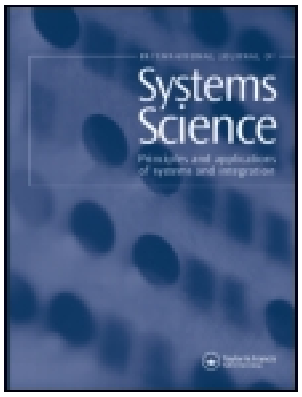

International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tsys20

On-line genetic algorithm-based fuzzy-neural sliding mode controller using improved adaptive bound reduced-form genetic algorithm

Ping-Zong Lin [a] , Wei-Yen Wang [b] , Tsu-Tian Lee [c] & Chi-Hsu Wang [a]
[a] Department of Electrical and Control Engineering , National Chiao Tung University , Hsinchu, Taiwan
[b] Department of Applied Electronics Technology , National Taiwan Normal University , Taipei, 106 Taiwan
[c] Department of Electrical Engineering , National Taipei University of Technology , Taipei, 106 Taiwan
Published online: 23 Jun 2009.

International Journal of Systems Science
Vol. 40, No. 6, June 2009, 571–585

On-line genetic algorithm-based fuzzy-neural sliding mode controller using improved adaptive bound reduced-form genetic algorithm

Ping-Zong Lin[a], Wei-Yen Wang[b]*, Tsu-Tian Lee[c] and Chi-Hsu Wang[a]

[a]*Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan;* [b]*Department of Applied Electronics Technology, National Taiwan Normal University, Taipei, 106 Taiwan;* [c]*Department of Electrical Engineering, National Taipei University of Technology, Taipei, 106 Taiwan*

(*Received 10 October 2007; final version received 19 December 2008*)

In this article, a novel on-line genetic algorithm-based fuzzy-neural sliding mode controller trained by an improved adaptive bound reduced-form genetic algorithm is developed to guarantee robust stability and good tracking performance for a robot manipulator with uncertainties and external disturbances. A general sliding manifold, which can be non-linear or time varying, is used to construct a sliding surface and reduce control law chattering. In this article, the sliding surface is used to derive a genetic algorithm-based fuzzy-neural sliding mode controller. To identify structured system dynamics, a B-spline membership function fuzzy-neural network, which is trained by the improved genetic algorithm, is used to approximate the regressor of the robot manipulator. The sliding mode control with a general sliding surface plays the role of a compensator when the fuzzy-neural network does not approximate the dynamics regressor of the robot manipulator well in the transient period. The adjustable parameters of the fuzzy-neural network are tuned by the improved genetic algorithm, which, with the use of the sequential-search-based crossover point method and the single gene crossover, converges quickly to near-optimal parameter values. Simulation results show that the proposed genetic algorithm-based fuzzy-neural sliding mode controller is effective and yields superior tracking performance for robot manipulators.

Keywords: fuzzy-neural sliding mode controller; adaptive bound reduced-form genetic algorithm; robot manipulator; on-line genetic algorithm-based controller

1. Introduction

Recently, fuzzy-neural network theory has been a topic of interest to those involved in dealing with the imprecision, uncertainty and non-linearity in control systems. Fuzzy-neural networks have been trained using various learning algorithms (Meng 1993; Wang 1994; Marra and Walcott 1996; Leu and Lee 2000; Wang, Chien, Leu, and Lee 2008; Leu, Wang, and Lee 2005). In many studies (Meng 1993; Wang 1994; Marra and Walcott 1996; Leu and Lee 2000; Wang et al. 2008; Leu et al. 2005), fuzzy logic systems and/or neural networks were successfully applied to adaptive control systems. In fuzzy set theory, the construction and range of membership functions play a critical role. We apply B-spline membership functions (Wang, Wang, Lee, and Tseng 1995; Wang and Li 2003) to construct the fuzzy membership functions. B-spline membership functions possess the property of local control and have been successfully applied to fuzzy-neural control (Leu, Wang, and Lee 1999).

Genetic algorithms have been intensively studied during the past three decades. Control researchers have become increasingly interested in the use of genetic algorithms as a means to control various classes of systems. Genetic algorithms (Goldberg 1989; Marra and Walcott 1996; Michalewicz 1996; Yang, Hachino and Tsuji 1996; Gen and Cheng 1997; Ferreira, Lopes, and Saraiva 2000) are a stochastic search technique that guides a population of solutions towards a global optimum using the principles of evolution and natural genetics. Thanks to this probabilistic search procedure, genetic algorithms are highly effective and robust over a broad spectrum of problems (Lawrence 1991; Kristinsson and Dumont 1992; Gonzalez and Perez 2001). This motivates the use of genetic algorithms to overcome the problems encountered by conventional learning methods for fuzzy-neural networks (Caponetto, Fortuna, Graziani, and Xibilia 1993; Farag, Quintana and Germano 1998; Lin and Jou 2000; Wang, Liu, and Lin 2001; Wang and Li 2003). In Wang and Li (2003), a genetic algorithm is used to automatically tune the adjustable parameters (control points and weights) of a B-spline membership function fuzzy-neural network to approximate non-linear functions.

*Corresponding author. Email: wywang@ntnu.edu.tw

ISSN 0020–7721 print/ISSN 1464–5319 online
© 2009 Taylor & Francis
DOI: 10.1080/00207720902750011
http://www.informaworld.com

However, it is not easy to determine the bounds of the parameters of genetic algorithms, which may affect the speed of the convergence of the learning algorithms when we apply the genetic algorithm to train parameters. To address this problem, Wang, Tao, and Chang (2004) use an adaptive bound algorithm to aid and speed up search in the reduced-form genetic algorithm (Wang and Li 2003). The two simulation examples in Wang et al. (2004) show that the searching speed of the adaptive bound algorithm is superior to that of the reduced-form genetic algorithm with fixed bounds. Nevertheless, the enlarging condition and operating rules of the adaptive bound algorithm in Wang et al. (2004) are not complete. Hence, in this article, we propose the improved adaptive bound algorithm to enhance the ability of the original adaptive bound algorithm. With the use of the sequential-search-based crossover point method and the improved adaptive bound algorithm, an improved adaptive bound reduced-form genetic algorithm, called Modified Adaptive bound Reduced-form Genetic algorithm (MARG), is proposed. The proper bounds of the adjustable parameters can be obtained automatically in the training process by applying MARG. It has a fast convergence speed in searching for near-optimal solutions.

In previous studies (Meng 1993; Leu and Lee 2000), fuzzy and/or neural network adaptive controllers were proposed for robot manipulators with unknown dynamics to identify structured system dynamics. These approaches take advantage of the regressor dynamics of robot manipulators to design adaptive controllers. However, adaptive fuzzy-neural control is traditionally trained by using gradient-based methods, which may fall into a local minimum during the learning process. In addition, sliding mode control or so-called variable structure system control (Young 1978; Yeung and Chen 1988; Pei and Zhou 1991; Su and Stepanenko 1993; Hsu 1998; Leu et al. 2005) has been frequently used for robust control systems due to its fast response, and insensitivity to plant parameter variation and/or external disturbances. The variable structure system control of robot manipulators has been the subject of considerable interest (Young 1978; Yeung and Chen 1988; Pei and Zhou 1991; Su and Stepanenko 1993). In particular, Su and Stepanenko (1993) uses a general sliding surface, which can be non-linear or time varying, for a robot manipulator. However, the studies (Young 1978; Yeung and Chen 1988; Pei and Zhou 1991; Su and Stepanenko 1993) do not consider robot manipulators with unknown dynamics. Using the proposed MARG, in this article, we propose a novel on-line genetic algorithm-based fuzzy-neural sliding mode controller for a robot manipulator with uncertainties and external disturbances. Our controller not only can deal with robot manipulators with unknown dynamics, but can also overcome the problem of traditional adaptive fuzzy-neural control. Moreover, unlike many genetic algorithm-based controllers, ours is on-line. Additionally, our sliding mode controller with a general sliding surface is like a supervisory controller. It plays the role of a compensator when the B-spline membership function fuzzy-neural network does not approximate the regressor dynamics of the robot manipulator well in the transient period. Consequently, the proposed controller tuned by MARG can guarantee robust stability and good tracking performance of the robot manipulator despite uncertainties and external disturbances. The proper bounds of the adjustable parameters can be obtained automatically in the training process by applying MARG. Pre-determining the initial (non-zero) bounds of the adjustable parameters is unnecessary.

The rest of this article is organised as follows. Section 2 provides a brief overview of the structure of the robot dynamic system. Section 3 provides a brief description of fuzzy-neural networks. Section 4 describes the improved adaptive bound reduced-form genetic algorithm, which tunes the control points and weights of the B-spline membership function fuzzy-neural networks. Details of the sequential-search-based crossover point method, single-gene crossover and the reduced-form genetic algorithm are also given in this section. Section 5 gives details of the on-line genetic algorithm-based fuzzy-neural sliding mode controller design. The simulation results are shown in Section 6. Finally, conclusions are drawn in Section 7.

2. System description

The model of robot dynamics can be found in the literature (Yeung and Chen 1988), which is written as

$$H\ddot{\Theta} + C\dot{\Theta} + G = u + \Delta_d \qquad (1)$$

where $H(\Theta) \in R^{m\times m}$ is a symmetric positive matrix of the manipulator inertial, $C(\Theta, \dot{\Theta})\dot{\Theta} \in R^{m\times 1}$ is a vector of centripetal and Coriolis torques, $G(\Theta) \in R^{m\times 1}$ is a vector of gravitational torques, $u \in R^{m\times 1}$ is a vector of applied joint torques, $\Delta_d = [\Delta_{d1}, \Delta_{d2}, \dots, \Delta_{dm}] \in R^{m\times 1}$ is an unknown vector of uncertainties, torque disturbances, etc. and $\Theta = [q_1, q_2, \dots, q_m]^T \in R^{m\times 1}$, $\dot{\Theta}$ and $\ddot{\Theta}$ are vectors of joint positions, velocities and accelerations, respectively.

The dynamic structure (1) has two important properties. One property is that the matrix $(\dot{H} - 2C)$

is skew-symmetric theoretically. The other property is that the robot dynamic structure is linear in terms of a suitably selected set of its parameters.

Since the matrices H, C and G are linear in terms of robot manipulator parameters and $\Theta_d = [q_{1d}, q_{2d}, \ldots, q_{md}]^T \in R^{m\times 1}$, $\dot{\Theta}_d$ and $\ddot{\Theta}_d$ are vectors of desired joint positions, velocities and accelerations, respectively, there exists a vector Γ with components depending on manipulator parameters, such that

$$H\dot{\Lambda} + C\Lambda + G = \Sigma\Gamma \tag{2}$$

where $\Sigma(\Theta, \dot{\Theta}, \Lambda, \dot{\Lambda}) \in R^{m\times h}$ is called the regressor (Kelly, Carelli, and Ortega 1989; Lu and Meng 1993), which is independent of the unknown dynamic parameters, $\Gamma \in R^{h\times 1}$ is a vector of unknown manipulator and load parameters, and $\Lambda(\Theta, \Theta_d, \dot{\Theta}_d, t) \in R^{m\times 1}$ is a vector of smooth functions to be chosen by the designer.

Since Γ is a vector of unknown manipulator and load parameters, we define the outputs of the inverse robot system dynamics as

$$Y^* = H\dot{\Lambda} + C\Lambda + G \tag{3}$$

where $Y^* = [y_1^*, y_2^*, \ldots, y_m^*]$ is the torque vector computed from the set of trial data of the robot dynamics in (2).

Now, we can use a matrix W and a vector φ to calculate the value of $W\varphi$ to approximate the value of Y^*. The estimated model of robot dynamics can be rewritten as

$$\hat{H}\dot{\Lambda} + \hat{C}\Lambda + \hat{G} = W\varphi \tag{4}$$

where $W = [w_1 \; w_2 \; \ldots \; w_m]^T \in R^{m\times h}$ is a matrix of adjustable parameters, $w_i \in R^{h\times 1}$, $\varphi \in R^{h\times 1}$ is a vector of specified functions, and $\hat{H}$, $\hat{C}$ and $\hat{G}$ are auxiliary matrices of the robot manipulator parameters. For the purpose of using W and φ to approximate Y^*, we use a B-spline membership function fuzzy-neural network as an approximator, and an improved adaptive bound reduced-form genetic algorithm to adjust the control points and weightings of the B-spline membership function fuzzy-neural network. Then we can obtain the outputs, $W\varphi$, of the B-spline membership function fuzzy neural network, such that Y^* is approximated by $W\varphi$ as close as possible.

Since the robot dynamics in (1) includes unknown uncertainties and torque disturbances Δ_d, designing the controller u becomes difficult. To solve this problem, a universal approximator (B-spline membership function fuzzy-neural network) is used to approximate the robot dynamics. Equation (4) is the estimated model. Therefore, instead of (1), using (4) to design the controller becomes easier. In addition, to overcome the problem of traditional adaptive fuzzy-neural controllers, a novel on-line genetic algorithm-based fuzzy-neural sliding mode controller is proposed. Unlike many genetic algorithm-based controllers, ours is on-line. Also, our sliding mode controller with a general sliding surface is like a supervisory controller. It plays the role of a compensator when the B-spline membership function fuzzy-neural network does not approximate the regressor dynamics of the robot manipulator well in the transient period. Consequently, the proposed controller tuned by MARG can guarantee robust stability and good tracking performance of the robot manipulator despite uncertainties and external disturbances. The proper bounds of the adjustable parameters can be obtained automatically in the training process by applying MARG. Pre-determining the initial (non-zero) bounds of the adjustable parameters is unnecessary.

In order to derive the on-line genetic algorithm-based fuzzy-neural sliding mode controller, the following assumptions are required.

Assumption 1: *The effects of uncertainties and torque disturbances are assumed to satisfy* $|\Delta_{di}| \le \beta_i$ *and* $\beta_i > 0$, $i = 1, \ldots, m$.

Assumption 2: *Since* H, G *and* C *are bounded in* Θ *and* C *is linear in* $\dot{\Theta}$, *we assume that there exist positive constants* α_1, α_2 *and* α_3, *such that* $\|\tilde{H}\| \le \alpha_1$, $\|\tilde{C}\| \le \alpha_2\|\dot{\Theta}\|$ *and* $\|\tilde{G}\| \le \alpha_3$, *where* $\tilde{H} = H - L_2\hat{H}$, $\tilde{C} = C - L_2\hat{C}$, $\tilde{G} = G - L_2\hat{G}$ *and* $L_2 = \mathrm{diag}[\ell_{21}, \ell_{22}, \ldots, \ell_{2m}]$, $0 \le \ell_{2i} \le 1$.

3. B-spline membership function fuzzy-neural networks

Fuzzy-neural networks are typically fuzzy inference systems constructed from a neural network structure. Learning algorithms are used to adjust the weights of the fuzzy inference system. Figure 1 shows the configuration of the B-spline membership function fuzzy-neural network, which has a total of four layers. Nodes at layer I are input nodes (linguistic nodes) that represent input linguistic variables. Nodes at layer II are term nodes which act as membership functions to represent the terms of the respective linguistic variables. Each node at layer III is a fuzzy rule. Layer IV is the output layer.

3.1. *B-spline membership functions*

For δ order and r control points, the B-spline basis functions have the knot vector $\{\tau_i, i = 1, 2, \ldots, r+\delta\}$ with $\tau_1 < \tau_2 < \cdots < \tau_{r+\delta}$. We choose that the order

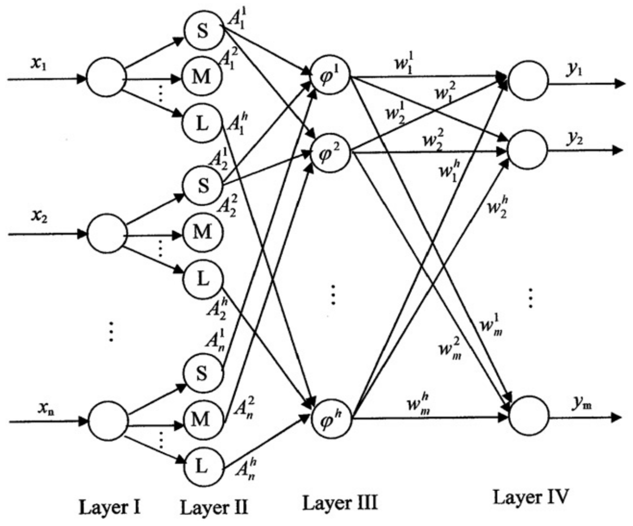

Figure 1. Configuration of the fuzzy-neural network.

is three or above, and that the type of the knot vector is set to open uniform. An element τ_i of the knot vector is defined as

$$\tau_i = \begin{cases} x_1 & \text{if } i \le \delta \\ \tau_{i-1} + \dfrac{x_n - x_1}{r - \delta + 2} & \text{if } \delta < i \le r \\ x_n & \text{if } i > r \end{cases} \tag{5}$$

where $x = \{x_q, q = 1, 2, \ldots, n; n > r\}$ is the data vector of input. For r control points, $\{c_1, c_2, \ldots, c_r\}$, the i-th B-spline blending function of δ order is denoted by $N_{i,\delta}(x)$. Hence, the B-spline curve $B(x)$ is defined as follows:

$$B(x_q) = \sum_{i=1}^{r} c_i N_{i,\delta}(x_q), \quad 1 \le \delta \le r,$$

and

$$N_{i,1}(x_q) = \begin{cases} 1, & \text{if } \tau_i \le x_q < \tau_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{aligned} N_{i,\delta}(x_q) = {} & \left(\frac{x_q - \tau_i}{\tau_{i+\delta-1} - \tau_i}\right) N_{i,\delta-1}(x_q) \\ & + \left(\frac{\tau_{i+\delta} - x_q}{\tau_{i+\delta} - \tau_{i+1}}\right) N_{i+1,\delta-1}(x_q). \end{aligned} \tag{6}$$

In this article, the B-spline membership function $\mu_A(x_q)$ introduced in the papers of Wang et al. (1995) and Wang, Lee and Liu (1997) is modified to satisfy the condition $0 \le \mu_A \le 1$, as follows:

$$\mu_A(x_q) = S\left[\sum_{i=1}^{r} c_i N_{i,\delta}(x_q)\right], \tag{7}$$

where A is a fuzzy set, and

$$S(\xi) = \begin{cases} 1, & \text{if } \xi > 1, \\ \xi, & \text{if } 0 \le \xi \le 1, \\ 0, & \text{if } \xi < 0. \end{cases}$$

We adopt B-spline membership functions as the fuzzy membership functions and use the reduced-form genetic algorithm with improved adaptive bounds (to be introduced in Section 4) to obtain a set of near-optimal control points for the B-spline membership functions. To avoid excessive numbers of control points, we use B-spline membership functions having a fixed number of control points.

3.2. *Fuzzy inference and formulisation*

Given the input data x_q, $q = 1, 2, \ldots, n$, and the output data y_p, $p = 1, 2, \ldots, m$, the i-th fuzzy rule has the following form:

$$\begin{aligned} R^i\!: & \text{ if } x_1 \text{ is } A_1^i \quad \text{and} \quad \cdots \quad x_n \quad \text{is } A_n^i \\ & \text{then} \quad y_1 = w_1^i \quad \text{and} \quad \cdots \quad y_m = w_m^i \end{aligned} \tag{8}$$

where i is a rule number, A_q^i are the fuzzy sets of the antecedent part and w_p^i are real numbers of the consequent part. When the inputs $x = [x_1\ x_2\ \cdots\ x_n]^T$ are given, the output y_p of the fuzzy inference can be derived from

$$y_p(x) = \frac{\sum_{i=1}^{h} w_p^i \left(\prod_{q=1}^{n} \mu_{A_q^i}(x_q)\right)}{\sum_{i=1}^{h} \left(\prod_{q=1}^{n} \mu_{A_q^i}(x_q)\right)} = \varphi^T w_P \tag{9}$$

where $\mu_{A_q^i}(x_q)$ is the B-spline membership function of A_q^i, h is the number of fuzzy rules and $w_p = [w_p^1\ w_p^2\ \cdots\ w_p^h]^T$ is a weighting vector related to the p-th output $y_p(x)$. $\varphi = [\varphi^1\ \varphi^2\ \cdots\ \varphi^h]^T$ is a set of fuzzy basis functions defined as:

$$\varphi^i(x) = \frac{\prod_{q=1}^{n} \mu_{A_q^i}(x_q)}{\sum_{i=1}^{h} \left(\prod_{q=1}^{n} \mu_{A_q^i}(x_q)\right)}, \quad i = 1, 2, \ldots, h. \tag{10}$$

Assume that each input has z fuzzy sets (B-spline membership functions). If there are h rules in the fuzzy rule base, then the adjustable set of all the control points is defined as

$$\begin{aligned} c &= \left[c_1^{1^T} c_1^{2^T} \cdots c_1^{z^T} c_2^{1^T} c_2^{2^T} \cdots c_2^{z^T} \cdots c_n^{z^T}\right]^T \\ &= \{c_{q,j_1}^f \,|\, f = 1, 2, \ldots, z, q = 1, 2, \ldots, n, \\ &\qquad j_1 = 1, 2, \ldots, r\}. \end{aligned} \tag{11}$$

By adjusting the weighting values w_p^i and control points $c_q^f = [c_{q,1}^f c_{q,2}^f \cdots c_{q,r}^f]^T$ of the B-spline membership function fuzzy-neural network, a learning algorithm can be derived to minimise the error function:

$$e_p(w_p, c) = \frac{1}{2}\left(y_p - y_p^*\right)^2 = \frac{1}{2}\left(\varphi^T w_p - y_p^*\right)^2, \quad p = 1, 2, \ldots, m. \tag{12}$$

For fuzzy-neural networks with multiple outputs, where

$$Y = [y_1 \ y_2 \ \cdots \ y_m] \tag{13}$$

and

$$Y^* = \left[y_1^* \ y_2^* \ \cdots \ y_m^*\right]$$

represent the m-dimensional vector of the actual outputs and the desired outputs from the B-spline membership function fuzzy neural network, respectively, the error function can be defined as

$$E(W, c) = \frac{1}{2}\| Y - Y^* \|^2 \tag{14}$$

where $W = [w_1 \, w_2 \ldots w_m]^T$ is defined in (4). Moreover, we define a new vector including all weights and control points of the B-spline membership function fuzzy-neural network adjusted by MARG as

$$\varpi = \left[w_1^T \ w_2^T \ \cdots \ w_m^T \ c_1^{1^T} \ c_1^{2^T} \ \cdots \ c_1^{z^T} c_2^{1^T} c_2^{2^T} \ \cdots \ c_2^{z^T} \ \cdots \ c_n^{z^T}\right] \in R^{1 \times \bar{\kappa}} \tag{15}$$

with a length of $\bar{\kappa} = m \times h + (n \times r) \times z$ for a fuzzy-neural network of m outputs.

4. Improved adaptive bound reduced-form genetic algorithm (MARG)

Basically, genetic algorithms are probabilistic algorithms which maintain a population of individuals (chromosomes), $\Psi(t) = \{\phi^1(t), \phi^2(t), \ldots, \phi^k(t)\}$, for iteration t. Each chromosome φ^ℓ represents a potential solution to the problem at hand, and is evaluated to give some measure of its "fitness". Then, a new population is formed by selecting the more fit individuals. Some members of the new population undergo transformations by means of genetic operators to form new solutions. After some number of generations, it is hoped that the system converges to a near-optimal solution.

Recently, a reduced-form genetic algorithm (Wang and Li 2003) was proposed to evolutionarily obtain the near-optimal weighting vector for a fuzzy-neural network. It is characterised by three simplified processes. First, the population size is fixed and can be reduced to a minimum of 4. Second, the crossover operator is simplified to be a single-gene crossover. Finally, only one chromosome in a population is selected for mutation. Furthermore, the search bounds of the genes (which correspond to the adjustable parameters in the B-spline membership function fuzzy-neural network) are determined by an adaptive bound method (Wang et al. 2004). We use the RGA and the adaptive bound algorithm in this article, but modify the enlarging condition and operating rules. Thus, the proper bounds of the adjustable parameters can be obtained automatically in the training process. Pre-determining the initial (non-zero) bounds of the adjustable parameters is unnecessary. Details are discussed as follows.

4.1. *Population initialisation*

Assume that the adjustable vector ϖ in (15) of the fuzzy-neural network is embedded in a chromosome that represents a potential solution to the problem and the ℓ-th chromosome is defined as:

$$\begin{aligned}\phi^\ell &= \left[\varpi \, \phi_{\bar{\kappa}+1}^\ell\right] = \left[w_1^T \ w_2^T \ \cdots \ w_m^T \ c^T \ \phi_{\bar{\kappa}+1}^\ell\right] \\ &= \left[\phi_1^\ell \ \phi_2^\ell \ \cdots \ \phi_{\bar{\kappa}}^\ell \ \phi_{\bar{\kappa}+1}^\ell\right] \in R^{\bar{\kappa}+1}\end{aligned} \tag{16}$$

where ϖ consists of all the weights and control points, as defined in (15). The weights, w_i, range over the interval $D_1 = [w_{\min}, w_{\max}] \subseteq R$, and the control point set, c, ranges over the interval $D_2 = [c_{\min}, c_{\max}] \subseteq R$. Our hope is that $E(W, c)$ in (14) is reduced to a minimum by searching for the optimal solution.

In this paper, because the search bounds, D_1 and D_2, are unknown beforehand, the initial bounds are all set to zero at initialisation. The initial chromosomes are also set to zero due to the zero bounds. A population with k chromosomes as defined in (16) is represented as:

$$\Psi = \begin{bmatrix} \phi^1 \\ \phi^2 \\ \vdots \\ \phi^k \end{bmatrix} = \begin{bmatrix} \phi_1^1 & \phi_2^1 & \cdots & \phi_{\bar{\kappa}}^1 & \phi_{\bar{\kappa}+1}^1 \\ \phi_1^2 & \phi_2^2 & \cdots & \phi_{\bar{\kappa}}^2 & \phi_{\bar{\kappa}+1}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi_1^k & \phi_2^k & \cdots & \phi_{\bar{\kappa}}^k & \phi_{\bar{\kappa}+1}^k \end{bmatrix}. \tag{17}$$

Each chromosome comprising an adjustable vector ϖ for the B-spline membership function fuzzy-neural network has $\bar{\kappa} + 1$ elements and is a candidate solution for the problem. As to be shown later in this section, $\phi_{\bar{\kappa}+1}^\ell$ serves as a dummy gene for the single-gene crossover operator (Wang and Li 2003) and has no effect on fitness evaluation in the population. It is expected that one of the candidate solutions, φ^ℓ, can be evolutionarily obtained to form a set of near-optimal parameters for the fuzzy-neural network. Note that the size of the population, k, needs to be an even number and larger than 3 as required by the single-gene crossover method (Wang and Li 2003).

After initialisation, three genetic operations, *crossover*, *sorting* and *mutation* are performed during procreation.

4.2. *Fitness function*

The performance of each chromosome is evaluated according to its fitness. After generations of evolution, it is expected that the genetic algorithm converges and a best chromosome with largest fitness representing the optimal solution to the problem is obtained. The fitness function is chosen as follows:

$$\text{fitness} = \frac{1}{E(W, c)} \tag{18}$$

where $E(W, c)$ is the estimation error function defined in (14).

4.3. *Single-gene crossover*

In order to deal with the adjustable parameters, we use the single-gene crossover (Wang and Li 2003). Figure 2 shows the difference between the traditional crossover methods and the applied single-gene crossover method. Figure 2(a) and (b) demonstrates the traditional methods, which adopt one crossover point and two crossover points, respectively. Although the applied single-gene crossover shown in Figure 2(c) has two crossover points, the distance between the two chosen crossover points is only one gene (parameter). For each generation, the crossover operator will act on parents to give offspring. The single-gene crossover operator is defined as:

$$\hat{\Psi} = \mathrm{Crs}(\Psi; j) = \begin{bmatrix} & \hat{\phi}^1_{j+1} & \\ \Delta & \hat{\phi}^2_{j+1} & \Delta \\ & \vdots & \\ & \hat{\phi}^k_{j+1} & \end{bmatrix} = \begin{bmatrix} \hat{\phi}^1 \\ \hat{\phi}^2 \\ \vdots \\ \hat{\phi}^k \end{bmatrix} \tag{19}$$

where j is the crossover point determined by a sequential-search-based crossover point method (Wang and Li 2003), Δ denotes the elements of offspring which remain the same as those of their parents, and

$$\hat{\phi}^i_{j+1} = \begin{cases} \phi^i_{j+1} * (1-a) + \phi^{i+(k/2)}_{j+1} * a, \\ \quad \text{if } i = 1, 2, \ldots, k/2, \\ \phi^i_{j+1} * (1-a) + \phi^{i-(k/2)}_{j+1} * a, \\ \quad \text{if } i = (k/2)+1, (k/2)+2, \ldots k \end{cases} \tag{20}$$

The single-gene crossover operator Crs(●; ●) generates new genes, $\hat{\phi}^i_{j+1}$, only at the position $j+1$ for all chromosomes with a linear combination of ϕ^i_{j+1} and $\phi^{(k/2)+i}_{j+1}$. a is a constant between 0 and 1. $\hat{\Psi}$ is a new population.

To determine a good crossover point j for the single-gene crossover, we use the sequential-search-based crossover point method (Wang and Li 2003). The crossover point j is determined via a sequential-search based on the fitness function (18) before the single-gene crossover actually takes place. The search algorithm is similar to the sequential search of a database. If there is no satisfactory crossover point in the current generation, then the crossover point is designated as $j = \bar{\kappa}$, so that the single-gene crossover is performed on the dummy gene, $\phi^{\ell}_{\bar{\kappa}+1}$, and the fitness values of chromosomes in the population will not be affected (Caponetto et al. 1993).

4.4. *Improved adaptive bound search for control points and weights*

The control points and weights of the B-spline fuzzy-neural network trained by the RGA in Wang and Li (2003) had fixed bounds. However, how do we know these bounds are proper? These bounds will affect the speed of the convergence of the learning algorithms and the response of the system. For this reason, we use a modified version of the adaptive bound method

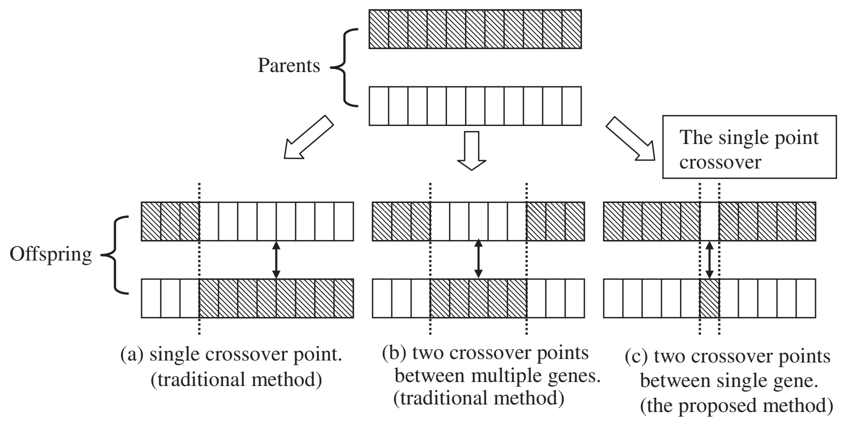
Parents
The single point crossover
Offspring
(a) single crossover point. (traditional method)
(b) two crossover points between multiple genes. (traditional method)
(c) two crossover points between single gene. (the proposed method)

Figure 2. Traditional crossover methods and the proposed single-gene crossover method.

(Wang et al. 2004) to adjust the bounds of the control points and weights. We can set the initial bounds to zero for all cases and the modified method will enlarge the bounds if some required conditions are satisfied. This leads to a gradual increase of the bounds. Figure 3 shows the pseudo code for the modified adaptive bound algorithm, where e is the exponential function, the value of η_s is an inverse function of the error function E in (14), and the multipliers, m_{w} or m_{c}, are adaptive factors for weights or control points.

When do we enlarge the boundaries of the adjustable parameters? There are two conditions shown in Figure 4. First, if crossover does not improve the fitness values of the population, i.e. the crossover point $j=\bar{\kappa}$, then we assume there is no solution within these bounds at this generation, and so we enlarge the boundaries. Second, if fitness $(\hat{\phi}^1)-$fitness (φ^1) $<\varepsilon/t$, where t is the iteration number and ε is a specified positive number, then we force the boundaries to be enlarged. Since at initialisation, all parameters and bounds are set zero, enlarging the boundaries is necessary. As t (i.e. learning iteration number) increases, the influence of this condition (i.e. fitness $(\hat{\phi}^1)-$fitness (φ^1) $<\varepsilon/t$) on adjusting the boundaries decreases. The bounds are enlarged by $e^{-\eta_s}$ as shown in Figure 3. As η_s approaches infinity

```
Procedure [$c_{\min}$,$c_{\max}$]= improved adaptive bound method(j) % also for $w_i$
Begin
        $\eta_s = E^{-1}$   %E is the error function in (3-10)
        Perform  $c_{\min} = c_{\min} - m_c e^{-\eta_s}$ , $c_{\max} = c_{\max} + m_c e^{-\eta_s}$ ;
End
```

Figure 3. Pseudo code for the improved adaptive bounds of adjustable parameters.

```
Procedure Bound Adjustment Sequential-Search-based Crossover Point algorithm(j)
Begin
    Let j=0; % i=0 at initialization
    Repeat
        Perform  $\hat{\Psi} = Crs(\Psi; i)$;
        Evaluate  $fitness(\hat{\phi}^1)$ and $fitness(\phi^1)$ by (4-3);
        i=i+1;
    Until  $fitness(\hat{\phi}^1) > fitness(\phi^1)$ or $i = \bar{\kappa}$ ;

    % To decide the crossover point
    If  $fitness(\hat{\phi}^1) > fitness(\phi^1)$ Then  $j = i$;
    Else  $j = i = \bar{\kappa}$ ;
    Return j=i;

    % To judge whether to enlarge the bound or not
    If  $i = \bar{\kappa}$  or  $(fitness(\hat{\phi}^1) - fitness(\phi^1) < \varepsilon / t)$
    Then execute adaptive bounds method(j) in Fig. 3;
End
```

Figure 4. Pseudo code for the bounds adjustable sequential-search-based crossover point method and additional conditions.

(error approaches zero), the bounds of the adjustable parameters become fixed ($e^{-\eta_s}\to 0$ as $\eta_s\to\infty$). The pseudo code for the bound adjustment sequential-search-based crossover point method is shown in Figure 4. The improved adaptive bound method has advantages over the traditional one (Wang et al. 2004). First, pre-determining the initial (non-zero) boundaries is unnecessary. Second, the variable η_s (an arbitrary constant in Wang et al. (2004)) is determinate now. Third, the speed of convergence is faster.

4.5. *Sorting*

After crossover, the newly generated population is sorted by fitness, resulting in $E(\hat{\phi}^1)\le E(\hat{\phi}^2)\le\cdots\le E(\hat{\phi}^k)$. Obviously, the first chromosome $\hat{\phi}^1$ of the sorted population $\hat{\Psi}=[\hat{\phi}^1\ \hat{\phi}^2\ \cdots\ \hat{\phi}^k]^T$ has the smallest error value.

4.6. *Mutation*

After sorting, the first chromosome, $\hat{\phi}^1$, is the best one in the population in terms of fitness. Genes within the $(k/2+1)$-*th* chromosome are randomly selected for mutation, according to the mutation rate p_m. Note that mutation on the selected genes is performed based on the fittest chromosome, i.e. the first chromosome $\hat{\phi}^1$ of the sorted population $\hat{\Psi}$. That is, genes $\hat{\phi}_j^{(k/2+1)}$ selected for mutation within the $(k/2+1)$-th chromosome $\hat{\phi}^{(k/2+1)}$ are altered by the following mutation operator as

$$\hat{\phi}_j^{(k/2+1)}=\begin{cases}\phi_j^1+\Delta\left(t, w_{\max}-\phi_j^1\right) & \text{if } \delta>0.5,\\ \phi_j^1-\Delta\left(t,\phi_j^1-w_{\min}\right) & \text{if } \delta\le 0.5,\end{cases}\tag{21}$$

$$\Delta(t,y)=y^{*}\gamma^{*}(1-t/T)^{\gamma},\tag{22}$$

where $\delta\in[0,1]$ is a random value, t is the current iteration, $\gamma\in[0,1]$ is a random number, and T is the maximal generation number. $(1-t/T)$ is the degree of dependency on an iteration number. The function $\Delta(t,y)$ returns a value in the range of $[0,y]$ such that the probability $\Delta(t,y)$ being close to 0 increases as t increases. Pseudo code for MARG is shown in Figure 5.

The steps of the MARG are presented as follows:

Step 1: Set all genes (adjustable parameters) and their boundaries to zero.

Step 2: Perform SSCP until the crossover point is found.

Step 3: Perform crossover and compute the fitness value.

Step 4: Perform the improved adaptive bound algorithm to adjust the boundaries of genes.

```
Procedure MARG % Improved Adaptive Bound Reduced-form Genetic Algorithm
Begin
    Initialize $\Psi$ % generate an initial population
    While (not terminate-condition) do
        Perform Boundary Adjustment Sequential-Search-based Crossover Point
            algorithm (j) in Fig. 4;
        Perform $\hat{\Psi} = Crs(\Psi; j)$;
        Sort $\hat{\Psi}$;
        Mutate $\hat{\Psi}$; % only apply to the middle chromosome
    End While
End
```

Figure 5. Pseudo code for the improved adaptive bound reduced-form genetic algorithm with gene boundary adjustment.

Step 5: Perform sorting and mutation.
Step 6: Obtain the control signal using the currently optimal chromosomes (solution).
Step 7: Go to **[Step 2]** for next iteration.

5. On-line GA-based fuzzy-neural sliding mode controllers

Now, we choose a vector of general switching planes $S^T = [s_1 s_2 \cdots s_m] = 0^T$, which can be a non-linear or time-varying manifold, defined as

$$S = \dot{\Theta} - \Lambda \tag{23}$$

where $\Lambda(\Theta, \Theta_d, \dot{\Theta}_d, t)$ is a design vector, Θ_d represents the vector of the desired positions, $\Lambda = \dot{\Theta}_d - \vartheta(e)$, and $e = \Theta - \Theta_d$. $\vartheta(e) = [\delta_1(e_1)\ \delta_2(e_2)\ \cdots\ \delta_m(e_m)]^T$ is a vector of error functions designated by the designer. If e_i is zero, then we define δ_i to be zero. The aim of the control is to force the motion of the system to be along the intersection of the general switching planes $S = 0$. In this section, an adaptive sliding mode controller with an on-line MARG fuzzy neural approximator is proposed. Based on the sliding manifold in (23), the controller is chosen as

$$u = -KS + \begin{bmatrix} L_1 & L_2 \end{bmatrix} \begin{bmatrix} u_{eq1} \\ u_{eq2} \end{bmatrix} - u_\Delta \tag{24}$$

where K is a pre-specified positive matrix, $L_1 = \mathrm{diag}[\ell_{11}, \ell_{12}, \ldots, \ell_{1m}]$, $\ell_{1i} = 1$, $L_2 = \mathrm{diag}[\ell_{21}, \ell_{22}, \ldots, \ell_{2m}]$, $0 \le \ell_{2i} \le 1$, $u_{eq1} = -[\hat{\alpha}_1 \|\dot{\Lambda}\| + \hat{\alpha}_2 \|\dot{\Theta}\| \|\Lambda\| + \hat{\alpha}_3]\mathrm{sgn}(S)$, $u_{eq2} = W\varphi$ (the output of the B-spline membership function fuzzy-neural approximator), and $u_\Delta = \{\hat{\beta}_i \mathrm{sgn}(s_i), i = 1, 2, \ldots, m\}$. $\hat{\alpha}_j, j = 1, 2, 3$ and $\hat{\beta}_i, i = 1, 2, \ldots, m$ are the estimations of α_j in Assumption 2 and β_i in Assumption 1, respectively. Therefore, the adaptive laws of $\hat{\alpha}_j$ and $\hat{\beta}_i$ are considered as

$$\dot{\hat{\alpha}}_1 = \zeta_1 \|S\| \|\dot{\Lambda}\|, \tag{25}$$

$$\dot{\hat{\alpha}}_2 = \zeta_2 \|\Lambda\| \|S\| \|\dot{\Theta}\|, \tag{26}$$

$$\dot{\hat{\alpha}}_3 = \zeta_3 \|S\|, \tag{27}$$

and

$$\dot{\hat{\beta}}_i = \eta_i |s_i|, \quad i = 1, \ldots, m. \tag{28}$$

Based on the above discussion, we have the following main theorem.

Theorem 1: *Consider the dynamic Equation* (1) *and that the Assumptions* 1 *and* 2 *are satisfied. If the vector of applied joint torques u is designed as* (24) *and the adaptive laws are chosen as* (25)–(28), *then the general sliding mode S will asymptotically converge to zero as time tends to infinity.*

Proof: Consider the candidate Lyapunov function

$$v = \frac{1}{2} S^T H S + \frac{1}{2} \tilde{A}^T \zeta \tilde{A} + \frac{1}{2} \tilde{B}^T \eta \tilde{B} \tag{29}$$

where

$$\tilde{A} = \begin{bmatrix} \alpha_1 - \hat{\alpha}_1 & \alpha_2 - \hat{\alpha}_2 & \alpha_3 - \hat{\alpha}_3 \end{bmatrix}^T,$$

$$\tilde{B} = \begin{bmatrix} \beta_1 - \hat{\beta}_1 & \beta_2 - \hat{\beta}_2 & ,\ldots, & \beta_m - \hat{\beta}_m \end{bmatrix}^T,$$

$$\zeta = \mathrm{diag}\left[\frac{1}{\zeta_1}, \frac{1}{\zeta_2}, \frac{1}{\zeta_3}\right], \quad \zeta_i > 0,\ i = 1, 2, 3,$$

and

$$\eta = \mathrm{diag}\left[\frac{1}{\eta_1}, \frac{1}{\eta_2}, \ldots, \frac{1}{\eta_m}\right], \quad \eta_i > 0, i = 1, \ldots, m.$$

Differentiating S with respect to time, we have

$$\dot{S} = \ddot{\Theta} - \dot{\Lambda}. \tag{30}$$

Multiplying the matrix H by (30) and inserting (1) for $H\ddot{\Theta}$ and $\dot{\Theta} = S + \Lambda$ yields

$$\begin{aligned} H\dot{S} &= H\ddot{\Theta} - H\dot{\Lambda} \\ &= u + \Delta_d - CS - \left(H\dot{\Lambda} + C\Lambda + G\right). \end{aligned} \tag{31}$$

Differentiating v with respect to time, we have

$$\dot{v} = S^T H \dot{S} + \frac{1}{2} S^T \dot{H} S + \dot{\tilde{A}}^T \zeta \tilde{A} + \dot{\tilde{B}}^T \eta \tilde{B}. \tag{32}$$

Inserting (31) in the above equation yields

$$\begin{aligned} \dot{v} = S^T &\left[u + \Delta_d - \left(H\dot{\Lambda} + C\Lambda + G\right)\right] \\ &+ S^T \left[\frac{1}{2}\dot{H} - C\right] S + \dot{\tilde{A}}^T \zeta \tilde{A} + \dot{\tilde{B}}^T \eta \tilde{B}. \end{aligned} \tag{33}$$

Because $[\frac{1}{2}\dot{H} - C]$ is a skew-symmetric matrix, the above equation become

$$\begin{aligned}\dot{v} &= S^T\big[u + \Delta_d - \big[H\dot{\Lambda} + C\Lambda + G\big]\big] + \dot{\tilde{A}}^T\zeta\tilde{A} + \dot{\tilde{B}}^T\eta\tilde{B} \\ &= S^T[u + \Delta_d - (\tilde{H}\dot{\Lambda} + \tilde{C}\Lambda + \tilde{G}) - L_2(\hat{H}\dot{\Lambda} + \hat{C}\Lambda + \hat{G})] \\ &\quad + \dot{\tilde{A}}^T\zeta\tilde{A} + \dot{\tilde{B}}^T\eta\tilde{B}. \end{aligned} \tag{34}$$

Since the robot dynamic structure is linear in terms of its parameters, the above equation can be expressed as

$$\begin{aligned}\dot{v} &= S^T\Big[u + \Delta_d - \Big[\tilde{H}\dot{\Lambda} + \tilde{C}\Lambda + \tilde{G}\Big] - L_2W\varphi\Big] \\ &\quad + \dot{\tilde{A}}^T\zeta\tilde{A} + \dot{\tilde{B}}^T\eta\tilde{B}\end{aligned} \tag{35}$$

where $\hat{H}\dot{\Lambda} + \hat{C}\Lambda + \hat{G} = W\varphi$ in (4), and

$$\begin{aligned}\dot{v} &\le S^Tu + \sum_{i=1}^{n}\beta_i|s_i| \\ &\quad + \big[\ \|S\|\|\dot{\Lambda}\| \quad \|S\|\|\Lambda\|\|\dot{\Theta}\| \quad \|S\|\ \big]\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} \\ &\quad - S^TL_2W\varphi + \dot{\tilde{A}}^T\zeta\tilde{A} + \dot{\tilde{B}}^T\eta\tilde{B}.\end{aligned} \tag{36}$$

Inserting (24) in the above equation yields

$$\begin{aligned}\dot{v} &\le -S^TKS + S^TL_1u_{eq1} + S^TL_2u_{eq2} - S^Tu_\Delta \\ &\quad + \sum_{i=1}^{m}\beta_i|s_i| + \big[\ \|S\|\|\dot{\Lambda}\| \quad \|S\|\|\Lambda\|\|\dot{\theta}\| \quad \|S\|\ \big]\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} \\ &\quad - S^TL_2W\varphi + \dot{\tilde{A}}^T\zeta\tilde{A} + \dot{\tilde{B}}^T\eta\tilde{B}.\end{aligned} \tag{37}$$

We choose

$$-\dot{\tilde{B}}^T\eta\tilde{B} = \sum_{i-1}^{m}(\beta_i - \hat{\beta}_i)|s_i| \tag{38}$$

and

$$-\dot{\tilde{A}}^T\zeta\tilde{A} = \big[\ \|S\|\|\dot{\Lambda}\| \quad \|S\|\|\Lambda\|\|\dot{\Theta}\| \quad \|S\|\ \big]\begin{bmatrix}\tilde{\alpha}_1\\ \tilde{\alpha}_2\\ \tilde{\alpha}_3\end{bmatrix} \tag{39}$$

where

$$-\dot{\tilde{B}} = [\eta_1|s_1|\eta_2|s_2|\cdots\eta_m|s_m|]^T \tag{40}$$

and

$$-\dot{\tilde{A}} = \big[\ \zeta_1\|S\|\|\dot{\Lambda}\| \quad \zeta_2\|S\|\|\Lambda\|\dot{\Theta}\|\| \quad \zeta_3\|S\|\ \big]^T. \tag{41}$$

Since $\dot{\tilde{B}} = -[\dot{\hat{}}\beta_1\ \ \dot{\hat{\beta}}_2\ \ \cdots\ \ \dot{\hat{\beta}}_m]^T$ and $\dot{\tilde{A}} = -[\dot{\hat{\alpha}}_1\dot{\hat{\alpha}}_2\dot{\hat{\alpha}}_3]^T$, we obtain

$$\Big[\dot{\hat{\beta}}_1\ \ \dot{\hat{\beta}}_2\ \ \cdots\ \ \dot{\hat{\beta}}_m\Big] = [\eta_1|s_1|\eta_2|s_2|\cdots\eta_m|s_m|] \tag{42}$$

and

$$[\dot{\hat{\alpha}}_1\dot{\hat{\alpha}}_2\dot{\hat{\alpha}}_3] = \big[\zeta_1\|S\|\|\dot{\Lambda}\|\zeta_2\|S\|\|\Lambda\|\|\dot{\Theta}\|\zeta_3\|S\|\big]. \tag{43}$$

From (42) and (43), we obtain (25) to (28).

Thus, the resulting expression of $\dot{v}$ is

$$\dot{v} \le -S^TKS \le 0. \tag{44}$$

Because K is a positive definite matrix, S approaches zero asymptotically. This completes the proof. □

6. Simulations

In this section, simulations of the proposed on-line genetic algorithm-based fuzzy-neural sliding mode control scheme trained by MARG for a two-link robotic manipulator model are presented. Two examples are employed to illustrate the performance of the control scheme. The two-link robotic manipulator is shown in Figure 6.

The initial adaptation gains, initial conditions for the robot, and design parameters are given by $\hat{\alpha}_1(0) = 0.002$, $\hat{\alpha}_2(0) = 0.005$, $\hat{\alpha}_3(0) = 10.2$, $\zeta_1 = 0.009$, $\zeta_2 = 0.005$, $\zeta_3 = 3.2$, and $K = \text{diag}[12\ 12]$. The external disturbances are square waves with the amplitude ± 0.1 and the period 2π.

For the two-link robotic manipulator, the sliding manifold defined in (23) is $\Lambda = \dot{\Theta}_d - \vartheta$, and

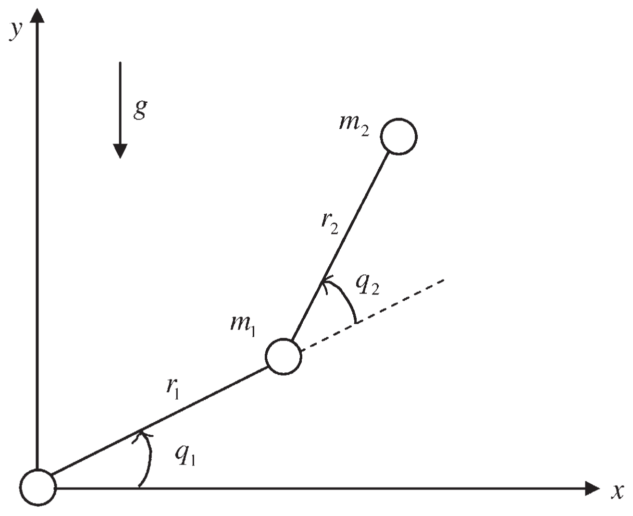

Figure 6. Two-link manipulator.

$\vartheta = [\,\delta_1 \quad \delta_2\,]^T$ is chosen as

$$\delta_i(e_i) = \begin{cases} \dfrac{e_m}{(1/\lambda_m) + K_e e_m} + \lambda_0(e_i - e_m), & \text{if } e_m < e_i \\ \dfrac{e_i}{(1/\lambda_m) + K_e e_i}, & \text{if } 0 < e_i \le e_m \\ 0, & \text{if } e_i = 0 \\ \dfrac{e_i}{(1/\lambda_m) - K_e e_i}, & \text{if } -e_m \le e_i < 0 \\ -\dfrac{e_m}{(1/\lambda_m) + K_e e_m} + \lambda_0(e_i + e_m), & \text{if } -e_m > e_i \end{cases},$$

for $i = 1,2$, where $e_m = [\sqrt{(\lambda_m/\lambda_0)} - 1]/(K_e\lambda_m)$, $\lambda_m > \lambda_0 > 0$, $\lambda_m = 16$, $\lambda_0 = 4$, $K_e = 0.2$ and $e_m = 1/3.2$. The torque vector $Y^* = [\,y_1^* \quad y_2^*\,]^T$ is given by

$$\begin{aligned} y_1^* &= [c_2(2\ddot{q}_{1d} + \ddot{q}_{2d}) - s_2\dot{q}_{2d}(\dot{q}_{2d} + 2\dot{q}_{1d})] \times m_2 l_1 l_2 \\ &\quad + (\ddot{q}_{1d} + \ddot{q}_{2d}) \times m_2 l_2^2 + g c_{12} \times m_2 l_2 \\ &\quad + g c_1 \times (m_1 + m_2) l_1 + \ddot{q}_{1d} \times (m_1 + m_2) l_1^2, \end{aligned}$$

$$\begin{aligned} y_2^* &= (c_2\ddot{q}_{1d} + s_2\dot{q}_{1d}^2) \times m_2 l_1 l_2 + (\ddot{q}_{1d} + \ddot{q}_{2d}) \times m_2 l_2^2 \\ &\quad + g c_{12} \times m_2 l_2, \end{aligned}$$

where $c_1 = \cos q_{1d}$, $c_2 = \cos q_{d2}$, $s_2 = \sin q_{d2}$, and $c_{12} = \cos(q_{d1} + q_{d2})$. The parameters of the two-link robotic manipulator are $m_1 = 0.5$ kg, $m_2 = 0.5$ kg, $l_1 = 1$ and $l_2 = 0.8$.

There are 120 adjustable parameters, including 50 weighting factors of fuzzy neural network and 70 control points of B-spline membership functions. Each input has five B-spline membership functions. Each B-spline membership function has seven control points. MARG, which is described in Section 4, is applied to find the optimal values and bounds of these parameters. The initial condition is set to be $[q_1\ \dot{q}_1\ q_2\ \dot{q}_2] = [0.2618\ 0\ 0.2618\ 0]$.

Example 1: In example 1, we refer to the trajectories in Leu and Lee (2000). Our objective is to control the outputs of the actual angles q_1 and q_2 of the system to track the desired angle trajectories $q_{1d} = \pi[1 - \exp(-t/2)]/8$ and $q_{2d} = -\pi[1 - \exp(-t/2)]/8$, respectively. An additional load ($m_L = 0.2$ kg) is added to m_2 after 14 s. The additional load $m_L = 0.2$ kg is used to show that the two-link robotic manipulator is robust while changing loads.

As previously discussed, if we do not use the proposed MARG, we must guess the bounds of the adjustable parameters (i.e. the bounds of the genes) before beginning the simulation. Figure 7 shows the tracking trajectories using the B-spline membership function fuzzy-neural network and guessed bounds. We set the bounds of all control points of the B-spline membership functions to $[-0.5, 0.5]$, and the bounds of all weights of the fuzzy-neural network to $[-2, 2]$. y_1^* and y_2^* are the ideal outputs of the inverse robot system dynamics, and $W\varphi_1$ and $W\varphi_2$ represent the outputs of the B-spline membership function fuzzy-neural network. We can see clearly that the fuzzy-neural network doesn't approximate the ideal outputs of the inverse robot system dynamics, $Y^* = [\,y_1^* \quad y_2^*\,]^T$, well because the bounds we guessed are improper.

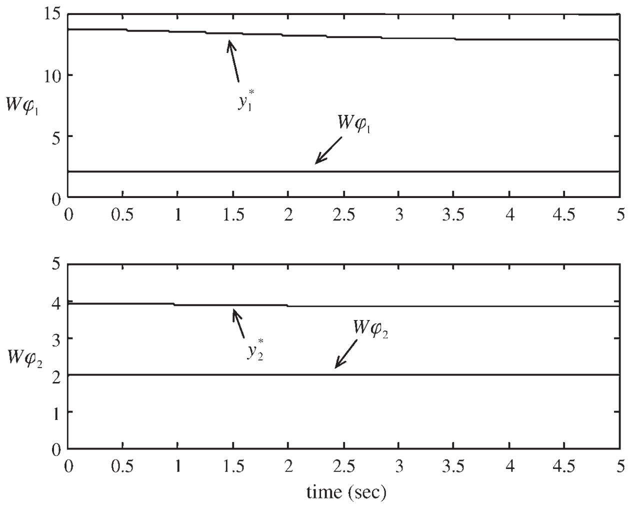

Figure 7. The tracking trajectories $W\varphi_1$ and $W\varphi_2$ by using the fuzzy-neural networks with guessed bounds $[-0.5, 0.5]$ and $[-2, 2]$ for Example 1.

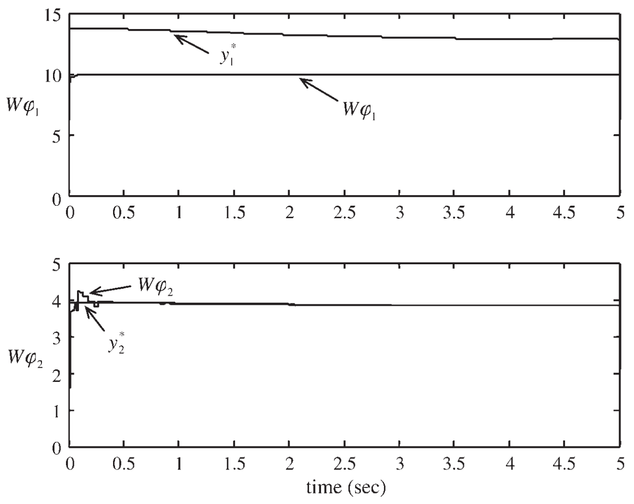

Figure 8. The tracking trajectories $W\varphi_1$ and $W\varphi_2$ by using the fuzzy-neural networks with guessed bounds $[-1, 1]$ and $[-10, 10]$ for Example 1.

In Figure 8, we reset the bounds of all control points of the B-spline membership functions to $[-1, 1]$ and the bounds of all weights to $[-10, 10]$. With these new bounds, $W\varphi_2$ traces y_2^* a little better but $W\varphi_1$ still does not trace y_1^* well. We have to keep adjusting these bounds until the fuzzy-neural network approximates the ideal trajectory well, a potentially time-consuming procedure.

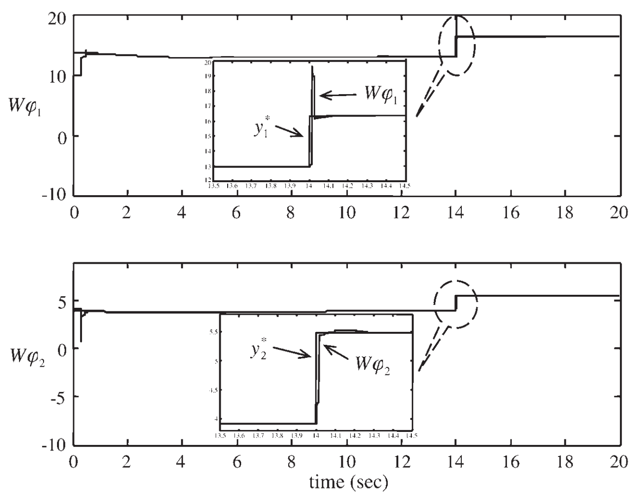

Figure 9. The tracking trajectories $W\varphi_1$ and $W\varphi_2$ by using the fuzzy-neural networks with MARG for Example 1.

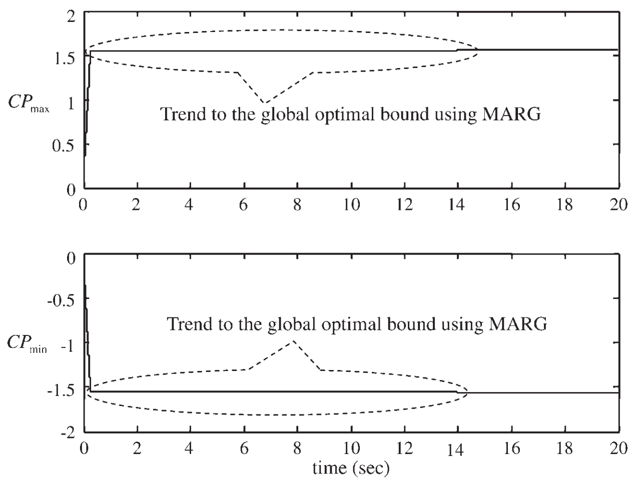

Figure 11. The final searched boundaries of the control points of B-spline membership function for Example 1.

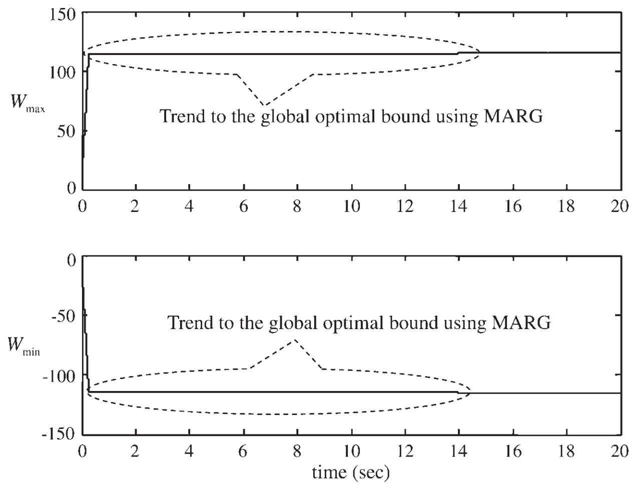

Figure 10. The final searched boundaries of the weights of the fuzzy-neural network for Example 1.

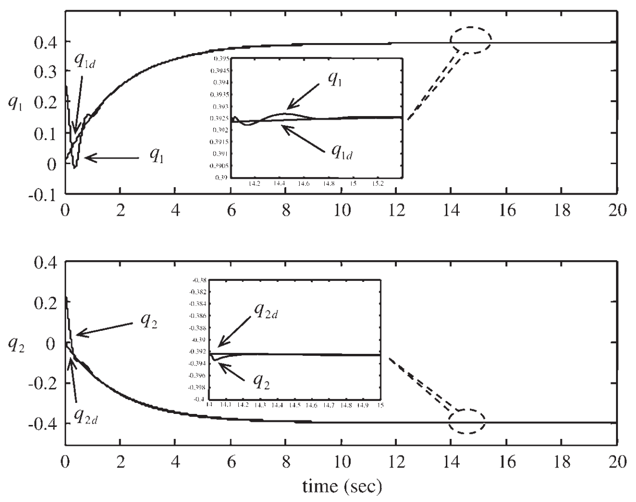

Figure 12. The actual angles q_1 and q_2 and the desired angles q_{1d} and q_{2d} using the proposed method for Example 1.

To overcome the above problem, we use MARG and redo the simulation. Pre-determining the initial (non-zero) boundaries is unnecessary. At initialisation, all parameters in (16) and bounds of the genes are set to zero. Figure 9 shows that the outputs of the fuzzy neural network, $W\varphi$, approximate the ideal outputs of the inverse robot system dynamics Y^*. It shows good approximating performance after 0.8 s in the transient period. When the load is added to the system at 14 s, it also shows good approximating performance after 14.3 s. We can see this fact from the insets.

Although all parameters and bounds of the genes are set to zero at initialisation, Figures 10 and 11 show that the final searched bounds using MARG are $[-115, 115]$ and $[-1.56, 1.56]$ for the weights of the fuzzy-neural network and the control points of B-spline membership functions, respectively. This example shows that MARG can overcome the problem of pre-determining parameter bounds in genetic algorithms.

Figure 12 presents the simulation results of the tracking control by MARG. The proposed controller has good control capability over the exponential waves, q_{1d} and q_{2d}. The simulation results in this case demonstrate that the tracking trajectories of the actual outputs q_1 and q_2 also follow the reference signals q_{1d} and q_{2d}, respectively, very well even when an additional load is added after 14 seconds. In order to specifically demonstrate our proposed control scheme is still effective under the additional load, we magnify Figure 12

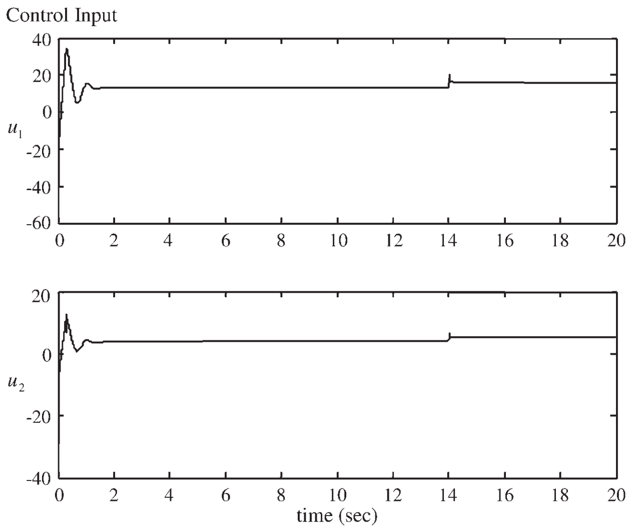

Figure 13. The control inputs for Example 1.

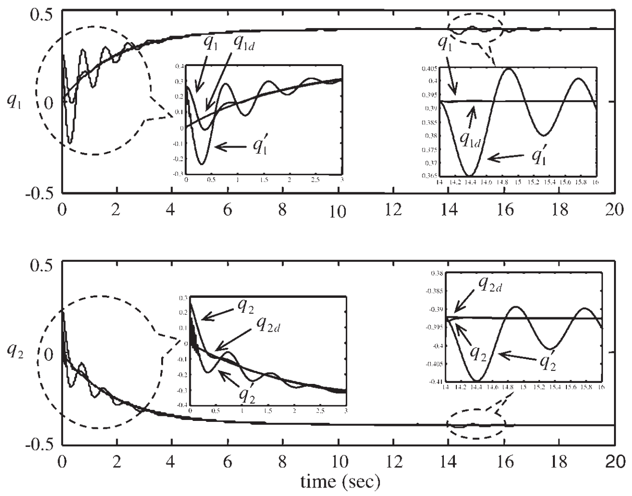

Figure 15. Compare the result of our proposed method with the one of the reference paper method (Leu and Lee 2000) for Example 1.

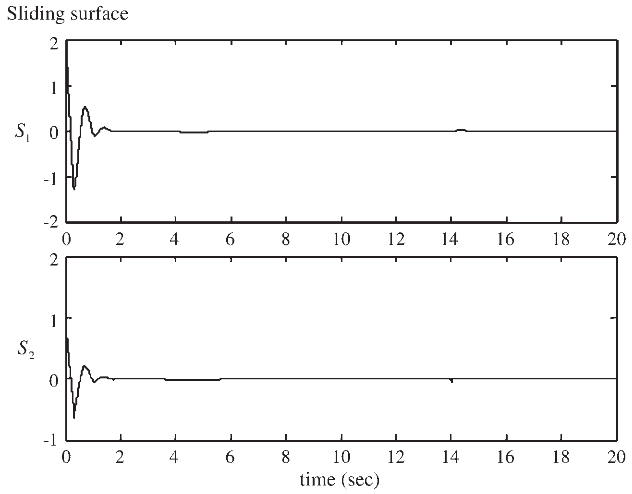

Figure 14. The sliding surfaces for Example 1.

around the 14 s region. We see clearly that the actual trajectories q_1 and q_2 quickly track the desired trajectories q_{1d} and q_{2d} after encountering the additional load. Our proposed controller demonstrates good robust performance.

Figure 13 shows the control inputs. It shows that using the general sliding manifold diminishes control law chattering. Figure 14 shows the sliding surfaces. The general sliding mode S will asymptotically converge to zero as time tends to infinity except immediately after encountering the extra load. Although the trajectory S immediately moves away from the convergence trajectory, it does not diverge and recovers to convergence quickly. This result is because of our proposed control scheme. From this example, we can see that our presented theorem and proof behave as expected and make for a practical robot system.

We also compared our proposed controller with the observer-based adaptive fuzzy-neural controller in Leu and Lee (2000). In Figure 15, q_{1d} and q_{2d} are the reference trajectories, q_1 and q_2 represent the trajectories of our proposed method, and q_1' and q_2' represent the trajectories of the reference paper's method (Leu and Lee 2000). It clearly shows that the trajectories of our proposed method trace the reference trajectories better than those of the reference paper's method (Leu and Lee 2000) in the transient period. Even when the load is added to the system at 14 seconds, our proposed method still shows good robust ability. We can see this fact from the insets of Figure 15. Hence, our proposed controller is better than the observer-based adaptive fuzzy-neural controller in Leu and Lee (2000).

Example 2: In example 2, our objective is to control the outputs of the actual angles q_1 and $q_2 q_2$ of the two-link planar manipulator to track the desired angle trajectories $q_{1m} = (\pi/30)\sin(t)$ and $q_{2m} = (\pi/30)\cos(t)$, respectively. At initialisation, all parameters in (16) and bounds of the genes for the improved adaptive bound reduced-form genetic algorithm are also set to zero. Figure 16 shows that the outputs of the fuzzy neural network, $W\varphi$, approximate the ideal outputs of the inverse robot system dynamics Y^*. We magnify the transient part to demonstrate the action of the outputs of the fuzzy neural network. It shows good approximating performance after 0.5 s. It is unnecessary to guess the bounds of the parameters. Although all parameters and bounds of the genes are set to zero at

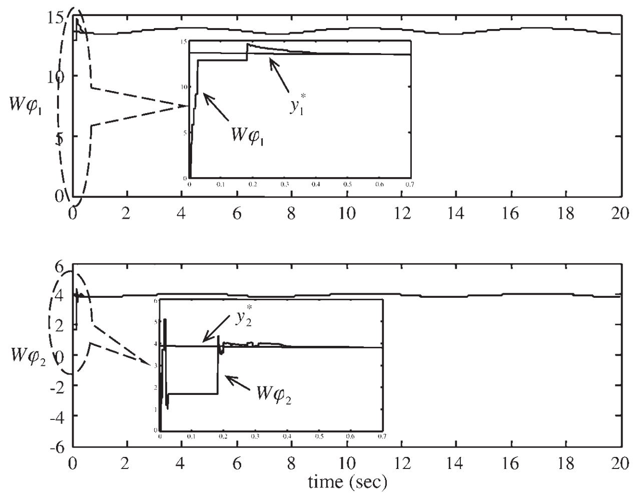

Figure 16. The tracking trajectories $W\varphi_1$ and $W\varphi_2$ by using the fuzzy-neural networks with MARG for Example 2.

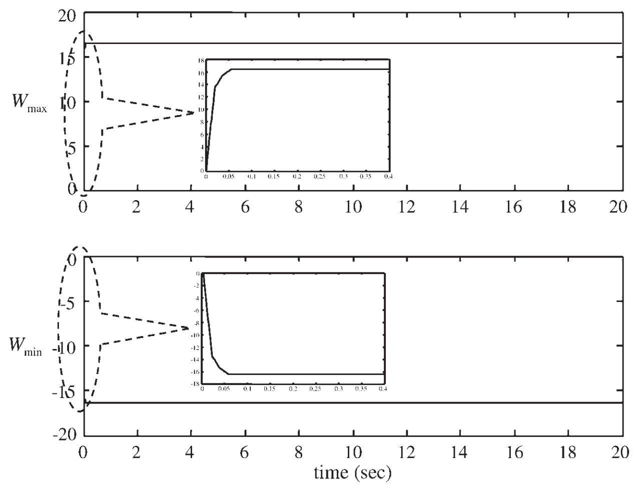

Figure 17. The final searched boundaries about the weights of BMF fuzzy-neural network for Example 2.

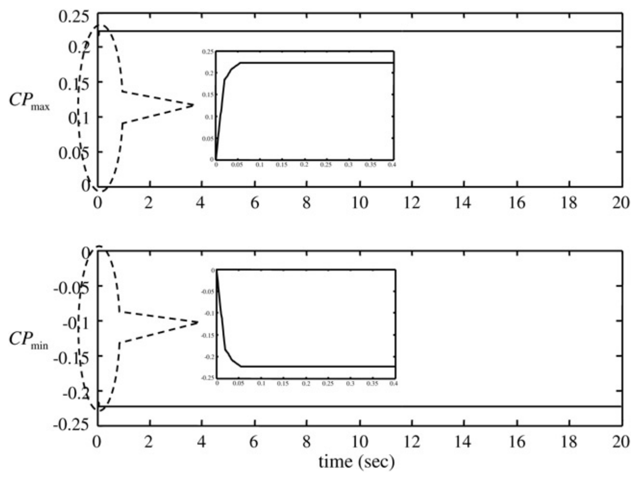

Figure 18. The final searched boundaries about the control points of B-spline membership function for Example 2.

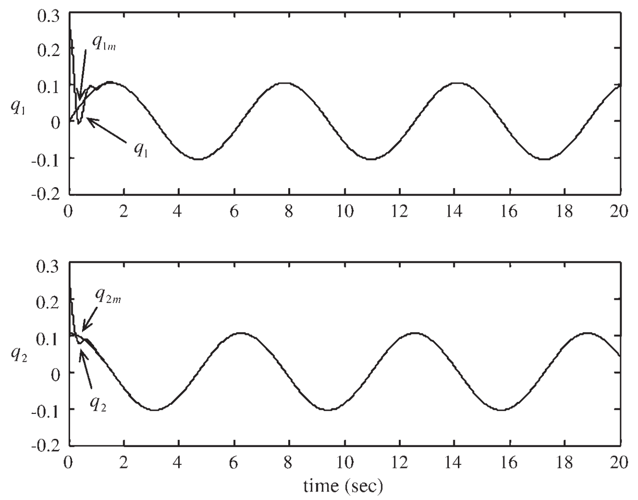

Figure 19. The actual angles q_1 and q_2 and the desired angles q_{1m} and q_{2m} using the proposed method for Example 2.

initialisation, Figures 17 and 18 show that the final searched bounds are $[-16.4, 16.4]$ and $[-0.23, 0.23]$ using MARG for the weights of the fuzzy-neural network and the control points of B-spline membership functions, respectively. Again, this example shows that MARG can overcome the problem of pre-determining parameter bounds in genetic algorithms.

Figure 19 presents the simulation results of the tracking control by MARG. Like example 1, the proposed controller also has good control capability over the sine and cosine waves, q_{1m} and q_{2m}. The actual trajectories trace the ideal trajectories well after 1.5 s.

Table 1 shows 25 (out of a total of 120) weights of the B-spline membership function fuzzy-neural network using fixed bounds and using the improved adaptive bound algorithm. We also give $E_o = |q - q_d|^2$, which represents the square error of the system output, and $E_e = |Y - Y^*|^2$, which represents the square error of the estimation of the torque vector. As we can see in Table 1, the proposed improved adaptive bound reduced-form genetic algorithm is more effective than the traditional method using fixed bounds. If the improper bounds for adjustable parameters are pre-determined, some optimal parameters are not discovered in the improper bounds. The proper bounds of the adjustable parameters can be obtained automatically in the training process by applying MARG. Therefore, the tracking error can be, and is, reduced significantly.

In this article, an on-line evolutionary controller is derived. In the papers of Leu et al. (1999, 2005), they present fuzzy-neural controllers with adaptive learning algorithms, but do not use genetic algorithms and so

Table 1. Twenty-five weights of the B-spline membership function fuzzy-neural network using fixed bounds and using the improved adaptive bound algorithm.

Selected adjustable values $[w_1^1 \cdots w_1^{25}]$ (remaining values are omitted to save space)	Final adjustable values for fixed bound algorithm with $D_1 = [w_{\min}, w_{\max}] \subseteq R = [-10, 10]$	Final adjustable values for improved adaptive bound algorithm. Final searched bound is $D_1 = [w_{\min}, w_{\max}] \subseteq R = [-16.4, 16.4]$
$w = [w_1^T\ w_2^T \cdots w_m^T]$	3.6331	−10.0428
$w = [w_1^T\ w_2^T \cdots w_m^T]$	10.0000	−5.6908
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.0037	−2.0548
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.7692	9.6143
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.8252	14.6507
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.9272	0.3456
$w = [w_1^T\ w_2^T \cdots w_m^T]$	10.0000	14.3441
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.8430	13.0843
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.7232	2.6597
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.4509	−1.3903
$w = [w_1^T\ w_2^T \cdots w_m^T]$	10.0000	−7.8490
$w = [w_1^T\ w_2^T \cdots w_m^T]$	10.0000	8.9022
$w = [w_1^T\ w_2^T \cdots w_m^T]$	10.0000	11.1237
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.9968	15.4167
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.9993	14.8704
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.2926	14.0115
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.9978	14.3402
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.6344	15.0663
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.6993	5.4284
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.5565	−3.3603
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.5731	14.0449
$w = [w_1^T\ w_2^T \cdots w_m^T]$	10.0000	14.5239
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.4595	12.5299
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.2363	5.7721
$w = [w_1^T\ w_2^T \cdots w_m^T]$	9.8978	−2.1470
$E_o = \lvert q - q_d \rvert^2$	8.3362e-005	2.1593e-011
$E(w, c) = \frac{1}{2} \lVert Y - Y^* \rVert^2$	11.5904 (not converged)	6.2627e-008

can fall into local minima. Our system uses genetic algorithms to overcome this important problem. Moreover, unlike many genetic algorithm-based controllers, ours is on-line. In the previous studies (Wang and Li 2003; Wang et al. 2004), Wang et al. present genetic algorithms and an adjustable bounds algorithm. However, the studies Wang and Li (2003) and Wang et al. (2004) do not implement controllers, and the adjustable bounds algorithm has been improved in this article.

7. Conclusion

A novel on-line genetic algorithm-based fuzzy-neural sliding mode controller trained by MARG is developed for robot manipulators with uncertainties and external disturbances. The general sliding manifold is employed to construct the sliding surface and reduce control law chattering; the sliding surface can be non-linear or time-varying. The sliding mode controller can eliminate uncertainties and external disturbances. The B-spline membership function fuzzy-neural network with MARG can approximate the regressor well. The proper bounds of the adjustable parameters can be obtained automatically in the training process by applying MARG. The proposed on-line genetic algorithm-based fuzzy-neural sliding mode controller can guarantee robust stability and good tracking performance of a robot manipulator with uncertainties and external disturbances. Results of the computer simulation applied to a two-link robotic manipulator following the desired trajectories demonstrate the effectiveness of the proposed approach.

Acknowledgement

This work was supported by the National Science Council, Republic of China, under Grants NSC 96-2221-E-027-116.

References

Caponetto, R., Fortuna, L., Graziani, S., and Xibilia, M. (1993), 'Genetic Algorithms and Applications in System Engineering: A Survey', *Transactions of the Institute of Measurement and Control*, 15, 143–156.

Farag, W.A., Quintana, V.H., and Germano, L.T. (1998), 'A Genetic-Based Neuro-Fuzzy Approach for Modeling and Control of Dynamical Systems', *IEEE Transactions on Neural Networks*, 9, 756–767.

Ferreira, J.R., Lopes, J.A.P., and Saraiva J.T. (2000), 'A Real Time Approach to Identify Actions to Prevent Voltage Collapse Using Genetic Algorithms and Neural Networks', *IEEE Power Engineering Society Summer Meeting* (Vol. 1), Washington, U.S.A., pp.255–260.

Gao, Y., and Er, M.J. (2001), 'Robust Adaptive Fuzzy Neural Control of Robot Manipulators', in *Proceedings of International Joint Conference on Neural Networks* (*IJCNN '01*) (Vol. 3), pp. 2188–2193.

Gen, M., and Cheng, R. (1997), *Genetic Algorithms and Engineering Design*, New York: A Wiley-Interscience Publication, John Wiley & Sons, INC.

Goldberg, D. (1989), *Genetic Algorithms in Search, Optimisation, and Machine Learning*, New York: Addison-Wesley.

Gonzalez, A., and Perez, R. (2001), 'Selection of Relevant Features in a Fuzzy Genetic Learning Algorithm', *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31, 417–425.

Hsu, K.C. (1998), 'Sliding Mode Controllers for Uncertain Systems with Input Nonlinearities', *Journal of Guidance, Control, and Dynamics*, 21, 666–668.

Kelly, R., Carelli, R., and Ortega, R. (1989), 'Adaptive Motion Control Design of Robot Manipulators: An Input–Output Approach', *International Journal of Control*, 50, 2563–2581.

Kristinsson, K., and Dumont, G.A. (1992), 'System Identification and Control using Genetic Algorithms', *IEEE Transactions on Systems, Man, and Cybernetics*, 22, 1033–1046.

Lawrence, D. (1991), *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold.

Leu, Y.G., Wang, W.Y., and Lee, T.T. (2005), 'Observer-Based Direct Adaptive Fuzzy-Neural Control for Nonaffine Nonlinear Systems', *IEEE Transactions on Neural Networks*, 16, 853–861.

Leu, Y.G., Wang, W.Y., and Lee, T.T. (1999), 'Robust Adaptive Fuzzy-Neural Controllers for Uncertain Nonlinear Systems', *IEEE Transactions on Robotics and Automation*, 15, 805–817.

Leu, Y.G., and Lee, T.T. (2000), 'Observer-Based Adaptive Fuzzy-Neural Control for a Class of MIMO Nonlinear Systems', in *IEEE IECON 26th Annual Conference* (Vol. 1), pp. 178–183.

Lin, C.T., and Jou, C.P. (2000), 'GA-Based Fuzzy Reinforcement Learning for Control of a Magnetic Bearing System', *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30, 276–289.

Lu, W.S., and Meng, Q.H. (1993), 'Regressor Formulation of Robot Dynamics: Computation and Applications', *IEEE Transactions on Robotics and Automation*, 9, 323–333.

Marra, M.A., and Walcott, B.L., (1996), 'Stability and Optimality in Genetic Algorithm Controllers', in *Intelligent Control, Proceedings of the 1996 IEEE International Symposium on 15–18 September*, pp. 492–496.

Meng, Q.H., (1993), 'A Neural Network Controller for Robots with Unknown Dynamics', in *Proceedings of 1993 International Joint Conference on Neural Networks*, Nagoya, Japan, pp. 1769–1772.

Michalewicz, Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, New York: Springer-Verlag.

Pei, H.L., and Zhou, Q.J. (1991), 'Variable Structure Control of Linearizable Systems with Applications to Robot Manipulator', in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 522–527.

Su, C.Y., and Stepanenko, Y. (1993), 'Adaptive Sliding Mode Control of Robot Manipulators with General Sliding Manifold', in *Proceedings of IEEE Conference on Intelligent Robots and Systems*, pp. 1255–1259.

Wang, C.H., Liu, H.L., and Lin, C.T. (2001), 'Dynamic Optimal Learning Rates of a Certain Class of Fuzzy Neural Networks and its Applications with Genetic Algorithm', *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31, 467–475.

Wang, C.H., Wang, W.Y., Lee, T.T., and Tseng, P.S. (1995), 'Fuzzy B-Spline Membership Function (BMF) and its Applications in Fuzzy-Neural Control', *IEEE Transactions on Systems, Man, and Cybernetics*, 25, 841–851.

Wang, W.Y., and Li, Y.H. (2003), 'Evolutionary Learning of BMF Fuzzy-Neural Networks Using a Reduced-Form Genetic Algorithm', *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33, 966–976.

Wang, W.Y., Lee, T.T., and Liu, C.L. (1997), 'Function Approximation Using Fuzzy Neural Networks with Robust Learning Algorithm', *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 27, 740–747.

Wang, W.Y., Tao, C.W., and Chang, C.G. (2004), 'Adaptive Bound Reduced-Form Genetic Algorithms for B-Spline Neural Network Training', *IEICE Transactions on Information and Systems*, 87-D, 2479–2488.

Wang, L.X. (1994), '*Adaptive Fuzzy Systems and Control*', New Jersey: Prentice Hall.

Yang, Z.J., Hachino, T., and Tsuji, T. (1996), 'Model Reduction with Time Delay Combining the Least-Squares Method with the Genetic Algorithm', *IEE Proceedings-Control Theory and Applications*, 143, 247–254.

Yeung, K.S., and Chen, Y.P. (1988), 'A new controller design for manipulators using the theory of variable structure systems', *IEEE Transactions on Automatic Control*, 33, 200–206.

Young, K.D. (1978), 'Controller Design for a Manipulator using Theory of Variable Structure System', *IEEE Transactions on Systems, Man, and Cybernetics*, 8, 101–109.