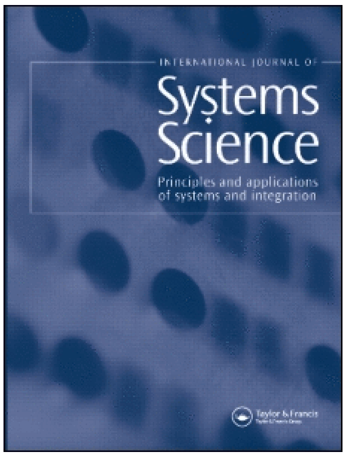

International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713697751

Modelling, control, and stability analysis of non-linear systems using generalized fuzzy neural networks

Yang Gao [a]; Meng Joo Er [a]
[a] Instrumentation and System Engineering Lab, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Online Publication Date: 15 May 2003

To cite this Article Gao, Yang and Er, Meng Joo(2003)'Modelling, control, and stability analysis of non-linear systems using generalized fuzzy neural networks',International Journal of Systems Science,34:6,427 — 438
To link to this Article: DOI: 10.1080/00207720310001612855
URL: http://dx.doi.org/10.1080/00207720310001612855

International Journal of Systems Science
volume 34, number 6, 15 May 2003, pages 427–438

Modelling, control, and stability analysis of non-linear systems using generalized fuzzy neural networks

YANG GAO* and MENG JOO ER

This paper presents an adaptive fuzzy neural controller (AFNC) suitable for modelling and control of MIMO non-linear dynamic systems. The proposed AFNC has the following salient features: (1) fuzzy neural control rules can be generated or deleted dynamically and automatically; (2) uncertain MIMO non-linear systems can be adaptively modelled on line; (3) adaptation and learning speed is fast; (4) expert knowledge can be easily incorporated into the system; (5) the structure and parameters of the AFNC can be self-adaptive in the presence of uncertainties to maintain a high control performance; and (6) the asymptotical stability of the system is established using the Lyapunov approach. Simulation studies on a two-link robot manipulator show that the performance of the proposed controller is better than that of some existing fuzzy/neural methods.

1. Introduction

In the last few decades, much research effort has been directed towards the design of intelligent controllers using fuzzy logic and neural networks. Fuzzy logic provides human reasoning capabilities to capture uncertainties, which cannot be described by precise mathematical models. Neural networks offer exciting advantages such as adaptive learning, parallelism, fault tolerance, and generalization. They have been proven to be very powerful techniques in the area of system modelling and control, especially when the system is difficult to model mathematically, or when the system has large uncertainties and strong non-linearities. Therefore, fuzzy logic and neural networks have been greatly adopted in model-free adaptive control of non-linear dynamic systems (Nie and Linkens 1995, Wang 1997). Moreover, a few hybrid techniques were applied to adaptation of parameters in fuzzy or neural controllers, such as sliding mode technique (Ohtani and Yoshimura 1996), Bayesian probability (MacKay 1995), genetic algorithms (Park *et al*. 1994), and various types of neural networks (Jin *et al*. 1995, Lin *et al*. 1999). However, it turns out that only adjustment of parameters will not be sufficient in many cases. For example, if the number of fuzzy rules or number of hidden layers and neurons is very large, real-time implementation will be difficult, if not impossible. More importantly, this reduces the flexibility and numerical-processing capability of the controller and results in redundant or inefficient computation. Therefore, the controller structure needs to be adaptive so that a compact fuzzy or neural control system can be obtained. Some techniques have been attempted to adapt fuzzy or neural control structure, such as genetic algorithms (Jin 1998), evolution strategies (Jin *et al*. 1999), Back-Propagation (BP) neural networks (Nie and Linkens 1995), wavelets (Cannon and Slotine 1995), and Kalman filter algorithms (Chak *et al*. 1998). However, they have difficulties in the initialization of control structures and the associated parameters, and the learning and adaptation speeds are slow due to the use of BP, wavelets, or Kalman filter algorithms.

This motivates us to investigate a new structure and parameter learning algorithm for constructing a fuzzy or neural control system systematically and automatically. The resulting intelligent controller must have a fast on-line adaptability to guarantee good real-time control performance. In line with this objective, a superior adaptive fuzzy neural controller (AFNC) is designed

Received 28 February 2003. Revised 3 June 2003. Accepted 6 August 2003.

Instrumentation and System Engineering Lab, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798.
* To whom correspondence should be addressed.
e-mail: ygao@pmail.ntu.edu.sg

and developed in this paper. The proposed AFNC is built based on a generalized fuzzy neural network (G-FNN). The G-FNN controller has the advantages of both fuzzy logic, e.g. human-like thinking and ease of incorporating expert knowledge, and neural networks, e.g. learning abilities and connectionist structures. In this way, we can incorporate low-level learning and computational power of neural networks into the fuzzy logic system on one hand and provide high-level human-like thinking and reasoning of fuzzy logic systems into neural networks on the other hand. The G-FNN algorithm offers a fast on-line learning algorithm, which can recruit or delete fuzzy control rules or neurons dynamically without predefinition. Its outstanding computational efficiency in terms of learning speed, adaptability, and generalization has been verified in some of our latest work (Gao and Er 2001; Er and Gao 2003). In essence, the G-FNN algorithm enables the G-FNN controller to successfully model the non-linear system dynamics and its uncertainties online.

The rest of the paper is organized as follows. Section 2 introduces the dynamic model of MIMO non-linear systems under consideration. Section 3 presents non-linear system modelling using the G-FNN, which includes the G-FNN architecture, learning algorithm and modelling method. This is followed by Section 4, which describes the design procedure of the G-FNN controller and the AFNC in details. Convergence of the G-FNN controller and global stability of the closed-loop system are proven using the Lyapunov theory. Section 5 presents the simulation results and discussions on a two-link robot manipulator. Finally, Section 6 concludes the paper. Detailed mathematical descriptions of the G-FNN learning algorithm are shown in the Appendix.

2. MIMO non-linear system dynamics

The class of nth-order MIMO non-linear systems considered in this paper, termed *companion from* or *controllability canonical form*, is given by (Slotine and Li 1991):

$$\mathbf{z}^{(n)} = \mathbf{F}(\underline{\mathbf{z}}) + \mathbf{G}(\underline{\mathbf{z}})\mathbf{u} + \mathbf{D}, \tag{1}$$

where $\mathbf{u} \in \Re^{n_i \times 1}$ and $\mathbf{z} \in \Re^{n_o \times 1}$ are the input and output vectors of the MIMO non-linear system respectively, with n_i and n_o being the total number of system inputs and outputs, respectively, $\underline{\mathbf{z}} = [\mathbf{z}^T \dot{\mathbf{z}}^T \cdots \mathbf{z}^{(n-1)T}]^T \in \Re^{n_o n \times 1}$ is the state vector of the system, $\mathbf{z}^{(n)} \in \Re^{n_o \times 1}$ denotes the nth derivative of the column vector $\mathbf{z}$, $\mathbf{z}^{(n-1)T}$ denotes the transpose of the column vector $\mathbf{z}^{(n-1)}$, $\mathbf{F}(\underline{\mathbf{z}}) \in \Re^{n_o \times 1}$ and $\mathbf{G}(\underline{\mathbf{z}}) \in \Re^{n_o \times n_i}$ represent smooth non-linearities of the dynamic system, and $\mathbf{D} \in \Re^{n_o \times 1}$ is an unknown function representing system uncertainties.

Assumption 2.1: *Since we require that the MIMO non-linear system* (1) *is controllable, the input gain* $\mathbf{G}(\underline{\mathbf{z}})$ *needs to be invertible for all* $\underline{\mathbf{z}} \in U_c \subset \Re^{n_o n \times 1}$. *Furthermore, the function* $\mathbf{G}$ *is assumed to be known and bounded, and the functions* $\mathbf{F}$ *and* $\mathbf{D}$ *are assumed to be bounded.*

3. Fuzzy neural modelling scheme

The G-FNN is a newly developed neural-network-based fuzzy logic control and decision system. The G-FNN learning algorithm provides an efficient way of constructing the G-FNN in real time and combining structure learning and parameter learning simultaneously within the FNN. The salient characteristics of the G-FNN are: (1) dynamic fuzzy neural structure; (2) fast on-line adaptation and learning ability; and (3) ease of incorporating expert knowledge. The G-FNN can serve as a generalized FNN for system modelling and control.

In the context of using the G-FNN directly for non-linear control, the G-FNN is viewed as a means of system modelling, or even a framework for knowledge representation. The knowledge about system dynamics and mapping characteristics are implicitly stored within the network. Therefore, training a G-FNN using input–output data from a non-linear system becomes a central issue to its use in control. In this section, the G-FNN architecture, learning algorithm, and modelling method will be elaborated.

3.1. *G-FNN architecture*

The G-FNN is constructed based on the extended RBF-NN, which is functionally equivalent to the TSK-type fuzzy inference system. The G-FNN (see figure 1) has a total of four layers. Layer one transmits values of the input linguistic variables x_i, $i = 1 \ldots N_i$ to the next layer directly. Layer two performs membership functions to the input variables. The membership function is chosen as a Gaussian function of the following form:

$$mf_{ij}(x_i) = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right], \tag{2}$$

where c_{ij}, $i = 1 \ldots N_i$, $j = 1 \ldots N_r$ and σ_{ij}, $i = 1 \ldots N_i$, $j = 1 \ldots N_r$ are, respectively, the centre (or mean) and the width (or variance) of the Gaussian function of the jth term of the ith input linguistic variable x_i. Layer three is the rule layer. The number of nodes (or neurons) in this layer indicates the number of

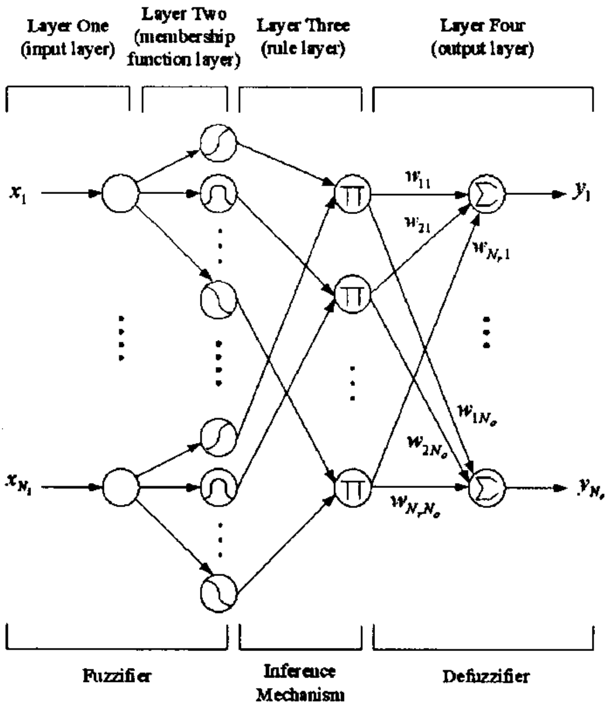

Figure 1. Architecture of the G-FNN.

fuzzy rules. The outputs are given by

$$\phi_j = \prod_{i=1}^{N_i} mf_{ij}. \tag{3}$$

Layer-four nodes define output linguistic variables. Each output variable y_k, $k = 1 \ldots N_o$ is the weighted sum of incoming signals, which performs the TSK-type defuzzification, i.e.

$$y_k = \sum_{j=1}^{N_r} \phi_j w_{jk}, \tag{4}$$

and the weight or the consequent is

$$w_{jk} = K_0^{jk} + K_1^{jk} x_1 + \ldots + K_{N_i}^{jk} x_{N_i}, \tag{5}$$

where K_is are real-valued parameters. It is not difficult to see that

$$\mathbf{y} = [y_1\ y_2 \cdots y_{N_o}]^T = \mathbf{W}^T \Phi, \tag{6}$$

where

$$\mathbf{W} = \begin{bmatrix} K_0^{11} K_1^{11} \ldots K_{N_i}^{11} \ldots\ldots K_0^{N_r1} K_{N_i}^{N_r1} \ldots K_{N_i}^{N_r1} \\ \vdots \qquad \vdots \qquad \vdots \\ K_0^{1N_o} K_1^{1N_o} \ldots K_{N_i}^{1N_o} \ldots\ldots K_0^{N_rN_o} K_1^{N_rN_o} \ldots K_{N_i}^{N_rN_o} \end{bmatrix}^T \tag{7}$$

$$\mathbf{\Phi} = \left[\phi_1\ \phi_1 x_1 \ldots \phi_1 x_{N_i} \ldots\ldots \phi_{N_r}\ \phi_{N_r} x_1 \ldots \phi_{N_r} x_{N_i}\right]^T. \tag{8}$$

3.2. *G-FNN learning algorithm*

The G-FNN algorithm is basically an on-line supervised structure and parameter learning algorithm which constructs the FNN automatically and dynamically. Parameters learning is performed by combining the semi-closed fuzzy set concept for the membership learning and the linear least-squared (LLS) method for weight learning. For structure learning, two criteria are proposed for generation of fuzzy rules, and an error reduction ratio (ERR) concept is proposed for pruning rules. Interested readers may refer to the Appendix for detailed mathematical descriptions of the G-FNN learning algorithm.

3.3. *Inverse modelling using G-FNN*

Inverse modelling of dynamical systems plays a crucial role in a range of control problems, which will become apparent in the next section. This paper attempts to model a system's inverse dynamics by the G-FNN. It can be easily derived from (1) that the inverse dynamics of the non-linear system is given by:

$$\mathbf{u}(\bar{\mathbf{z}}) = \mathbf{G}(\underline{\mathbf{z}})^{\dagger}[\mathbf{z}^{(n)} - \mathbf{F}(\underline{\mathbf{z}}) - \mathbf{D}] = \mathbf{\Omega}(\bar{\mathbf{z}}), \tag{9}$$

where $\mathbf{G}(\underline{\mathbf{z}})^{\dagger}$ is the pseudoinverse of $\mathbf{G}$ if $n_i \neq n_o$ or the inverse of $\mathbf{G}$ if $n_i = n_o$, and $\bar{\mathbf{z}} = [\underline{\mathbf{z}}^T\ \mathbf{z}^{(n)T}]^T \in \Re^{n_o(n+1)\times 1}$. The G-FNN is trained to obtain an estimate of the inverse dynamics, $\hat{\mathbf{\Omega}}$, i.e. the one-to-one mapping relationship from $\bar{\mathbf{z}}$ to $\mathbf{u}$. This is achieved by applying the G-FNN learning algorithm, which is capable of estimating the mapping relationship by determining the appropriate structure and parameters of a fuzzy neural system.

The resulting G-FNN can therefore be used to estimate the input signal $\mathbf{u}_{\text{G-FNN}}$ of the non-linear system given the desired output vector $\bar{\mathbf{z}}$. From (6) and (9), the output of the G-FNN can be shown to be

$$\mathbf{u}_{G-FNN}(\bar{\mathbf{z}}|\mathbf{W}) = \hat{\mathbf{\Omega}}(\bar{\mathbf{z}}|\mathbf{W}) = \mathbf{W}^T \mathbf{\Phi}(\bar{\mathbf{z}}), \tag{10}$$

where $\mathbf{W} \in \Re^{N_r(N_i+1)\times N_o}$ is the weight matrix of the G-FNN, $\mathbf{\Phi} \in \Re^{N_r(N_i+1)\times 1}$ is the regressor vector associated with $\bar{\mathbf{z}}$, $N_i = n_o(n+1)$ and $N_o = n_i$ are the number of inputs and outputs of the G-FNN, respectively, and N_r is the total number of fuzzy control rules inside G-FNN.

The optimal parameter $\mathbf{W}^*$ for a given regressor vector, $\mathbf{\Phi}(\bar{\mathbf{z}})$, is defined as follows:

$$\mathbf{W}^* \triangleq \arg\min\left[\sup_{\bar{z}\in U_c} \|\mathbf{\Omega}(\bar{\mathbf{z}}) - \hat{\mathbf{\Omega}}(\bar{\mathbf{z}}|\mathbf{W})\|\right]. \tag{11}$$

The *minimum approximation error vector*, $\varepsilon \in \Re^{n_i \times 1}$ can be defined as

$$\varepsilon \triangleq \mathbf{\Omega}(\bar{\mathbf{z}}) - \hat{\mathbf{\Omega}}(\bar{\mathbf{z}}|\mathbf{W}^*). \tag{12}$$

As a consequence, (9) can be established by the G-FNN as follows:

$$\mathbf{u}(\bar{\mathbf{z}}) = \hat{\mathbf{\Omega}}(\bar{\mathbf{z}}|\mathbf{W}^*) + \varepsilon \tag{13}$$

$$= \mathbf{W}^{*T}\mathbf{\Phi}(\bar{\mathbf{z}}) + \varepsilon. \tag{14}$$

Assumption 3.1: *In this paper, we assume that ϵ is upper-bounded by $|\varepsilon| \leqslant \varepsilon_N$, where ε_N is a known vector containing positive constants.*

4. Fuzzy neural control scheme

4.1. *Adaptive fuzzy neural controller (AFNC)*

The objective of this paper is to design an AFNC for MIMO non-linear systems in companion form (1), which guarantees boundedness of all closed-loop variables and tracking of a given desired signal $\mathbf{z}_d(t)$. We define the tracking error, e, as follows:

$$\mathrm{e} \triangleq \mathbf{z}_d - \mathbf{z} \in \Re^{n_o \times 1}. \tag{15}$$

To facilitate the following development, we define the dynamic tracking error, $\mathbf{s}$, as follows:

$$\mathbf{s} \triangleq \mathbf{z}_r^{(n-1)} - \mathbf{z}^{(n-1)} \in \Re^{n_o \times 1}, \tag{16}$$

where $\mathbf{z}_r^{(n-1)}$ is the so-called 'reference' value of $\mathbf{z}^{(n-1)}$ obtained by modifying $\mathbf{z}_d^{(n-1)}$ according to the tracking error as follows:

$$\mathbf{z}_r^{(n-1)} = \mathbf{z}_d^{(n-1)} + \mathbf{\Upsilon}_{n-2}\mathbf{e}^{(n-2)} + \ldots + \mathbf{\Upsilon}_0\mathbf{e}. \tag{17}$$

Here, $\mathbf{\Upsilon}_i \in \Re^{n_o \times n_o}$, $i = 0, 1, \ldots, n-2$ is a diagonal matrix whose entries are positive constants. Substituting (17) into (16), we have

$$\mathbf{s} = \mathbf{e}^{(n-1)} + \mathbf{\Upsilon}_{n-2}\mathbf{e}^{(n-2)} + \ldots + \mathbf{\Upsilon}_0\mathbf{e}. \tag{18}$$

To achieve asymptotically perfect tracking of a desired configuration $\mathbf{z}_d$ with all $\dot{\mathbf{z}}_d \ldots \mathbf{z}_d^{(n)}$ known and bounded, the perfect control law $\mathbf{u}^*$ can be designed as follows:

$$\mathbf{u}^* = \mathbf{G}(\underline{\mathbf{z}})^{\dagger}[\mathbf{z}_r^{(n)} - \mathbf{F}(\underline{\mathbf{z}}) - \mathbf{D}] + \mathbf{K_D}\mathbf{s} \tag{19}$$

$$= \mathbf{\Omega}(\bar{\mathbf{z}}_r) + \mathbf{K_D}\mathbf{s}, \tag{20}$$

where the function $\mathbf{\Omega}(\cdot)$ represents the inverse dynamics of the non-linear system, $\bar{\mathbf{z}}_r = [\underline{\mathbf{z}}^T \mathbf{z}_r^{(n)T}]^T \in \Re^{n_o(n+1) \times 1}$, and $\mathbf{K_D} \in \Re^{n_i \times n_o}$.

If all the parameters of the system are known, this control law leads to the following tracking error dynamics:

$$\mathbf{G}(\underline{\mathbf{z}})^{\dagger}\dot{\mathbf{s}} + \mathbf{K_D}\mathbf{s} = 0, \tag{21}$$

which gives exponential convergence of $\mathbf{s}$ by proper choice of $\mathbf{K_D}$, which in turn guarantees the convergence of $\mathbf{e}$.

However, the perfect control law cannot be implemented due to the imprecision and uncertainties in the system dynamics. To circumvent this problem, the G-FNN is proposed to generate the optimal torque to approximate the perfect control law. By substituting (13) and (14) into (20), the perfect control law can be executed using the G-FNN as follows:

$$\mathbf{u}^* = \hat{\mathbf{\Omega}}(\bar{\mathbf{z}}_r|\mathbf{W}^*) + \varepsilon + \mathbf{K_D}\mathbf{s} \tag{22}$$

$$= \mathbf{W}^{*T}\mathbf{\Phi}(\bar{\mathbf{z}}_r) + \varepsilon + \mathbf{K_D}\mathbf{s}. \tag{23}$$

Our proposed AFNC is therefore designed to mimic the perfect control law (23). As depicted in figure 2, a G-FNN controller is connected in parallel with a linear controller to generate a compensated control signal. The control law of the AFNC is given by

$$\mathbf{u}_c = \hat{\mathbf{\Omega}}(\bar{\mathbf{z}}_r|\mathbf{W}) + \varepsilon_N \cdot \mathrm{sgn}(\mathbf{G}^T\mathbf{P}\mathbf{s}) + \mathbf{K_D}\mathbf{s} \tag{24}$$

$$= \mathbf{W}^T\mathbf{\Phi}(\bar{\mathbf{z}}_r) + \varepsilon_N \cdot \mathrm{sgn}(\mathbf{G}^T\mathbf{P}\mathbf{s}) + \mathbf{K_D}\mathbf{s}, \tag{25}$$

where the operator '$\cdot$' denotes element-by-element multiplication of the vector ε_N and $\mathrm{sgn}(\mathbf{G}^T\mathbf{P}\mathbf{s})$, and $\mathbf{P} \in \Re^{n_o \times n_o}$ is a symmetric positive definite matrix.

In other words, the G-FNN controller is formed in such a way that it captures the inverse dynamic of the controlled system, i.e. the mapping relationship from $\bar{\mathbf{z}}$ to $\mathbf{u}$. This is achieved by applying the G-FNN learning algorithm, which is able to determine the appropriate structure and parameters of a fuzzy neural system to estimate the desired mapping relationship. Its online learning ability enables structure and parameters of

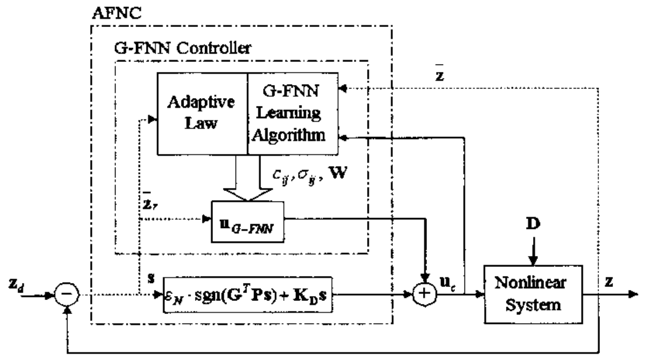

Figure 2. Adaptive fuzzy neural control structure.

the G-FNN, like N_r, c_{ij}, σ_{ij}, and $\mathbf{W}(0)$, to be obtained during real-time control of the plant. After the initial value of the weight matrix, $\mathbf{W}(0)$, is obtained from the G-FNN learning algorithm, $\mathbf{W}$ is further adjusted by the adaptive law derived in Section 4.2. This is to compensate for any modelling errors in the G-FNN learning algorithm and fulfil the stability requirements of the entire control system. The stability of the AFNC system is proven in Section 4.3.

4.2. *Convergence analysis of AFNC*

From (1), (22), and (24), the system tracking error equation can be shown to be

$$\dot{\mathbf{s}} = \mathbf{A}\mathbf{s} + \mathbf{G}(\underline{\mathbf{z}})(\mathbf{u}^* - \mathbf{u}_c), \tag{26}$$

where $\mathbf{A} = -\mathbf{G}(\underline{\mathbf{z}})\mathbf{K_D} \in \Re^{n_o \times n_o}$ is designed to be a Hurwitz matrix.

Substituting (23) and (25) into (26), we have

$$\dot{\mathbf{s}} = \mathbf{A}\mathbf{s} + \mathbf{G}(\underline{\mathbf{z}})[(\mathbf{W}^* - \mathbf{W})^T \mathbf{\Phi}(\bar{\mathbf{z}}_r) + \varepsilon - \varepsilon_N \cdot \operatorname{sgn}(\mathbf{G}^T\mathbf{P}\mathbf{s})]. \tag{27}$$

Equation (27) shows that only $\mathbf{W}$ needs to be further adjusted to minimize the tracking error. The adaptive law of $\mathbf{W}$ is designed as follows:

$$\dot{\mathbf{w}}_{jk} = a_j\lambda_j\mathbf{\Phi}_j\mathbf{s}^T\mathbf{P}\mathbf{g}_k - (1-a_j)\eta_{jk}\operatorname{sgn}(\mathbf{w}_{jk}): \quad j = 1\ldots N_r, \quad k = 1\ldots N_o, \tag{28}$$

where $\mathbf{w}_{jk} \in \Re^{(N_i+1)\times 1}$ is the weight vector associated with the jth rule for the kth output variable, i.e.

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_{11} & \mathbf{w}_{12} & \cdots & \mathbf{w}_{1N_o} \\ \mathbf{w}_{21} & \mathbf{w}_{22} & \cdots & \mathbf{w}_{2N_o} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{w}_{N_r1} & \mathbf{w}_{N_r2} & \cdots & \mathbf{w}_{N_rN_o} \end{bmatrix} \text{(refer to (7))},$$

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_1 \\ \mathbf{\Phi}_2 \\ \vdots \\ \mathbf{\Phi}_{N_r} \end{bmatrix} \text{(refer to (8))},$$

and $\mathbf{G} = [\mathbf{g}_1\ \mathbf{g}_2 \ldots \mathbf{g}_{N_o}]$.

The term $\lambda_j > 0$ is the learning rate for the weight vector associated with the jth rule. The term $\eta_{jk} = \lambda_j\hat{\eta}_{jk} > 0$ is the time-varying decay rate, which is chosen so that

$$\hat{\eta}_{jk} \geqslant \operatorname{sum}|\mathbf{\Phi}_j\mathbf{s}^T\mathbf{P}\mathbf{g}_k|, \tag{29}$$

where the operator 'sum' denotes summation of all the elements of the vector $|\mathbf{\Phi}_j\mathbf{s}^T\mathbf{P}\mathbf{g}_k|$.

The variable $a_j \in [0, 1]$ is a *structural adaptation parameter*, which is obtained from the G-FNN learning algorithm. It reflects the generation and deletion of fuzzy control rules associated with the G-FNN in the following manner. The jth rule is introduced into the G-FNN at time t by setting $a_j(t) = 1$, and the corresponding weight vector is initialized to $\mathbf{w}_{jk}(0)$ acquired by the G-FNN learning algorithm. The jth rule is targeted for deletion from the G-FNN at time t by setting $a_j(t) = 0$, which causes the adaptation mechanism to drive each of the output weights associated with the jth rule node to zero in a time bounded by $\max(\mathbf{w}_{jk})/\eta$. When $a_j(t) = 0$ and weight vectors of the jth rule node have reached zero, the rule is considered deleted from the G-FNN, and its associated memory and computational resources may be freed for reuse.

The symmetric positive definite matrix $\mathbf{P}$ is designed to satisfy the following relationship:

$$\mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} = -\mathbf{Q}, \tag{30}$$

where $\mathbf{Q}$ is a positive definite matrix and is selected by the user.

To guarantee the stability of the control system, the G-FNN must converge, which requires the parameters of the G-FNN to be bounded. Equation (6) shows that the outputs of the G-FNN are bounded if the weights $\mathbf{W}$ are bounded. Define the constraint set $\mathbf{\Gamma}$ for $\mathbf{W}$ as follows:

$$\mathbf{\Gamma} \triangleq \{\|\mathbf{w}_{jk}\| \leqslant \|\mathbf{w}_{jk}(0)\|\}: \quad j = 1\ldots N_r, \quad k = 1\ldots N_o, \tag{31}$$

where $\|\cdot\|$ denotes two-norm of a vector. According to the projection algorithm of (Wang 1997), the adaptation law (28) can be modified as follows:

$$\dot{\mathbf{w}}_{jk} = \begin{cases} a_j\lambda_j\mathbf{\Phi}_j\mathbf{s}^T\mathbf{P}\mathbf{g}_k - (1-a_j)\eta_{jk}\operatorname{sgn}(\mathbf{w}_{jk}) \\ \text{if } (\|\mathbf{w}_{jk}\| < \|\mathbf{w}_{jk}(0)\| \text{ or } (\|\mathbf{w}_{jk}\| = \|\mathbf{w}_{jk}(0)\| \text{ and} \\ \qquad \mathbf{w}_{jk}^T\mathbf{\Phi}_j\mathbf{s}^T\mathbf{P}\mathbf{g}_k \leqslant 0) \\ a_j\lambda_j\left(\mathbf{I} - \dfrac{\mathbf{w}_{jk}\mathbf{w}_{jk}^T}{\|\mathbf{w}_{jk}\|^2}\right)\mathbf{\Phi}_j\mathbf{s}^T\mathbf{P}\mathbf{g}_k - (1-a_j)\eta_{jk}\operatorname{sgn}(\mathbf{w}_{jk}) \\ \text{if } (\|\mathbf{w}_{jk}\| = \|\mathbf{w}_{jk}(0)\| \text{ and } \mathbf{w}_{jk}^T\mathbf{\Phi}_j\mathbf{s}^T\mathbf{P}\mathbf{g}_k > 0), \end{cases} \tag{32}$$

Concerning the boundedness of the weights of the G-FNN, we have

Theorem 4.1: *If the initial values of the weights* $\mathbf{w}_{jk}(0) \in \mathbf{\Gamma}$*, the adaptation law* (32) *guarantees* $\mathbf{w}_{jk}(t) \in \mathbf{\Gamma}$, $\forall t > 0$.

Proof: Consider the following Lyapunov function

$$V_{jk} = \frac{1}{2}\mathbf{w}_{jk}^T\mathbf{w}_{jk}. \tag{33}$$

Taking the derivative of the Lyapunov function with respect to time

$$\dot{V}_{jk} = \mathbf{w}_{jk}^T\dot{\mathbf{w}}_{jk}. \tag{34}$$

When ($\|\mathbf{w}_{jk}\| = \|\mathbf{w}_{jk}(0)\|$ and $\mathbf{w}_{jk}^T\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k \leqslant 0$), $\dot{V}_{jk} \leqslant 0$. Thus, it can be guaranteed that $\|\mathbf{w}_{jk}\| \leqslant \|\mathbf{w}_{jk}(0)\|$. When ($\|\mathbf{w}_{jk}\| = \|\mathbf{w}_{jk}(0)\|$ and $\mathbf{w}_{jk}^T\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k > 0$), $\dot{V}_{jk} \leqslant 0$. Thus, $\|\mathbf{w}_{jk}\| \leqslant \|\mathbf{w}_{jk}(0)\|$ is also guaranteed. Since the initial value $\mathbf{w}_{jk}(0) \in \mathbf{\Gamma}$, $\mathbf{w}_{jk}$ is bounded by the constraint set $\mathbf{\Gamma}$ for all $t \geqslant 0$. □

4.3. *Stability analysis of AFNC*

Concerning the stability of the closed-loop system, we have the following theorem

Theorem 4.2: *Consider the MIMO non-linear dynamic system represented by* (1). *If the control law of* (24) *and the adaptive law of* (32) *are applied, asymptotic stability is guaranteed.*

Proof: We consider the following Lyapunov function candidate, which is based on (27),

$$V(t) = \frac{1}{2}\mathbf{s}^T\mathbf{Ps} + \frac{1}{2}tr[(\mathbf{W}^* - \mathbf{W})^T\mathbf{\Lambda}^{-1}(\mathbf{W}^* - \mathbf{W})], \tag{35}$$

where $\mathbf{\Lambda} \in \Re^{N_r(N_i+1)\times N_r(N_i+1)}$ is a diagonal matrix whose jth $(N_i + 1)$ entries are the learning rate λ_j.

Taking the derivative of the Lyapunov function in (35) and using (27) and (30), we have

$$\begin{aligned}\dot{V}(t) &= \frac{1}{2}\dot{\mathbf{s}}^T\mathbf{Ps} + \frac{1}{2}\mathbf{s}^T\mathbf{P}\dot{\mathbf{s}} - tr[(\mathbf{W}^* - \mathbf{W})^T\mathbf{\Lambda}^{-1}\dot{\mathbf{W}}]\\ &= \frac{1}{2}\mathbf{s}^T(\mathbf{PA} + \mathbf{A}^T\mathbf{P})\mathbf{s} + \mathbf{s}^T\mathbf{PG}[\varepsilon - \varepsilon_N\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})]\\ &\quad + \mathbf{s}^T\mathbf{PG}(\mathbf{W}^* - \mathbf{W})^T\mathbf{\Phi}\\ &\quad - \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T\lambda_j^{-1}\dot{\mathbf{w}}_{jk}\\ &= -\frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{PG}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})]\\ &\quad + \sum_{k=1}^{N_o}\mathbf{s}^T\mathbf{Pg}_k\sum_{j=1}^{N_r}(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T\mathbf{\Phi}_j\\ &\quad - \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T\lambda_j^{-1}\dot{\mathbf{w}}_{jk}.\end{aligned} \tag{36}$$

Under condition 1 of (32), (36) becomes

$$\begin{aligned}\dot{V}(t) &= -\frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{P}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})]\\ &\quad + \sum_{k=1}^{N_o}\mathbf{s}^T\mathbf{Pg}_k\sum_{j=1}^{N_r}(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T\mathbf{\Phi}_j\\ &\quad - \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}(\mathbf{w}_{jk}^* - \mathbf{w}_{jk\lambda^{-1}})^T\lambda_j^{-1}[a_j\lambda_j\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k\\ &\quad - (1 - a_j)\eta_{jk}\mathrm{sgn}(\mathbf{w}_{jk})]\\ &= -\frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{PG}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})]\\ &\quad + \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}(1 - a_j)(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T[\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k + \hat{\eta}_{jk}\mathrm{sgn}(\mathbf{w}_{jk})]\\ &= \frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{PG}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})] + \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}v_{jk}.\end{aligned} \tag{37}$$

When $a_j = 1$, i.e. the jth rule is generated, $v_{jk} = 0$. When $a_j = 0$, i.e. the jth rule is going to be deleted, and $\mathbf{w}_{jk}^* = 0$, $v_{jk} = -\mathbf{w}_{jk}^T[\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k + \hat{\eta}_{jk}\mathrm{sgn}(\mathbf{w}_{jk})] \leqslant 0$ using (29). Therefore, (37) becomes

$$\dot{V}(t) \leqslant -\frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{PG}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})] \leqslant 0. \tag{38}$$

Under condition 2 of (32), (36) becomes

$$\begin{aligned}\dot{V}(t) &= -\frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{PG}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})]\\ &\quad + \sum_{k=1}^{N_o}\mathbf{s}^T\mathbf{Pg}_k\sum_{j=1}^{N_r}(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T\mathbf{\Phi}_j\\ &\quad - \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T\lambda_j^{-1}\\ &\quad \left[a_j\lambda_j\left(\mathbf{I} - \frac{\mathbf{w}_{jk}\mathbf{w}_{jk}^T}{\|\mathbf{w}_{jk}\|^2}\right)\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k - (1 - a_j)\eta_{jk}\mathrm{sgn}(\mathbf{w}_{jk})\right]\\ &= -\frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{PG}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})]\\ &\quad + \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}(1 - a_j)(\mathbf{w}_{jk}^* - \mathbf{w}_{jk})^T\\ &\quad [\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k + \hat{\eta}_{jk}\mathrm{sgn}(\mathbf{w}_{jk})]\\ &\quad - \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}a_j\left(1 - \frac{\mathbf{w}_{jk}^{*T}\mathbf{w}_{jk}}{\|\mathbf{w}_{jk}\|^2}\right)\mathbf{w}_{jk}^T\mathbf{\Phi}_j\mathbf{s}^T\mathbf{Pg}_k\\ &= -\frac{1}{2}\mathbf{s}^T\mathbf{Qs} - \mathbf{s}^T\mathbf{PG}[(\varepsilon_N \pm |\varepsilon|)\cdot\mathrm{sgn}(\mathbf{G}^T\mathbf{Ps})]\\ &\quad + \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}v_{jk} - \sum_{k=1}^{N_o}\sum_{j=1}^{N_r}u_{jk}.\end{aligned} \tag{39}$$

Since $\mathbf{w}_i^* \in \mathbf{\Gamma}$ and $1 - [(\mathbf{w}_i^{*T}\mathbf{w}_i)/(\|\mathbf{w}_i\|^2)] \geqslant 0$, when $a_j = 1$, i.e. the jth rule is generated, $v_{jk} = 0$ and $u_{jk} \geqslant 0$. When $a_j = 0$, i.e. the jth rule is going to be deleted and $\mathbf{w}_{jk}^* = 0$, $v_{jk} \leqslant 0$ and $u_{jk} = 0$. Therefore, (39) becomes

$$\dot{V}(t) \leqslant -\frac{1}{2}\mathbf{s}^T\mathbf{Q}\mathbf{s} - \mathbf{s}^T\mathbf{P}\mathbf{G}[(\varepsilon_N \pm |\varepsilon|) \cdot \operatorname{sgn}(\mathbf{G}^T\mathbf{P}\mathbf{s})] \leqslant 0. \tag{40}$$

Equations (35), (38), and (40) show that $V(t) \geqslant 0$ and $\dot{V}(t) \leqslant 0$. Furthermore, (38) and (40) imply that $\dot{V}(t) = 0$ if, and only if, $V(t) = 0$. Therefore, global stability is guaranteed by the Lyapunov theorem. By using Barbalat's lemma (Slotine and Li 1991), it can be shown that $s(t) \to 0$ as $t \to \infty$. As a result, the control system is asymptotically stable. Moreover, the tracking error of the system will converge to zero.

$\square$

5. Simulation studies

In this section, a simulation example is given to demonstrate the effectiveness of the proposed AFNC for non-linear systems. Comparisons with some newly developed adaptive fuzzy or neural controllers further demonstrate its advantages in both transient and steady-state performances.

In this simulation, the performance of the proposed AFNC in tracking control of a two-link robot manipulator (refer to figure 3) is evaluated. The dynamic equation is given by (Slotine and Li 1991)

$$\ddot{\theta} = -\mathbf{M}(\theta)^{-1}\mathbf{H}(\theta, \dot{\theta}) + \mathbf{M}(\theta)^{-1}\tau - \mathbf{M}(\theta)^{-1}\tau_{\mathbf{d}} \tag{41}$$

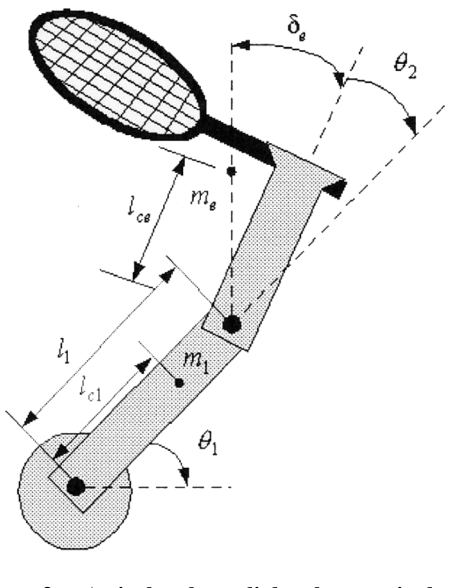

Figure 3. Articulated two-link robot manipulator.

and

$$\begin{aligned}
\mathbf{M}_{11} &= (I_1 + m_1 l_{c1}^2 + I_e + m_e l_{ce}^2 + m_e l_1^2) \\
&\quad + 2(m_e l_1 l_{ce}\cos\delta_e)\cos\theta_2 + 2(m_e l_1 l_{ce}\sin\delta_e)\sin\theta_2 \\
\mathbf{M}_{12} &= \mathbf{M}_{21} = (I_e + m_e l_{ce}^2) + (m_e l_1 l_{ce}\cos\delta_e)\cos\theta_2 \\
&\quad + (m_e l_1 l_{ce}\sin\delta_e)\sin\theta_2 \\
\mathbf{M}_{22} &= I_e + m_e l_{ce}^2 \\
\mathbf{H}_{11} &= -[(m_e l_1 l_{ce}\cos\delta_e)\sin\theta_2 - (m_e l_1 l_{ce}\sin\delta_e)\cos\theta_2]\dot{\theta}_1\dot{\theta}_2 \\
&\quad - [(m_e l_1 l_{ce}\cos\delta_e)\sin\theta_2 - (m_e l_1 l_{ce}\sin\delta_e)\cos\theta_2] \\
&\quad (\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 \\
\mathbf{H}_{21} &= [(m_e l_1 l_{ce}\cos\delta_e)\sin\theta_2 - (m_e l_1 l_{ce}\sin\delta_e)\cos\theta_2]\dot{\theta}_1^2,
\end{aligned} \tag{42}$$

where $\mathbf{M}$ is the 2×2 manipulator inertia matrix, which is symmetric positive definite, $\mathbf{H}$ is the 2×1 vector of centrifugal, Coriolis, friction forces, and gravity, τ is the 2×1 vector of input torque generated by the joint motor, $\ddot{\theta}$, $\dot{\theta}$ and θ are 2×1 vectors of output link acceleration, velocity, and position, respectively, and $\tau_{\mathbf{d}}$ is the 2×1 vector of unknown terms arising from unmodelled dynamics and external disturbances.

Initial conditions are chosen as $\theta_1(0) = \theta_2(0) = 0$ *rad* and $\dot{\theta}_1(0) = \dot{\theta}_2(0) = 0\,rad/s$. The desired trajectory is an ellipse in the $\theta_1 - \theta_2$ plane not starting from the initial position with $\theta_{d1} = 1.5\sin(2\pi t)\,rad$ and $\theta_{d2} = 0.7\cos(2\pi t)\,rad$. Disturbances are chosen as $\tau_{d1} = 100\sin(2\pi t)\,Nm$ and $\tau_{d2} = 50\sin(2\pi t)Nm$, which are comparable with the control torques of the robot manipulator.

The G-FNN learning algorithm operates at a sampling rate of 50 *sample/s*, whose predefined thresholds are designed as: $e_{\max} = 0.1$, $e_{\min} = 0.005$, $d_{\max} = \sqrt{\ln(1/0.5)}$, $d_{\min} = \sqrt{\ln(1/0.8)}$, $K_{s,\min} = 0.9$, $K_{mf} = 0.5$, $K_{err} = 0.01$, and $N_d = 100$. The remaining part of the AFNC operates at a sampling rate of 1000 *sample/s*. The parameters of the AFNC are designed as follows: $\Upsilon_0 = diag[25/7, 25/7]$, $\mathbf{K_D} = diag[7, 7]$, $\mathbf{P} = diag[8, 8]$, $\varepsilon_N = [0.1\ 0.05]^T$, and $\lambda_j = 0.4$.

Using the G-FNN learning algorithm, the fuzzy neural structure and parameters are generated simutaniously and automatically. In this simulation study, a total of four fuzzy rules were generated online, as shown in figure 4. The corresponding Gaussian fuzzy membership functions with respect to the input training variable as shown in figure 5 were obtained. It can be seen that the membership functions are evenly distributed over the input training interval. This is in line with the aspiration of 'local representation' in fuzzy logic. Figure 6 shows the control responses of the two links. The results clearly demonstrate that the two links were able to track the desired trajectories from the second trial onwards. It should be highlighted that for θ_2,

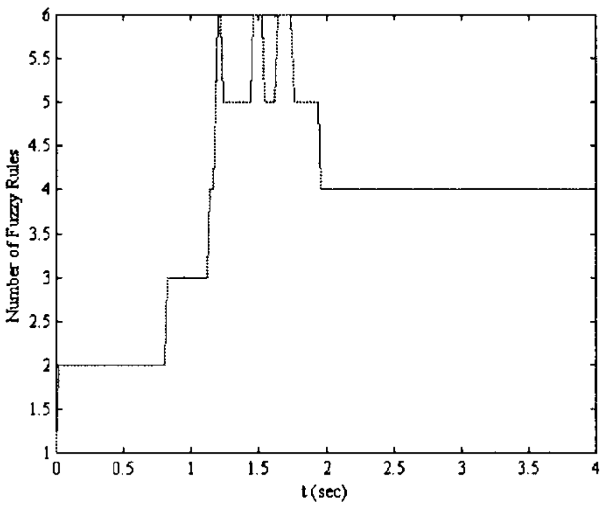

Figure 4. Number of fuzzy rules generated.

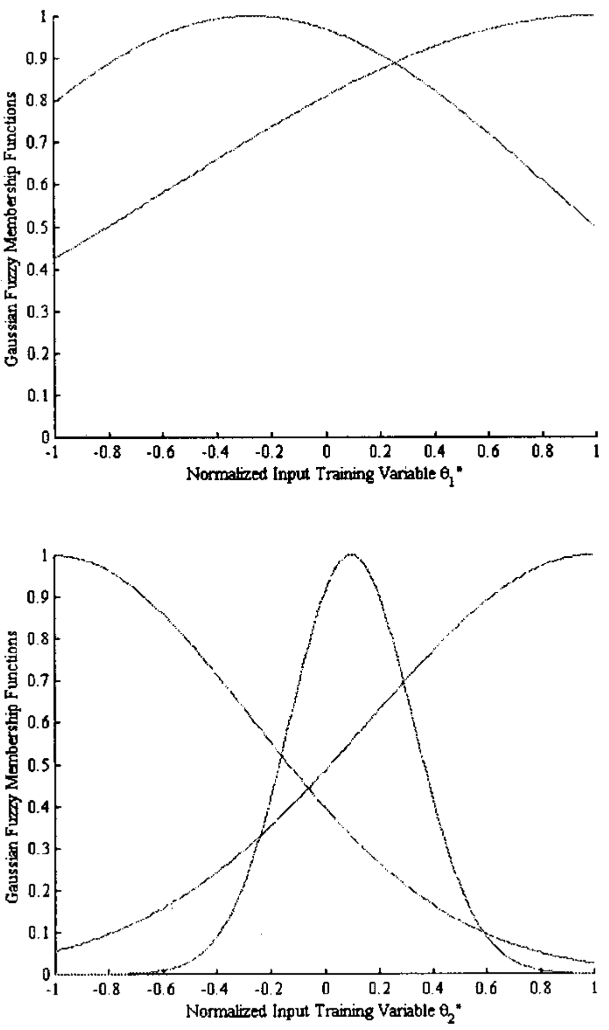

Figure 5. Gaussian fuzzy membership functions w.r.t input training variables.

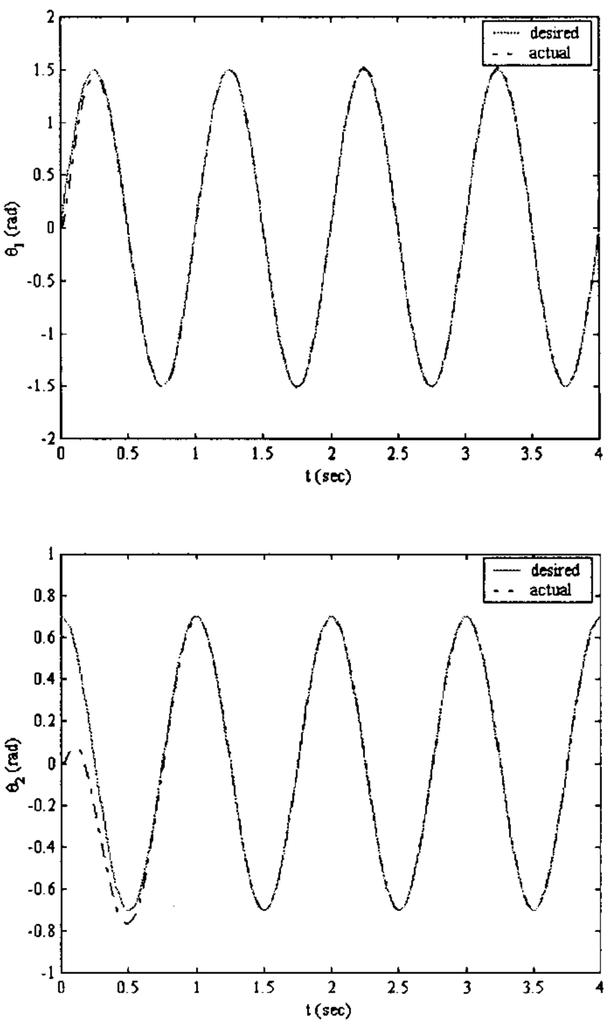

Figure 6. Tracking responses using AFNC.

whose initial position was set to be 100% error from the desired initial link position, i.e. $\theta_2(0) = 0$, the link was still able to track the desired trajectory very quickly without any overshoots.

The proposed AFNC has some advantages over normal fuzzy or neural controllers. For example, in normal fuzzy or neural controller design, certain controller parameters must be tuned by trial and error. Such parameters include the number of membership functions, centre/width of a single membership function, and number of fuzzy control rules or hidden neurons. On the contrary, the AFNC based on the dynamic model estimation will allow the parameters to be tuned automatically. In addition, the performance of the AFNC is also considered to be excellent.

In this example, the performance of the AFNC was compared with three other controllers that were developed in the similar control structure, i.e. (1) a computed torque controller (CTC) of (Slotine and Li 1991),

Table 1. Comparison of AFNC with other controllers.

	Robot model	$\mathbf{K_P}$	$\mathbf{K_V}$	Number of fuzzy rules/hidden neurons
CTC (Slotine and Li 1991)	Required	25	10	0 (fixed)
ANC (Feng 1997)	Required	25	10	121 (fixed)
AFC (Er and Chin 2000)	Required	25	35	4 (fixed)
AFNC	Not required	25	7	4 (adaptive)

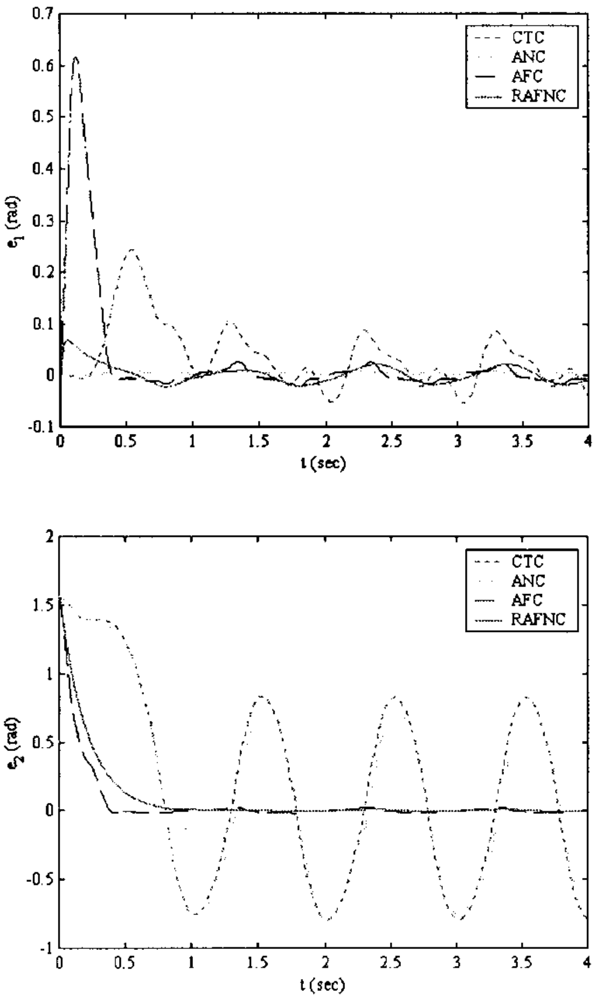

Figure 7. Tracking errors using CTC, ANC, AFC, and AFNC.

(2) an adaptive neural controller (ANC) of (Feng, 1997), and (3) an adaptive fuzzy controller (AFC) of (Er and Chin 2000). Table 1 shows the properties of the designed AFNC with respect to the other three controllers. Figure 7 compares the control performance of the AFNC with the other controllers. It clearly shows that the AFNC can achieve better transient and steady-state control results, even with a smaller feedback gain.

Remark 5.1: Concerning the control law (25) of the AFNC, besides the robust term containing ϵ_N, the AFNC for second-order *square* systems consists of a feedforward G-FNN controller and a feedback PD controller with $\mathbf{K_P} = \mathbf{K_D}\Upsilon_0$ and $\mathbf{K_V} = \mathbf{K_D}$.

6. Conclusions

In this paper, an adaptive fuzzy neural modelling and control scheme for non-linear systems was proposed, and its adaptive capability to handle modelling errors and external disturbances was demonstrated. The error convergence rate with the AFNC was found to be fast. Asymptotic stability of the control system is established using the Lyapunov approach. Computer simulation studies of a two-link robot manipulator verify the flexibility, adaptation, and tracking performance of the proposed AFNC.

Appendix: Mathematical descriptions of G-FNN learning algorithm

A.1. *Two criteria of fuzzy-rule generation*

1. *System error.* The system error is defined as the difference between the output of the G-FNN and the teaching signal. It is an important factor in determining whether a new rule should be added. For each training data pair $[\mathbf{x}(t_s),\ \mathbf{t}(t_s)]$: $t_s = 1 \ldots N_d$, where $\mathbf{x}(t_s) = [x_1(t_s)\ x_2(t_s) \ldots x_{N_i}(t_s)]^T$ is t_sth input vector, $\mathbf{t}(t_s) = [t_1(t_s)\ t_2(t_s) \ldots t_{N_o}(t_s)]^T$ is the t_sth desired output or the supervised teaching signal, and N_d is the total number of training data, compute the G-FNN output $\mathbf{y}(t_s)$ for the existing structure using (2)–(8). Define the t_sth system error, $e(t_s)$, as the Euclidean distance between vectors $\mathbf{t}(t_s)$ and $\mathbf{y}(t_s)$, i.e.

$$e(t_s) = \|\mathbf{e}(t_s)\| = \|\mathbf{t}(t_s) - \mathbf{y}(t_s)\|. \tag{A1}$$

If

$$e(t_s) > K_e, \tag{A2}$$

a new fuzzy rule should be considered. K_e is designed based on desired accuracy of the G-FNN.

2. *System ϵ-completeness*

Definition A1 ϵ-completeness of fuzzy rule: For any input within the operating range, there exists at least one fuzzy rule such that the match degree (or firing strength) is not less than ϵ (Wang, 1997). The minimum of ϵ is usually selected as $\epsilon_{\min} = 0.5$.

A fuzzy rule is a local representation over the region of input space. If a new pattern falls within the accommodation boundary of the existing rules, i.e. it satisfies ϵ-completeness, G-FNN will not generate a new rule but update parameters of the existing rules, and vice versa. According to ϵ-completeness, when the t_sth observation $[\mathbf{x}(t_s), \mathbf{t}(t_s)]$ enters the system, we calculate the regularized Mahalanobis distance $md_j(t_s)$: $j = 1 \ldots N_r$ between $\mathbf{x}(t_s)$ and the jth existing hyperellipsoidal region as

$$md_j(t_s) = \sqrt{[\mathbf{x}(t_s) - \mathbf{c}_j]^T \Sigma_j [\mathbf{x}(t_s) - \mathbf{c}_j]}, \tag{A3}$$

where $\mathbf{c}_j = [c_{1_j}\ c_{2_j} \ldots c_{N_{ij}}]^T$, $\Sigma_j = diag(1/\delta_{1j}^2\ 1/\delta_{2j}^2 \ldots 1/\delta_{N_{ij}}^2)$, and N_r is the number of existing rules. Find

$$\tilde{j} = \arg\min md_j(t_s). \tag{A4}$$

If

$$d(t_s) = md_{\tilde{j}}(t_s) > K_d, \tag{A5}$$

where $K_d = \sqrt{\ln(1/\epsilon)}$, a new rule should be considered because the existing system is not satisfied by ϵ-completeness. Otherwise, the new input training data can be represented by the nearest existing rule.

A.2. *Pruning of fuzzy rules*

If inactive hidden units can be deleted as learning progresses, a more parsimonious network topology can be achieved. Therefore, pruning becomes imperative for modelling of non-linear systems with time-varying dynamics. In the G-FNN learning algorithm, the error reduction ratio (ERR) concept proposed in Chen *et al.* (1991) is adopted for rule sensitivity measure and rule pruning of MIMO G-FNNs.

As the t_sth observation $[\mathbf{x}(t_s), \mathbf{t}(t_s)]$ enters the G-FNN, all the existing t_s data are memorized by the G-FNN as $(\mathbf{X}, \mathbf{T})$ where $\mathbf{X} = [\mathbf{x}(1)\ \ \mathbf{x}(2) \ldots \mathbf{x}(t_s)]$ and $\mathbf{T} = [\mathbf{t}(1)\ \mathbf{t}(2) \ldots \mathbf{t}(t_s)]$. From (6) and (A1), we have

$$\mathbf{T} = \mathbf{W}^T\boldsymbol{\Theta} + \mathbf{E}, \tag{A6}$$

where $\mathbf{T} = [\mathbf{t}_1^T\ \mathbf{t}_2^T \ldots \mathbf{t}_{N_o}^T]^T \in \Re^{N_o \times t_s}$ is the teaching signal, $\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2 \ldots \mathbf{w}_{N_o}] \in \Re^{v \times N_o}$ denotes the real parameters, $\boldsymbol{\Theta} = [\boldsymbol{\Phi}(1)\ \boldsymbol{\Phi}(2) \ldots \boldsymbol{\Phi}(t_s)] \in \Re^{v \times t_s}$ is known as the regressor, $\mathbf{E} = [\mathbf{e}_1\ \mathbf{e}_2 \ldots \mathbf{e}_{N_o}]^T \in \Re^{N_o \times t_s}$ is the system error, which is assumed to be uncorrelated with regressor $\boldsymbol{\Theta}$, and $v = N_r(N_i + 1)$.

For simplicity, we use a single output system to illustrate the ERR method. Therefore, for the kth output variable becomes

$$\mathbf{t}_k = \mathbf{w}_k^T\boldsymbol{\Theta} + \mathbf{e}_k\colon\ k = 1 \ldots N_o. \tag{A7}$$

Taking the transpose at both sides, we have

$$\mathbf{t}_k^T = \boldsymbol{\Theta}^T\mathbf{w}_k + \mathbf{e}_k^T, \tag{A8}$$

where $\mathbf{t}_k^T$, $\mathbf{e}_k^T \in \Re^{t_s \times 1}$, $\Theta^T \in \Re^{t_s \times v}$, and $\mathbf{w}_k \in \Re^{v \times 1}$. For any matrix $\boldsymbol{\Theta}^T$, if its row number is larger than column number, i.e. $t_s \geqslant v$, it can be transformed into a set of orthogonal basis vectors by QR decomposition. Thus

$$\boldsymbol{\Theta}^T = \mathbf{QR}, \tag{A9}$$

where $\mathbf{Q} = [\mathbf{q}_1\ \mathbf{q}_2 \ldots \mathbf{q}_v] \in \Re^{t_s \times v}$ has orthogonal columns, $\mathbf{R} \in \Re^{v \times v}$ is an upper triangular matrix. Therefore, (A8) can then be written as

$$\mathbf{t}_k^T = \mathbf{QR}\mathbf{w}_k + \mathbf{e}_k^T = \mathbf{Qg} + \mathbf{e}_k^T, \tag{A10}$$

where $\mathbf{g} = [g_1\ g_2 \ldots g_v]^T$, and the LLS method gives

$$\mathbf{g} = (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\mathbf{t}_k^T \quad \text{or} \quad g_\gamma = (\mathbf{q}_\gamma^T\mathbf{q}_\gamma)^{-1}\mathbf{q}_\gamma^T\mathbf{t}_k^T : \gamma = 1 \ldots v. \tag{A11}$$

As $\mathbf{Q}$ has orthogonal columns, the energy function of $\mathbf{t}_k^T$ can be represented as (Chen *et al.* 1991)

$$\mathbf{t}_k\mathbf{t}_k^T = \sum_{\gamma=1}^{v} g_\gamma^2\mathbf{q}_\gamma^T\mathbf{q}_\gamma + \mathbf{e}_k\mathbf{e}_k^T. \tag{A12}$$

If $\mathbf{t}_k^T$ is the desired output after its mean has been removed, the variance of $\mathbf{t}_k^T$ would be

$$t_s^{-1}\mathbf{t}_k\mathbf{t}_k^T = t_s^{-1}\sum_{\gamma=1}^{v} g_\gamma^2\mathbf{q}_\gamma^T\mathbf{q}_\gamma + t_s^{-1}\mathbf{e}_k\mathbf{e}_k^T, \tag{A13}$$

where $t_s^{-1}\sum g_\gamma^2\mathbf{q}_\gamma^T\mathbf{q}_\gamma$ is the variance that can be explained by the regressor $\boldsymbol{q}_\gamma$, while $t_s^{-1}\mathbf{e}_k\mathbf{e}_k^T$ is the unexplained variance $\mathbf{t}_k^T$. Thus, $t_s^{-1}\sum g_\gamma^2\mathbf{q}_\gamma^T\mathbf{q}_\gamma$ is the increment to the explained desired output variance introduced by $\mathbf{q}_\gamma$. Thus, an error reduction ratio (ERR) due to $\mathbf{q}_\gamma$ on the kth output variable can be defined as (Chen *et al.* 1991)

$$err_\gamma^k = \frac{g_\gamma^2\mathbf{q}_\gamma^T\mathbf{q}_\gamma}{\mathbf{t}_k\mathbf{t}_k^T}. \tag{A14}$$

Substituting g_γ with (A11), we have

$$err_\gamma^k = \frac{\left(\mathbf{q}_\gamma^T \mathbf{t}_k^T\right)^2}{\mathbf{q}_\gamma^T \mathbf{q}_\gamma \mathbf{t}_k \mathbf{t}_k^T}. \tag{A15}$$

The ERR offers a simple and effective means of seeking a subset of significant regressors. The term err_γ in (A15) reflects the similarity of $\mathbf{q}_\gamma$ and $\mathbf{t}_k^T$ or the inner product of $\mathbf{q}_\gamma$ and $\mathbf{t}_k^T$. A larger err_γ implies that $\mathbf{q}_\gamma$ is more significant to output $\mathbf{t}_k^T$.

We can further define the total error reduction ratio $Terr_j$: $j = 1 \ldots N_r$ corresponding to the jth rule as

$$Terr_j = \sqrt{\frac{(\mathbf{err}_j)^T \mathbf{err}_j}{(N_i + 1)N_o}} \tag{A16}$$

$$\mathbf{err}_j = [err^1_{0\times N_r+j} \cdots err^1_{N_i\times N_r+j} \cdots err^{N_o}_{0\times N_r+j} \cdots err^{N_o}_{N_i\times N_r+j}]^T. \tag{A17}$$

If

$$Terr_j < K_{err}, \tag{A18}$$

where K_{err} is a predefined threshold, the jth fuzzy rule should be deleted, and vice versa.

A.3. *Determination of premise parameters*

Definition A2 Semi-closed fuzzy set: For input variable x_i:$i = 1 \ldots N_i$, if each fuzzy Gaussian membership function $mf_{ij}(c_{ij}, \sigma_{ij})$ in the operating interval $[x_{i,\min}, x_{i,\max}]$ satisfies the following boundary conditions:

$$\begin{aligned}
&mf_{ij}\left(c_{ij} = x_{i,\min}, \sigma_{ij} = \frac{|x_{i,\max} - x_{i,\min}|}{\ln(1/\varepsilon)}\right) \quad |x_i - x_{i,\min}| \leqslant \delta\\
&mf_{ij}\left(c_{ij} = x_{i,\max}, \sigma_{ij} = \frac{|x_{i,\max} - x_{i,\min}|}{\ln(1/\varepsilon)}\right) \quad |x_i - x_{i,\max}| \leqslant \delta\\
&mf_{ij}\left(c_{ij} = x_i, \sigma_{ij} = \frac{\max(|c_{ij} - c_{ij_a}|, |c_{ij} - c_{ij_b}|)}{\ln(1/\varepsilon)}\right)\\
&\qquad |x_i - x_{i,\min}| \geqslant \delta \quad \text{or} \quad |x_i - x_{i,\max}| \geqslant \delta,
\end{aligned} \tag{A19}$$

where δ is a tolerable small value, c_{ija} and c_{ij_b} are the centres of two most neighbouring membership functions $mf_{ij}(c_{ij},\sigma_{ij})$, then in this case, mf_{ij} is a semi-closed fuzzy set which satisfies ϵ-completeness of fuzzy rules.

The concept of a semi-closed fuzzy set is then used to allocate premise parameters under the following three cases.

1. $e(t_s) > K_e$ and $d(t_s) > K_d$: Multidimensional input vector $\mathbf{x}(t_s)$ is first projected to the ith axis as $x_i(t_s)$: $i = 1 \ldots N_i$. The compute Euclidean distance between $x_i(t_s)$ and boundary point $b_{ij_n}: i = 1 \ldots N_i$, $j_n = 1, 2, \ldots, N_r + 2$:

$$ed_{ij_n}(t_s) = \|x_i(t_s) - b_{ij_n}\|, \tag{A20}$$

where $b_{ij_n} \in \{c_{i1}, c_{i2}, \ldots, c_i N_r, x_{i,\min}, x_{i,\max}\}$, and N_r is the number of existing rules. Find

$$\tilde{j}_n = \arg\min ed_{ij_n}(t_s). \tag{A21}$$

If

$$ed_{i\tilde{j}n}(t_s) < K_{mf}, \tag{A22}$$

where K_{mf} is predetermined threshold which implies the similarity of neighbouring membership function. In cooperation with the first two conditions in (A19), we have

$$c_i(N_r + 1) = b_{i\tilde{j}_n}, \qquad \sigma_i(N_r + 1) = \sigma_{i\tilde{j}_n}. \tag{A23}$$

Otherwise, in cooperation with the third condition in (A19), we choose

$$\begin{aligned}
&c_i(N_r + 1) = x_i(t_s),\\
&\sigma_i(N_r + 1) = \left\{\begin{matrix}(\max(|c_i(N_r+1) - c_i(N_r+1)_a|,\\ |c_i(N_r+1) - c_i(N_r+1)_b|).)\end{matrix}\right\} \Big/ (\ln(1/\varepsilon))
\end{aligned} \tag{A24}$$

2. $e(t_s) > K_e$ but $d(t_s) \leqslant K_d$: $\mathbf{x}(t_s)$ can be clustered by the adjacent fuzzy rule, but the rule is not sufficiently significant to accommodate all the patterns covered by its ellipsoidal field. Therefore, the ellipsoidal field needs to be decreased to obtain a better local approximation. A simple method to reduce the Gaussian widths is designed as

$$\sigma_{i\tilde{j}_{new}} = K_s \times \sigma_{i\tilde{j}_{old}}, \tag{A25}$$

where K_s is a reducing factor which depends on the sensitivity of the input variables. The sensitivity measure of the ith input variable in the jth fuzzy rule is denoted as s_{ij}, which is calculated using (A15), i.e.

$$s_{ij} = \frac{\sum_{k=1}^{N_o} err^k_{i\times N_r+j}}{\sum_{i=1}^{N_i}\sum_{k=1}^{N_o} err^k_{i\times N_r+j}}. \tag{A26}$$

The threshold K_s could be designed with a minimum value of $K_{s,\min}$, when $s_{ij} = 0$, and a maximum value of 1 when s_{ij} is greater than or equal to the average sensitivity level of the input variables in jth rule, i.e.

$$K_s = \begin{cases} \frac{1}{1+((1-K_{s,\min})N_r^2/(K_{s,\min}))(s_{ij}-(1/N_i))^2} & s_{ij} < \frac{1}{N_i} \\ 1 & \frac{1}{N_i} \leqslant s_{ij} \end{cases}, \tag{A27}$$

3. $e(t_s) \leqslant K_e$ but $d(t_s) > K_d$; or $e(t_s) \leqslant K_e$ and $d(t_s) \leqslant K_d$: The system has a good generalization, and nothing needs to be done except adjusting the weight.

A.4. *Determination of consequent parameters*

TSK-type consequence for each rule is determined using the Linear Least Squared (LLS) method. The LLS method is employed to find the weight vector, $\mathbf{W}$, such that the error energy $trace(\mathbf{E}^T\mathbf{E})$ is minimized in (A6). Furthermore, the LLS method provides a computationally simple but efficient procedure for determining weight so that it can be computed very quickly and used for real-time control. The weight vector is calculated as

$$\mathbf{W} = (\mathbf{T}\boldsymbol{\Theta}^{\dagger})^T, \tag{A28}$$

where $\boldsymbol{\Theta}^{\dagger}$ is the pseudo-inverse of $\boldsymbol{\Theta}$

$$\boldsymbol{\Theta}^{\dagger} = (\boldsymbol{\Theta}^T\boldsymbol{\Theta})^{-1}\boldsymbol{\Theta}^T. \tag{A29}$$

References

Cannon, M. R., and Slotine, J. J. E., 1995, Space-frequency localized basis function networks for nonlinear system estimation and control. *Neurocomputing*, **9,** 293–342.

Chak, C. K., Feng, G., and Ma, J., 1998, An adaptive fuzzy neural network for mimo system model approximation in high-dimensional spaces. *IEEE Transactions on Systems, Man and Cybernetics*, **28,** 436–446.

Chen, S., Cowan, C. F. N., and Grant, P. M., 1991, Orthogonal least squares learning algorithm for radial basis function network. *IEEE Transactions on Neural Networks*, **2,** 302–309.

Er, M. J., and Chin, S. M., 2000, Hybrid adaptive fuzzy controllers of robot manipulators with bounds estimation. *IEEE Transactions on Industrial Electronics*, **47,** 1151–1160.

Er, M. J., and Gao, Y., 2003, Robust adaptive control of robot manipulators using generalized fuzzy neural networks. *IEEE Transactions on Industrial Electronics*, **50,** 620–628.

Feng, G., 1997, A new stable tracking control scheme for robotic manipulators. *IEEE Transactions on System, Man and Cybernetics*, **27,** 510–516.

Gao, Y., and Er, M. J., 2001, Adaptive fuzzy neural control of multiple-link robot manipulators. *International Journal of Robotics and Automation*, **16,** 172–182.

Jin, Y., 1998, Decentralized adaptive fuzzy control of robot manipulators. *IEEE Transactions on System, Man, and Cybernetics*, **28,** 47–58.

Jin, Y., Jiang, J., and Zhu, J., 1995, Neural network based fuzzy identification with application to modelling and control of complex systems. *IEEE Transactions on System, Man, and Cybernetics*, **25,** 990–997.

Jin, Y., Seelen, W. V., and Sendhoff, B., 1999, On generating fc^3 fuzzy rule systems from data using evolution strategies. *IEEE Transactions on System, Man, and Cybernetics*, **29,** 829–845.

Lin, F. J., Hwang, W. J., and Wai, R. J., 1999, A supervisory fuzzy neural network control system for tracking periodic input. *IEEE Transactions on Fuzzy Systems*, **7,** 41–52.

MacKay, D. J. C., 1995, Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research*, **354,** 73–80.

Nie, J., and Linkens, D., 1995, *Fuzzy-Neural Control: Principles, Algorithms and Applications* (Englewood Cliffs: Prentice Hall).

Ohtani, Y., and Yoshimura, T., 1996, Fuzzy control of a manipulator using the concept of sliding mode. *International Journal of Systems Science*, **27,** 1727–1731.

Park, D., Kandel, A., and Langholz, G., 1994, Genetic-based new fuzzy reasoning models with application to fuzzy control. *IEEE Transactions on System, Man, and Cybernetics*, **24,** 39–47.

Slotine, J. J. E., and Li, W., 1991, *Applied Nonlinear Control* (Englewood Cliffs: Prentice Hall).

Wang, L. X., 1997, *A Course in Fuzzy Systems and Control* (Englewood Cliffs: Prentice Hall).