# Robust self-tuning rotated fuzzy basis function controller for robot arms

C.-K. Lin
S.-D. Wang

*Indexing terms: Adaptive fuzzy controllers, Robot arms, Self-tuning fuzzy control*

**Abstract:** An adaptive fuzzy controller is developed for a serial-link robot arm. The proposed rotated fuzzy basis function (RFBF) controller is a more flexible fuzzy basis function expansion to approximate unknown functions of the robot model. All parameters of RFBF network can be tuned online when the number of rules is determined. In the control design, the unmodelled dynamics are considered. Moreover, the stability analysis shows that the states and tracking errors of the robot arm are uniformly bounded. Simulations of the proposed controller on the PUMA-560 robot arm demonstrate the effectiveness.

## 1 Introduction

Fuzzy control has been an active research field in the past decade. Recently, it has also been related to adaptive control based on fuzzy basis function expansion addressed in [1–3]. Although adaptive fuzzy controllers in [2, 3] can guarantee the global stability and tracking performance, they can only tune a part of parameters of the fuzzy system. The robustness property of the controller was also not mentioned.

The area of self-tuning fuzzy control has been studied by many researchers, with a few of the notable ones surveyed here. Although self-organising, self-learning and self-tuning fuzzy control are something different, they are all treated as self-tuning fuzzy control here. Procyk and Mamdani [4] proposed a self-organised fuzzy control system which employs a mechanism to modify or create a fuzzy rule base. Since the pioneer work of Mamdani, considerable research works about self-tuning fuzzy control have appeared. Consequential papers which improve Procyk–Mamdani's controller were proposed in [5, 6]. However, such self-organised fuzzy control systems cannot determine the rule number, shapes of membership functions, global stability and the tracking performance. Many fuzzy-neural paradigms have been proposed recently to confront these deficiencies. For example, Yamaguchi *et al.* [7] proposed a fuzzy associative memory system called FAMOUS. In the IF-part of the knowledge pairs of FAMOUS, the membership function is automatically generated from input data by an unsupervised learning algorithm. Another class of adaptive fuzzy systems is based on feedforward five-layer neural networks. Lin and Lee [8] proposed an unsupervised learning neural network-based fuzzy logic control systems employing a reinforcement learning algorithm to tune parameters. This fuzzy controller can be represented by the ad hoc structure of a five-layer neural network which has a direct correspondence from fuzzy control system. Jang [9] also proposed a five-layer self-learning fuzzy control architecture whose learning algorithm is gradient descent. These two five-layer self-tuning fuzzy neural control systems are not capable of guaranteeing stability and tracking performance.

Fuzzy Basis Function Network is a new fuzzy-neural scheme which integrates fuzzy reasoning systems with radial basis function networks. Nie and Linkens [10] proposed a fuzzified self-organised radial basis function network employing a simplified fuzzy control algorithm (SFCA) to tune all parameters and construct the network. SFCA, which is an unsupervised learning algorithm, cannot guarantee the stability and performance. In the past, a fuzzy controller has not been viewed as a rigorous control approach owing to a lack of formal synthesis techniques that guarantee the very basic requirements of global stability and acceptable performance. Wang [2, 3] proposed an adaptive fuzzy controller which can tune the parameters of consequent part by adaptation law using Lyapunov stability theory. This controller can also cope with the unstructured dynamics by adaptive robust techniques.

In this paper, the proposed controller is only based on a general three-layer neural network that implements a fully adaptive fuzzy radial basis function expansion. The architecture of the neural network is basically a multilayer neural network where radial functions are used instead of sigmoid functions in hidden neurons. The hidden-to-output layer interconnection weights correspond to the consequence part of the fuzzy rule base and the input-to-hidden layer interconnection weights can be mapped to the parameters of fuzzy basis functions. The major differences between RFBF networks and three-layer neural networks are the processing elements of neurons and the input-to-hidden layer interconnection weights. The input of the sigmoid (kernel) function of RFBF networks is the square norm of the difference of input vector and input-to-hidden layer weight vector. The use of RFBF controller in direct closed-loop controllers



*IEE Proceedings* online no. 19971000

Paper received 7th March 1996

The authors are with the Department of Electrical Engineering, National Taiwan University, 1 Roosevelt Rd., Sec. 4, Taipei 106, Taiwan

can guarantee the global stability and tracking performance.

## 2 Preliminaries

### 2.1 Robot arm dynamics

The dynamic equation of an $n$ degree of freedom robot manipulator is given by:

$$M(\underline{\theta})\ddot{\underline{\theta}} + V_m(\underline{\theta}, \dot{\underline{\theta}})\dot{\underline{\theta}} + G(\underline{\theta}) + F\dot{\underline{\theta}} + \underline{\tau}_d = \underline{\tau} \qquad (1)$$

where vector $\underline{\theta} \in R^n$ is the joint position vector; $M(\underline{\theta}) \in R^{n\times n}$ is a symmetric positive definite inertia matrix; $V_m$ ($\underline{\theta}$, $\dot{\underline{\theta}}$) is a vector of Coriolis and centripetal torques; $G(\underline{\theta}) \in R^n$ represents the gravitational torques; $F = K_w + V_f \in R^{n\times n}$ is a diagonal matrix consisting of the back emf coefficient matrix $K_w$ and the viscous friction coefficients matrix $K_f$; $\underline{\tau}_d \in R^{n\times 1}$ is the unmodelled disturbances vector; and $\underline{\tau} \in R^{n\times 1}$ is the vector of control input torques. A planar two-link robot arm used for illustration is shown in Fig. 1. The structural properties of the robot such as boundedness of $V_m$ ($\underline{\theta}$, $\dot{\underline{\theta}}$) and skew-symmetry of matrix $\dot{M} - V_m$ are well-known and not stated here.

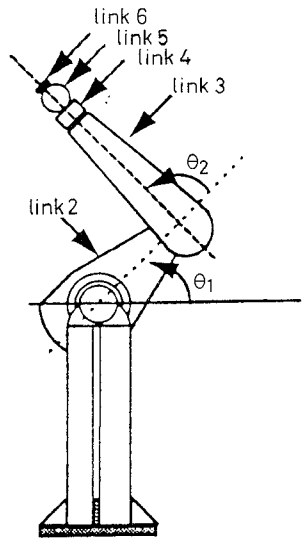


**Fig. 1** *A two-link robot manipulator with links 4, 5 and 6 fixed*

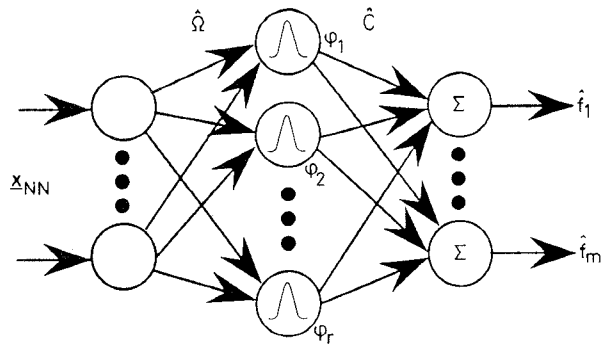


**Fig. 2** *Network representation of RFBF expansion system*

### 2.2 Rotated fuzzy basis function networks

A three-layer network representation of an RFBF network is shown in Fig. 2. The RFBF network can perform the procedures of fuzzification, fuzzy inference and defuzzification. Let input space $X \subset R^n$ be a compact product space. Assume that there are $r$ rules in the rule base and each of which has the following form:

$R_j$:If $x_1$ is $A_{1j}$ and $x_2$ is $A_{2j}$ and ... and $x_n$ is $A_{nj}$ then $y_1$ is $B_{j1}$ and $y_2$ is $B_{j2}$ and ... $y_m$ is $B_{jm}$,

where $j = 1, 2, ..., r$, the input vector $\underline{x} = (x_1, x_2, ..., x_n) \in X$, $x_i$ ($i = 1, 2, ..., n$) contains the input variables to the fuzzy system, $y_k$ ($k = 1, 2, ..., m$) are the output variables of the fuzzy system, and $A_{ij}$ and $B_{jk}$ are linguistic terms characterised by their corresponding fuzzy membership functions $\mu_{Aij}(x_i)$ and $\mu_{Bjk}(y_k)$, respectively.

As in [2], we consider a subset of fuzzy systems with singleton fuzzification, product inference, and Gaussian membership functions. Hence, such fuzzy system can be written as

$$f_k = \sum_{j=1}^{r} c_{jk} \left( \prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \right), \quad k = 1, \ldots, m \qquad (2)$$

where $f_k$: $X \subset R^n \rightarrow R^m$, $\mu_{Bjk}$ ($c_{jk}$) = 1, and $\mu_{Aij}$ ($x_i$) is the Gaussian membership function of the form

$$\mu_{A_{ij}} = \exp\left[ -\left( \frac{x_i - \xi_{ij}}{\sigma_{ij}} \right)^2 \right] \qquad (3)$$

where $\xi_{ij}$ and $\sigma_{ij}$ are real, and vector $\underline{\xi}_j = [\xi_{1j} \ldots \xi_{nj}]^T$ is an $n$-dimension vector.

From the point of view of geometry, the rule base of an RFBF network can be written as [13]: $R_j$: If ($\xi_{1j}$, $\sigma_{1j}$) and ($\xi_{2j}$, $\sigma_{2j}$) and ... and ($\xi_{nj}$, $\sigma_{nj}$), then ($c_{j1}$, $\sigma^y_{j1}$) and ($c_{jm}$, $\sigma^y_{jm}$), where $\underline{\sigma}^y_j = [\sigma^y_{j1}, \sigma^y_{j2}, ..., \sigma^y_{jm}]^T$ is the radius vector of THEN part of the $j$th rule, and $\underline{\sigma}_j = [\sigma_{1j}, \sigma_{2j}, ..., \sigma_{nj}]^T$ and $\underline{\xi}_j = [\xi_{1j}, \xi_{2j}, ..., \xi_{nj}]^T$ are radius and centre vector of IF part of the $j$th rule, respectively.

Definition 1: (Fuzzy Basis Function) Define fuzzy basis functions (FBFs) as

$$\phi_j\left( \|\underline{x} - \underline{\xi}_j\|, \underline{\sigma}_j \right) = \prod_{i=1}^{n} \mu_{A_{ij}}(x_i) = e^{-(\underline{x}-\underline{\xi}_j)^T S_j (\underline{x}-\underline{\xi}_j)}$$
$$j = 1, 2, \ldots, r \qquad (4)$$

where $\mu_{Aij}$ ($x_i$) are Gaussian membership functions (eqn. 3), $\underline{x} = [x_1, x_2, ..., x_n]^T \in X$ and $S_j = \text{diag}(1/\sigma^2_{1j}, 1/\sigma^2_{2j}, ..., 1/\sigma^2_{nj})$.

Motivated by [11, 12], the basis functions can be rotated to achieve better performance. We use the rotated fuzzy basis function which is defined as follows.

Definition 2: (Rotated Fuzzy Basis Function) Define rotated fuzzy basis functions (RFBFs) as

$$\phi_j\left( \|\underline{x} - \underline{\xi}_j\|, \underline{\sigma}_j \right) = e^{-(\underline{x}-\underline{\xi}_j)^T \Re_j^T S_j \Re_j (\underline{x}-\underline{\xi}_j)} = e^{-\underline{z}_j^T S_j \underline{z}_j}$$
$$j = 1, 2, \ldots, r \qquad (5)$$

where $\Re_j$ is the rotation matrix and $\mathbf{z}_j = \Re_j(\mathbf{x} - \underline{\xi}_j)$.

The RFBF expansion can be defined in the following definition.

Definition 3: (Rotated Fuzzy Basis Function Expansion) The set of a fuzzy system with rotated fuzzy basis functions consists of all functions of the form

$$y = \sum_{j=1}^{r} c_j \phi_j = \sum_{j=1}^{r} c_j e^{-\underline{z}_j^T S_j \underline{z}_j} \qquad (6)$$

where $c_j$ are weights of defuzzifier, $\phi_j$ are rotated fuzzy basis functions.

The concepts of input patterns and rule patterns are introduced in [10]. The rule pattern of the $j$th rule is defined by the centre $\underline{\xi}_j$ and radius $\underline{\sigma}_j$ of the $j$th rule. From the above definition, the $n$-dimensional vector $\Re_j \underline{\xi}_j$ will be referred to be a rotated input pattern and the rule pattern is the same as [10]. In other words, the difference between RFBF and FBF networks is just in that the inputs are multiplied by a rotated matrix for

each rule. Fig. 3 shows the difference of two-dimension elliptic isocontours of FBFs and RFBFs and the latter has more flexibility in structure. In [2, 3], the parameters of $\mu_{A_{ij}}(x_i)$ are fixed and the weights $\underline{c}_j$ are adjustable. However, in this paper, the parameters of $\mu_{A_{ij}}(x_i)$, $\xi_{ij}$ and $\sigma_{ij}$ are also adjustable and the parameter updated law will be stated in a later Section.

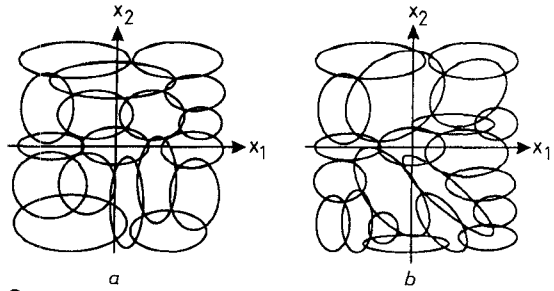


**Fig. 3** *Isocontours of fuzzy basis functions which has two input variables*
(*a*) Isocontours of fuzzy basis functions
(*b*) Isocontours of rotated fuzzy basis functions

Another type of rotated fuzzy basis functions we consider has an easily calculated form:

$$\phi_j = e^{-(\omega_{j0}+\omega_{j1}x_1+\cdots+\omega_{jn}x_n)^2} + \alpha_j \underline{x}_{Net}^T \underline{x}_{Net}$$
$$= e^{-\underline{x}_{Net}^T(\underline{\omega}_j\underline{\omega}_j^T+\alpha_j I)\underline{x}_{Net}}$$
$$j = 1, 2, \ldots, r \quad (7)$$

where $\omega_{j0}$, $\omega_{j1}$, ..., $\omega_{jn}$ are adjustable parameters, $\underline{\omega}_j = [\omega_{j0}, \omega_{j1}, ..., \omega_{jn}]^T$ and $\mathbf{x}_{Net} = [1\ \mathbf{x}^T]^T$.

It is easy to show that using the definition 2 eqn. 7 can be put into an RFBF form as in eqn. 5. Since the matrix is $\underline{\omega}_j\underline{\omega}_j^T + \alpha_j I$ a positive-definite matrix, there exists

$$\underline{\omega}_j\underline{\omega}_j^T + \alpha_j I = \Gamma_j^T \Lambda_j \Gamma_j, \quad j = 1, 2, \ldots, r \quad (8)$$

where $\Lambda_j = \text{diag}(\lambda_0, \lambda_1, \lambda_2, ..., \lambda_n)$. The matrix $\Gamma_j$ can be divided into four blocks:

$$\Gamma_j = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{21} & \Gamma_{22} \end{bmatrix} \quad (9)$$

where $\Gamma_{11}$ is 1 by 1, $\Gamma_{12}$ is 1 by $n$, $\Gamma_{21}$ is $n$ by 1 and $\Gamma_{22}$ is $n$ by $n$ matrix. Thus equation eqn. 5 can be rewritten as:

$$\phi_j = e^{-\lambda}0(\Gamma_{11} + \Gamma_{12}\underline{x})^2 e^{-(\Gamma_{21}+\Gamma_{22}\underline{x})^T \Lambda_j'(\Gamma_{21}+\Gamma_{22}\underline{x})} \quad (10)$$
$$j = 1, 2, \ldots, r$$

where $S_j = \Lambda_j' = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$, $\underline{\xi}_j = -\Gamma_{22}^{-1}\Gamma_{21}$ and $= \Re_j = \Gamma_{22}$. Therefore, eqn. 7 is a rotated fuzzy basis function multiplied by a magnitude modification term $e^{-\lambda}(\Gamma_{11} + \Gamma_{12}\underline{x})^2$.

Each fuzzy rule is described by membership functions of linguistic variables. In other words, in fuzzy basis function expansion, the parameters $\xi_{ij}$ and $\sigma_{ij}$ determine the IF part of rule and $c_j$ determine THEN part of the rule. From the previous derivation, while $\underline{\omega}_j$ is determined, we can find $\Gamma_j$ to determine $\xi_{ij}$ and $\sigma_{ij}$ by $S_j = \Lambda_j'$ and $\Gamma_{22}\ \underline{\xi}_j = -\Gamma_{21}$. Therefore, the IF parts of rules can be reconstructed from rotated fuzzy basis functions. The weights $\mathbf{c}$ of defuzzifier can be tuned by learning algorithm.

*Rule-base initialisation*: To initialise the input pattern and rule pattern of a fuzzy system [10] is to specify the centre and radius of each FBF. For RFBF, besides the specification of centre $\underline{\xi}_j$ and radius $\underline{\sigma}_j$, a rotated matrix $\Re_j$ is needed to be determined. As the rule base is initialised, the initial weight vector $\mathbf{w}$ can be determined by the following equations.

$$\Gamma_{22} = \Re_j \quad (11)$$

$$\Gamma_{21} = -\Gamma_{22}\underline{\xi}_j \text{ and} \quad (12)$$

$$\Gamma^T\Lambda\Gamma = \begin{bmatrix} \Gamma_{21}^T\Lambda'\Gamma_{21} & \Gamma_{21}^T\Lambda'\Gamma_{22} \\ \Gamma_{22}^T\Lambda'\Gamma_{21} & \Gamma_{22}^T\Lambda'\Gamma_{22} \end{bmatrix} \quad (13)$$

Therefore, the initial weight vector $\underline{w}$ can be calculated from $\omega_{j0}^2 = {\Gamma_{21}}^T\Lambda\Gamma_{21}$ and

$$\omega_{j0}\underline{\omega} = \begin{bmatrix} \Gamma_{21}^T\Lambda'\Gamma_{21} \\ \Gamma_{22}^T\Lambda'\Gamma_{21} \end{bmatrix}$$

In comparison with general self-organising fuzzy systems as [10], the size of the RFBF network is smaller. For an RFBF network, there are only $(n + 1) \times r + r \times m$ parameters to be tuned and $2n \times r + r \times m$ parameters for [10]. The parameters $\xi_{ij}$, $\sigma_{ij}$ determine the IF part and $\mathbf{c}_j$ determine the THEN part of the $j$th rule. Therefore, these parameters are stored and tuned in a compressed form. Another advantage of RFBF is that the experts' experience can be incorporated into the RFBF network by eqns. 11–13. The lack of experts' knowledge is the major difference of Gaussian neural networks to FBF systems. In general FBF systems, only weights $\mathbf{c}$ of defuzzification can be tuned by learning algorithm [1, 2]. However, the RFBF network can tune both the IF parts and THEN parts of rules. In essential, the RFBF network is a multilayer neural networks with Gaussian activation functions. The sigmoid functions may result in local errors producing global change for function mapping, however, Gaussian activation functions are capable of eliminating this global interaction [13].

## 3 Robot RFBF controller design

### 3.1 RFBF-based controller

The control objective is to design a robust RFBF-based controller so that the movement of robot arms follow the desired trajectory even in the presence of disturbances. Given a desired robot manipulator trajectory $\underline{\theta}_d(t)$, the tracking error vector is $\underline{e}(t) = \underline{\theta}_d(t) - \underline{\theta}(t)$ and error metric $\underline{s}(t)$ can be defined as

$$\underline{s}(t) = \dot{\underline{e}}(t) + \Lambda\underline{e}(t) \quad (14)$$

where $\Lambda = \Lambda^T > 0$. The RFBF network is used to approximate unknown nonlinear function

$$\underline{f} = M(\underline{\theta})(\ddot{\underline{\theta}}_d + \Lambda\dot{\underline{e}}) + V_m(\underline{\theta}, \dot{\underline{\theta}})(\underline{\theta}_d + \Lambda\underline{e} - \Delta sat(\underline{s})) + G(\underline{\theta}) + F\dot{\underline{\theta}} \quad (15)$$

The approximation error and unmodelled uncertainties are inevitable. Therefore, two methods are adopted to compensate them: one is the deadzone function and another is a robust term. Deadzones can be incorporated into error metrics by defining continuous function $\underline{s}_\Delta$ and $\underline{s}_\Delta^a(t)$ as:

$$\underline{s}_\Delta(t) = \underline{s}(t) - \Delta sat(\underline{s}(t)/\Delta) \text{ and} \quad (16)$$

$$\mathbf{s}_\Delta^a(t) = [s_1^a \quad \cdots \quad s_m^a]^T \quad (17)$$

where *sat* is the saturation function:

$$sat(z) = \begin{cases} 1, & z > 1 \\ z, & |z| \le 1 \\ -1, & z < -1 \end{cases}$$

Deadzone functions, which are specified around the zero of their corresponding error metrics, will be used in the adaptation law to tolerate the parameter errors and approximator errors. The architecture of the controller is shown in Fig. 4. The control law is given by

$$\underline{\tau}(t) = \underline{\nu}_{RFBF}(t) + \hat{\underline{d}}(t) + K\underline{s}_\Delta \quad (18)$$

where $K$ is a positive definite diagonal matrix, $\underline{\nu}_{RFBF}(t) = \hat{C}\underline{\phi}$ and $\underline{\hat{d}}(t)$ is the robust term. There are three main reasons for using rotated fuzzy basis function expansion system as the basic component of an adaptive fuzzy controller. First, the rotated fuzzy basis function expansion can be represented by a three-layer neural network as shown in Fig. 3 rather than five-layer neural network as in [2, 3]. The general learning algorithm can be used to train the network. Secondly, the experiences of human experts can be incorporated into the controller by the linguistic IF-THEN rules. The third is that the fuzzy system is a universal approximator. It has been proven in [7] that for any given real function $f$ over $X$, there exists a fuzzy system in the fuzzy basis function expansion form of (eqn. 16) such that it can uniformly approximate $f$ on the compact set $X$ to arbitrary accuracy. Accordingly, we have the following assumption:

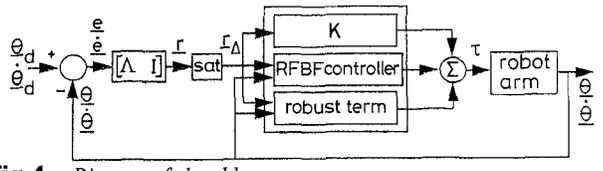


**Fig. 4** *Diagram of closed-loop system*

*Assumption 1*: There exist matrices $\Omega^*$ and $C^*$ such that $\underline{\hat{f}}$ approximate $\underline{f}$ with arbitrary accuracy $\underline{\varepsilon}$ over a compact set $X$, i.e.

$$\exists \Omega^* \text{ and } C^* \text{ s.t. } |\underline{f}(\underline{x}(t)) - \underline{\hat{f}}(\Omega^*, C^*, \underline{x}(t))| \leq \underline{\varepsilon}$$

Hence, eqn. 1 can be rewritten as:

$$M\underline{\dot{s}}(t) = -(V_m + K)\underline{s}_\Delta(t) + \underline{\hat{f}}(\Omega^*, C^*, \underline{x}(t)) - \underline{\nu}_{RFBF}(t) - \underline{\hat{d}}(t) + \underline{\delta}(t) + \underline{\tau}_d \quad (19)$$

where the disturbance $\underline{\delta}(t) = \underline{f}(\underline{x}(t)) - \underline{\hat{f}}(\Omega^*, C^*, \underline{x}(t))$ satisfies $|\delta_i(t)| \leq \varepsilon_i$.

The architecture of RFBF controller as an approximator is shown in Fig. 2. Denote the approximator having $(n + 1)$ inputs, $r$ rules and $m$ outputs as follows:

$$\underline{\hat{f}}(\underline{x}_{Net}, \hat{\Omega}, \hat{C}) = \underline{\nu}_{RFBF}(t) = \hat{C}\phi(\hat{\Omega}\underline{x}_{Net}) \quad (20)$$

where $\hat{\Omega} \in R^{r\times(n+1)}$ and $\hat{C} \in R^{m\times r}$ are RFBF network weight matrix and defuzzification weight matrix, respectively, and RFBF vector is

$$\phi(\hat{\Omega}\mathbf{x}_{Net}) = \begin{bmatrix} \phi(\hat{\omega}_1^T\mathbf{x}_{Net}) \\ \vdots \\ \phi(\hat{\omega}_r^T\mathbf{x}_{Net}) \end{bmatrix} \text{ and } \hat{\Omega} = \begin{bmatrix} \hat{\omega}_1^T \\ \vdots \\ \hat{\omega}_r^T \end{bmatrix} \quad (21)$$

For simplicity, we define $\underline{\hat{\phi}} = \phi(\hat{\Omega}\underline{x}_{Net})$ and $\underline{\phi}^* = \phi(\Omega^*\underline{x}_{Net})$. Thus, eqn. 19 becomes

$$M\underline{\dot{s}}(t) = -(V_m + K)\underline{s}_\Delta(t) + C^*\underline{\phi}^* - \hat{C}\underline{\hat{\phi}} - \underline{\hat{d}}(t) + \underline{\delta}(t) + \underline{\tau}_d \quad (22)$$

The RFBF can be expressed by Taylor series expansion [3] as

$$\underline{\phi}^* = \underline{\hat{\phi}} + \hat{\phi}'\tilde{\Omega}\underline{x}_{Net} + O((\tilde{\Omega}\underline{x}_{Net})^2) \text{ or} \quad (23)$$

$$\underline{\tilde{\phi}} = \hat{\phi}'\tilde{\Omega}\underline{x}_{Net} + O((\tilde{\Omega}\underline{x}_{Net})^2) \quad (24)$$

where $\underline{\tilde{\phi}} = \underline{\phi}^* - \underline{\hat{\phi}}$, $\tilde{\Omega} = \Omega^* - \hat{\Omega}$ and $\hat{\phi}' = \text{diag}(\hat{\phi}_1, \ldots, \hat{\phi}_r)$. Substituting eqn. 23 into eqn. 22, we can get

$$M\underline{\dot{s}}(t) = -(V_m + K)\underline{s}_\Delta(t) + \tilde{C}\underline{\hat{\phi}} + \hat{C}\hat{\phi}'\tilde{\Omega}\underline{x}_{Net} + \underline{d} - \underline{\hat{d}} \quad (25)$$

where $\tilde{C} = C^* - \hat{C}$ and $\underline{d} = \tilde{C}\hat{\phi}'\tilde{\Omega}\underline{x}_{Net} + C^*O((\tilde{\Omega}\underline{x}_{Net})^2) + \underline{\delta}(t) + \underline{\tau}_d$. In the following, the norm of vector or matrix, $\|\bullet\|$, is Frobenius norm [14]. The upper bound of the norms of $O((\tilde{\Omega}\underline{x}_{Net})^2)$ is given by:

$$\begin{aligned} \left\|O((\tilde{\Omega}\underline{x}_{Net})^2)\right\| &= \left\|\underline{\tilde{\phi}} - \hat{\phi}'\tilde{\Omega}\underline{x}_{Net}\right\| \\ &\leq \left\|\underline{\tilde{\phi}}\right\| + \left\|\hat{\phi}'\right\| \cdot \left\|\tilde{\Omega}\right\| \cdot \|\underline{x}_{Net}\| \\ &\leq \kappa_1 + \kappa_2 \cdot \left\|\tilde{\Omega}\right\| \cdot \|\underline{x}_{Net}\| \end{aligned} \quad (26)$$

where $\|\underline{\tilde{\phi}}\| \leq \kappa_1$ and $\|\hat{\phi}'\| \leq \kappa_2$. And, the upper bound of the norm of $\underline{d}$ is as follows:

$$\begin{aligned} \|d_i\| &\leq \left\|\tilde{C}\hat{\phi}'\tilde{\Omega}\underline{x}_{NN} + C^*O((\tilde{\Omega}\underline{x}_{Net})^2) + \underline{\delta}(t) + \underline{\tau}_d\right\| \\ &\leq \left\|\tilde{C}\right\| \cdot \kappa_2 \cdot \left\|\tilde{\Omega}\right\| \cdot \|\underline{x}_{Net}\| \\ &\quad + \kappa_C \cdot (\kappa_1 + \kappa_2 \left\|\tilde{\Omega}\right\| \cdot \|\underline{x}_{Net}\|) \\ &\quad + \|\underline{\delta}(t)\| + \|\underline{\tau}_d\| \\ &\leq (\kappa_C + \left\|\hat{C}\right\|) \cdot \kappa_2 \cdot (\kappa_\Omega + \left\|\hat{\Omega}\right\|) \cdot \|\underline{x}_{Net}\| \\ &\quad + \kappa_C \cdot (\kappa_1 + \kappa_2 \cdot (\kappa_\Omega + \left\|\hat{\Omega}\right\|) \cdot \|\underline{x}_{Net}\|) \\ &\quad + \|\underline{\delta}(t)\| + \|\underline{\tau}_d\| \\ &= A_{\alpha i}\underline{\beta} \end{aligned} \quad (27)$$

where $d_i$ is the $i$th element of $\underline{d}$, $\|\tilde{\Omega}\| \leq \|\Omega^*\| + \|\hat{\Omega}\|$, $\|\tilde{C}\| \leq \|C^*\| + \|\hat{C}\|$, $\|\Omega^*\| \leq \kappa_\Omega$, $\|C^*\| \leq \kappa_C$, $\|\underline{\phi}\| \leq \kappa_1$, $\|\hat{\phi}'\| \leq \kappa_2$, $\underline{\beta}^T = [1\ \|\underline{x}_{Net}\|\ \|\hat{\Omega}\|\cdot\|\underline{x}_{Net}\|\ \|\hat{C}\|\cdot\|\underline{x}_{Net}\|\ \|\hat{\Omega}\|\cdot\|\hat{C}\|\cdot\|\underline{x}_{Net}\|]$ ($\kappa_1$, $\kappa_2$, $\kappa_\Omega$, $\kappa_C$ are constants) and $A_\alpha i$ is the $i$th row of $A_\alpha$. The robustifying term is as

$$\underline{\hat{d}}(t) = Z\hat{A}_\alpha\underline{\beta}$$

where $Z = \text{diag}(\text{sgn}(s_{1\Delta}), \text{sgn}(s_{2\Delta}), \text{sgn}(s_{3\Delta}))$ and sgn is a sign function:

$$\text{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

The parameters $\hat{C}$, $\hat{\Omega}$ and $\hat{A}_\alpha$ are updated by the following adaptation law which is similar to the back-propagation algorithm:

$$\dot{\hat{C}}^T = K_C\underline{\hat{\phi}}\mathbf{s}_\Delta^T \quad (28)$$

$$\dot{\hat{\Omega}}^T = K_\Omega\underline{x}_{Net}\underline{s}_\Delta^T\hat{C}\hat{\phi}' \quad (29)$$

$$\dot{\hat{A}}_\alpha^T = K_\alpha\underline{\beta}\underline{s}_\Delta^T Z \quad (30)$$

where $K_C$, $K_\Omega$ and $K_\alpha$ are positive symmetric constant matrices determining the adaptation rate.

### 3.2 Stability analysis

A stability theorem is presented for the tuning algorithm eqns. 28–30.

*Theorem 1*: Consider the dynamic eqn. 1 with the control law eqn. 18 and weight tuning algorithm eqns. 28–30. All states in eqn. 31 will remain bounded and the tracking errors will approach zero.

*Proof*: Consider the Lyapunov function candidate

$$V(t) = \frac{1}{2}\left(\underline{s}_\Delta^T M\underline{s}_\Delta + tr\left(\tilde{\Omega}^T K_\Omega^{-1}\tilde{\Omega}\right) + tr\left(\tilde{C}^T K_C^{-1}\tilde{C}\right) + tr\left(\tilde{A}_\alpha^T K_\alpha^{-1}\tilde{A}_\alpha\right)\right) \quad (31)$$

Evaluating the time derivative of $V(t)$ along the trajectories of the tuning law eqns. 28–30, then

$$\dot{V} = 0, \text{ when } |s_i| \leq \Delta, \ i = 1, 2$$

When $|s_i| > \Delta$ for each $i$, by the property of skew-symmetry of $\dot{M} - 2V_m$, we can get

$$\begin{aligned}\dot{V} &= \underline{s}_\Delta^T \left(-K\underline{s}_\Delta + \tilde{C}\hat{\underline{\phi}} + \hat{C}\hat{\phi}'\tilde{\Omega}\underline{x}_{Net} + \underline{d} - \hat{\underline{d}}\right) \\ &\quad - tr\left(\tilde{\Omega}\underline{x}_{Net}\underline{s}_\Delta^T\hat{C}\hat{\phi}'\right) - tr\left(\tilde{C}\hat{\underline{\phi}}\underline{s}_\Delta^T\right) - tr\left(\tilde{A}_\alpha^T\underline{\beta}\underline{s}_\Delta^T Z\right) \\ &\leq -\underline{s}_\Delta^T K\underline{s}_\Delta + \underline{s}_\Delta^T Z\tilde{A}_\alpha\underline{\beta} - tr\left(\tilde{A}_\alpha^T\underline{s}_\Delta^T\underline{\beta}Z\right) \\ &= -\underline{s}_\Delta^T K\underline{s}_\Delta \leq 0 \end{aligned} \quad (32)$$

Therefore, if $\underline{s}_\Delta$, all $\tilde{c}_{ij}$ and all $\tilde{\omega}_{ij}$ are bounded at initial time $t = 0$, they will remain bounded for all time $t > 0$. If $\tilde{\underline{y}}_i(0)$ is bounded, then $\tilde{\underline{y}}_i(t)$ is also bounded for all time $t$, and since $\underline{y}_{id}(t)$ is bounded specified, $\underline{y}_i(t)$ is bounded as well. Next, we will show that $\mathbf{s}_\Delta \to 0$ as $t \to \infty$. It is easy to show by Barbalat's lemma:

$$V_1(t) = V(t) - \int_0^t \left[\dot{V}(\tau) + \underline{s}_\Delta^T K\underline{s}_\Delta\right] d\tau \text{ with} \quad (33)$$

$$\dot{V}_1(t) = -\underline{s}_\Delta^T K\underline{s}_\Delta \quad (34)$$

Thus, we have shown that every term in eqn. 31 is bounded; hence $\underline{s}_\Delta$ is bounded and $\underline{s}$ is bounded as well. This implies that $\dot{V}_1(t)$ is a uniformly continuous function of time. Since $V_1$ is bounded by 0, and $\dot{V} \leq 0$ for all time $t$, Barbalat's lemma can be applied to prove that $\dot{V}_1 \to 0$ and hence $\underline{s}_\Delta \to 0$ as $t \to \infty$. Q.E.D.

## 4 Simulation results

In this section, PUMA-560 is taken as the robot manipulator to be controlled. As in [15], the fourth, fifth and sixth links of PUMA-560 were fixed, and the angles of the second and third links were considered to be $\theta_1$ and $\theta_2$, respectively. The numerical values of the robot model can refer to [15]. The desired trajectories for $\theta_1$ and $\theta_2$ were chosen as:

$$\theta_{d1} = 0.5 + 0.2(\sin t + \sin 2t) \quad (rad) \text{ for } \theta_1 \text{ and}$$

$$\theta_{d2} = 1.3 - 0.1(\sin t + \sin 2t) \quad (rad) \text{ for } \theta_2$$

The proposed RFBF controller was compared with the well-known Slotine–Li's adaptive controller. The Slotine–Li's method [12] estimated nine parameters; the proposed method approximated nonlinear functions eqn. 15. The derivative gains of Slotine–Li's method were $K_D$ = diag(250, 250). For the RFBF network, there were 20 rules in the rule base and the parameters of RFBF network were given by eqns. 28–30. The adaptation rates were specified as $K_C = 100.0\mathrm{I}_{r\times r}$, $K_\Omega = 50.0\mathrm{I}_{(n+1)\times(n+1)}$ and $K_\alpha = 0.05\mathrm{I}_{5\times 5}$ ($\mathrm{I}_{p\times p}$ is a $p \times p$ identity matrix). Figs. 5 and 6 show the desired trajectories and trajectories obtained from RFBF and Slotine–Li's controller. The maximum tracking errors of $\theta_1$ and $\theta_2$ after the first two seconds of movement of the robot arm using the Slotine–Li's method were 0.72° and 0.60°. Using the proposed RFBF controller, the errors were found to be 0.39° and 0.08°, respectively. This comparison shows the proposed controller can obtain more accurate tracking performance due to the good approximation capability of the RFBF network as shown in Figs. 8–11 show the simulation results with bounded disturbances $\underline{\tau}_d$. These results imply the robustness of the proposed RFBF controller.

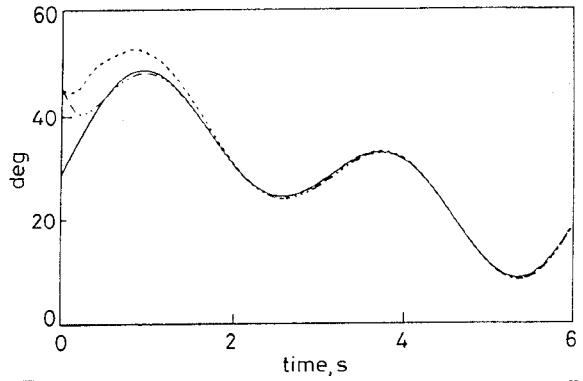


**Fig. 5** *Simulations for $\theta_1(t)$ using Slotine–Li's and RFBFN controller*
——— desired trajectory
····· RFBFN controller
– – · · · – – Slotine–Li's controller

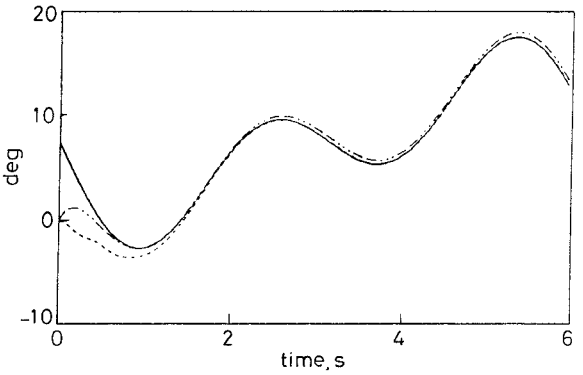


**Fig. 6** *Simulations for $\theta_2(t)$ using Slotine–Li's and RFBFN controller*
——— desired trajectory
····· RFBFN controller
– – · · · – – Slotine–Li's controller

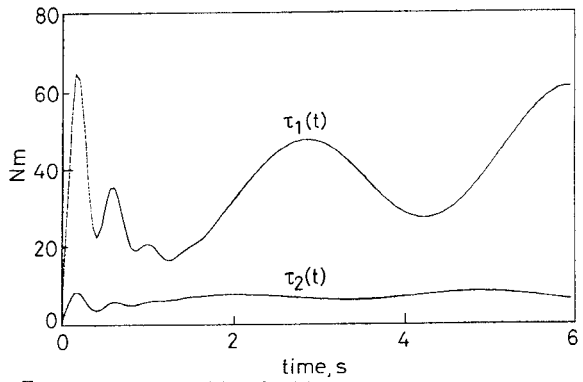


**Fig. 7** *Input torques: $\tau_1(t)$ and $\tau_2(t)$*

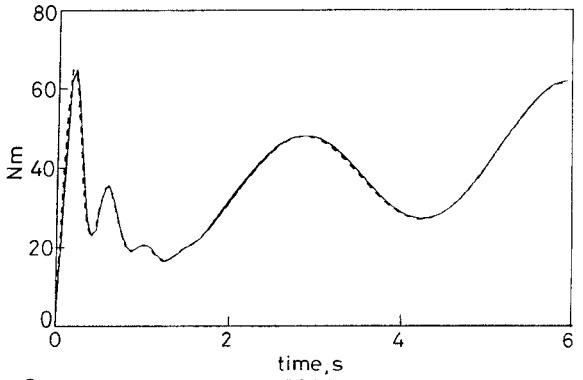


**Fig. 8** *Function approximation of $f_1(t)$*
——— desired trajectory
····· RFBFN controller

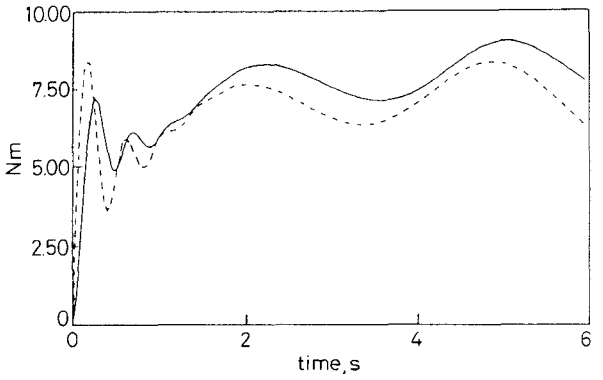


**Fig. 9** *Function approximation of $f_2(t)$*
——— desired trajectory
···· RFBFN controller

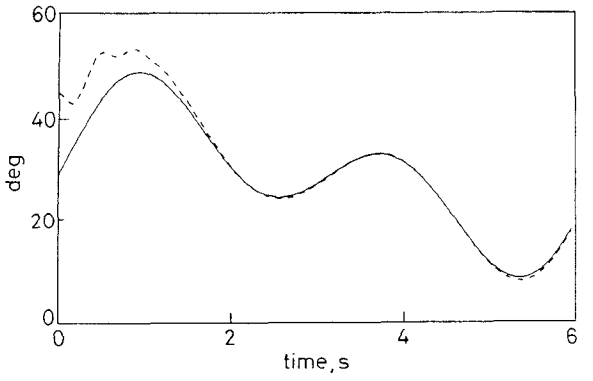


**Fig. 10** *Simulations for $\theta_1(t)$ using RFBFN with disturbances*
——— desired trajectory
···· RFBFN controller

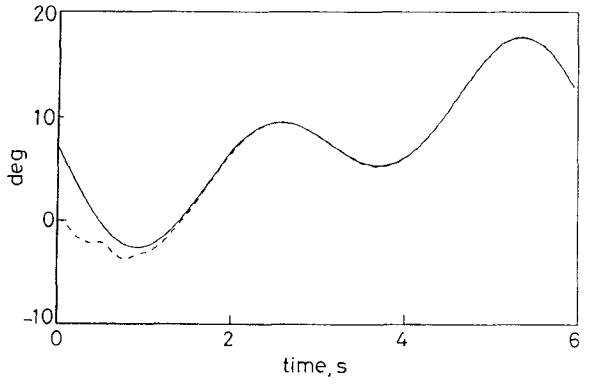


**Fig. 11** *Simulations for $\theta_2(t)$ using RFBFN with disturbances*
——— desired trajectory
···· RFBFN controller

## 5 Conclusions

A self-tuning rotated fuzzy basis function network has been proposed. The controller design is based on the proposed RFBF which is capable of incorporating experts' experience into a controller. The self-tuning rotated fuzzy basis function control system used a smaller network size than other neural fuzzy systems to achieve the desired performance. The controller is flexible because all parameters of the RFBF network can be tuned by adaption law once the rule number is determined. By combining the deadzone functions and robustness techniques, we can show that the controller is robust. Sound tracking results could be obtained by the proposed controller. Simulation shows that the RFBF controller is more accurate than the well-known Slotine–Li's controller. The stability analysis shows that the control system can be guaranteed to be asymptotically stable.

## 6 References

1 SU, C.-Y., and STEPANENKO, Y.: 'Adaptive fuzzy control of a class of nonlinear systems with fuzzy logic', *IEEE Trans. Fuzzy Syst.*, 1994, **2**, (4), pp. 285–294
2 WANG, L.X.: 'Stable adaptive fuzzy control of nonlinear systems', *IEEE Trans. Fuzzy Syst.*, 1993, **1**, (2), pp. 146–155
3 WANG, L.X.: 'Design and analysis of fuzzy identifiers of nonlinear dynamics systems', *IEEE Trans. Autom. Control*, 1995, **40**, (1), pp. 11–23
4 PROCYK, T.J., and MAMDANI, E.H.: 'A linguistic self-organizing process controller', *Automat.*, 1979, **15**, (1), pp. 15–30
5 LIN, C.–K.: 'A fuzzy self-organizing controller for robust control'. M.S. thesis, Electrical Engineering, National Taiwan Univ., Taiwan, 1991
6 SHAO, S.: 'Fuzzy self-organizing controller and its application for dynamic processes', *Fuzzy Sets Syst.*, 1988, **26**, pp. 151–164
7 YAMAGUCHI, T., TAKAGI, T., and MITA, T.: 'Self-organizing control using fuzzy neural networks', *Int. J. Control*, 1992, **56**, (2), pp. 415–439
8 LIN, C.T., and LEE, C.S.G.: 'Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems', *IEEE Trans. Fuzzy Syst.*, 1995, **2**, (1), pp. 46–63
9 JANG, J.-S.R.: 'Self-learning fuzzy controllers based on temporal back propagation', *IEEE Trans. Neural Netw.*, 1992, **3**, (5), pp. 714–723
10 NIE, J., and LINKENS, D.A.: 'Learning control using fuzzified self-organizing radial basis function network', *IEEE Trans. Fuzzy Syst.*, 1993, **1**, (4), pp. 280–287
11 SANNER, R.M., and SLOTINE, J.-J.E.: 'Stable recursive identification using radial basis function networks'. Proceedings of American Control conference, 1992, pp. 1829–1833
12 SLOTINE, J.-J.E., and LI, W.: 'Adaptive manipulator control: a case study', *IEEE Trans. Autom.. Control*, 1988, **33**, (11), pp. 995–1003
13 KOFFMAN, S.J., and MECKL, P.H.: 'Gaussian network variants: a preliminary study'. Proceedings of international joint conference on *Neural networks*, 1993, pp. 523–528
14 FIERRO, R., and LEWIS, F.L.: 'Control of a nonholonomic mobile robot using neural networks'. IEEE international symposium on *Intelligent control*, 1995, pp. 415–421
15 ERLIC, M., and LU, W.-S.: 'A reduced-order adaptive velocity observer for manipulator control', *IEEE Trans. Robot. Autom.*, 1995, **11**, (2), pp. 293–303