Decentralized Adaptive Fuzzy Control of Robot Manipulators

Yaochu Jin

***Abstract*—This paper develops a decentralized adaptive fuzzy control scheme for robot manipulators via a combination of genetic algorithm and gradient method. The controller for each link consists of a feedforward fuzzy torque-computing system and a feedback fuzzy PD system. The feedforward fuzzy system is trained and optimized off-line by an improved genetic algorithm, that is to say, not only the parameters but also the structure of the fuzzy system are self-organized. Because genetic algorithm can operate successfully without the system model, no exact inverse dynamics of the robot system are required. The feedback fuzzy PD system, on the other hand, is tuned on-line using gradient method. In this way, the proportional and derivative gains are adjusted properly to keep the closed-loop system stable. The proposed controller has the following merits: 1) it needs no exact dynamics of the robot systems and the computation is time-saving because of the simple structure of the fuzzy systems; and 2) the controller is insensitive to various dynamics and payload uncertainties in robot systems. These are demonstrated by analyses of the computational complexity and various computer simulations.**

I. Introduction

DYNAMIC control of robot manipulators is one of the most important topics in robotics. Various modern control strategies have been widely investigated to deal with the high nonlinearity and strong coupling of the robot dynamics. These controllers are generally designed assuming an exact knowledge about the model structure and do not include nonlinear friction, backlash and other uncertainties in robot systems.

Among the model-based robot controllers reported in the literature that have proved to be effective, the PD control with feedforward torque computation is the most promising one [1], [2]. It is also, perhaps, the simplest controller that can be implemented for practical position control of robot manipulators. Furthermore, it has proved that the PD control with computed feedforward is locally exponentially stable provided that the proportional and derivative gains are sufficiently large [3], [4]. However, it needs the full inverse dynamics of the robot and the computation is time-consuming. In addition, sufficiently large gains are undesirable in practice. To deal with the latter problem, [5] develops a design procedure that sets the lower bounds on the proportional and derivative gains that guarantee stability of the closed-loop system. The procedure is carried out under the assumption that the robot dynamic model is known and the desired trajectory is properly chosen. That is to say, if there exists large amount of uncertainties, the method will probably fail.

Since the inverse dynamics of the robot can not be exactly known, conventional adaptive feedforward compensation systems are suggested in [6], [7]. Besides, artificial neural networks are introduced to derive the feedforward torque because of their learning ability and universal approximating capability [8]–[10]. An alternative to the artificial neural network is the fuzzy system. It has been shown that the fuzzy systems are also universal approximators and capable of learning. Reference [11] tries to model each parameter in the manipulator dynamic equations with a set of fuzzy rules. However, since the robot dynamics are coupled and there are too many parameters to be modeled, the proposed method is very hard to implement in real systems. It is noticed that for a conventional fuzzy system, the number of fuzzy rules grows exponentially when the number of input variables increases. Therefore, it is very important to design a decentralized fuzzy controller for a multivariable coupled system.

This paper aims at building a position controller for robot manipulators, which not only exhibits strong robustness in the presence of a variety of uncertainties, but also is computationally very efficient. This is realized by proposing a decentralized adaptive fuzzy controller, which is composed of a feedforward fuzzy system that compensates the nonlinearity of the robot and a feedback fuzzy controller that is adaptive to the uncertainties in the robot system. Because a decentralized strategy is suggested and a simple input–output form is adopted, the number of fuzzy rules in the system is greatly reduced and thus the computational complexity of the algorithm is significantly simplified. The feedforward fuzzy systems are made more compact through an optimization of the rule structure. Since the PD gains are tuned on-line, it is not necessary to design them in advance and the closed-loop system keeps stable even in the presence of large uncertainties.

The remainder of the paper is organized as follows. Section II describes the off-line training and optimization of the feedforward fuzzy systems using genetic algorithms. This is followed by Section III that deals with the on-line tuning of the gains of the fuzzy PD controller using gradient method. In Section IV, the computational complexity of the controller is carefully considered and it is shown that the proposed controller is computationally very efficient. Simulation research is carried out in Section V, which focuses its attention on the performance of the controller in the presence of various parameter uncertainties, unmodeled nonlinear friction and unknown payloads. A summary of the significance of the paper is provided in Section VI.

Manuscript received September 4, 1995; revised October 1, 1996 and April 5, 1997.

The author is with Department of Electrical Engineering, Zhejiang University, Hangzhou 310027, China.

Publisher Item Identifier S 1083-4419(98)00217-9.

II. Off-Line Training and Optimization of the Feedforward Fuzzy Systems Via Genetic Algorithms

The successful application of the fuzzy control depends on the parameters and the structure of the concerned fuzzy rule system. However, they are usually decided upon by rule of thumb. Therefore, it is receiving increasing attention to build self-organizing fuzzy rule bases. By a self-organizing fuzzy system, we mean it should embrace the following two characteristics: 1) the fuzzy system is capable of setting up an optimal fuzzy rule structure so that the fuzzy rule base is complete, consistent and compact; and 2) the fuzzy system is able to automatically adjust the parameters both in condition part and consequence part of the rules. Until now, a lot of efforts have been made to realize such features. To provide the fuzzy system with the first function abovementioned, Yamaguchi *et al*. [11] and Nie *et al*. [12] use BAM and CPN neural network, respectively, whilst Nakamori *et al*. [13] select the clustering technique. As for the second function, backpropagation neural networks [14], pi–sigma neural network [15] and simulated annealing [16] are attempted for supervised parameter learning, while a neuron-like structure is used as reinforcement learning [17]. Karr [18] first introduces genetic algorithm into the self-learning of the fuzzy systems. Park *et al*. [19] apply GA to the optimization of fuzzy relation matrix and fuzzy membership functions. Generally, these two functions are realized separately.

This section discusses the training of the parameters and the optimization of the feedforward fuzzy systems based on genetic algorithms. We first describe the fuzzy model of the robot dynamics, and then apply the genetic algorithms to adjusting the parameters of the fuzzy model with a standard structure form. Further investigation shows that the trained fuzzy system has much room for improvement. To this end, we use the genetic algorithm to optimize the parameters and the structure of the rules simultaneously. In this way, the number of the fuzzy rules is further reduced.

A. Decentralized Fuzzy Modeling of a Robot Manipulator

Two main approaches are used by most researchers to derive the dynamic model of a manipulator—the Lagrange–Euler (L–E) and the Newton–Euler (N–E) formulations. From the control point of view, the L–E formulation is very desirable. For an n-link robot arm, the Lagrange equation of motion is as follows:

$$\tau_i = \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} C_{ijk}(q)\dot{q}_j\dot{q}_k + G_i(q), \qquad i = 1, 2, \ldots n \quad (1)$$

where τ_i is the torque exerted on joint i, q_i is the joint displacement, D_{ii} and D_{ij} are the effective inertia and coupling inertia, C_{ijk} stands for the centrifugal and Coriolis forces, and G_i is the gravity loading. It should be noticed that, for a planned trajectory, the desired torque depends not only on the trajectory, the geometric and inertia parameters of the link itself, but also on the parameters of other links and the payload at the end effector.

In order to model the dynamics of each link with a fuzzy system, it is necessary to choose proper input and output variables. For the sake of computational simplicity, it is necessary and feasible to select a noninteractive fuzzy system. In our case, only position and velocity are selected as two input variables and naturally the feedforward torque is selected as the output. Consequently, the fuzzy rules in the feedforward system are expressed in the following form:

$$\text{If } q^d(k) \text{ is } A_1^i \text{ and } \dot{q}^d(k) \text{ is } A_2^i, \text{ then } \tau \text{ is } \tau_0^i \quad (2)$$

where A_1^i and A_2^i are the fuzzy sets for q^d and $\dot{q}^d$, τ_0^i is the crisp output of each fuzzy rule and k is the time instant. Note that the premise variables do not appear in the consequence part of the rules, because it is found that they do not make much sense for improving the precision of the fuzzy model. What is worse, they sometimes complicate the algorithm seriously. According to Takagi and Sugeno [20], if the rule base has M rules altogether, the final output of the fuzzy model is calculated as follows:

$$\tau(k) = \frac{\sum_{i=1}^{M}\{w^i(k)\tau_0^i\}}{\sum_{i=1}^{M} w^i(k)} \quad (3)$$

$$w^i(k) = \min\{A_1^i(q^d(k)), A_2^i(\dot{q}^d(k))\}. \quad (4)$$

Given a set of input–output data, the premise and consequence parameters can be determined by use of a complex search algorithm and a recursive least-square algorithm [21].

No doubt, the performance of the fuzzy model is dependent on the structure and the parameters of the fuzzy rule base resulted from the learning procedure, which is the subject of the following part of this section.

B. Genetic Algorithm-Based Parameter Learning

Genetic algorithm (GA) is a stochastic optimization technique mimicking the natural selection, which consists of three main operations, namely, reproduction, crossover, and mutation [22]. For the implementation procedure, please refer to Fig. 1. Although genetic algorithms were developed a few decades ago, concrete theoretical analyses of the algorithm have not been provided until in the recent years. Reference [23] concludes that the canonical genetic algorithm can not always find the optimal solution within definite time. The so-called premature convergence is perhaps an illustration of this conclusion. Furthermore, the paper points out that if the chromosome with the best performance in each generation is reserved for the next generation, the algorithm will globally converge. Inspired by these conclusions, we introduce the following two measures to improve the convergence property of the genetic algorithm.

1) In reproduction, we stochastically introduce a randomly generated gene at a probability of R_h to replace one of the two parents selected for reproduction.
2) Select the best performed genes in the current population at a rate of R_e and place them directly in the next generation.

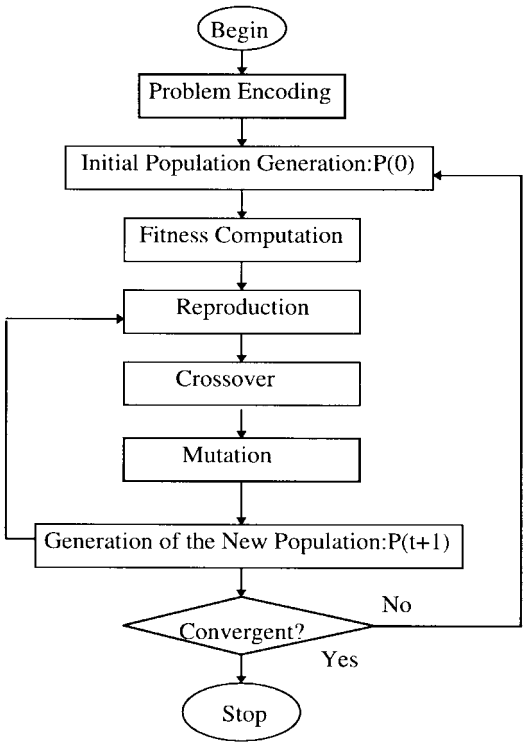

Fig. 1. Implementation flow chart of GA.

These two measures can be further explained with the results in [23]. If the reproduction is carried out in the traditional way, the best gene will probably be lost and thus the convergence can not be guaranteed. However, if we only adopt the second measure, the procedure of the evolution is no longer a Markov process and thus it does not satisfy the assumption of the convergence theory. Despite our successful application of these measures, mathematical analyses of them still lack. The parameter R_h and R_e are adjusted in the following way. At the beginning of the learning, R_h is relatively large and R_e is relatively small, and later, vice versa.

In order that the feedforward fuzzy system can realize the mapping of the robot inverse dynamics, the following quadratic form of performance index is established:

$$J = \sum_{k=0}^{P} [\tau^d(k) - \tau(k)]^2 \tag{5}$$

where $\tau^d(k)$ and $\tau(k)$ are the desired torque and torque computed from the feedforward fuzzy system, respectively, P is the number of training samples. Because the genetic algorithm endeavors to maximize the fitness function, and because our aim is to minimize the above performance index, the fitness function of each gene is calculated as follows:

$$F = \frac{1}{1+J} \tag{6}$$

where J is the performance index and 1 is introduced to prevent the fitness function from becoming infinitely large. Suppose the membership functions in fuzzy system (2) take a Gaussian form as

$$A_j^i(x) = \exp\left(-b_j^i(x - a_j^i)^2\right); \quad j = 1, 2 \tag{7}$$

where a_j^i and b_j^i are the center and width of the Gaussian function. For simplicity, the membership function (MF) in (7) is notated as (a_j^i, b_j^i).

The coding of the parameters to be adjusted can be arranged as follows if each variable is partitioned into n fuzzy subspaces:

$$a_1^1\ b_1^1\ \cdots\ a_1^n\ b_1^n\ a_2^1\ b_2^1\ \cdots\ a_2^n\ b_2^n\ \tau_0^1\ \tau_0^2\ \cdots\ \tau_0^M$$

where, $M = n^2$. In the above list, each element stands for a certain number of binary bits that encode the corresponding parameter. In order to reduce the dimension of the searching space, the whole length of each gene should be limited as short as possible. To this end, each parameter to be optimized is normalized to a certain range.

What the GA's are concerned, we can make the following observations.

1) The searching of the genetic algorithm starts from multiple initial states simultaneously and proceeds in all of the parameter subspaces in parallel, which provides GA an excellent parallel processing ability and an inherent global optimization capacity.
2) GA requires almost no prior knowledge of the concerned system, which enables it to deal with the completely unknown systems that other optimization methods may fail.
3) GA can not evaluate the performance of a system properly at one step. For this reason, it can generally not be used as an on-line optimization strategy and is more suitable for fuzzy modeling rather than for fuzzy control.

Without the loss of generality, we take a two-link rigid robot as an example for simulation. It should be pointed out that in training the feedforward fuzzy system, the algorithm does not require full knowledge of the robot inverse model because the optimization is completely data-driven. In practice, the training data can be obtained by experimentation or by establishment of an ideal model. By experimentation, we can exert a bounded random torque on the robot to be controlled. It may also be possible to derive an ideal mathematical model. This is theoretically feasible and helpful for training and checking of the fuzzy system, despite that the derived model is not the same as the real one. In computer simulation, we need a model to emulate the behavior of a robot to collect training data. The robot model used in simulation is shown in Section V, where $[m_1, m_2]^T = [2, 1]^T$, $[l_1, l_2]^T = [0.223, 0.2]^T$. At the training stage, no nonlinear friction and payloads are considered. The trajectory for the off-line training is as follows:

$$q_1^d(k) = 0.5\pi(1 - e^{-k}) \text{ (rad)} \tag{8}$$

$$q_2^d(k) = \pi(1 - e^{-k}) \text{ (rad)}. \tag{9}$$

At first, both input variables in each link are partitioned into four fuzzy subsets and thus 16 fuzzy rules in the standard form of (2) are set up for each link. Then, genetic algorithm is used to tune the parameters so that the fuzzy system can realize the mapping of the inverse robot dynamics. The population size of GA is 50 and the length of each gene is 160. The crossover and mutation probabilities are set to 0.96 and 0.1, respectively.

TABLE I
STANDARD RULE BASE OF LINK 1

IF		THEN	IF		THEN
q^d	$\dot{q}^d$	τ_0^i	q^d	$\dot{q}^d$	τ_0^i
(0,3.85)	(0.0625,5.85)	4.375	(1.06,4.35)	(0.0625,5.85)	2.75
(0,3.85)	(0.25,1.35)	6.5	(1.06,4.35)	(0.25,1.35)	5.75
(0,3.85)	(1.0625,5.6)	2.375	(1.06,4.35)	(1.0625,5.6)	5.625
(0,3.85)	(1.4375,7.6)	-0.875	(1.06,4.35)	(1.4375,7.6)	2.75
(0.25,3.6)	(0.0625,5.85)	2.5	(1.0,3.6)	(0.0625,5.85)	4.125
(0.25,3.6)	(0.25,1.35)	6.5	(1.0,3.6)	(0.25,1.35)	6.875
(0.25,3.6)	(1.0625,5.6)	3.375	(1.0,3.6)	(1.0625,5.6)	6.625
(0.25,3.6)	(1.4375,7.6)	-0.75	(1.0,3.6)	(1.4375,7.6)	5.0

TABLE II
STANDARD RULE BASE OF LINK 2

IF		THEN	IF		THEN
q^d	$\dot{q}^d$	τ_0^i	q^d	$\dot{q}^d$	τ_0^i
(0.375,5.85)	(0.219,2.35)	0.75	(2.25,7.6)	(0.219,2.35)	-0.6875
(0.375,5.85)	(1.25,5.1)	0.625	(2.25,7.6)	(1.25,5.1)	1.4375
(0.375,5.85)	(1.875,6.85)	1.5	(2.25,7.6)	(1.875,6.85)	1.6825
(0.375,5.85)	(2.843,2.35)	0.875	(2.25,7.6)	(2.843,2.35)	0.5
(1.0,4.6)	(0.219,2.35)	-1.25	(2.843,2.35)	(0.219,2.35)	-1.9375
(1.0,4.6)	(1.25,5.1)	1.5	(2.843,2.35)	(1.25,5.1)	-1.5
(1.0,4.6)	(1.875,6.85)	1.75	(2.843,2.35)	(1.875,6.85)	1.6875
(1.0,4.6)	(2.843,2.35)	1.5	(2.843,2.35)	(2.843,2.35)	-1.4375

After about 60 generations of learning, the GA searching process converges. The termination condition is decided by an average mapping error criterion determined in advance. The fuzzy models for link 1 and link 2 resulted from the best chromosome are shown in Tables I and II, respectively. For example, the first fuzzy rule in Table I is

$$\text{If } q^d \text{ is } (0, 3.85) \text{ and } \dot{q}^d \text{ is } (0.0625, 5.85), \text{ then } \tau \text{ is } 4.375.$$

Fig. 2 shows the approximating results of the fuzzy systems. The average approximating errors are 0.134 and 0.082, respectively.

If the performance of the fuzzy systems is only evaluated by the approximating precision, the above fuzzy systems with a standard structure are acceptable. However, we find in simulation that the average firing rate of the rules are very low. For example, on average only 50% and 22% of the fuzzy rules in the rule base of link 1 and link 2 are fired at each time instant. It indicates that the fuzzy systems are not compact enough and the structure of the fuzzy rules needs to be optimized.

C. Genetic Algorithm Based Structure Optimization and Parameter Learning

It is straightforward to optimize the structure and parameters of the fuzzy rules simultaneously using genetic algorithms. Each fuzzy system is represented as a string composed of two substrings. The first substring, which has the same form illustrated as in Section II-B, is to optimize the parameters of the fuzzy systems. The second substring encodes the structure of the fuzzy rule such that one integer number represents one membership function (MF) in the space of input variable in question. The MF's in the first substring are numbered in ascending order according to their centers. For example, a number "1" represents the MF with the lowest center. Since each variable is supposed to have at most four subspaces, the valid numbers in the second substring are 0, 1, 2, 3, and 4. The number "0" implies that this variable does not appear in the premise part of the rule. If both variable take a value of "0" in the second substring, then this rule is deleted from the rule base. It is also possible that more than one rule in the rule base has the same premise. In this case, only the rule that appears first is kept, so that the rules are consistent. An example of the second substring is given as follows:

$$\underbrace{3\,4}_{\text{rule 1}} \quad \underbrace{0\,1}_{\text{rule 2}} \cdots \underbrace{0\,0}_{\text{rule 16}}.$$

The corresponding fuzzy rules are:

R^1: If $q^d(k)$ is (a_1^3, b_1^3) and $\dot{q}^d$ is (a_2^4, b_2^4), then τ is τ_0^1
R^2: If $\dot{q}^d$ is (a_2^1, b_2^1), then τ is τ_0^2
$\vdots$
R^{16}: (Deleted).

In order to optimize the structure, the performance index in (5) is rewritten as

$$J = \sum_{k=0}^{P} [\tau^d(k) - \tau(k)]^2 + \lambda J_{\text{COM}} \tag{10}$$

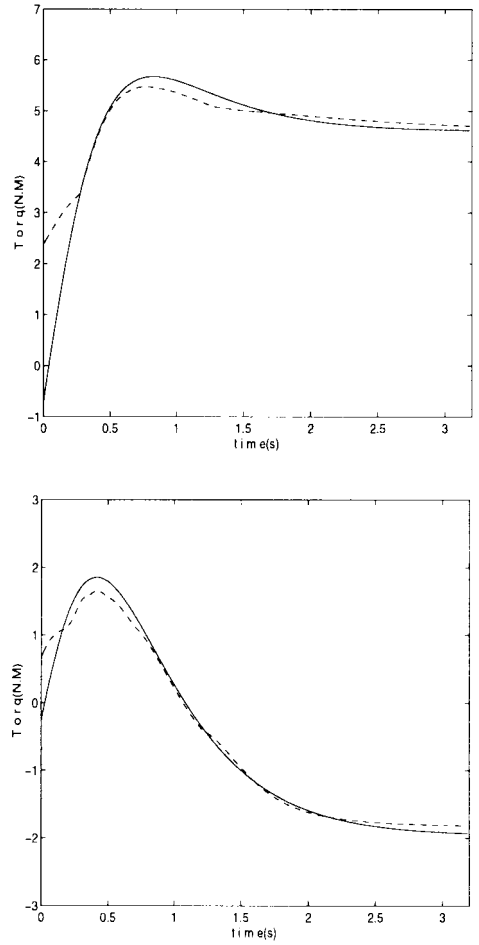

Fig. 2. Off-line training of the inverse dynamics (without structure optimization).

where λ is the weighting constant, $J_{\rm COM}$ is the penalty for model complexity and is expressed as:

$$J_{\rm COM} = \frac{\text{the total number of rules in the rule base}}{\text{the average number of the fired rules}}. \quad (11)$$

The value of λ is set to 0.1 for link 1 and 0.25 for link 2. We suppose a rule is fired when the membership grad is greater than 0.05. In case no rules are fired or there are no rules in the rule base, $J_{\rm COM}$ will be set to a very large value.

The simulation results are inspiring. The optimized rule bases for link 1 and link 2 have eight and 11 rules, respectively, and the firing rates are raised to about 75% and 32%, respectively. The rule bases for the two links are listed in Tables III and IV, and the approximating results are demonstrated in Fig. 3. The average approximating errors are 0.132 and 0.122, respectively. We see that the approximating errors are quite satisfying, although the number of the fuzzy rules are reduced.

III. Design of the Adaptive Feedback Fuzzy PD Controllers

If the feedforward fuzzy system compensates the nonlinear dynamics of the robot to a certain accuracy, the closed-loop system can be regulated by a PD controller with fixed gains. However, if the uncertainties in the robot system increase, the performance of the controller deteriorates seriously. In order to solve this problem, an adaptive fuzzy feedback controller is suggested in this section. The feedback fuzzy controller has a PD-like structure, and is able to adjust its parameters when the uncertainties in the robot system vary. The adaptation mechanism is driven by the gradient method based on a quadratic performance index that is widely adopted in optimal control. Fortunately, the algorithm depends on neither the inverse dynamics nor the full perturbation model of the robot. Paper [24] shows that the gradient method based learning algorithm can work just if the sensitive model of the controlled system is available. Moreover, the sensitive model can be replaced with its sign [25].

This section first derives the decentralized perturbation model of each link to show how the sensitive model can be obtained. Then the learning algorithm for the PD gains is presented.

A. Decentralized Perturbation Model of the Robot

The dynamics of a robot in (1) can be expressed in the following form [26]:

$$\dot{x}(t) = f(x(t), u(t)) \quad (12)$$

where $x = [q, \dot{q}]^T \in R^{2n}$, $u = \tau \in R^n$, $q, \dot{q} \in R^n$ are the position and velocity vectors, respectively, and n is the number of the degree of the freedom. With this formulation, we intended to find a feedback control law such that the closed-loop system is asymptotically stable and tracks the desired trajectory as close as possible.

Since the inverse dynamics are learned by the feedforward fuzzy systems, the desired torques can be computed from the fuzzy models. These computed torques can be treated as nominal torque values. Because of the approximating errors of the feedforward fuzzy systems and because the working conditions of the robot varies, there exists joint errors if only the feedforward torques are exerted on the robot. Nevertheless, if the errors between the computed torques and the real nominal torques are small, then the joint errors will be small and the dynamic equations of the robot can be expanded in the vicinity of the desired trajectory set points to obtain the associated perturbation equations.

Given the desired trajectory and provided the feedforward torques are acceptably accurate, (12) can be linearized along the nominal trajectory

$$\delta\dot{x}(t) = \nabla_x f|_{x=x^d}\delta x(t) + \nabla_u f|_{x=x^d}\delta u(t) \quad (13)$$

where $\nabla_x f|_{x=x^d}$ and $\nabla_u f|_{x=x^d}$ are the gradients of $f(x,u)$ evaluated at x^d and u^d, respectively, $\delta x(t) = x(t) - x^d$, $\delta u(t) = u(t) - u^d$, and x^d and u^d are the nominal values of x and u. Let $A(t) = \nabla_x f|_{x=x^d}$ and $B = \nabla_u f|_{x=x^d}$, then we have the following perturbation equation for the robot system:

$$\delta\dot{x}(t) = A(t)\delta x(t) + B(t)\delta u(t). \quad (14)$$

As a result, the control problem reduces to producing a proper $\delta u(t)$ so that $\delta x(t)$ converges to zero. Rewrite (14) in the

TABLE III
THE OPTIMIZED FUZZY RULE BASE OF LINK 1

IF		THEN	IF		THEN
q^d	$\dot{q}^d$	τ_0^i	q^d	$\dot{q}^d$	τ_0^i
(1.75,5.85)	(0.375,0.35)	6.875	(1.375,5.85)	(0.125,5.6)	6.25
(1.0625,5.85)	(0.125,5.6)	6.375	(0.4375,5.1)	(0.125,5.6)	1.5
(0.4375,5.1)	(1.0,7.85)	-0.125	(0.4375,5.1)	(1.25,4.1)	5.75
(1.0625,5.85)	(0.375,0.35)	6.75	(0.4375,5.1)		6.625

TABLE IV
THE OPTIMIZED FUZZY RULE BASE OF LINK 2

IF		THEN	IF		THEN
q^d	$\dot{q}^d$	τ_0^i	q^d	$\dot{q}^d$	τ_0^i
(2.8125,4.35)	(0.1563,5.6)	-1.875	(0.2188,7.35)	(1.375,1.85)	-0.0625
(1.4375,2.1)	(1.375,1.85)	1.375	(2,8125,4.35)	(1.9375,3.6)	0.4375
(1.625,5.1)	(2.5313,1.35)	-1.75	(2.8125,4.35)	(2.5313,1.35)	0.8125
(1.625,5.1)	(1.9375,3.6)	-1.25	(1.4375,2.1)	(1.9375,3.6)	-1.5
(1.4375,2.1)	(0.1563,1.65)	1.625	(1.4375,2.1)		-1.25
	(0.1563,1.65)	0.5625			

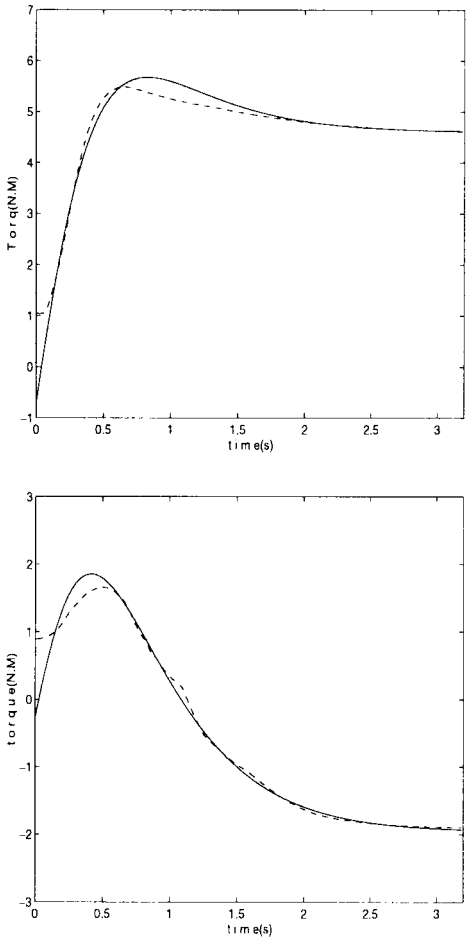

Fig. 3. Off-line training of the inverse dynamics (with structure optimization).

discrete-time form and let $\delta u(t) = u(t)$ and $\delta x(t) = x(t)$, then we have

$$x(k+1) = A(k)x(k) + B(k)u(k) \tag{15}$$

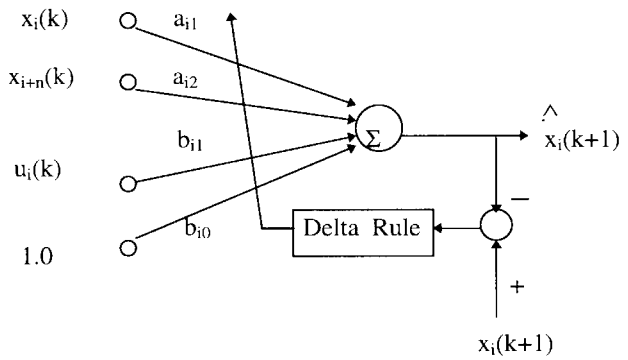

Fig. 4. Neural network structure for identification of the perturbation model.

where $A(k)$ and $B(k)$ are $2n \times 2n$ and $2n \times n$ matrices, respectively. However, this linear perturbation model is still coupled among the links. To make the proposed controllers completely noninteractive, we rewrite (15) in the form of n decentralized subsystems

$$x_i(k+1) = a_{i1}x_i(k) + a_{i2}x_{i+n}(k) + b_{i1}u_i(k) + b_{i0} \tag{16}$$

where, b_{i0} accounts for the coupled terms, i denotes the ith link of the robot. Since the inverse dynamics are unknown, the above model can not be obtained analytically. Usually it can be identified by the recursive least-square algorithm. However, [27] suggests that a linear system can be expressed by a linear two-layer neural network (see Fig. 4) and its parameters can be estimated efficiently using δ-rule. For example, the adaptation algorithm for the sensitive model parameter b_{i1} has the following form:

$$\Delta b_{i1}(k) = \xi u_i(k)\delta_i(k) \tag{17}$$

where $\delta_i(k) = x_i(k+1) - \hat{x}_i(k+1)$, $\hat{x}_i(k+1)$ is the predicted value of $x_i(k+1)$, and ξ is the learning constant. It is obvious that this type of learning algorithm is much simpler than that of the least-square method.

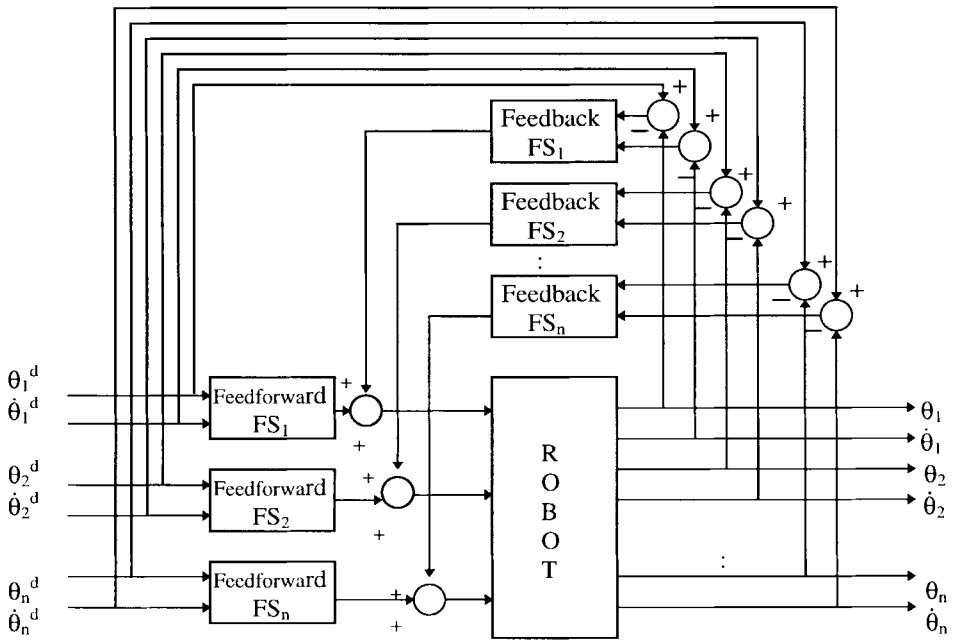

Fig. 5. Diagram of the proposed decentralized adaptive fuzzy controller.

B. Design of the Feedback Controller

The fuzzy rules in the feedback fuzzy controller appear as follows:

$$\text{If } x_i(k) \text{ is } B_1^j \text{ and } x_{i+n}(k) \text{ is } B_2^j, \quad \text{then } u_i(k) = K_p^j x_i + K_d^j x_{i+n}. \tag{18}$$

Obviously, it is in the form of a PD controller. Consequently, the final output of the feedback controller is given by

$$u_i = \frac{\sum_{j=1}^{N} \{v^j (K_p^j x_i + K_d^j x_{i+n}^j)\}}{\sum_{j=1}^{N} v^j} \tag{19}$$

$$v^j = \min\{B_1^j(x_i(k)), B_2^j(x_{i+n}(k))\} \tag{20}$$

where N is the number of the feedback fuzzy rules, B_1 and B_2 are the fuzzy subsets defined on the universe of x_i and x_{i+n}. The fuzzy PD gains are adjusted to minimize the following quadratic performance index:

$$J_i = \frac{1}{2}\{r_1[x_i(k)]^2 + r_2[u_i(k)]^2\} \tag{21}$$

where r_1 and r_2 are weighting constants. According to the gradient method, the learning algorithm of the parameters in the feedback fuzzy system can be derived as follows:

$$\Delta K_p^j = -\frac{\partial J_i}{\partial K_p^j} = -\frac{\partial J_i}{\partial u_i}\frac{\partial u_i}{\partial K_p^j} \tag{22}$$

$$= -\left\{r_1 x_i \frac{\partial x_i}{\partial u_i} + r_2 u_i\right\} v^j x_i \Big/ \sum_{j=1}^{N} v^j. \tag{23}$$

In (23), the sensitive model can be derived from (16). Therefore, we have

$$\Delta K_p^j = -\{r_1 x_i b_{i1} + r_2 u_i\} v^j x_i \Big/ \sum_{j=1}^{N} v^j. \tag{24}$$

Similarly, K_d^j can be adjusted by the following algorithm:

$$\Delta K_d^j = -\{r_1 x_i b_{i1} + r_2 u_i\} v^j x_{i+n} \Big/ \sum_{j=1}^{N} v^j. \tag{25}$$

How to compute the sensitive model of the controlled plant is worth discussing. If the system is known, it is very easy to obtain such a model. When the system is unknown and if the concerned system is SISO, then there are three choices proposed by Psaltis *et al*. [24], Saerens *et al*. [25], and Wu *et al*. [28], respectively. In the MIMO situation, Jin *et al*. [29] use the approximating reasoning to obtain the partial derivatives. It is argued that in some cases, the use of sensitive model is better than the use of its sign. This is the reason why we try to identify the sensitive model instead of utilizing its sign. The closed-loop system is shown in Fig. 5.

IV. Computational Complexity

Whether a designed controller is practical or not depends greatly on its computational complexity because the computing capacity of a low-cost microprocessor is limited. This section provides the complexity of the feedforward and feedback computation of the controller proposed in this paper. Torque computing methods based on robot inverse dynamics and the fuzzy feedforward system are compared. The computational complexity of the feedback controller is also compared with that of the adaptive controller proposed in [26]. We show that the proposed controller is computationally very efficient.

Generally speaking, the computational burden can be evaluated in terms of required mathematical multiplication and addition operations. The controller developed in this paper consists of a feedforward torque compensation system and a feedback fuzzy PD regulator. The computation of the feedforward fuzzy system has three stages: computation of the membership functions, computation of the contribution of each

TABLE V
COMPUTATIONAL COMPLEXITY OF THE FEEDFORWARD SYSTEM

	Inverse Dynamics	Standard Fuzzy System	Optimized Fuzzy Syetem
Addition	$117n-24$	$54n$	66
Multiplication	$103n-21$	$33n$	53

TABLE VI
COMPUTATIONAL COMPLEXITY OF THE FEEDBACK CONTROLLER

	Optimal Feedback Controller	Feedback Fuzzy PD Controller
Addition	$8n^3+34.5n^2-2.5n$	$37n$
Multiplication	$8n^3+37n^2+10n+3$	$43n$

TABLE VII
TRACKING ERRORS DUE TO PARAMETER UNCERTAINTIES (IN RADIANS)

	Link 1			Link 2		
Percentage of Error in Link Length and Mass	Average	Maximum	Final	Average	Maximum	Final
a)10%, 10%	0.0024	0.0054	0.0017	0.0006	0.0019	0.0007
b)15%,50%	0.0058	0.0081	0.0052	0.0022	0.0045	0.0030
c)20%, 100%	0.0099	0.0141	0.0093	0.0052	0.0094	0.0071

rule and computation of the final output of the fuzzy system. The results are provided in Table V, where n is the freedom of the manipulator. For the standard fuzzy system, each variable is supposed to have at most four subsets and therefore there are eight fuzzy membership functions involved for each link. For the optimized fuzzy system, we list the total number of the addition and multiplication operations of the two fuzzy systems obtained in Section II. Clearly, the computation of the optimized fuzzy system is simpler compared with that of the standard fuzzy system. It should be clarified that each minimum operation is treated as one addition operation in this paper. Table V denotes that the computation burden of the proposed fuzzy torque computing system is significantly reduced compared with the conventional torque computing method, especially when the freedom of the robot increases.

The computation of the adaptive feedback fuzzy controller can also be divided into three parts: computation of control effect fanned out by each feedback fuzzy system, computation of identification of the sensitive model and computation of adaptation of the PD gains. In this paper, the error and change of error are both partitioned into three subspaces and the standard fuzzy structure is adopted. That is to say, there are nine rules in the feedback fuzzy rule base for each link. As a comparison, we list the computational complexity of our scheme together with the adaptive feedback control proposed in [26] (see Table VI). We make this comparison because the feedback controller in [26] has essentially a PD structure. The differences reside in the fact that, paper [26] identifies the coupled perturbation model in (15) using a recursive least-square method, while we identify the decentralized model in (16) using a simple neural network.

V. SIMULATION RESULTS

The purpose of the simulation is to investigate the robustness of the proposed controller. The robot system considered in simulation is a two-link rigid robot, and its inverse dynamic model is expressed as follows:

$$\begin{aligned}\tau_1 &= \left[(m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2\cos(q_2)\right]\ddot{q}_1 \\ &\quad+\left[m_2l_2^2+m_2l_1l_2\cos(q_2)\right]\ddot{q}_2-2m_2l_1l_2\sin(q_2)\dot{q}_1\dot{q}_2 \\ &\quad-m_2l_1l_2\sin(q_2)\dot{q}_2^2+(m_1+m_2)gl_1\cos(q_1) \\ &\quad+m_2gl_2\cos(q_1+q_2) \\ \tau_2 &= \left[m_2l_2^2+m_2l_1l_2\cos(q_2)\right]\ddot{q}_1+m_2l_2^2\ddot{q}_2 \\ &\quad+m_2l_1l_2\sin(q_2)\dot{q}_1^2+m_2gl_2\cos(q_1+q_2)\end{aligned}$$

where m_1 and m_2 are the mass of each link, l_1 and l_2 are the length, and g is the gravity. In order to observe how the controller behaves in presence of various uncertainties, three types of uncertainties are considered, namely, parameter variations, unmodeled nonlinear friction and unknown payloads.

A. Parameter Variations

By parameter variations, we mean here the mass and length errors of the links. At the training stage, we suppose $[m_1,m_2]=[2,1]$, $[l_1,l_2]=[0.223,0.2]$. In this section, three cases are taken into account. The parameter errors, the maximum, average and final joint tracking errors are listed in Table VII. Fig. 6 shows the position tracking errors in the three cases.

B. Unmodeled Friction

At the off-line training stage of our simulation, we obtain the training samples from the robot model, which does not consider the nonlinear friction. In order to examine the performances of the controller in the presence of unmodeled nonlinear friction, the following unmodeled nonlinear friction is added at the control stage:

$$f_i=f_{q_i}(\dot{q}_i,\tau_i)+f_{v_i}(\dot{q}_i)\quad(i=1,2) \tag{26}$$

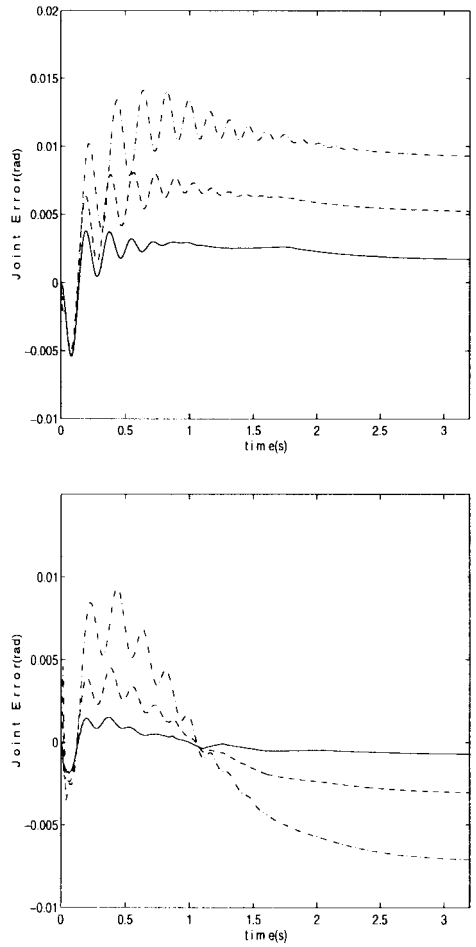

Fig. 6. Position tracking errors in the presence of parameter uncertainties.

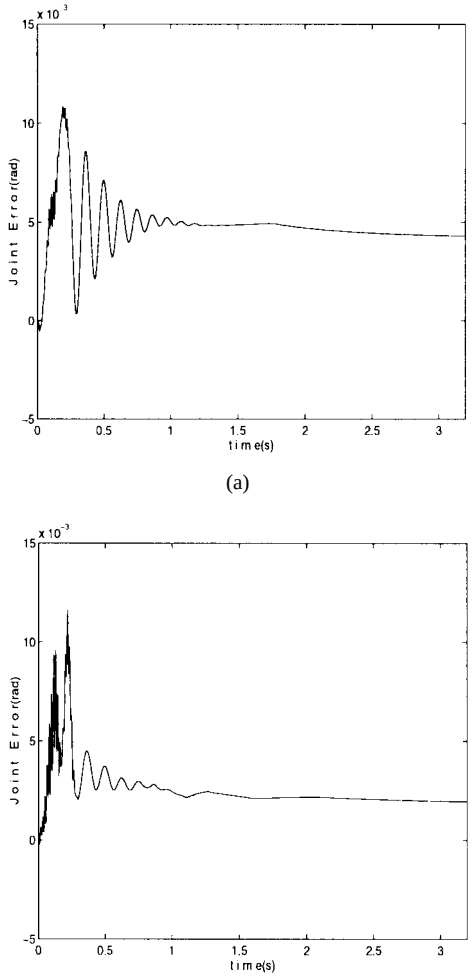

Fig. 7. Position tracking errors in the presence of unmodeled nonlinear friction. (a) Link 1, (b) Link 2.

where f_{q_i} and f_{v_i} are the Columbus and viscous friction, respectively, which can be expressed by

$$f_{q_i}(\dot{q}_i, \tau_i) = \begin{cases} k_i \operatorname{sgn}(\dot{q}), & |\dot{q}_i| > 0 \\ k_i \operatorname{sgn}(\tau_i), & |\dot{q}_i| = 0, \quad |\tau_i| > k_i \\ \tau_i, & |\dot{q}_i| = 0, \quad |\tau_i| < k_i \end{cases} \tag{27}$$

$$f_{v_i}(\dot{q}_i) = C_i \dot{q}_i \tag{28}$$

where, k_i and C_i are constants. In simulation, we set $[k_1, k_2] = [2.5, 1.5]$, $[C_1, C_2] = [0.1, 0.1]$. The tracking errors of link 1 and link 2 are given in Fig. 7. The final errors of the two links are 0.0043 (rad) and 0.0019 (rad), respectively.

C. Unknown Payloads

In robot systems, the unknown payload is one of the major dynamic uncertainties. Compared with the parameter uncertainties and unmodeled friction, the influence of unknown payload is much greater. In simulation, the robot has a payload of 1 kg, 2 kg, and 3 kg, respectively, which are supposed to be unknown. The simulation results are provided in Fig. 8 and Table VIII.

From the simulation results provided above, we draw the conclusion that the proposed controller is quite insensitive to various uncertainties in the robot dynamics. However, if the uncertainties become unreasonably large, the controller will collapse. This can be explained that under such situations, the torques computed from the trained feedforward fuzzy systems are no longer near the nominal torques, and consequently, (13) does not hold.

In order to discuss the relationship between the final values of K_p^i and K_d^i and the amount of uncertainties in the system, Table IX provides the learning results of the fuzzy PD gains when the unknown payloads are 1.0 kg and 3.0 kg, respectively. It demonstrates that the fuzzy controller can vary its parameters properly with the amount of the uncertainty. Remember that in the feedback fuzzy systems, the number of the fuzzy subsets for x_i and x_{i+2} $(i = 1, 2)$ in (18) are both set to three, namely P for positive, Z for zero and N for negative.

VI. Conclusion

In this paper, a decentralized fuzzy control scheme for robot manipulators is developed. The controller for each link has a feedforward fuzzy torque computing system and an adaptive feedback controller. Due to the simple structure of the fuzzy systems, the on-line computational burden for nonlinear feedforward compensation is greatly relaxed. Genetic algorithm is applied to fuzzy system training because it is fully data-driven and is able to optimize the structure of the fuzzy system

TABLE VIII
TRACKING ERRORS DUE TO UNKNOWN PAYLOADS (IN RADIANS)

	Link 1			Link 2		
Unknown Payload	Average	Maximum	Final	Average	Maximum	Final
a)1.0kg	0.0038	0.0112	0.0000	0.0037	0.0064	0.0053
b)2.0kg	0.0048	0.0176	0.0006	0.0049	0.0096	0.0069
c)3.0kg	0.0059	0.0207	0.0009	0.0057	0.0106	0.0083

TABLE IX
FUZZY PD GAIN ADAPTATION IN THE PRESENCE OF UNKNOWN PAYLOADS. (†: THE UNKNOWN PAYLOAD IS 1.0 kg; ‡: THE UNKNOWN PAYLOAD IS 3.0 kg)

Link 1						Link 2					
Premise		PD Gains†		PD Gains‡		Premise		PD Gains†		PD Gains‡	
x_1	x_3	K_p	K_d	K_p	K_d	x_2	x_4	K_p	K_d	K_p	K_d
P	P	68.6	2.9	135.6	4.6	P	P	3.2	0.1	188.5	4.5
P	ZO	78.4	4.5	172.9	4.5	P	ZO	3.2	0.1	185.5	4.5
P	N	72.1	6.7	140.8	4.4	P	N	89.2	3.6	202.4	4.5
ZO	P	72.6	1.8	189.5	4.7	ZO	P	96.9	3.1	248.8	4.6
ZO	ZO	529.4	3.2	902.6	4.4	ZO	ZO	91.9	3.1	949.6	4.5
ZO	N	81.9	10.7	222.6	4.1	ZO	N	679.8	3.5	275.9	4.4
N	P	65.2	3.4	120.5	4.5	N	P	100.8	3.1	185.4	4.5
N	ZO	70.2	4.6	141.4	4.5	N	ZO	87.5	3.6	175.4	4.5
N	N	67.4	6.3	124.6	4.4	N	N	92.2	3.1	195.1	4.4

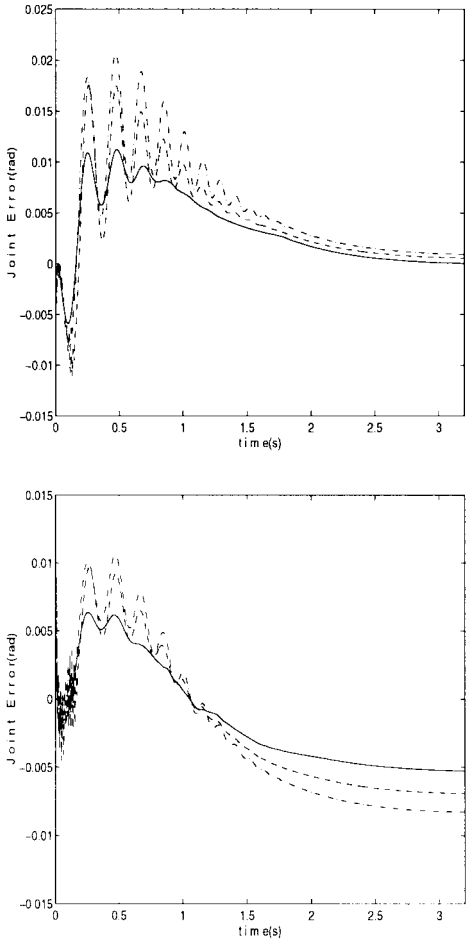

Fig. 8. Position tracking errors in the presence of unknown payloads.

simultaneously. The training samples can be collected by doing experiments or by establishing an ideal model. We believe that such a model is theoretically available and is helpful in training and checking the fuzzy systems. We have demonstrated that the proposed controller works quite well, even if the ideal model is not so in concordance with the real inverse dynamics. The feedback controller is also composed of a set of fuzzy rules. The rules have a PD-like structure and their gains are tuned on-line based on gradient method. The computational complexity is greatly reduced compared with that of a self-tuning controller. Various simulation results prove that the proposed controller is effective.

ACKNOWLEDGMENT

The author would like to thank the editor and reviewers for their inspiring encouragement and constructive comments, which have contributed much to the improvement of the clarity and presentation of this paper. He would also like to thank Prof. J. P. Jiang, Department of Electrical Engineering, Zhejiang University, and Prof. W. von Seelen, Institut für Neuroinformatik, Ruhr-Universität Bochum, for their support at different stages of this paper.

REFERENCES

[1] C. H. An, C. G. Atkeson, and T. M. Hollerbach, *Model-Based Control of Robot Manipulators*. Cambridge, MA: MIT Press, 1988.
[2] T. J. Tarn, A. K. Bejczy, G. T. Marth, and A. K. Ramadori, "Performance comparison of four manipulator servo controllers," *IEEE Contr. Syst. Mag.*, vol. 13, pp. 22–29, 1993.
[3] J. T. Wen and D. S. Bayard, "New class of control law for robotic manipulators, Part 1: Nonadaptive case," *Int. J. Contr.*, vol. 47, no. 5, pp. 1361–1385, 1988.
[4] J. T. Wen, "A unified perspective on robot: The energy Lyapunov function approach," *Int. J. Adaptive Contr. Signal Process.*, vol. 4, no. 6, pp. 487–500, 1990.
[5] R. Kelly and R. Salgado, "PD control with computed feedforward of robot manipulators: A design procedure," *IEEE Trans. Robot. Automat.*, vol. 10, no. 4, pp. 566–571, 1994.
[6] H. Serji, "A new approach to adaptive control of manipulators," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, vol. 109, pp. 193–201, 1987.
[7] A. A. Goldenberg, J. A. Apkarian, and H. W. Smith, "An approach to adaptive control of robot manipulators using computed torque tech-

nique," *Trans. ASME, J. Dynamic Syst., Meas. Contr.*, vol. 111, p. 8, 1989.
[8] W. Miller, "Real-time application of neural network for sensor based control of robots with vision," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 5, pp. 825–831, 1989.
[9] M. Kawato, M. Uno, M. Isobe, and R. Suzuki, "Hierachical neural network model for voluntary movement with application to robotics," *IEEE Contr. Syst. Mag.*, vol. 8, no. 2, pp. 8–15, 1988.
[10] R. Carelli, E. F. Camaho, and D. Patino, "A neural network based feedforward adaptive control for robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 1281–1287, Jan. 1995.
[11] T. Yamaguchi, T. Takagi, and T. Mita, "Self-organizing control using fuzzy neural networks," *Int. J. Contr.*, vol. 56, no. 2, pp. 415–439, 1992.
[12] J. Nie and D. A. Linkens, "Fast self-learning multivariable fuzzy controllers constructed from a modified CPN network," *Int. J. Contr.*, vol. 60, no. 3, pp. 369–393, 1994.
[13] Y. Nakamori and M. Ryoke, "Identification of fuzzy prediction models through hyperellipsoidal clustering," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 1153–1173, Aug. 1994.
[14] C. T. Lin and C. S. G. Lee, "Neural network based fuzzy logic control and decision systems," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, 1991.
[15] Y. C. Jin, J. P. Jiang, and J. Zhu, "Neural network based fuzzy identification with application to modeling and control of complex systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 990–997, June 1995.
[16] S. Isaka and A. V. Sebald, "An optimization approach for fuzzy controller design," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1469–1472, Nov./Dec. 1992.
[17] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy controllers through reinforcement," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 724–739, 1992.
[18] C. Karr, "Applying genetics to fuzzy logic," *IEEE AI Expert*, vol. 6, pp. 26–33, 1992.
[19] D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 39–47, Jan. 1994.
[20] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. 15, no. 1, pp. 116–132, 1985.
[21] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its application to prediction of complex systems," *Fuzzy Sets Syst.*, vol. 42, pp. 315–324, 1994.
[22] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[23] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 96–101, 1994.
[24] D. Psaltis, A. Sideris, and A. Yamakura, "Neural controllers," in *Proc. 1st Int. Conf. Neural Networks*, 1987, vol. 4, pp. 551–558.
[25] M. Saerens and A. Soquet, "A neural controller," in *Proc. 1st IEEE Conf. Artificial Neural Networks*, 1989, pp. 211–215.
[26] C. S. G. Lee and M. J. Chung, "An adaptive control strategy for computer based manipulators," in *Proc. 21st IEEE Conf. Decision Control*, 1982, pp. 95–100.
[27] Y. Takahashi, "Neural network adaptive control," *Meas. Contr.*, vol. 29, no. 8, pp. 35–39, 1990 (in Japanese).
[28] Q. Wu, B. W. Hogg, and C. W. Irwen, "A neural network for turbogenerators," *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 95–100, 1992.
[29] Y. C. Jin, J. P. Jiang, and J. Zhu, "State estimation and adaptive control of mutivariable systems via neural network and fuzzy logic," *AMSE Advances Modeling Anal.*, vol. 43, no. 2, pp. 15–22, 1994.

Yaochu Jin was born in Wujiang, Jiangsu Province, China, on August 31, 1966. He received the B.Sc. and M.Sc. degrees in electrical engineering from Zhejiang University, Hangzhou, China, in 1988 and 1990, respectively.

He joined the Electrical Engineering Department, Zhejiang University, as a Teaching Assistant in 1991. From 1993 to 1996, he was a Lecturer and offered a course of Intelligent Control for graduate students in the Electrical Engineering Department. Since November 1996, he has been an Associate Professor. Currently, he is on leave with the Institut für Neuroinformatik, Ruhr-Universität Bochum. He has published more than 20 technical papers and coauthored two books and one translation. His research interests include fuzzy and neural systems, rule-based knowledge extraction, evolutionary computation and their application to modeling and control of complex systems.

Mr. Jin is a member of the Chinese Society of Electrical Engineering, Chinese Society of Automation, and Association of Science and Technology of Zhejiang Province. He received a Science and Technology Progress Award from the State Education Commission of China and Education Commission of Zhejiang Province in 1995 and 1996, respectively.