International Journal of Innovative Computing, Information and Control
Volume **8**, Number **3(A)**, March **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 1955–1971

SELF-ORGANIZING FUZZY RADIAL BASIS FUNCTION NEURAL-NETWORK CONTROLLER FOR ROBOTIC MOTION CONTROL

Ruey-Jing Lian[1,*] and Chung-Neng Huang[2]

[1]Department of Management and Information Technology
Vanung University
No. 1, Wanneng Rd., Jhongli City, Toayuan County 32061, Taiwan
[*]Corresponding author: rjlian@vnu.edu.tw; rjlian@ms79.hinet.net

[2]Graduate Institute of Mechatronic System Engineering
National University of Tainan
No. 33, Sec. 2, Shu-Lin St., Tainan 70005, Taiwan
kosono@mail.nutn.edu.tw; ngshs01@gmail.com

Received December 2010; revised April 2011

Abstract. *Robotic systems are complicated, nonlinear, multiple-input multiple-output (MIMO) systems, which make the design of model-based controllers for robotic systems particularly difficult. Moreover, to achieve reasonable control performance, the dynamic coupling effects between degrees of freedom (DOFs) of the robotic systems must be overcome during the control process. Although a model-free, self-organizing fuzzy controller (SOFC) can be applied to the manipulation of complex and nonlinear systems, its parameters are difficult to select appropriately, and it mainly focuses on controlling single-input single-output systems rather than MIMO systems. To address these problems, this study developed a self-organizing fuzzy radial basis function neural-network controller (SFRBFNC) for robotic systems. The SFRBFNC introduces a radial basis function neural-network into the SOFC to compensate for the dynamic coupling effects between the DOFs of the robotic system, as well as solve the problem caused by inappropriate selection of parameters in designing an SOFC. The SFRBFNC demonstrated control performance superior to the SOFC, as shown in experimental results from motion control tests of a 6-DOF robot.*

Keywords: Multiple-input multiple-output (MIMO) systems, Radial basis function neural-network, Robotic motion control, Self-organizing fuzzy controller (SOFC)

1. **Introduction.**

Research and development of industrial automation are important to improve productivity and quality. The robot is one of the most effective machines used in industrial automation. Flexible multifunctional robots are ideal candidates to replace humans for high-risk jobs in unsafe environments or those requiring repetitive motions. The design of the controller for robotic trajectory motion needs to be efficient and accurate since many industrial applications require robots to perform tasks with high precision. Robots are complicated, nonlinear, multiple-input multiple-output (MIMO) systems. It is difficult to design model-based controllers to manipulate such systems. Thus, it is necessary to develop model-free control strategies for the control of complex and nonlinear robots.

Fuzzy logic control, which does not require a mathematical model of the system, has been successfully applied to robotic systems to improve their control performances [1-3]. However, a fuzzy logic controller (FLC) for practical applications has difficulties in determining suitable membership functions and fuzzy rules. Moreover, the main problem in the design of an FLC is that both the inference table and the knowledge base of the FLC, which are constructed using an expert's knowledge or the experience of a skilled operator, are fixed after selection. To solve the problems of the FLC implementation, Procyk and Mamdani [4] first proposed a self-organizing fuzzy controller (SOFC). This control strategy involves the use of online learning, rather than human thinking, to establish fuzzy control rules, thereby simplifying the procedures for designing an FLC. Subsequently, Shao [5] and Zhang and Edmunds [6] developed modified learning methods to further simplify the design of the SOFC. However, the construction of the modified learning scheme is based on a performance decision table proposed by Procyk and Mamdani [4] and the design of a performance decision table is as difficult as the design of a fuzzy rule table. Therefore, to overcome this problem, Yang [7], Huang and Lee [8], and Lin and Lian [9,10] used the output error and the error change of the system to establish a learning algorithm that can adjust the linguistic fuzzy rule table of the SOFC directly, so that it can be generated without any initial fuzzy rules. The SOFC eliminates the difficulty of finding appropriate membership functions and fuzzy rules in designing an FLC. Under the disk operating system, Huang and Lee [8] employed this SOFC to control a robot with 5 degrees of freedom (DOFs) and evaluated its trajectory tracking performance.

The SOFC has demonstrated its superior learning ability in controlling complicated and nonlinear systems in practical applications [8-10]; however, both the learning rate and weighting distribution in the SOFC must be carefully chosen and are fixed once selected. Unfortunately, inappropriate selection of either the weighting distribution or the learning rate (or both) in the SOFC will substantially affect the output response of the system and may result in an unstable system. Moreover, the SOFC is primarily designed for controlling single-input single-output systems, so the use of the SOFC to control robotic systems, which are MIMO systems, cannot eliminate the dynamic coupling effects between the DOFs of the robotic systems.

Neural networks for robotic system control have attracted the attention of many researchers because they have model-free features and learning abilities. Neural networks have been used to control complicated robotic systems and their control performances have been demonstrated in previous studies [11,12]. However, in practical applications, convergence rates of neural networks are too slow to compensate for the dynamic coupling effects between the DOFs of robotic systems. Noticeably, it is difficult to achieve satisfactory control performance when using a neural network to control robotic systems, unless the convergence algorithm and the training speed of the neural network are significantly improved.

The design and implementation of fuzzy control systems with learning capacities introduced by neural networks have become very active areas of research in recent years [13-15]. Such a synergism between neural networks and fuzzy logic systems, integrated into a functional system, has provided a new way of realizing intelligent systems for various applications. Most of the hybrid fuzzy-logic and neural network control strategies make use of neural networks to determine the membership functions and use the determined membership functions to design appropriate fuzzy control rules, which are fixed once decided. Nevertheless, the design of these control strategies is very complicated and impractical for practical industrial applications. The robot is an example of an MIMO system. It is characterized by complicated dynamic coupling effects between the DOFs of the robot. Controller development for robotic systems needs to overcome these coupling effects to improve the overall control performances. However, in implementing robotic system control, few considerations [16,17] are usually entertained to deal with the dynamic coupling effects between the DOFs of robotic systems.

As mentioned previously, when the SOFC is used to handle robotic systems, its parameters are difficult to choose appropriately and the dynamic coupling effects between the DOFs of the robotic systems cannot be improved. To eliminate the problem, this study develops a self-organizing fuzzy radial basis function neural-network controller (SFRBFNC) for robotic systems. The SFRBFNC introduces a radial basis function neural-network (RBFN) [18-20], as the training algorithm of a neural network, into an SOFC to compensate for the dynamic coupling effects between the DOFs of the robotic system and overcome the problem encountered by the SOFC with inappropriately selected parameters. Therefore, the use of the SFRBFNC to manipulate robotic systems not only solves the problem of an SOFC implementation, but also alleviates the dynamic coupling effects between the DOFs of the robotic systems. Most of the work in this field involves computer simulations of simple robotic system models. This study explores the control performance of the SFRBFNC for motion control of a 6-DOF robot experimentally.

2. Robotic System.

A 5-DOF robot (type RV-MI), made by the Mitsubishi Company, had sliding track equipment added to its bottom in order to increase its horizontal movement capability and expand its workspace. Moreover, the robot was retrofitted, allowing an individual to use a personal-computer-based controller to control it. The retrofitted robot has 6 DOFs and each of its joints is driven by a direct current servo-motor.

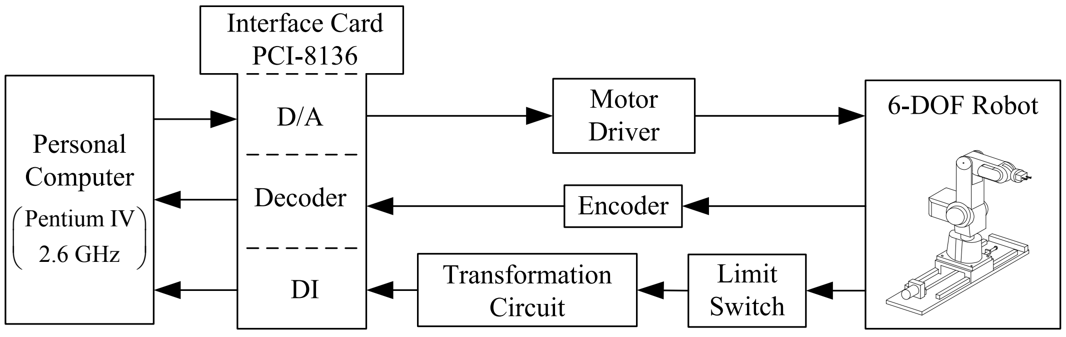


Figure 1. Experimental set-up of the retrofitted robotic control system. Digital-to-analog (D/A); digital-input (DI)

Figure 1 presents an experimental set-up of the retrofitted robotic control system. To evaluate the performance of the proposed controller for the control of the robotic system, an appropriate reference trajectory was first planned to convert the points $\overline{\theta}_i(k)$ into a continuously desired trajectory $\theta_{ri}(k)$. This study used the joint-space trajectory for a cubic interpolation polynomial [21] to plan the desired smooth motion for each joint of the robot.

Table 1. Coordinate parameters of the 6-DOF robot

Joint i	θ_i (deg.)	d_i (mm)	a_i (mm)	α_i (mm)
1	0	148	300	0
2	90	152	0	90
3	90	0	250	0
4	–90	0	–160	0
5	0	0	–72	–90
6	0	0	0	0

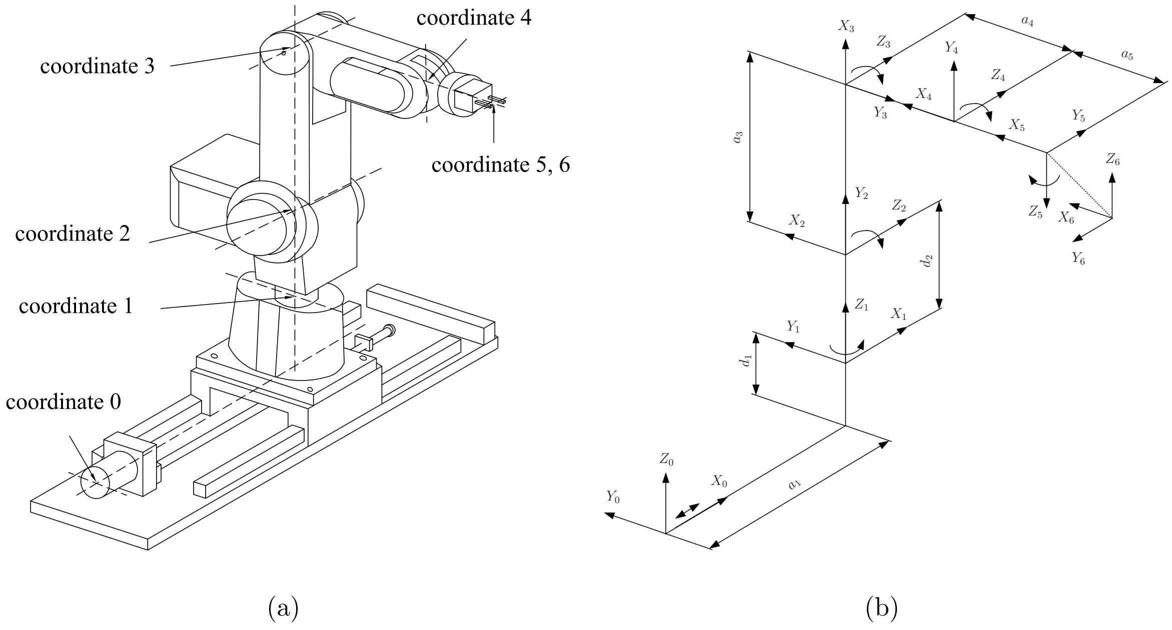


FIGURE 2. Coordinate system of the 6-DOF robot

Figure 2 shows the defined coordinate system and the joint parameters for the 6-DOF robot using the Denavit-Hartenberg (D-H) indicative method [22]. To determine the control performance of the robotic trajectory tracking, the kinematics equation and the inverse kinematics equation are first established for the relationship between the joint-space and the Cartesian-space. The joint parameters, defined as per the rules established by the D-H representation, are listed in Table 1. Similar to the derived process from Fu et al. [22] and Huang and Lian [16], the kinematics equation and the inverse kinematics equation of the 6-DOF robot can be determined.

The 6-DOF robotic system, in this study, exhibits nonlinear, time-varying characteristics because of backlash, friction, Coriolis coupling, gravity force, saturation of the actuator, and other factors. Traditional model-based controllers are difficult to implement on this complex robotic system. Although a model-free SOFC can be employed to control such systems, its learning rate and weighting distribution are arduous to select appropriately. When the SOFC with inappropriately chosen parameters is used to manipulate a robotic system, it may excessively modify its fuzzy rules during the control process so that the system's output response generally results in oscillatory phenomena or becomes unstable. In addition, the use of the SOFC to handle robotic systems, the dynamic coupling effects between the DOFs of the robotic systems cannot be compensated. To solve these problems, this study developed an SFRBFNC to control the 6-DOF robot for improving the control performance of the robot.

3. **Controller Design.**

3.1. **Self-organizing fuzzy algorithm.**

The self-organizing part is introduced into an FLC to constitute an SOFC, as depicted in Figure 3, and consists of three steps: performance measure, model estimation, and rule modification [7-10]. A generalized form for the self-organizing fuzzy algorithm has been proposed [9,10] as

$$\begin{aligned}\overline{u}^l(k+1) &= \overline{u}^l(k) + \Delta\overline{u}^l(k) \\ &= \overline{u}^l(k) + w_e^l w_{ec}^l \frac{\gamma}{M} \times [(1-\varsigma)e(k) + \varsigma ec(k)]\end{aligned} \tag{1}$$

where $\overline{u}^l(k+1)$ is the control input of the lth fuzzy rule on $k+1$ -step sampling interval. $\overline{u}^l(k)$ and $\Delta\overline{u}^l(k)$ are the control input and the control input correction of the lth fuzzy rule on k -step sampling interval, respectively. w is an excitation strength of each fuzzy rule. It is represented as a triangular membership function and is calculated using a linear interpolation algorithm [7]. $e(k)$ and $ec(k)$ are the output error and the error change of the system on k -step sampling interval, respectively. M is the direct forward system gain in the control system. It is generally set as 1 to eliminate the identification procedure of the system and to reduce the computational time required during implementation [7-10]. γ is a learning rate; ς is a weighting distribution.

For a multiple DOF self-organizing fuzzy control system, Equation (1) can be modified as

$$\begin{aligned}\overline{u}_i^l(k+1) &= \overline{u}_i^l(k) + \Delta\overline{u}_i^l(k) \\ &= \overline{u}_i^l(k) + w_{e_i}^l w_{ec_i}^l \frac{\gamma_i}{M} \times [(1-\varsigma_i)e_i(k) + \varsigma_i ec_i(k)]\end{aligned} \tag{2}$$

where i is used for indicating the DOF of the control system. A more complete description and discussion concerning the self-organizing fuzzy algorithm can be found in [9,10].

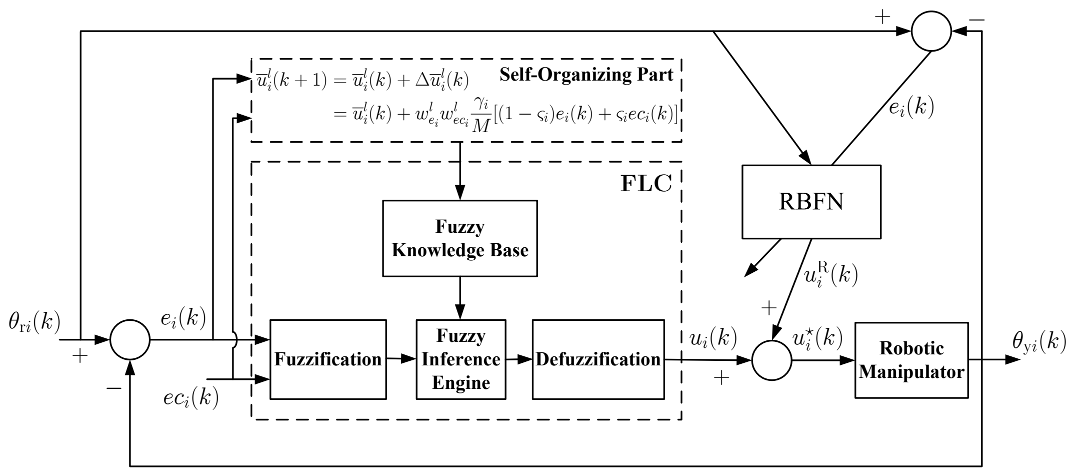


Figure 3. SFRBFNC for a robotic manipulator

3.2. **RBFN algorithm.** The RBFN has a feed forward structure consisting of an input layer with m inputs, a single hidden layer with s locally tuned neurons, and an output layer with n linear neurons. The s locally tuned neurons of the single hidden layer are fully interconnected to the n linear neurons of the output layer. All hidden neurons simultaneously receive the m-dimensional input vector X (see Figure 4). After the hidden layer receives the data from the input layer, the Gaussian basis function in the hidden layer is used to transform the data nonlinearly, and the function responses are then linearly combined to construct the output layer data.

Figure 4 shows a structural diagram of the RBFN. The algorithm of the RBFN can be described [18-20] as

$$O_j = \sum_{i=1}^{s} \overline{w}_{ij} h_i = \sum_{i=1}^{s} \overline{w}_{ij} R\left(\|X - Z_i\|\right),\ j = 1, 2, \cdots, n \tag{3}$$

and

$$h_i = R\left(\|X - Z_i\|\right),\ i = 1, 2, \cdots, s$$

$$R\left(\|X - Z_i\|\right) = \exp\left(-\frac{\|X - Z_i\|^2}{2\sigma_i^2}\right),\ i = 1, 2, \cdots, s$$
$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}^{\mathrm{T}}$$
$$Z_i = \begin{bmatrix} z_{i1} & z_{i2} & \cdots & z_{im} \end{bmatrix}^{\mathrm{T}},\ i = 1, 2, \cdots, s$$

where $\overline{w}_{ij}$ is a weighting, representing the strength of the connection from the hidden neuron h_i to the output neuron O_j; the jth output neuron of the output layer is a nonlinear function of the output value of the hidden layer neurons. The index i runs over all connections to the jth neuron. Z_i and σ_i are the central position and the standard deviation of the ith neuron receptive field, respectively; the norm, $\|\cdot\|$, is the 2-dimensional Euclidean space; $R(\cdot)$ is a Gaussian function.

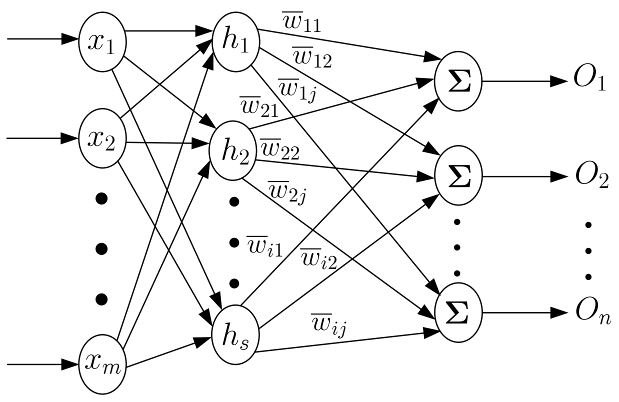


FIGURE 4. RBFN

The desired output is $T_j^\star$ and the corresponding output of the output layer is O_j. Then, a cost function of the RBFN is defined as

$$E = \frac{1}{2} \sum_{j=1}^{n} \left(T_j^\star - O_j\right)^2 = \frac{1}{2} \sum_{j=1}^{n} v_j^2 \tag{4}$$

To accelerate the convergence rate for the correction value of the weighting in the neural network, this study proposes a Levenberg-Marquardt (LM) algorithm [17,23,24], instead of the steepest descent method, to minimize the cost function of the neural network.

A generalized form for the LM algorithm has been proposed [23,24] as

$$\Delta x_k = -[J^{\mathrm{T}}(x_k) J(x_k) + \mu_k I]^{-1} J^{\mathrm{T}}(x_k) v(x_k) \tag{5}$$

where $\nabla E(x)|_{x=x_k} = J^{\mathrm{T}}(x_k) v(x_k)$. $J(x_k)$ is the Jacobian matrix of $v(x)$ at x_k; $E(x)$ is an objective function, $v(x_k)$ is a residual error, μ_k is a parameter, and I is an identity matrix. The LM algorithm has a very useful feature. It approaches the steepest descent algorithm with a small convergence rate as μ_k is increased,

$$x_{k+1} = x_k - \frac{1}{\mu_k} J^{\mathrm{T}}(x_k) v(x_k) = x_k - \frac{1}{\mu_k} \nabla E(x) \text{ for large } \mu_k \tag{6}$$

When μ_k is decreased to zero, the LM algorithm becomes a Gauss-Newton method. The LM algorithm begins from a small μ_k value (such as $\mu_k = 0.01$). If a step does not yield a smaller value for $E(x)$, the step will be repeated with μ_k multiplied by some factor $\eta > 1$ (such as $\eta = 10$). Ultimately, $E(x)$ should be reduced because a small step is taken in the direction of the steepest descent. If a step does not produce a smaller value for $E(x)$, μ_k will be divided by η for the next step. This is done so that the LM algorithm will approach the Gauss-Newton method, which should provide faster convergence [23].

According to Equations (3)-(6), the weighting correction ($\Delta\overline{w}_{ij}$), central position correction (ΔZ_i), and standard deviation correction ($\Delta\sigma_i$) of the RBFN can be separately determined as

$$\Delta\overline{w}_{ij} = -\left[J(\overline{w}_{ij})^{\mathrm{T}} J(\overline{w}_{ij}) + \mu_k I\right]^{-1} J^{\mathrm{T}}(\overline{w}_{ij}) v_j(\overline{w}_{ij}) \tag{7}$$

$$\Delta Z_i = -\left[J(Z_i)^{\mathrm{T}} J(Z_i) + \mu_k I\right]^{-1} J^{\mathrm{T}}(Z_i) v_j(Z_i) \tag{8}$$

$$\Delta\sigma_i = -\left[J(\sigma_i)^{\mathrm{T}} J(\sigma_i) + \mu_k I\right]^{-1} J^{\mathrm{T}}(\sigma_i) v_j(\sigma_i) \tag{9}$$

The output value of the RBFN has to be maintained within an appropriate range for this control strategy implementation, so a nonlinear transformation layer is introduced between the hidden layer and the output layer of the RBFN to regulate the output value of the RBFN to achieve the aforementioned goal. The procedure for determining the weighting correction $\Delta\overline{w}_{ij}$ of the nonlinear transformation layer is similar to the procedure for determining that of the RBFN between the hidden layer and the output layer.

To derive the weighting correction of the aforementioned RBFN, a batch learning method [25] was employed and an objective function of the RBFN for the step p was defined as

$$E_p^{\star} = \frac{1}{2} \sum_j (\theta_{\mathrm{r}pj} - \theta_{\mathrm{y}pj})^2 = \frac{1}{2} \sum_j e_{pj}^2 \tag{10}$$

where $\theta_{\mathrm{r}pj}$ and $\theta_{\mathrm{y}pj}$ express the desired set-points and the system outputs in the step p, respectively. When $E_p^{\star}$ approaches zero, the mapping between inputs and outputs of each step p is realized. Similar to the derived process of Equations (7)-(9), the weighting correction, central position correction, and standard deviation correction of the RBFN in the step p can be individually determined as

$$\Delta\overline{w}_{ij} = -\sum_p \left[J_p(\overline{w}_{ij})^{\mathrm{T}} J_p(\overline{w}_{ij}) + \mu_{kp} I\right]^{-1} J_p^{\mathrm{T}}(\overline{w}_{ij}) e_{pj}(\overline{w}_{ij})$$

$$\Delta Z_i = -\sum_p \left[J_p(Z_i)^{\mathrm{T}} J_p(Z_i) + \mu_{kp} I\right]^{-1} J_p^{\mathrm{T}}(Z_i) e_{pj}(Z_i)$$

$$\Delta\sigma_i = -\sum_p \left[J_p(\sigma_i)^{\mathrm{T}} J_p(\sigma_i) + \mu_{kp} I\right]^{-1} J_p^{\mathrm{T}}(\sigma_i) e_{pj}(\sigma_i)$$

If the input data at k -step is $X(k)$, the updated rules for the aforementioned corrections can be separately described as

$$\overline{w}_{ij}(k+1) = \overline{w}_{ij}(k) + \Delta\overline{w}_{ij}(k) \tag{11}$$

$$Z_i(k+1) = Z_i(k) + \Delta Z_i(k) \tag{12}$$

$$\sigma_i(k+1) = \sigma_i(k) + \Delta\sigma_i(k) \tag{13}$$

3.3. **Design of an SFRBFNC.** The SOFC is composed of a self-organizing part and an FLC, as shown in Figure 3. Its learning algorithm was presented in the preceding section. Design procedure of an FLC is described here. The structure of an FLC design consists of the following: the definition of input-output fuzzy variables, decision-making related to fuzzy control rules, fuzzy inference logic, and defuzzification. From Figure 3, the control variables of the system are defined as

$$e_i(k) = \theta_{\mathrm{r}i}(k) - \theta_{\mathrm{y}i}(k) \tag{14}$$

$$ec_i(k) = e_i(k) - e_i(k-1) \tag{15}$$

where $e_i(k)$ and $e_i(k-1)$ are the output errors of the system on k -step and $k-1$ -step sampling intervals, respectively; $ec_i(k)$ is the error change of the system on k -step sampling interval; $\theta_{\mathrm{r}i}(k)$ and $\theta_{\mathrm{y}i}(k)$ represent the reference input and the output response of the system on k -step sampling interval respectively. The subscript $(i = 1, 2, \ldots, 5)$ is employed to represent each joint of the robot.

A triangular membership function, depicted in Figure 5, is employed to convert these input variables ($e_i(k)$ and $ec_i(k)$) and the output variable ($u_i(k)$) into linguistic control variables (NB, NM, …, PB), where β_i^j is a scaling factor. The subscript i has been described previously. The superscript $(j = 1, 2, 3)$ is used to express the system's output error, error change, and control input.

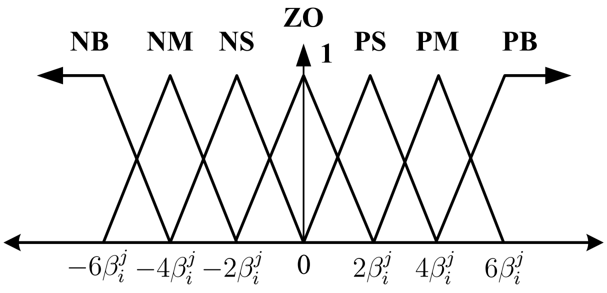


Figure 5. Membership function of the FLC. Negative big (NB), negative medium (NM), negative small (NS), positive big (PB), positive medium (PM), positive small (PS), and zero (ZO)

This study employed fuzzy control rules of the state evaluation [26] for controlling the inherently complicated and nonlinear robot. To prevent the change of the antecedent suitability of the fuzzy-inference logic to be unsmooth, the fuzzy-inference logic used an algebraic product [27], instead of the Max − Min product composition [26], to operate the fuzzy rules. Finally, this study applied the height method [26] to defuzzify the output variables to obtain accurate control inputs for controlling this system. The aforementioned design process yields the following actual control input of the actuator for this self-organizing fuzzy control system,

$$u_i(k) = u_i(k-1) + \Delta u_i(k) \tag{16}$$

where $\Delta u_i(k)$ indicates the control input increment of the system on k -step sampling interval. $u_i(k)$ and $u_i(k-1)$ represent the control inputs of the system on k -step and $k-1$ -step sampling intervals, respectively.

Figure 3 shows an SFRBFNC for a robotic manipulator, in which the variables of the input layer of the RBFN are $\theta_{\mathrm{r}i}(k)$ and $e_i(k)$. The variable of the output layer of the RBFN is $u_i^{\mathrm{R}}(k)$, which represents the control effort generated from the RBFN operation, to compensate for the dynamic coupling effects between the DOFs of the robotic system and to solve the problem caused by the inappropriate selection of the parameters in designing an SOFC. Accordingly, the total control input of each DOF of a robotic system $u_i^\star(k)$, obtained by combining an SOFC with an RBFN, can be represented as

$$u_i^\star(k) = u_i(k) + u_i^{\mathrm{R}}(k) \tag{17}$$

Moreover, it is important to determine the stability and robustness of the proposed controller for control applications. However, the stability and robustness of the SFRBFNC are difficult to demonstrate using a rigorously mathematical proof. This is because the proof requires complex mathematical operations and might be impossible to obtain at present. One of the methods such as the state-space approach [17], which does not need complicated mathematical operations, can be employed to determine the stability and robustness of the proposed controller. Therefore, the stability and robustness of the SFRBFNC can be demonstrated and guaranteed by following the derived process from Lin and Lian [17] using the state-space approach.

4. **Experimental Results.**

Figure 1 presents an experimental set-up of the retrofitted robotic control system. This retrofitted robot was controlled using a compatible IBM PC Pentium IV (2.6 GHz) central processing unit for processing all of the system input-output data, as well as the control parameters. The requisite interface is a PCI-8136 card comprising a digital-to-analog part with 6 channels, an analog-to-digital part with 6 channels, a digital-input part with 19 channels, and 6 decoding channels. The card came from the ADLINK Company.

This study introduced an RBFN into the SOFC to construct an SFRBFNC for robotic systems. To conveniently manipulate the robotic manipulator in practical industrial applications, a controller with a friendly graphical-user-interface (GUI) should be developed. To achieve this goal, this study used the Borland C^{++} Builder programming language to code the proposed controllers (SOFC and SFRBFNC) with friendly GUIs. The coding and the experimental tests were done under a Windows XP operating system.

The following experiments were performed to evaluate the effectiveness and feasibility of the proposed control strategy. The learning rate and the weighting distribution of the SOFC was selected as $\gamma_i = 0.7$ and $\varsigma_i = 0.5$ $(i = 1, 2, \ldots, 5)$, respectively, according to the experience in previous experiments on robotic system control. Figure 5 presents the membership function of the FLC. According to the dynamic characteristics of the system, the scaling factor β_i^j $(i = 1, 2, \ldots, 5;\ j = 1, 2, 3)$ for the parameters of both FLCs (within the SOFC and within the SFRBFNC, respectively) was chosen separately in Table 2. The sampling frequency in the experiments was 200 Hz during all control processes.

Table 2. Parameters of the FLC within both the SOFC and SFRBFNC

$\beta_1^1 = 100$	$\beta_1^2 = 1$	$\beta_1^3 = 0.5$
$\beta_2^1 = 10$	$\beta_2^2 = 0.5$	$\beta_2^3 = 0.5$
$\beta_3^1 = 50$	$\beta_3^2 = 0.8$	$\beta_3^3 = 0.5$
$\beta_4^1 = 10$	$\beta_4^2 = 2$	$\beta_4^3 = 0.3$
$\beta_5^1 = 10$	$\beta_5^2 = 2$	$\beta_5^3 = 1$

There is no special method that can be used to select an appropriate number of hidden neurons of the RBFN in practical applications. Thus, according to the dynamic characteristics of the system and the experimental tests, this study selected the number of hidden neurons of the RBFN as 40. Therefore, the RBFN has 10 input neurons, 40 hidden neurons, and 5 output neurons. Its input vector, $X(k)$, can be expressed as

$$X(k) = \left[\begin{array}{ccccccc} \theta_{\mathrm{r}1}(k) & e_1(k) & \theta_{\mathrm{r}2}(k) & e_2(k) & \cdots & \theta_{\mathrm{r}5}(k) & e_5(k) \end{array}\right]^{\mathrm{T}}$$

and its output neuron O_j, for $j = 1, 2, \ldots, 5$, can be expanded and represented as $O_1 = u_1^{\mathrm{R}}(k)$, $O_2 = u_2^{\mathrm{R}}(k)$, …, $O_5 = u_5^{\mathrm{R}}(k)$, whose initial values are all set to 0 to represent the beginnings of the dynamic coupling effects of the robotic system control from zero. According to the dynamic characteristics of the system, this study set all initial values of the central position Z_i to 25, the standard deviation σ_i to 20, and the weighting $\overline{w}_{ij}(k)$ to 0.2.

Case 1: Motion control of the joint-space trajectory planning. The desired motion trajectory is a cubic interpolation polynomial based on five desired positions on a sinusoidal curve. Because the desired trajectory and the output response of the joint-space of the robot are too close to be differentiated from each other, this study shows only tracking errors of the joint-space trajectory planning for the robotic motion control. Figures 6(a) and (b) displays the tracking errors of the learning process of the 3rd joint-space trajectory planning using the SOFC and the SFRBFNC, respectively. It can be seen that the SOFC needs five learning cycles to achieve reasonable control performance for the control of the joint-space trajectory planning; however, the SFRBFNC needs only two learning cycles.

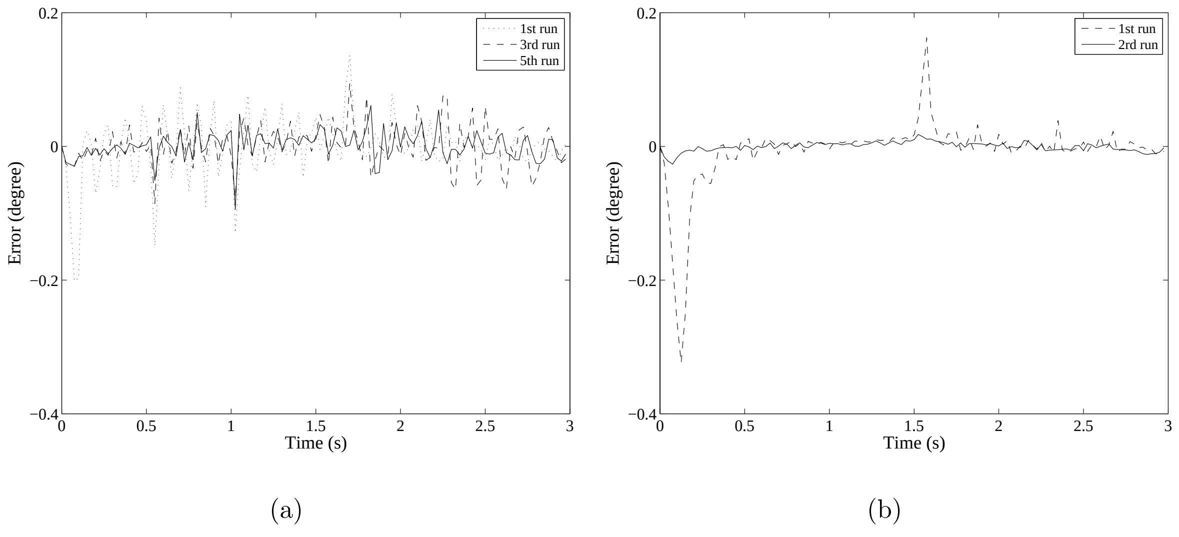


FIGURE 6. Tracking error histories of the 3rd joint-space trajectory planning: (a) with the SOFC and (b) with the SFRBFNC

To compare the control performance between the SOFC and the SFRBFNC further, Figure 7(a) plots the tracking errors of the 3rd joint-space trajectory planning using the SOFC with five learning cycles and the SFRBFNC with two learning cycles. Figure 7(b) shows their corresponding control efforts. From Figure 7(a), the maximum tracking error of the 3rd joint-space trajectory planning is 0.0781° with the use of the SOFC, as compared with 0.0178° with the use of the SFRBFNC. The control effort of the SFRBFNC is smoother than that of the SOFC, as seen in Figure 7(b). This implies that the use of the SFRBFNC to manipulate the robot provides longer actuator service life than the use of the SOFC. Moreover, the root-mean-square (RMS) error of the 3rd joint-space trajectory planning using the SOFC is 0.0215°, while the use of the SFRBFNC reduced the RMS error to 0.0068°. Clearly, the use of the SFRBFNC exhibits better control performance

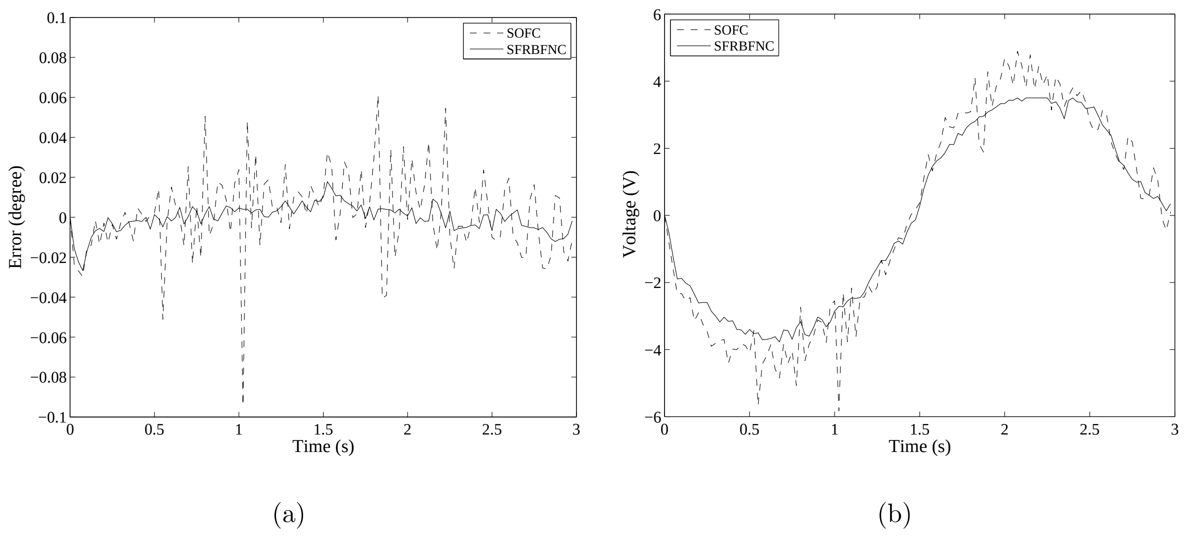


Figure 7. (a) Tracking errors and (b) control voltages of the 3rd joint-space trajectory planning using the SOFC (with five learning cycles) and the SFRBFNC (with two learning cycles)

than the use of the SOFC for manipulating the joint-space trajectory planning, in terms of reducing the maximum tracking errors and RMS errors of the joint-space trajectory, reducing the number of learning cycles, and producing fewer control command variations.

In this study, the SOFC is used to control the robotic system. It needs five learning cycles. This is in contrast to the use of the SFRBFNC, which needs only two learning cycles. Although an SFRBFNC with more than two learning cycles can be employed to control the robotic system, its control performance is almost the same as that of the SFRBFNC with two learning cycles. Noticeably, the SFRBFNC's performance cannot be improved significantly by adding more learning cycles because the two initial cycles are sufficient to achieve a reasonable performance.

Case 2: Joint-space trajectory tracking for point-to-point control in the workspace. The desired motion trajectory is a cubic interpolation polynomial from one point (300, −240, 300) mm to another point (500, −480, 570) mm in the workspace. The trajectories of the reference joints were calculated using the inverse kinematics equation based on this polynomial. The robotic manipulator was moved rapidly to the objective point (within 2.5 s). Subsequently, the robot was kept at that position for 2.5 s.

In Case 1, the SOFC and the SFRBFNC produced their respective reasonable fuzzy rule tables for each joint of the robot after the 5th run of the SOFC and the 2nd run of the SFRBFNC were performed. In this case, the learning processes of both the SOFC and the SFRBFNC for the control of each joint of the robot began from their respective reasonable fuzzy rule tables, which were obtained from Case 1, rather than from empty fuzzy rule tables. The initial values of the central position Z_i, the standard deviation σ_i, and the weighting $\overline{w}_{ij}(k)$ in the SFRBFNC were obtained from the final step of the 2nd run of the SFRBFNC in Case 1. The SOFC and SFRBFNC require respectively one learning cycle, in the learning process, for the point-to-point control of the robot.

Figures 8 and 9 illustrate the trajectory tracking errors with control efforts of the 1st and the 2nd joint, respectively, for the point-to-point control of the robot when the SOFC and the SFRBFNC were applied separately. From Figures 8(a) and 9(a), it can be seen that, in the case of the SOFC, the maximum errors and RMS errors of the joint-space

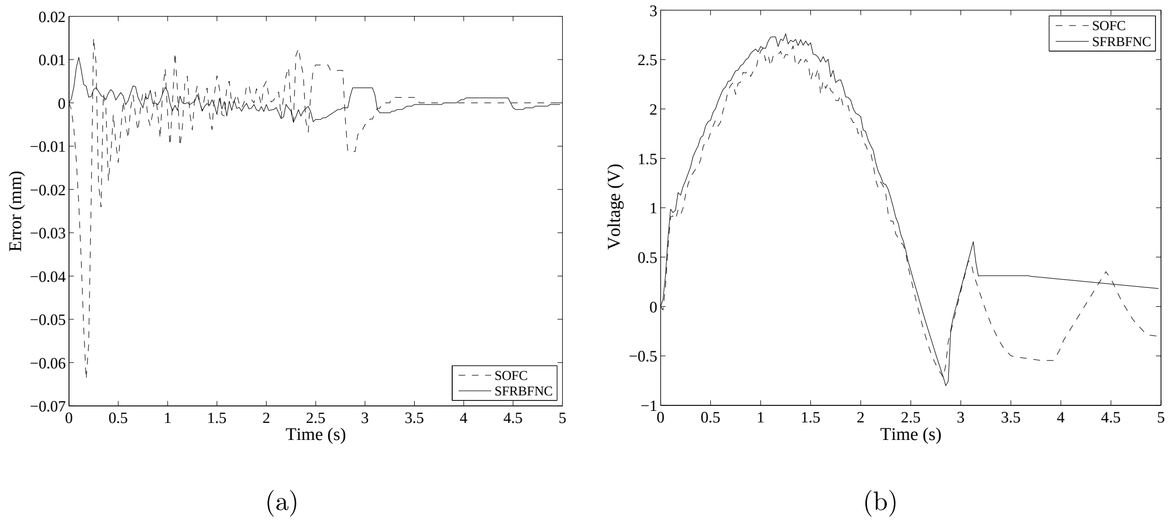


Figure 8. (a) Errors and (b) control voltages of the joint-space trajectory tracking at the 1st joint for point-to-point control using the SOFC and the SFRBFNC

trajectory tracking are 0.0171 mm and 0.0095 mm at the 1st joint, respectively, and 0.0614° and 0.0167° at the 2nd joint, respectively. However, when the SFRBFNC was applied, the maximum errors and RMS errors of the joint-space trajectory tracking were reduced to 0.0105 mm and 0.0021 mm at the 1st joint, respectively, and 0.0131° and 0.0049° at the 2nd joint, respectively. Furthermore, from Figures 8(b) and 9(b), it can be observed that the SFRBFNC has fewer control command variations than the SOFC. Clearly, the use of the SFRBFNC to manipulate the robot supplies longer actuator service life than that of the SOFC.

For the point-to-point control of the robot, the SFRBFNC is clearly superior to the SOFC, in terms of reducing the maximum errors and RMS errors of the joint-space trajectory tracking in the workspace and reducing control command variations.

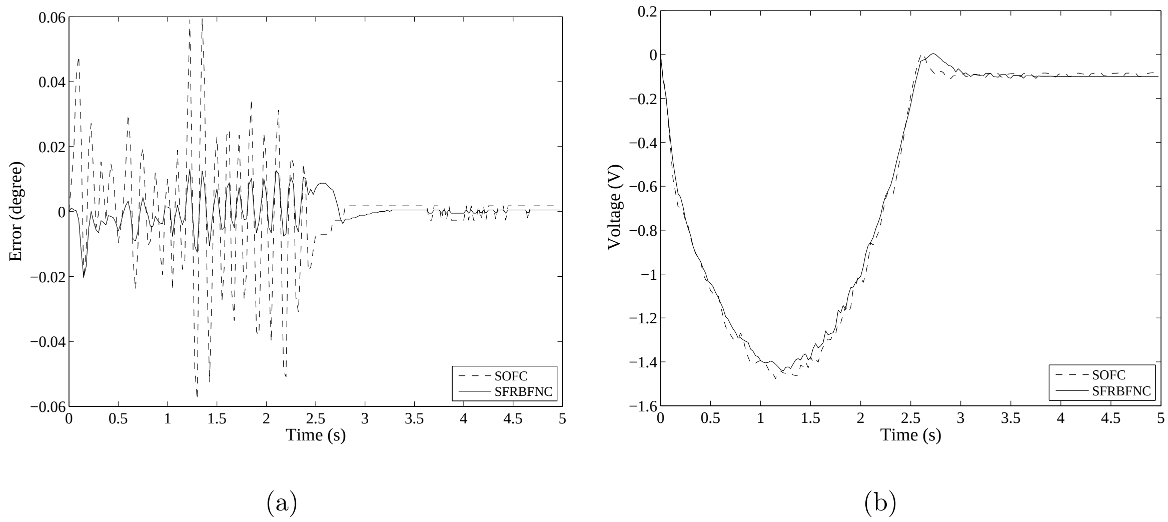
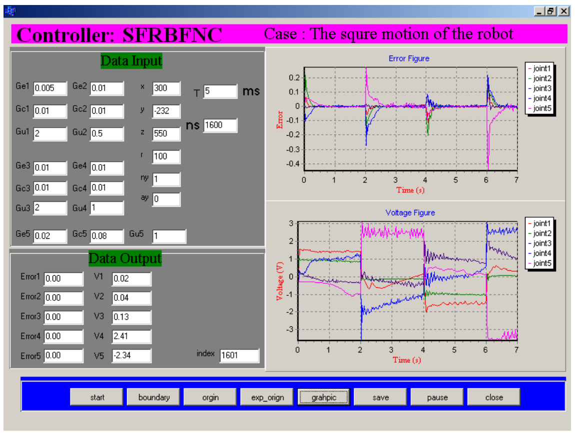


Figure 9. Same as those in Figure 8, except the 2nd joint instead of the 1st joint

Case 3: Joint-space trajectory tracking for square-path control in the workspace. In this case, the operating conditions of the system control are similar to Case 2, varying only in implementation of the square-path control of the robot instead of the point-to-point control of the robot. The square-path, which is a desired tracking trajectory in the workspace, has a side length of 10 cm.



Controller: SFRBFNC Case : The squre motion of the robot

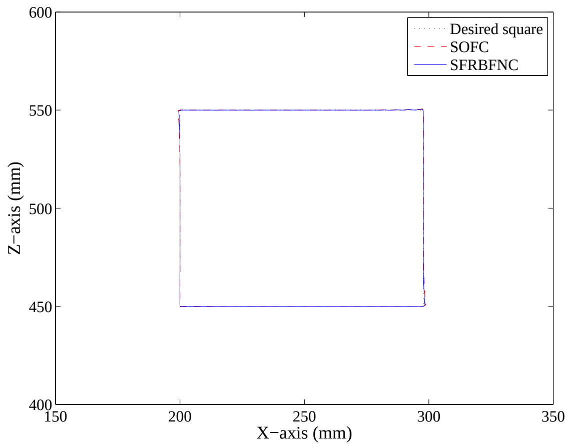
Data Input

Ge1	0.005	Ge2	0.01	x	300	T	5 ms
Gc1	0.01	Gc2	0.01	y	-232		
Gu1	2	Gu2	0.5	z	550	ns	1600
				r	100		
Ge3	0.01	Ge4	0.01	ny	1		
Gc3	0.01	Gc4	0.01	ay	0		
Gu3	2	Gu4	1				
Ge5	0.02	Gc5	0.08	Gu5	1		

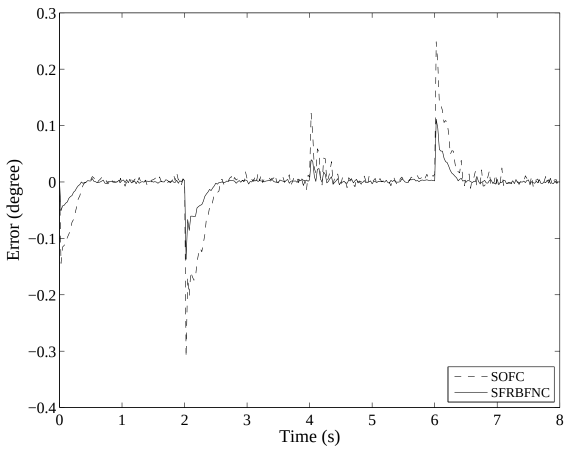
Data Output

Error1	0.00	V1	0.02		
Error2	0.00	V2	0.04		
Error3	0.00	V3	0.13		
Error4	0.00	V4	2.41		
Error5	0.00	V5	-2.34	index	1601

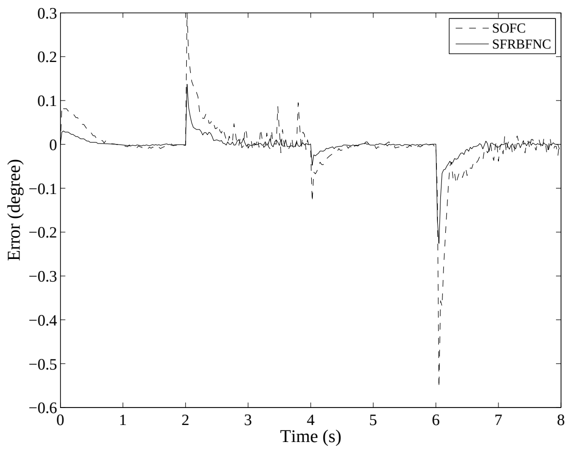
(a)



(b)



(c)

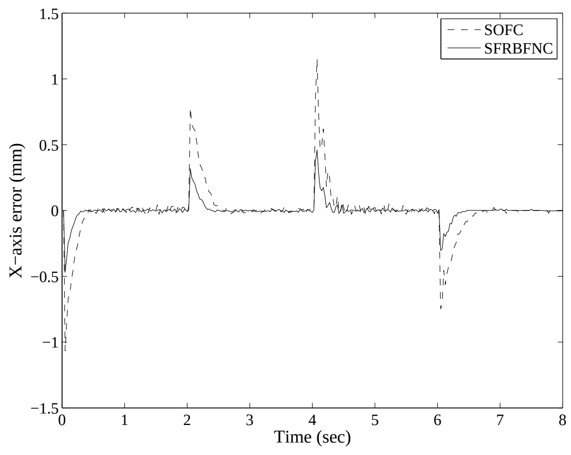


(d)

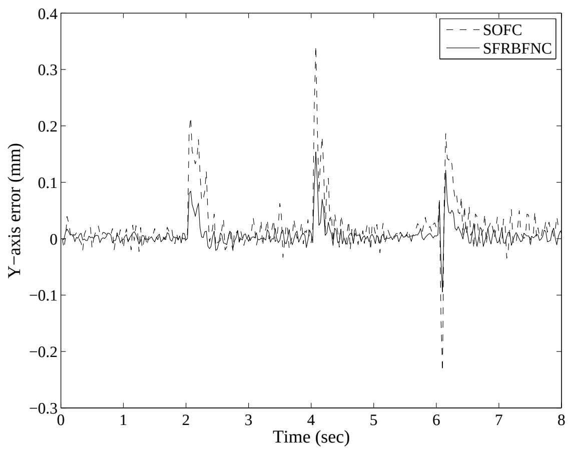
FIGURE 10. Robotic square-path control in the workspace. (a) The friendly GUI for the SFRBFNC. (b) Square trajectory output responses (side length = 10 cm). Trajectory tracking errors of (c) the 4th joint and (d) the 5th joint

Figure 10(a) displays a friendly GUI for the SFRBFNC, which is one of several interfaces used in controlling the robotic manipulator. Such an interface facilitates the setting of the parameters of the proposed controller. It is also capable of monitoring the trajectory tracking errors and the control effort of each joint of the robot in real time. Figure 10(b) shows a comparison between the square tracking trajectories with side lengths of 10 cm using the SOFC and the SFRBFNC. Noticeably, the output responses of the robot using both the SOFC and the SFRBFNC are too close to be distinguished. Figures 10(c) and

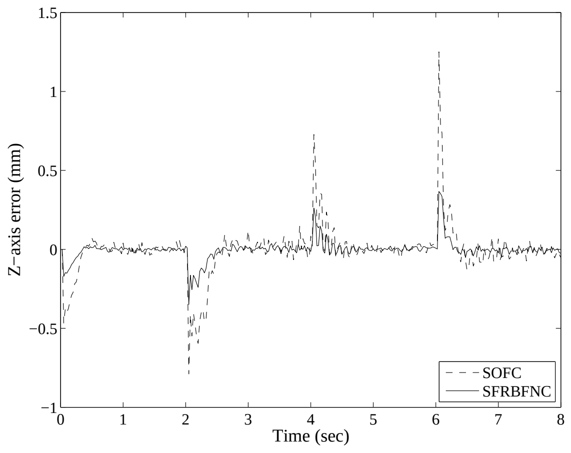
(d) presents the trajectory tracking errors of the 4th and the 5th joint, respectively, for the square-path control of the robot in the workspace using the SOFC and the SFRBFNC. Clearly, the maximum errors and RMS errors of the joint-space trajectory tracking using the SOFC are 0.2485° and 0.0480° at the 4th joint, respectively, and 0.2998° and 0.0629° at the 5th joint, respectively. However, when the SFRBFNC was applied, the aforementioned maximum errors and RMS errors were significantly reduced to 0.1370° and 0.0186° at the 4th joint, respectively, and 0.2258° and 0.0239° at the 5th joint, respectively.



(a)



(b)



(c)

FIGURE 11. Positioning trajectory tracking errors for the square-path control of the robot in the workspace: (a) X-axis, (b) Y-axis and (c) Z-axis

Figure 11 illustrates the positioning trajectory tracking errors, in the X, Y and Z directions of the Cartesian space, for the square-path control of the robot using the SOFC and the SFRBFNC. Table 3 summarizes the maximum errors and the RMS errors of the positioning trajectory tracking shown in Figure 11. It is noted that, for the square-path control of the robot in the workspace, the SFRBFNC has better control performance than the SOFC, in reducing the joint-space trajectory tracking errors and the positioning trajectory tracking errors.

Table 3. Maximum errors and RMS errors of the positioning trajectory tracking for the square-path control of the robot in the workspace

	SOFC		SFRBFNC	
Axis	Max. error (mm)	RMS error (mm)	Max. error (mm)	RMS error (mm)
X	1.1508	0.2012	0.4690	0.0727
Y	0.3418	0.0525	0.1538	0.0212
Z	1.2507	0.1680	0.3665	0.0629

It can be clearly seen from the aforementioned three cases that the SFRBFNC has better control performance than the SOFC for robotic motion control. Noticeably, it is difficult to gain a perfect control performance for manipulating robotic systems when applying the SOFC. Therefore, to achieve the perfect performances at any operating condition, the use of the SFRBFNC to control the robotic systems is best choice.

In addition, the RBFN, in the proposed SFRBFNC, employs the LM algorithm rather than the steepest descent method, which was used in the hybrid fuzzy-logic and neural-network controller (HFNC), designed by Huang and Lian [16], to minimize the objective function for determining the correction value of the weighting of the RBFN. The HFNC consists of an FLC, which was designed to control each DOF of a robot individually, and an additional coupling BP neural network, which was incorporated into an FLC to compensate for the dynamic coupling effects between the DOFs of the robot. Although the HFNC also used a BP neural network [16] to compensate for the dynamic coupling effects between the DOFs of the robot, it needed four learning cycles after training one learning cycle offline. The required data for training the BP neural network offline was obtained by the FLC when the FLC had been used to control the robot. As mentioned previously, the design of the FLC has some limitations. Moreover, the convergence rate of the BP neural network with the steepest descent method is too slow to compensate for the dynamic coupling effects between the DOFs of the robot in practical control applications. However, the LM algorithm has a faster convergence rate than the steepest descent method, so it can improve the problem of slow convergence rates of neural networks. Although Lin and Lian [17] suggested a modified HFNC, which used the LM algorithm instead of the steepest descent method to minimize the cost function of the BP neural network, to improve the performance of the HFNC proposed by Huang and Lian [16], the problem of the FLC application still exists. Clearly, the learning capability of the SFRBFNC outperforms those of the HFNC designed by Huang and Lian [16] and the modified HFNC designed by Lin and Lian [17] for robotic motion control. Neural networks have been adopted in several robotic control approaches [11,12] to compensate for the effects of unknown nonlinearities, but most of their control tests were implemented in simulations instead of actual experimentations. It is evident that the proposed SFRBFNC for robotic motion control is efficient in practical applications.

5. Conclusion.

This study has successfully retrofitted a 6-DOF robotic manipulator. Since the SOFC may have inappropriately chosen parameters, being able to produce an unstable system when it is applied, and the robotic control system is subject to dynamic coupling effects between its DOFs, this study developed an SFRBFNC to eliminate or alleviate the problems faced by the practical application of the SOFC in robotic motion control. Under the Windows XP operating system, the proposed controllers (SOFC and SFRBFNC) with friendly GUIs were coded using the Borland C^{++} Builder programming language to conveniently manipulate the 6-DOF robot. Experimental results verified that the SFRBFNC has better control performance than the SOFC for robotic motion control, in terms of (1) reducing the number of learning cycles, (2) reducing the maximum tracking errors and RMS errors for the joint-space trajectory planning in the workspace, (3) decreasing the maximum errors and RMS errors of the joint-space trajectory tracking for point-to-point control and the positioning trajectory tracking for square-path motion control in the workspace, and (4) decreasing variations of the control command, thereby enhancing the actuator service life of the robotic system.

Acknowledgment. The authors would like to thank the National Science Council of Taiwan for financially supporting this research under Contract No. NSC 100-2221-E-238-011.

REFERENCES

[1] M. K. Chang and T. H. Yuan, Experimental implementations of adaptive self-organizing fuzzy sliding mode control to 3-DOF rehabilitation robot, *International Journal of Innovative Computing, Information and Control*, vol.5, no.10(B), pp.3391-3404, 2009.
[2] M. M. Fateh and A. Azarfar, Improving fuzzy control of robot manipulators by increasing scaling factors, *ICIC Express Letters*, vol.3, no.3(A), pp.513-518, 2009.
[3] G. R. Yu and L. W. Huang, Design of LMI-based fuzzy controller for robot arm using quantum evolutionary algorithms, *ICIC Express Letters*, vol.4, no.3(A), pp.719-724, 2010.
[4] T. J. Procyk and E. H. Mamdani, A linguistic self-organizing process controller, *Automatica*, vol.15, pp.15-30, 1979.
[5] S. Shao, Fuzzy self-organizing controller and its application for dynamic process, *Fuzzy Sets and Systems*, vol.26, pp.151-164, 1988.
[6] B. S. Zhang and J. M. Edmunds, Self-organizing fuzzy logic controller, *Proc. of IEE-D Control Theory and Applications*, vol.139, no.5, pp.460-464, 1992.
[7] C. Z. Yang, *Design of Real-Time Linguistic Self-organizing Fuzzy Controller*, Master Thesis, Department of Mechanical Engineering, National Taiwan University, Taiwan, 1992.
[8] S. J. Huang and J. S. Lee, A stable self-organizing fuzzy controller for robotic motion control, *IEEE Transaction on Industrial Electronics*, vol.47, no.2, pp.421-428, 2000.
[9] J. Lin and R. J. Lian, Self-organizing fuzzy controller for injection molding machines, *Journal of Process Control*, vol.20, no.5, pp.585-595, 2010.
[10] J. Lin and R. J. Lian, Self-organizing fuzzy controller for gas-injection molding combination systems, *IEEE Transactions on Control Systems Technology*, vol.18, no.6, pp.1413-1421, 2010.
[11] V. Kroumov, J. Yu and K. Shibayama, 3D path planning for mobile robots using simulated annealing neural network, *International Journal of Innovative Computing, Information and Control*, vol.6, no.7, pp.2885-2899, 2010.
[12] Y. Zuo, Y. Wang, X. Liu, S. X. Yang, L. Huang, X. Wu and Z. Wang, Neural network robust control for a nonholonomic mobile robot including actuator dynamics, *International Journal of Innovative Computing, Information and Control*, vol.6, no.8, pp.3437-3449, 2010.
[13] R. J. Wai and C. M. Liu, Design of dynamic petri recurrent fuzzy neural network and its application to path-tracking control of nonholonomic mobile robot, *IEEE Transactions on Industrial Electronics*, vol.56, no.7, pp.2667-2683, 2009.
[14] A. Gajate, R. E. Haber, P. I. Vega and J. R. Alique, A transductive neuro-fuzzy controller: Application to a drilling process, *IEEE Transactions on Neural Networks*, vol.21, no.7, pp.1158-1167, 2010.
[15] L. Cai, A. B. Rad and W. L. Chan, An intelligent longitudinal controller for application in semiautonomous vehicles, *IEEE Transactions on Industrial Electronics*, vol.57, no.4, pp.1487-1497, 2010.
[16] S. J. Huang and R. J. Lian, A hybrid fuzzy logic and neural network algorithm for robot motion control, *IEEE Transactions on Industrial Electronics*, vol.44, no.3, pp.408-417, 1997.
[17] J. Lin and R. J. Lian, Hybrid fuzzy-logic and neural-network controller for MIMO systems, *Mechatronics*, vol.19, no.6, pp.972-986, 2009.
[18] S. J. Huang, K. S. Huang and K. C. Chiou, Development and application of a novel radial basis function sliding mode control, *Mechatronics*, vol.13, no.4, pp.313-329, 2003.
[19] S. Ferrari, F. Bellocchio, V. Piuri and N. A. Borghese, A hierarchical RBF online learning algorithm for real-time 3-D scanner, *IEEE Transactions on Neural Networks*, vol.21, no.2, pp.275-285, 2010.

[20] X. Yuan, Y. Wang, W. Sun and L. Wu, RBF networks-based adaptive inverse model control system for electronic throttle, *IEEE Transactions on Control Systems Technology*, vol.18, no.3, pp.750-756, 2010.

[21] F. L. Lewis, C. T. Abdallah and D. M. Dawson, *Control of Robot Manipulators*, Macmillan, New York, 1993.

[22] K. S. Fu, R. C. Gonzalez and C. S. G. Lee, *Robotics Control, Sensing, Vision, and Intelligence*, McGraw-Hill International Editions, 1987.

[23] V. Singh, I. Gupta and H. O. Gupta, ANN-based estimator for distillation using Levenberg-Marquardt approach, *Engineering Applications of Artificial Intelligence*, vol.20, no.2, pp.249-259, 2007.

[24] C. Ma and C. Wang, A nonsmooth Levenberg-Marquardt method for solving semi-infinite programming problems, *Journal of Computational and Applied Mathematics*, vol.230, no.2, pp.633-642, 2009.

[25] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink and D. L. Alkon, Accelerating the convergence of the back-propagation method, *Biological Cybernetics*, vol.59, no.2, pp.257-263, 1988.

[26] D. Driankov, H. Hellendoorn and M. Reinfrank, *An Introduction to Fuzzy Control*, Springer-Verlag, Berlin, 1993.

[27] S. J. Huang and R. J. Lian, A combination of fuzzy logic and neural network algorithms for active vibration control, *Proc. of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol.210, no.3, pp.153-167, 1996.