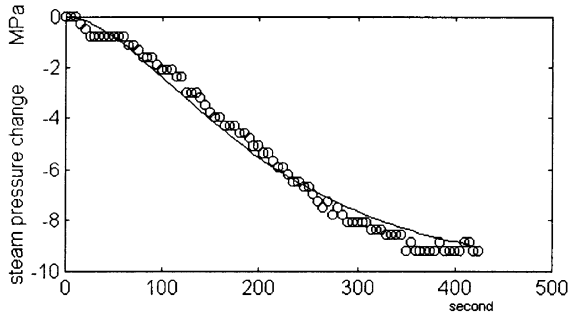


Fig. 8. Steam pressure for an increase in steam valve setting.

For a constant fuel consumption, the transfer function with steam pressure P_b as the output and the steam valve setting μ_T as the control is a second-order linear representation [2]

$$\frac{P_b(s)}{\mu_T(s)} = G(s) = \frac{K_p}{(T_1 s+1)(T_2 s+1).} \tag{26}$$

Rewrite (26) as

$$G(s) = \frac{K}{s^2 + as + b} \tag{27}$$

where $a = (T_1 + T_2)/T_1T_2, b = 1/T_1T_2, K = K_p/T_1T_2$. Note that K_p, T_1, and T_2 are a function of the operating point. From the data shown in Fig. 8, which are obtained when the system is operating at a load condition of 250 MW, the estimated parameters of (26) are $T_1 = 120$ s $T_2 = 100$ s, $K_p = 0.01$ kg/s/%.

The conventional model reference adaptive controller is chosen to be

$$u = \theta_1 r - \theta_2 y - \theta_3 \dot{y}. \tag{28}$$

From (27) and (28), the closed-loop output is given by

$$P_b(s) = \frac{K\theta_1}{s^2 + (a + K\theta_3)s + (b + K\theta_2)} \mu_T(s). \tag{29}$$

Let the reference model be chosen as

$$G_m(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{30}$$

where ξ and ω_m are the designed parameters. In this example, these parameters are chosen to be $\xi = 0.707$ $\omega_n = 2$. The parameters of the controller (28) are updated by the following equations:

$$\begin{aligned} \frac{d\theta_1}{dt} &= -ge\frac{r}{h^2 + 2\xi\omega_n h + \omega_n^2} \\ \frac{d\theta_2}{dt} &= ge\frac{y}{h^2 + 2\xi\omega_n h + \omega_n^2} \\ \frac{d\theta_3}{dt} &= ge\frac{hy}{h^2 + 2\xi\omega_n h + \omega_n^2} \end{aligned} \tag{31}$$

where h is the differential operator. The initial values of the estimated parameters are chosen to be: $\theta_{10} = \theta_{20} = \theta_{30} = 0.1$, and $g = 1$. The closed-loop response of steam pressure P_b is shown by the dashed line in Fig. 9 for a square-wave change in the set point. After a few cycles in the change of the set point, the controller is trained, giving good closed-loop response with a short setting time and small overshoots [see also the tracking

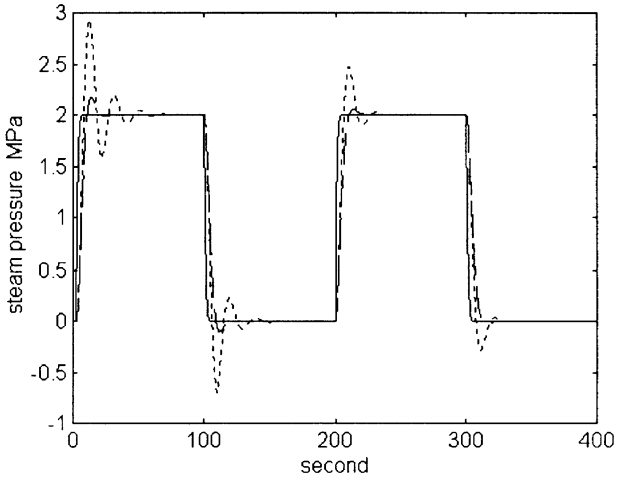


Fig. 9. Closed-loop output using linear MRAC and neurofuzzy network-based MRAC.

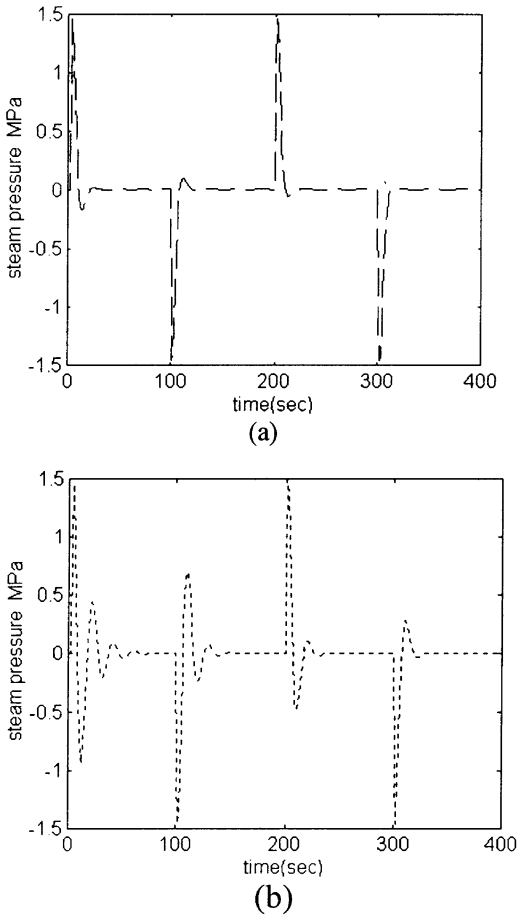


Fig. 10. Tracking error of (a) linear MRAC (dashed line) and (b) neurofuzzy network-based MRAC (dotted line).

error shown in Fig. 10(a)]. From the simulation result, the linear MRAC is quite acceptable for controlling the steam pressure under fixed working condition in practice.

The MRAC is then implemented using a neurofuzzy network, as described in Section IV. Each of the input is fuzzified by two triangular basis functions, representing the linguistic variables: "small" and "large". In this case, $v_y = v_{\dot{y}} = v_r = 2$, and

$w_y = w_{\dot{y}} = w_r = 0$. From (13), the number of weights of the neurofuzzy network p is: $p = 2^3 = 8$. The range of $y(t), \dot{y}(t)$ and $r(t)$ is chosen to be between -1 and 3. The initial weight $\hat{\theta}(0)$ is set to 0.1, same as that for the linear MRAC controller.

The closed-loop output using the MRAC based on the neurofuzzy network is shown by the dotted line in Fig. 9, and the tracking error is shown in Fig. 10(b). During the initial learning period, the overshoot is large, but becomes smaller as the training of the neurofyzzy network is progressed. Indeed, as the training increases, the perfromance of the neurofuzzy network-based MRAC is close to the linear one. The reason for the neurofuzzy newtwork-based MRAC to take longer to train is mainly due to the larger number of the parameters in the proposed controller.

B. Example 2: Nonlinear Model

Consider the following nonlinear system [4]:

$$y(k) = 0.3y(k-1) + 0.6y(k-2) + [u(k-1)]^{1/3} + \xi(k) \quad (32)$$

where $\xi(k)$ is a normally distributed white noise with zero mean and a variance of 0.1^2. The linear MRAC with the input $x(k) = [y(k), y(k-1), y(k-2), r(k)]$ is applied first to control the system. The reference input is again a square wave, and the following equations are used to update the parameters of the MRAC:

$$\begin{aligned}
\theta_1(k+1) &= \theta_1(k) - ge(k)r(k)[u(k-1)]^{-\frac{2}{3}} \\
\theta_2(k+1) &= \theta_2(k) - ge(k)y(k)[u(k-1)]^{-\frac{2}{3}} \\
\theta_3(k+1) &= \theta_3(k) - ge(k)y(k-1)[u(k-1)]^{-\frac{2}{3}} \\
\theta_4(k+1) &= \theta_4(k) - ge(k)y(k-2)[u(k-1)]^{-\frac{2}{3}}. \quad (33)
\end{aligned}$$

The initial values of these parameters are $\theta_{10} = \theta_{20} = \theta_{30} = \theta_{40} = 0.1$, and $g = 0.2$. The closed-loop output is poor with large overshoots and oscillations, as shown in Fig. 11.

The neurofuzzy network-based MRAC is implemented with the same input as in the linear case. Again, two triangular basis functions are used for each of the input, which represent the linguistic variables "small" and "large". In this case, $v_{y(k)} = v_{y(k-1)} = v_{y(k-2)} = v_{r(k)} = 2$, and $w_{y(k)} = w_{y(k-1)} = w_{y(k-2)} = w_{r(k)} = 0$. From (13), the number of weights of the neurofuzzy network is $p = 2^4 = 16$. The range of $y(k), y(k-1), y(k-2)$ and $r(k)$ is chosen to be between 0 and 12. The initial weight $\theta(0)$ is set to the same initial values as that for the linear MRAC, and the update of the weights is given by

$$\theta_i(k+1) = \theta_i(k) - gea_i(x(k))[u(k-1)]^{-\frac{2}{3}}, \quad \text{for } i = 1, 2, \ldots, 16. \quad (34)$$

The closed-loop output using the neurofuzzy network-based MRAC is shown in Fig. 12. In this case, the overshoots and the oscillations are much smaller, yielding a much better performance than the linear MRAC. The small oscillations that occur at the steady-state level of the closed-loop output arises from poor approximation of the nonlinearity in the system, as only two basis functions are used for each of the input. To reduce these oscillations, five triangular basis functions are used next, which represent the linguistic variables "positive large", "positive medium", "zero", "negative medium", and "negative large".

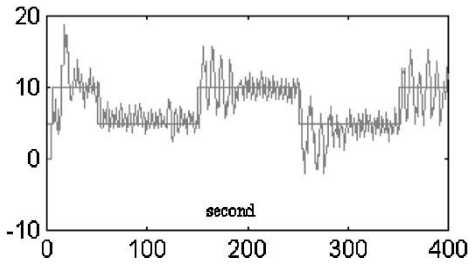


Fig. 11. Closed-loop output of nonlinear system with linear MRAC.

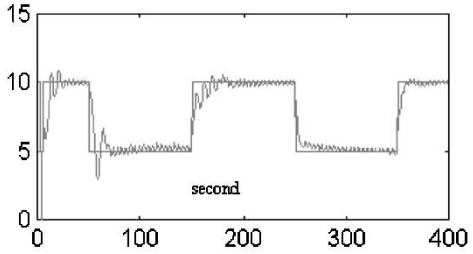


Fig. 12. Closed-loop output of nonlinear system with neurofuzzy network-based MRAC.

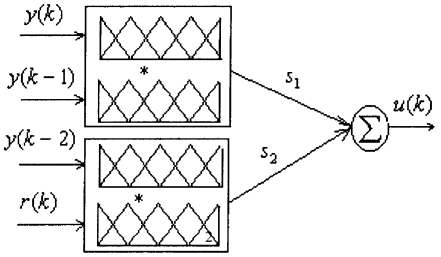


Fig. 13. Neurofuzzy controller implemented by lower-dimension submodels.

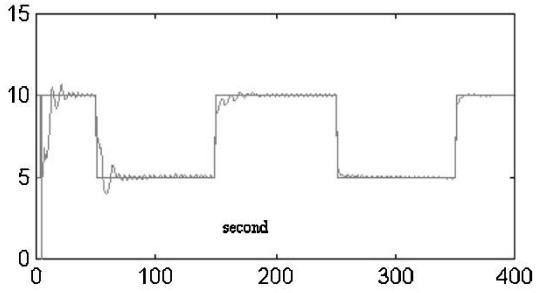


Fig. 14. Closed-loop output using the approximated MRAC.

With a typical B-spline network, this would require $5^4 = 625$ storage locations and each input would activate $2^4 = 16$ basis functions. We make the controller additively decomposed. The following approximation is used to implement the neurofuzzy network-based MRAC:

$$u(k) = s_1[y(k), y(k-1)] + s_2[y(k-2), r(k)]. \quad (35)$$

The neurofuzzy network for implementing the controller (35) is a linear combination of two 2-D subnetworks as shown in Fig. 13. The advantage of this approximation is that the number of weights is reduced drastically to 50 for each of the input. With the increase in the resolution of the fuzzification of the input, the steady-state oscillations are greatly reduced, as shown in Fig. 14.

VI. Conclusion

A neurofuzzy network-based MRAC is derived in this paper. Neurofuzzy networks are chosen here not only because of their ability to approximate nonlinear functions with arbitrary accuracy, but also the weights of the network can be readily updated online. The implementation of the neurofuzzy network-based MRAC is presented, and the local stability derived. The implementation and the performance of the proposed controller are demonstrated by two examples. The first example involves a local linear model of the steam pressure in the boiler of a 300-MW power-generation plant, whilst in the second, a nonlinear system. It is shown that on linear system, the performance of the neurofuzzy network-based MRAC is comparable to the linear MRAC, but longer learning time is required, as the number of weights is larger. However, the performance of the proposed controller is superior to the linear MRAC controller in the case of the nonlinear system. It is also shown that increasing the resolution of the fuzzification of the input can reduce oscillations in the closed-loop output. The problem of "curse of dimensionality" can be reduced using lower dimension submodels to implement the nonlinear controller.

References

[1] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] X.-J. Liu, Y.-M. Peng, and X. Zhou, "Identification of boiler models and its fuzzy logic strategy," in *Proc. 14th IFAC World Congr.*, vol. O, Beijing, China, July 1999, pp. 149–154.

[3] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, Jan. 1990.

[4] S. Bittanti and L. Piroddi, "GMV technique for nonlinear control with neural networks," *Proc. Inst. Elect. Eng.—Control Theory Applications*, vol. 141, no. 2, pp. 57–69, Mar. 1994.

[5] M. Yuan, A. N. Poo, and G. S. Hong, "Direct neural control system: Nonlinear extension of adaptive control," *Proc. Inst. Elect. Eng.—Control Theory Applications*, vol. 142, no. 6, pp. 661–667, Nov. 1995.

[6] G. Lightbody and G. W. Irwin, "Direct neural model reference adaptive control," *Proc. Inst. Elect. Eng.—Control Theory Applications*, vol. 142, no. 1, pp. 31–43, Jan. 1995.

[7] T. K. Yin and C. S. G. Lee, "Fuzzy model-reference-adaptive-control," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 1606–1615, Dec. 1995.

[8] C. S. Chen and W. L. Chen, "Robust model-reference-adaptive-control of nonlinear systems using fuzzy systems," *Int. J. Syst. Sci.*, vol. 27, no. 12, pp. 1435–1442, 1996.

[9] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Pretice-Hall, 1992.

[10] M. Brown and C. J. Harris, *Neurofuzzy Adaptive Modeling and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[11] J. Zhang and A. J. Morris, "Fuzzy neural networks for nonlinear systems modeling," *Proc. Inst. Elect. Eng.—Control Theory Applications*, vol. 142, no. 6, pp. 551–561, Nov. 1995.

[12] C. W. Chan, X.-J. Liu, and W. K. Yeung, "Neurofuzzy network based self-tuning control with offset elimination," *Int. J. Syst. Sci.*, vol. 34, no. 2, pp. 111–122, 2003.

[13] C. W. Chan, K. C. Cheung, and W. K. Yeung, "A computation-efficient on-line training algorithm for neurofuzzy networks," *Int. J. Sys. Sci.*, vol. 31, no. 3, pp. 297–306, 2000.

[14] S. G. Kong and B. Kosko, "Adaptive fuzzy systems for backing up a truck-and-trailer," *IEEE Trans. Neural Networks*, vol. 3, pp. 211–223, Mar. 1992.

[15] W. Pedrycz, "Processing in relational structures: Fuzzy relational equations," *Fuzzy Sets Syst.*, vol. 40, no. 1, pp. 77–106, 1991.

[16] D. E. Serborg, T. F. Edgar, and D. A. Mellichamp, *Process Dynamics and Control*. New York, NY: Wiley, 1989.

[17] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, Feb. 1985.

[18] S. Chen and S. A. Billings, "Representations of nonlinear systems: The NARMAX model," *Int. J. Control*, vol. 49, no. 3, pp. 1013–1032, 1989.

[19] H. Ying, "A nonlinear fuzzy controller with linear control rules is the sum of a global two-dimensional multilevel relay and a local nonlinear proportional-integral controller," *Automatica*, vol. 29, no. 2, pp. 499–505, 1993.

[20] X.-J. Liu, H.-S. Zhang, and T.-Y. Chai, "Structure analysis of three-dimensional fuzzy controller and its relationship to PID controller," *Proc. 36th IEEE Conf. Decision and Control*, pp. 3350–3351, Dec. 10–12, 1997.

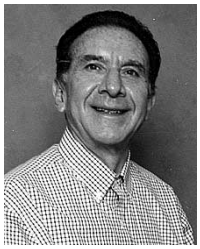
[21] X.-J. Liu and X. Zhou, "Structure analysis of fuzzy controller with gaussian membership function," in *Proc. 14th IFAC World Congr.*, vol. K, Beijing, China, July 5–9, 1999, pp. 201–206.

[22] H. Ying, "General analytical structure of typical fuzzy controller and their limiting structure theorems," *Automatica*, vol. 29, no. 4, pp. 1139–1143, 1993.

[23] C.-C. Wong, C.-H. Chou, and D.-L. Mon, "Studies on the output of fuzzy controller with multiple inputs," *Fuzzy Sets Syst.*, vol. 57, no. 2, pp. 149–158, 1993.

Xiang-Jie Liu received the Ph.D. degree in electrical and electronic engineering from the Research Center of Automation, Northeastern University, Shenyang, China, in 1997. He was then in the postdoctor program in the Electric Power Research Institute (EPRI), Beijing, China, until 1999.

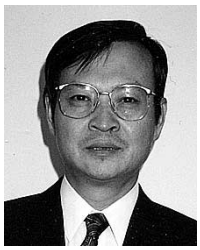
He was an Associate Professor at EPRI and a Research Associate with the University of Hong Kong. He joined the National University of Mexico, Mexico City, in 2001, where he is now a Professor with the Research Center of Instrumentation. His current research areas include fuzzy control, neural network, filtering, intelligent control theory and its application in the industrial process, and the development and application of DCS.



Felipe Lara-Rosano received the M.S. and Ph.D. degrees in mechanical and electrical engineering from the National University of Mexico, Mexico City, in 1970 and 1973, respectively. He received an honorary doctorate from the International Institute for Advanced Systems Research and Cybernetics, Windsor, ON, Canada.

He has been Director of the Center for Applied Science and Technological Development, University of Mexico, since 1997. His current research interests include artificial intelligence, expert systems, fuzzy logic, and neural networks.

Dr. Lara-Rosano is a member of the New York Academy of Sciences, the Mexican Academy of Sciences, the Mexican Academy of Engineering, and the Mexican Academy of Technology. He is a Fellow and Board Member of the International Institute for Advance Systems Research and Cybernetics.



C. W. Chan received the M.Sc. and Ph.D. degrees in control from the Institute of Science and Technology, University of Manchester, Manchester, U.K.

He has previously been with Weir Pumps Ltd., the National Engineering Laboratory, and Unilever Research Port Sunlight Laboratory. He is currently with the Department of Mechanical Engineering, University of Hong Kong. His research interests include design and analysis of compensation for actuator saturation, fuzzy logic, neural networks, adaptive neurofuzzy controllers, and fault detection and isolation. He is Regional Editor (Asia Pacific) of the *International Journal of Systems Science*.