Alexandria Engineering Journal (2013) 52, 583–594

FACULTY OF ENGINEERING
ALEXANDRIA UNIVERSITY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

# A computationally efficient fuzzy control scheme for a class of MIMO systems

**Abdel Badie Sharkawy** *

*Mechanical Engineering Department, Faculty of Engineering, Assiut University, 71516 Assiut, Egypt*

Received 22 November 2011; revised 9 July 2013; accepted 27 July 2013
Available online 8 September 2013

**KEYWORDS**

Robot manipulators;
Genetic algorithm (GA);
Feedforward fuzzy torque computing;
Fuzzy PD feedback control;
Closed-loop stability;
Computational complexity;
Parametric and payload uncertainties

**Abstract** This paper develops a decentralized fuzzy control scheme for MIMO nonlinear second order systems with application to robot manipulators via a combination of genetic algorithms (GAs) and fuzzy systems. The controller for each degree of freedom (DOF) consists of a feedforward fuzzy torque computing system and a feedback fuzzy PD system. The feedforward fuzzy system is trained and optimized off-line using GAs, whereas not only the parameters but also the structure of the fuzzy system is optimized. The feedback fuzzy PD system, on the other hand, is used to keep the closed-loop stable. The rule base consists of only four rules per each DOF. Furthermore, the fuzzy feedback system is decentralized and simplified leading to a computationally efficient control scheme. The proposed control scheme has the following advantages: (1) it needs no exact dynamics of the system and the computation is time-saving because of the simple structure of the fuzzy systems and (2) the controller is robust against various parameters and payload uncertainties. The computational complexity of the proposed control scheme has been analyzed and compared with previous works. Computer simulations show that this controller is effective in achieving the control goals.



## 1. Introduction

In many practical applications where high performance trajectory tracking is required, the control scheme in Fig. 1 is commonly used to enable the independent design of the feedforward and the feedback control [1].

The feedforward control $u_{FF}$ is applied to achieve the desired tracking performance of the output $\theta$, whereas the feedback control is designed such that the system $\Sigma$ is appropriately stabilized and robustified against model uncertainties. In comparison with the broad spectrum of available design methods for feedback control, only few methods are known for a systematical feedforward control design, which forms a contrast to the respective demand in industry. The reason for this methodological gab is related to the system inversion required in the course of the feedforward control design and to the respective difficulties arising with nonlinear systems

* Tel.: +20 1226840350; fax: +20 882335572.
E-mail address: ab.shark@aun.edu.eg.

Peer review under responsibility of Faculty of Engineering, Alexandria University.

Production and hosting by Elsevier

1110-0168 © 2013 Production and hosting by Elsevier B.V. on behalf of Faculty of Engineering, Alexandria University.
http://dx.doi.org/10.1016/j.aej.2013.07.008

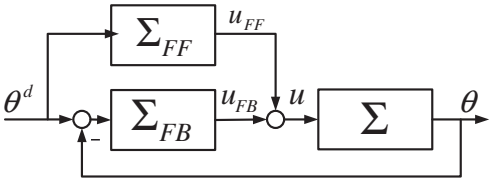


**Figure 1** Structure of the control scheme with system $\Sigma$, feedback control $\Sigma_{FB}$, and feedforward control $\Sigma_{FF}$.

[2]. Feedforward can also be made from disturbances [3,4], but this problem is different from feedforward from the set-point, and it is not treated in this paper.

Generally speaking, multiple-input multiple-output (MIMO) systems usually have characteristics of nonlinear dynamics coupling. Therefore, the difficulty in controlling MIMO systems is how to overcome the coupling effects between the degrees of freedom. The computational burden and dynamic uncertainty associated with MIMO systems make model-based decoupling impractical for real-time control. Adaptive control has been studied for many decades to deal with constant or slowly changing unknown parameters. Applications include manipulators, ship steering, aircraft control, and process control. Although the perfect knowledge of the inertia parameters can be relaxed via adaptive technique, its real practical usefulness is not really clear and the obtained controllers may be too complicated to be easily implemented [5]. Also, because many design parameters, like learning rates and initialization of the parameters to be adapted, have to be considered in controller construction, most existing methodologies have limitations. Moreover, owing to the different characteristics among design parameters, attaining a complete learning, while considering an overall performance goal, is an extremely difficult task.

Fuzzy controllers have demonstrated excellent robustness in both simulations and real-life applications [6]. They are able to function well even when the controlled system differs from the system model used by the designer. A customary for this phenomenon is that fuzzy sets, with their gradual membership property, are less sensitive to errors than crisp sets. Another explanation is that a design based on the "computing with words" paradigm is inherently robust; the designer forsakes some mathematical rigor but gains a very general model which remains valid even when the system parameters and structure vary.

However, it has been proved that standard fuzzy logic controllers are not suitable for loop controllers [7]. This fact is referred to that there are many tuning parameters in membership functions and control rules. Furthermore, standard fuzzy logic controller has a long computation time since it performs fuzzification, inference, and defuzzification processes in determining control inputs. Thus, it is difficult for control inputs of standard fuzzy logic control to be computed within the sampling time of a loop controller. For this reason, complexity reduction of fuzzy feedback controllers was the topic of many researchers [7,8].

In this paper, we focus on the design of appropriate fuzzy systems in feedforward and feedback paths. In the feedforward path, the capabilities of GAs are used off-line to determine the optimal parameters and structure of fuzzy systems which can approximate the inverse dynamics of the system. No mathematical model is needed. In the feedback path, a stable fuzzy feedback controller is designed based on the Lyapunov synthesis. Only four rules constitute the rule base for each DOF. Furthermore, the fuzzy feedback controller is decentralized and simplified leading to a computationally efficient fuzzy control scheme. A primary version of this feedback controller has been introduced in [9] by the author of this paper. In this paper, we revisit it and design an adaptive mechanism to determine its gains adaptively. To demonstrate the proposed approach, we use the example of robotics because it is a well-known example of nonlinear MIMO second order systems.

The paper is outlined as follows: in Section 2, the robot model and the nominal value of its parameter are introduced. This model is used to generate simulation data instead of experimental data from real robot platform. Section 3 explains the fuzzy models of the inverse dynamics of the robot. The models are two input one output fuzzy systems. They are used in the feedforward path. In Section 4, we explain how GAs can be used off-line to optimally determine parameters and structure of the fuzzy systems. In Section 5, the fuzzy feedback controller is derived based on the Lyapunov direct method. Furthermore, the controller is simplified, i.e., it has a closed form mathematical relation with only three parameters need to be tuned and the controller gain is adaptively determined online so as to minimize a performance index. Section 6 discusses the computational complexity of the proposed control scheme in comparison with previous works. Simulation results are demonstrated in Section 7. Finally, some concluding remarks are given in Section 8.

## 2. Robot modeling and the control statement

Without the loss of generality, we take the two-link rigid robot shown in Fig. 2, as an example to demonstrate the proposed control scheme. The inverse dynamic model is expressed as [10,11]:

$$u = M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) \tag{1}$$

where $\theta \in R^n$ is the joint angular position vector of the robot; $u \in R^n$ is the vector of applied joint torque (or force); $M(\theta) \in R^{n\times n}$ is the inertia matrix, positive definite; $C(\theta,\dot{\theta})\dot{\theta} \in R^n$ is the effect of Coriolis and centrifugal torque; and $G(\theta) \in R^n$ is the gravitational torque. The physical

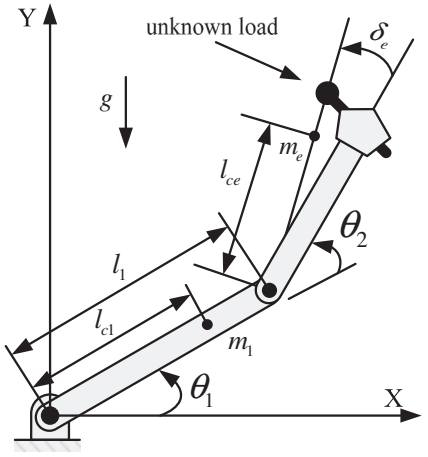


**Figure 2** An articulated two-link manipulator.

properties of the above model can be found in [12]; however, they are not needed here.

For the robot shown in Fig. 2 (1) can be rewritten as:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} -h\dot{\theta}_2 & -h(\dot{\theta}_1 + \dot{\theta}_2) \\ h\dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$$

where

$$M_{11} = a_1 + 2a_3\cos(\theta_2) + 2a_4\sin(\theta_2), M_{22} = a_2,$$
$$M_{21} = M_{12} = a_2 + a_3\cos(\theta_2) + a_4\sin(\theta_2),$$
$$h = a_3\sin(\theta_2) - a_4\cos(\theta_2), G_1 = b_1\cos(\theta_1) + b_2\cos(\theta_1 + \theta_2),$$
$$G_2 = b_2\cos(\theta_1 + \theta_2)$$

with

$$a_1 = I_1 + m_1 l_{c1}^2 + I_e + m_e l_{ce}^2 + m_e l_1^2, a_2 = I_e + m_e l_{ce}^2,$$
$$a_3 = m_e l_1 l_{ce}\cos(\delta_e), a_4 = m_e l_1 l_{ce}\sin(\delta_e)$$
$$b_1 = m_1 g l_{c1} + m_e g l_1, b_2 = m_e g l_{ce}$$

The nominal parameters of the two-link manipulator are chosen as follows:

$$m_1 = 5\text{ kg},\ m_e = 2.5\text{ kg},\ l_1 = 1.0\text{ m},\ l_{c1} = 0.5\text{ m},\ l_{ce} = 0.5\text{ m},\ \delta_e = 30^\circ,\ I_1 = 0.36\text{ kg m}^2,\ I_e = 0.24\text{ kg m}^2$$

Position control or also the so-called regulation problem is one of the most relevant issues in the operation of robot manipulators. This is a particular case of the motion control or trajectory control. The primary goal of motion control in joint space is to make the robot joints track a given time-varying desired joint position, $\theta^d = [\theta_1^d, \theta_2^d]^T$. Several control architectures related to robot control can be found in literature ranging from the simple PD, learning based, adaptive, and adaptive/learning hybrid controllers. The reader is referred to [12,13] and the references included. The main advantage of the PD controller is that it can easily be implemented on simple microcontroller architectures. On the other hand, the performance obtained from PD controllers is not satisfying for most of the sensitive applications [13,14]. Most of the other aforementioned types of controllers suffer from the complexities and the huge number of calculations needed to be carried out online.

## 3. Decentralized fuzzy system-based identification

It should be noticed that, for a planned trajectory, the desired torque depends not only on the trajectory, geometric, and inertia parameters of the link itself, but also on the parameters of the other links and the payload at the end effector; see (1). 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 computation to be as simple as possible, it is necessary to select a non-interactive fuzzy system. Here, only position and velocity are selected as two input variables and naturally the feedforward torque is selected as the output. Thus, the fuzzy rules are expressed in the following form:

$$\text{If } \theta^d(k) \text{ is } A_1^i \text{ and } \dot{\theta}^d(k) \text{ is } A_2^i \text{ then } u^d \text{ is } u_{FF}^i \tag{2}$$

where $A_1^i$ and $A_2^i$ are the fuzzy sets for $\theta^d$ and $\dot{\theta}^d$, $u_{FF}^i$ is the crisp output of each fuzzy rule, and $k$ is the time instant. The fuzzy system in (2) is called Sugeno zero-order model. Here, we call it as standard fuzzy system since it is widely used in the literature [7,15–17]. If the rule base has $M$ rules altogether, the final output of the fuzzy model is calculated as follows:

$$u_{FF}(k) = \frac{\sum_{i=1}^{M}[w^i(k)u_o^i]}{\sum_{i=1}^{M} w^i(k)} \tag{3}$$

$$w^i(k) = A_1^i(\theta^d(k)) \times A_2^i(\dot{\theta}^d(k)) \tag{4}$$

It has been proven [7] that the fuzzy system in (3) can approximate continuous function to an arbitrary degree of accuracy provided that enough number of rules are considered. In Section 4, the fuzzy system (3) is trained off-line and the optimal rule base is determined. 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 [16,18].

Naturally, the performance of the fuzzy model is dependent on the structure and the parameters of the fuzzy rule base resulted from some kind of learning procedure. Given a set of input–output data, the premise and consequence parameters can be determined by use of a complex search algorithms, recursive least square algorithm, and hybrid systems [16]. As mentioned earlier, in this paper, GAs are used to establish the feedforward fuzzy systems, which is the subject of the following Section.

## 4. Optimal selection of fuzzy systems using genetic algorithm

Genetic algorithms are derivative-free stochastic optimization methods based loosely on the concepts of natural selection and evolutionary processes. Their popularity can be attributed to their freedom from dependence on functional derivatives, and they are less likely to get trapped in local minima, which inevitably are present in any practical optimization application. Eventually, GAs can be used to determine the optimal parameters and structure of a fuzzy system given some optimality criterion.

The solution of an optimization problem begins with a set of potential solutions (fuzzy systems) or chromosomes (usually in the form of bit strings) that are randomly selected. The entire set of these chromosomes comprises a population. The chromosomes evolve during several iterations or generations. New generations (offsprings) are generated utilizing the crossover, mutation, and elitism technique. Crossover involves splitting two chromosomes and then combining one-half of each chromosome with the other pair. Mutation involves flipping a single bit of a chromosome. Elitism is a policy of always keeping a certain number of best members when each new population is generated. The chromosomes are then evaluated employing a certain fitness criteria, and the best ones are kept while the others are discarded. This process repeats until one chromosome has the best fitness and is taken as the optimum solution of the problem. Fig. 3 is a schematic diagram illustrating how a fuzzy system can be trained using GAs. A comprehensive review about GAs can be found in [19]. Refs. [20,21] also give other examples of using GAs to identify the fuzzy model parameters.

As the performance of a GA depends on its parameters, a parametric study has been carried out to determine the optimal

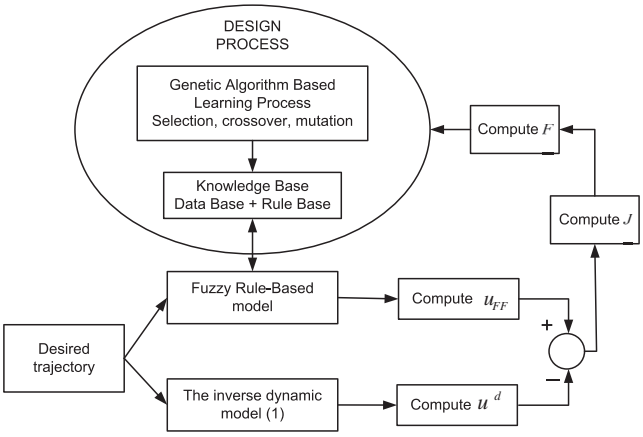


**Figure 3** Implementation flow chart of genetic algorithm.

set of parameters. These parameters are the population size, number of generations, number of bits of each variable, crossover rate, and the mutation rate. They are problem-dependent and should be selected carefully in order to achieve *good* results. For the problem under consideration, the following parameters are found to give the best results:

(a) number of generations is 150,
(b) population size is 50,
(c) single point crossover with a rate of 0.90,
(d) bitwise mutation with a rate of 0.1, and
(e) number of bits which represent each variable is 16.

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. 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 data. The robot model (1) in Section 2 with the nominal parameter values mentioned that there are used to emulate the robot motion. At the training stage, no parameter variations and nonlinear friction are considered. The trajectory for off-line training is chosen as follows:

$$\theta_1^d = 0.5\pi(1 - e^t) \text{ and } \theta_2^d = \pi(1 - e^t) \tag{5}$$

At first, both input variables in each joint are partitioned into four subsets and thus 16 fuzzy rules in the standard form of (2) are set up for each joint. Then, GA is used to tune the parameters of the fuzzy model within suitable ranges. In order to reduce the dimension of the searching space, the 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. The tuning ranges of the two fuzzy models are given in Table 1.

### 4.1. Parameter learning using genetic algorithm

During the training phase using GAs, the following quadratic form of performance index is established, so that the feedforward fuzzy model can realize the mapping of the robot inverse dynamics:

$$J = \frac{\sum_{k=1}^{P}[u^d(k) - u_{FF}(k)]^2}{P} \tag{6}$$

where $u^d(k)$ and $u_{FF}(k)$ are the desired torque computed from the model (system (1) or experimental data) and the torque computed from feedforward fuzzy model, respectively, and $P$ is the number of training samples. Since GAs guide the optimal solution to the direction of maximizing the fitness value, it is necessary to map the objective function (6) to the fitness function form by

$$F = \frac{1}{1 + J} \tag{7}$$

where $J$ is the performance index defined in (6) and 1 is introduced at the denominator to prevent the fitness function from becoming infinitely large.

The membership functions in fuzzy system (2) are taken as Gaussian which has the following form:

$$A_j^i(x) = \exp\left(\frac{-\left(x - c_j^i\right)^2}{\sigma_j^i}\right); \quad j = 1, 2, \ldots, 4 \tag{8}$$

where $c_j^i$ and $\sigma_j^i$ are the center and width of the Gaussian function. Here, the membership function in (8) is denoted as $(c_j^i, \sigma_j^i)$.

GAs represent the parameters for the given problem by the chromosome $S$ which may contain one or more substring(s). Each chromosome, therefore, contains a possible solution to the problem. A possible coding of the parameters to be tuned can be arranged as follows:

$$S_k = c_1^1 c_1^2 \cdots c_1^n \sigma_1^1 \sigma_1^2 \cdots \sigma_1^n c_2^1 c_2^2 \cdots c_2^n \sigma_2^1 \sigma_2^2 \cdots \sigma_2^n u_{FF}^1 u_{FF}^2 \cdots u_{FF}^M,$$
$$k = 1, 2, \ldots, N$$

where, $M = n^2$ is the number of rules and $N$ is the number of chromosomes in the generation.

After the training is completed, the fuzzy models for joint 1 and joint 2 resulted from the best chromosomes are shown in Tables 2 and 3, respectively. For instance, the first rule in Table 1 can be read as follows:

If $\theta^d(k)$ is $(0.08, 0.24)$ and $\dot{\theta}^d(k)$ is $(0.22, 0.59)$ then $u_{FF}$ is $-4.18$

The graphical representations of the two fuzzy models are depicted in Fig. 4. These figures show the complexity of a system

**Table 1** Ranges of the premise and consequent parameters for the two fuzzy models.

| Parameters | | Range |
|---|---|---|
| *Premise parameters for fuzzy models 1 and 2* | | |
| | $c_1^1, c_2^1$ | 0:1 |
| | $c_1^2, c_2^2$ | 1:2 |
| | $c_1^3, c_2^3$ | 2:3 |
| | $c_1^4, c_2^4$ | 3:4 |
| | $\sigma_1^1 \cdots \sigma_1^4, \sigma_2^1 \cdots \sigma_2^4$ | 0.1:3 |
| *Consequent parameters* | | |
| Fuzzy model 1 | $u_{FF}^i$, $i = 1:16$ | −20:80 |
| Fuzzy model 2 | $u_{FF}^i$, $i = 1:16$ | −20:50 |

**Table 2** Standard rule base of joint 1.

| IF | | THEN | IF | | THEN |
|---|---|---|---|---|---|
| $\theta^d$ | $\dot{\theta}^d$ | $u^i_{FF}$ | $\theta^d$ | $\dot{\theta}^d$ | $u^i_{FF}$ |
| (0.08, 0.24) | (0.22, 0.59) | −4.18 | (2.81, 1.51) | (0.22, 0.59) | 25.30 |
| (0.08, 0.24) | (1.82, 1.94) | 52.40 | (2.81, 1.51) | (1.82, 1.94) | −13.56 |
| (0.08, 0.24) | (2.14, 0.58) | 75.82 | (2.81, 1.51) | (2.14, 0.58) | 36.86 |
| (0.08, 0.24) | (3.93, 1.70) | 56.30 | (2.81, 1.51) | (3.93, 1.70) | −13.07 |
| (1.70, 2.30) | (0.22, 0.59) | −19.67 | (3.32, 2.93) | (0.22, 0.59) | −16.70 |
| (1.70, 2.30) | (1.82, 1.94) | 57.24 | (3.32, 2.93) | (1.82, 1.94) | −1.54 |
| (1.70, 2.30) | (2.14, 0.58) | 79.38 | (3.32, 2.93) | (2.14, 0.58) | 41.87 |
| (1.70, 2.30) | (3.93, 1.70) | 50.36 | (3.32, 2.93) | (3.93, 1.70) | 27.07 |

**Table 3** Standard rule base of joint 2.

| IF | | THEN | IF | | THEN |
|---|---|---|---|---|---|
| $\theta^d$ | $\dot{\theta}^d$ | $u^i_{FF}$ | $\theta^d$ | $\dot{\theta}^d$ | $u^i_{FF}$ |
| (0.23, 1.15) | (0.47, 1.35) | −19.51 | (2.87, 2.96) | (0.47, 1.35) | −15.84 |
| (0.23, 1.15) | (1.53, 2.31) | 17.32 | (2.87, 2.96) | (1.53, 2.31) | −16.07 |
| (0.23, 1.15) | (2.13, 0.35) | 18.14 | (2.87, 2.96) | (2.13, 0.35) | 6.87 |
| (0.23, 1.15) | (3.40, 0.68) | −3.13 | (2.87, 2.96) | (3.40, 0.68) | 14.28 |
| (1.57, 1.34) | (0.47, 1.35) | −15.83 | (3.45, 0.87) | (0.47, 1.35) | 11.51 |
| (1.57, 1.34) | (1.53, 2.31) | −11.22 | (3.45, 0.87) | (1.53, 2.31) | 18.43 |
| (1.57, 1.34) | (2.13, 0.35) | −10.22 | (3.45, 0.87) | (2.13, 0.35) | 39.34 |
| (1.57, 1.34) | (3.40, 0.68) | 37.54 | (3.45, 0.87) | (3.40, 0.68) | 8.08 |

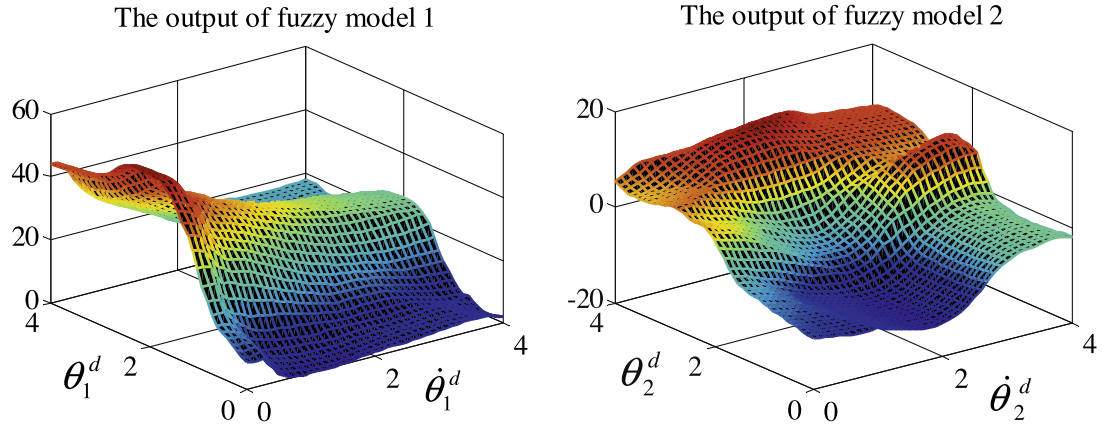


**Figure 4** The output surfaces of the two fuzzy models.

which can be represented by relatively simple fuzzy counterpart. Fig. 5 shows the approximating results of the fuzzy models. The average approximating errors are 0.8725 and 0.4146, respectively.

Because the performance of the fuzzy systems is evaluated by the approximating precision, the above fuzzy models with standard structure are acceptable. However, during simulation tests, it is found that the average firing rates of the two rule bases are relatively low. They are 42% for the first fuzzy model and 43% for the second one. It means that the fuzzy systems are not compact enough and the structure of the fuzzy rule bases needs to be optimized. This choice is reasonable since it leads to a reduced number of arithmetic operations which is needed to be performed online. Structure optimization is the subject of the coming Subsection.

### *4.2. Structure optimization using genetic algorithms*

In this Subsection, the structure and parameters of the fuzzy rules are simultaneously optimized using GAs. To this end, each fuzzy system (chromosome) contains two substrings. The first substring, which has the same form illustrated as in the previous Subsection, is to optimize the parameters of the fuzzy model. The second substring encodes the structure of the fuzzy rule base, such that one integer number represents one membership (MF) in the space of input variable in

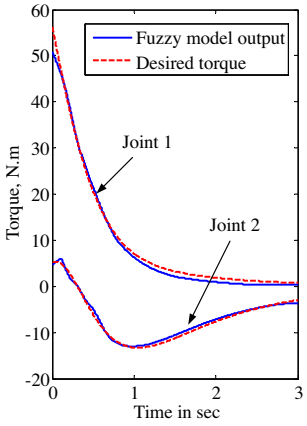


**Figure 5** Off-line training of the inverse dynamics (without structure optimization).

| 3 1 | 2 0 | ..... | 0 0 |
|---|---|---|---|
| Rule 1 | Rule 2 | ..... | Rule 16 |

**Figure 6** Example of the second substring of a chromosome.

question. Similar to the work of [18], the MFs of each input variable are numbered in ascending order according to their centers, i.e., a number "1" represents the MF with the lowest center and "4" for biggest one, since each variable is supposed to have at most four subspaces. The second substring may take one of the following numbers: 0, 1, 2, 3, and 4. Number "0" implies that this variable does not appear in the premise of the rule. If both variables 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 logically accepted. An example of the second substring is shown in Fig. 6.

The corresponding fuzzy rules are

$R^1$ : If $\theta^d(k)$ is $(c_1^3, \sigma_1^3)$ and $\dot{\theta}^d(k)$ is $(c_2^1, \sigma_2^1)$ then $u_{FF}$ is $u_{FF}^1$

$R^2$ : If $\theta^d(k)$ is $(c_1^2, \sigma_1^2)$ then $u_{FF}$ is $u_{FF}^2$:

$R^{16}$ : (Deleted)

The following performance index is used to optimize both the parameters and structure of the fuzzy models:

$$J = \frac{\sum_{k=0}^{P}[u^d(k) - u_{FF}(k)]}{P} + \lambda J_S \tag{9}$$

where $\lambda$ is the weighting constant, and $J_S$ is the penalty for model complexity and is expressed as:

$$J_S = \frac{\text{The total number of rules in the rule base}}{\text{The average number of active rules}} \tag{10}$$

In this work, $\lambda$ is set to 0.1 for joint 1 and 0.4 for joint 2. A rule is considered as active one when the $w^i$ in (4) is greater than 0.05.Simulation results show that the optimized rule bases for joint 1 and joint 2 have 9 and 11 rules, respectively, and the firing rates are raised to about 84% and 75%, respectively. The rule bases for the two joints are listed in Tables 4 and 5, and the graphical representation of the two rules is depicted in Fig. 7. The approximating results are demonstrated in Fig. 8. The average approximating errors are 0.9267 and 0.5349, respectively. So that, it can be concluded that the approximating errors are relatively small, which means that structure optimization is quite reasonable.

## 5. Decentralized fuzzy feedback control

The performance of any fuzzy logic controller is greatly dependent on its inference rules. In most cases, the closed-loop control performance and stability are enhanced if more rules are added to the rule base of the fuzzy controller. However, a large set of rules requires more online computational time and more parameters need to be adjusted. Adjustment of the fuzzy system may be achieved using GAs [21,22]. However, GAs cannot be used online and perfect mathematical model or experimental data should be available.

In this Section, a robust PD-type fuzzy feedback controller is driven for a class of MIMO second order nonlinear systems with application to tracking control problem of robotic manipulators [9]. The rule base consists of only four rules per each DOF. The approach implements fuzzy partition to the state variables based on Lyapunov synthesis. The resulting control law is stable and able to exploit the dynamic variables of the system in a linguistic manner.

### 5.1. Construction of fuzzy feedback controllers

In this Subsection, we apply the fuzzy synthesis to the design of stable controllers. To this end, consider a class of MIMO nonlinear second order systems whose dynamic equation can be expressed as:

$$\ddot{x}(t) = f(x, \dot{x}, u_{FB}), \tag{11}$$

where $f(x, \dot{x}, u_{FB})$ is an unknown continuous function, $u_{FB}$ is the feedback control input, and $x(t) = [x_1, x_2, \ldots, x_n]^T$ is the state vector and $\dot{x} = \frac{dx}{dt} = [\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n]^T$. We now seek a smooth Lyapunov function $V: R^n \to R^n$ for the continuous

**Table 4** The optimized fuzzy rule base for joint 1.

| IF | | THEN | IF | | THEN |
|---|---|---|---|---|---|
| $\theta^d$ | $\dot{\theta}^d$ | $u_{FF}^i$ | $\theta^d$ | $\dot{\theta}^d$ | $u_{FF}^i$ |
| (0.43,0.19) | (0.60,0.80) | 56.38 | (0.43,0.19) | (2.37,1.51) | 1.72 |
| – | (0.60,0.80) | −7.57 | (2.84,1.55) | (0.60,0.80) | −12.05 |
| (1.38,2.01) | (3.13,2.58) | 65.13 | – | (1.70,0.48) | 54.35 |
| (1.38,2.01) | (2.37,1.51) | 66.76 | (1.38,2.01) | – | −9.70 |
| (1.38,2.01) | (1.70,0.48) | 74.86 | | | |

**Table 5** The optimized fuzzy rule base for joint 2.

| IF | | THEN | IF | | THEN |
|---|---|---|---|---|---|
| $\theta^d$ | $\dot{\theta}^d$ | $u^i_{FF}$ | $\theta^d$ | $\dot{\theta}^d$ | $u^i_{FF}$ |
| (0.43, 0.19) | – | 15.63 | (2.85, 1.55) | – | −9.58 |
| (0.43, 0.19) | (2.37, 1.51) | −15.43 | – | (2.37, 1.51) | −18.40 |
| (1.38, 2.01) | (0.60, 0.80) | −3.17 | (0.43, 0.19) | (0.60, 0.80) | 7.92 |
| (0.43, 0.19) | (1.70, 0.48) | 28.68 | (2.85, 1.55) | (2.37, 1.51) | −17.59 |
| (1.38, 2.01) | – | −10.11 | (0.43, 0.19) | (3.13, 2.58) | 19.38 |
| – | (0.60, 0.80) | 10.86 | – | – | – |

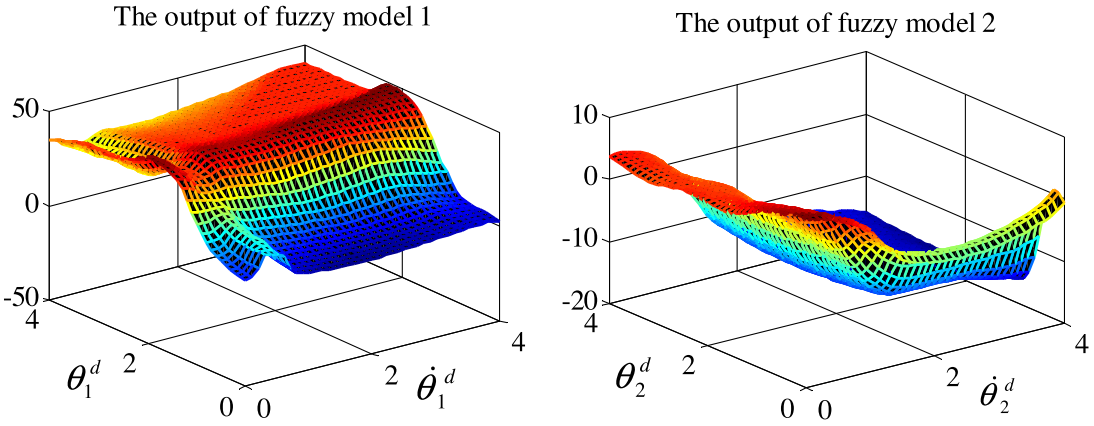


**Figure 7** The output surfaces of the two fuzzy models after structure optimization.

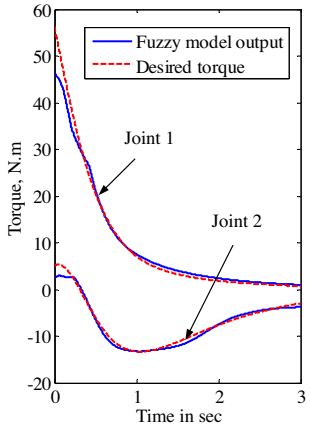


**Figure 8** Off-line training of the inverse dynamics (with structure optimization).

feedback model (1) that is positive definite, i.e., $V(x) > 0$ when $x \neq 0$ and $V(x) = 0$ when $x = 0$, and grows to infinity: $V(x) \to \infty$ as $x^T x \to \infty$. Obviously, this holds for a generalized Lyapunov candidate function of the following quadratic form:

$$V(x,t) = \frac{1}{2}x^T x + \frac{1}{2}\dot{x}^T \dot{x} \tag{12}$$

Differentiating (12) with respect to time gives

$$\dot{V}(x,t) = x_1\dot{x}_1 + x_2\dot{x}_2 + \cdots + x_n\dot{x}_n + \dot{x}_1\ddot{x}_1 + \dot{x}_2\ddot{x}_2 + \cdots + \dot{x}_n\ddot{x}_n$$

From which

$$\dot{V}(x,t) = (x_1\dot{x}_1 + \dot{x}_1\ddot{x}_1) + (x_2\dot{x}_2 + \dot{x}_2\ddot{x}_2) + \cdots + (x_n\dot{x}_n + \dot{x}_n\ddot{x}_n)$$

This is equal to

$$\dot{V}(x,t) = \dot{V}_1 + \dot{V}_2 + \cdots + \dot{V}_n \tag{13}$$

where

$$\dot{V}_i(x,t) = x_i\dot{x}_i + \dot{x}_i\ddot{x}_i, \quad i = 1,2,\ldots,n$$

Then, the standard results in Lyapunov stability theory imply that the dynamic system (11) has a stable equilibrium $x = x_e$ if each $\dot{V}_i$ in (13) is $\leqslant 0$ along the system trajectories. To achieve this, we have chosen the control $u_{FB_i}(x)$ to be proportional to $\ddot{x}_i$.

Next, our controller design is achieved if we determine a fuzzy control $u_{FB_i}(x)$ so that:

$$\dot{V}_i(x,t) = x_i\dot{x}_i + \alpha_i\dot{x}_i u_{FB_i}(x) \leqslant 0, \quad i = 1,2,\ldots,n \tag{14}$$

where $\alpha_i$ is a positive constant. The results of Wang [23] state that a fuzzy system that would approximate (14) exists. To this end, one would consider the state vector $x(t)$ and $\dot{x}(t)$ to be the inputs to the fuzzy system. The output of the fuzzy system is the feedback control $u_{FB}$. A possible form of the control rules is:

IF $x_i$ is $(lv)$ and/or $\dot{x}_i$ is $(lv)$THEN $u_{FB_i}$ is $(lv)$, $i = 1,2,\ldots,n$

where the $(lv)$ are linguistic values (e.g., *positive* and *negative*). These rules constitute the rule base for a Mamdani-type fuzzy controller.

In the above formulation, two basic assumptions have been made. They are the following:

- The knowledge of the state vector. It is assumed to be available from measurements.
- The control input, $u_{FB}$ is proportional to $\ddot{x}$. This assumption can be justified for a large class of second order nonlinear mechanical systems [23–26]. For instance, here in robotics, it means that the acceleration of links is proportional to the input torque.

These two assumptions represent the basic knowledge about the system which is needed to derive the control rules. Clearly, the exact mathematical model is not needed.

In the coming Subsection, we use this approach to design a PD-type fuzzy feedback tracking controller.

### 5.2. Fuzzy feedback tracking control

Robots are familiar examples of trajectory-following mechanical systems. Their nonlinearities and strong coupling of the robot dynamics present a challenging control problem. In practice, the load may vary while performing different tasks, the friction coefficients may change in different configurations, and some neglected nonlinearities as backlash may appear. Therefore, the control objective is to design a stable fuzzy controller, so that the link movement follows the desired trajectory in spite of such effects.

Consider a class of robots whose vector of generalized coordinates is denoted by $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T$ where $\theta_i$, $i = 1, \ldots, n$ are the joint parameters. We consider the state variables of the robot as $\theta(t)$ and $\dot{\theta}(t)$, which are usually available as feedback signals. Define the tracking error vectors $e(t)$ and $\dot{e}(t)$ as:

$$e(t) = \theta(t) - \theta^d(t), \text{ and } \dot{e}(t) = \dot{\theta}(t) - \dot{\theta}^d(t) \tag{15}$$

where $\theta^d$ and $\dot{\theta}^d$ are vectors of the desired joint position and velocity, respectively. Throughout this work, we assume that $\theta^d$ and its derivative are available for online control computation. In robot tracking tasks, the desired position history is generally planned ahead of time and its derivatives can be easily obtained.

We now apply the approach presented in the previous Subsection in order to find a fuzzy controller that achieves tracking to the robotic system under consideration. To this end, let us choose the following Lyapunov function candidate

$$V = \frac{1}{2}(e^T e + \dot{e}^T \dot{e}) \tag{16}$$

Differentiating with respect to time and using (13) gives

$$\dot{V}_i = e_i \dot{e}_i + \dot{e}_i \ddot{e}_i$$

To enforce asymptotic stability, it is required to find $u_{FB}$ so that

$$\dot{V}_i = e_i \dot{e}_i + \dot{e}_i \ddot{e}_i \leq 0 \tag{17}$$

in some neighborhood of the equilibrium of (16). Taking the control $u_{FB}$ to be proportional to $\ddot{e}$, Eq. (17) can be rewritten as:

$$\dot{V}_i = e_i \dot{e}_i + \alpha_i \dot{e}_i u_{FB_i} \leqslant 0 \tag{18}$$

**Table 6** Fuzzy rules for the fuzzy feedback controller.

| $e_i$ \ $\dot{e}_i$ | P | N |
|---|---|---|
| P | $u_N$ | $u_Z$ |
| N | $u_Z$ | $u_P$ |

$i = 1, \cdots n$

where $\alpha_i$ is positive constant, $i = 1, \ldots, n$. Sufficient conditions for (18) to hold can be stated as follows:

(a) if for each $i \in [1, \ldots, n]$, $e_i$ and $\dot{e}_i$ have opposite signs and $\alpha_i u_{FB_i}$ is zero, inequality (18) holds;
(b) if $e_i$ and $\dot{e}_i$ are both positive, then (18) will hold if $\alpha_i u_{FB_i}$ is negative; and
(c) if $e_i$ and $\dot{e}_i$ are both negative, then (18) will hold if $\alpha_i u_{FB_i}$ is positive. $i \in [1, \ldots, n]$ denotes the joint number.

Using these observations and assuming that $\alpha_i$ is positive small number, one can easily obtain the four rules listed below in Table 6.

In this table, P and N denote respectively positive and negative errors; $u_P$, $u_N$, and $u_Z$ are respectively positive, negative, and zero control inputs. These rules are simply the fuzzy partitions of $e_i$, $\dot{e}_i$ and $u_{FB_i}$ which follow directly from the stabilizing conditions of the Lyapunov function (16).

In concluding words, the presented approach transforms classical Lyapunov synthesis from the world of exact mathematical quantities to the world of words [26]. This combination provides us with a solid analytical basis from which the rules are obtained and justified. Relative to other works, this number of rules is quite small. For example, in [8], the rule base of a two-link robot consists of 625 rules. After introducing a rule base reduction approach, the authors in [8] reach to a rule base consists of 160 rules, which is hard to be implemented. Otherwise, the results obtained here contradict the conclusions of a recent survey on the industrial applications of fuzzy controllers. The authors' opinion there in [27] is that all fuzzy control applications should be tackled in the model-based design manner. They think that this is the way that enables systematic analyses of the structural properties of the fuzzy controllers such as stability, controllability, parametric sensitivity, and robustness. Remember that here, we did not use any information about the system model.

To complete the design, we must specify the fuzzy system with which the fuzzy feedback computes the control signal. Here, we use different fuzzy system than that mentioned in Section 3. The Gaussian membership defining the linguistic terms in the rule base is chosen as follows:

$$\mu_{positive}(x) = G(x, a_z) = e^{-(x-a_z)^2}$$

$$\mu_{negative}(x) = G(x, -a_z)$$

$$\mu_{zero}(x) = G(x, 0)$$

where $a_z > 0$ and $z$ stands for control variable, the product for "and" and center of gravity inferencing. For some positive constant $k_i$, $i \in [1, \ldots, n]$ denotes the joint number, the above four rules can be represented by the following mathematical expression:

$$u_{FB_i} = \frac{G(e_i, a_{1i})(-k_i) + G(e_i, -a_{1i})(k_i)}{G(e_i, a_{1i}) + G(e_i, -a_{1i})} + \frac{G(\dot{e}_i, a_{2i})(-k_i) + G(\dot{e}_i, -a_{2i})(k_i)}{G(\dot{e}_i, a_{2i}) + G(\dot{e}_i, -a_{2i})}$$

in more details

$$u_{FB_i} = -k_i \left[ \frac{\exp(-(e_i - a_{1i})^2) - \exp(-(e_i + a_{1i})^2)}{\exp(-(e_i - a_{1i})^2) + \exp(-(e_i + a_{1i})^2)} + \frac{\exp(-(\dot{e}_i - a_{2i})^2) - \exp(-(\dot{e}_i + a_{2i})^2)}{\exp(-(\dot{e}_i - a_{2i})^2) + \exp(-(\dot{e}_i + a_{2i})^2)} \right]$$

from which

$$u_{FB_i} = -k_i \left[ \frac{\exp(2a_{1i}e_i) - \exp(-2a_{1i}e_i)}{\exp(2a_{1i}e_i) + \exp(-2a_{1i}e_i)} + \frac{\exp(2a_{2i}\dot{e}_i) - \exp(-2a_{2i}\dot{e}_i)}{\exp(2a_{2i}\dot{e}_i) + \exp(-2a_{2i}\dot{e}_i)} \right]$$

This yields the fuzzy feedback controller

$$u_{FB_i} = -k_i[\tanh(2a_{1i}e_i) + \tanh(2a_{2i}\dot{e}_i)], \quad i = 1, \ldots, n \tag{19}$$

In (19), the inputs are the error in position $e_i$ and the error in velocity $\dot{e}_i$ and the output is the control input of joint $i$; i.e., it is a PD-type fuzzy feedback controller. The following remarks are in order:

- The fuzzy controller in (19) is a special case of fuzzy systems, where Gaussian membership functions are used to introduce the input variables ($e_i$ and $\dot{e}_i$) to the fuzzy network. Also, the fuzzification and defuzzification methods used in this study are not unique; see [16] for other alternatives. For example, using different membership functions (e.g., triangular, trapezoidal, etc.) will result in a different fuzzy controller. However, the controller in (19) is a simple one and the closed form relation between the inputs and the output makes it computationally inexpensive.
- Only three parameters per each DOF need to be tuned, namely, they are $k_i$, $a_{1i}$ and $a_{2i}$. This greatly simplifies the tuning procedure since the search space is quite small relative to other works. For instance, the fuzzy controller in [28] needs 45 parameters to be tuned for a one DOF system.
- This controller is inherently bounded since $|\tanh(x)| \leqslant 1$.
- Each joint has independent control input $u_{FB_i}$, $i = 1, 2, \ldots, n$.
- In the case of robotic control, this controller can be regarded as output feedback controller since the joint's position and velocity are usually the outputs.

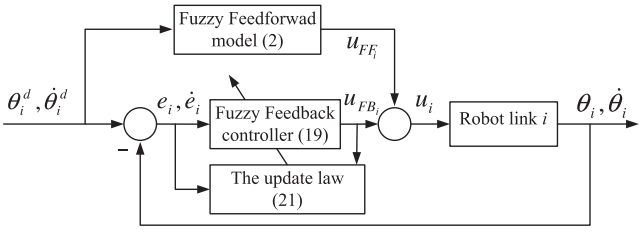


**Figure 9** Configuration of the proposed decentralized fuzzy control scheme of joint $i$.

Finally, the fuzzy PD gain, i.e., $k_i$, $i \in [1, \ldots, n]$ is chosen so as to minimize the following quadratic performance index:

$$J_i = \frac{1}{2}\left\{ r_i[u_{FB_i}(k)]^2 \right\} \tag{20}$$

where input $r_i$ is a constant. According to the gradient method, the learning algorithm of the parameter $k_i$ in the feedback fuzzy controller (19) can be derived as follows:

$$\Delta k_i = -\frac{\partial J_i}{\partial k_i} = -\frac{\partial J_i \partial u_{FB_i}}{\partial u_{FB_i} \partial k_i} = -r_i u_{FB_i}[\tanh(2c_{1i}e_i) + \tanh(2c_{2i}\dot{e}_i)] \tag{21}$$

Thus, the fuzzy feedback controller uses the $e_i$, $\dot{e}_i$ and $u_{FB_i}$ to compute (21) and update the control gain $k_i$ given that $k_i(0) \neq 0$. The overall closed-loop control system is shown in Fig. 9, where $u_i = u_{FB_i} + u_{FF_i}$ is the total input to joint $i$.

## 6. Computational aspects

In general, control algorithms for closed-loop control should have a small number of tuning parameters and short computation time due to limited memory of low-cost microprocessors. This Section discusses the complexity aspects of the feedforward and feedback computation of the control scheme proposed in this paper. The computational complexity of the feedback controller is compared with that of a self-tuning fuzzy controller proposed in [29]. Also, torque computing methods based on robot inverse dynamics are compared with the feedforward system. It is shown that the proposed control scheme is computationally very efficient.

Naturally, the computational burden can be evaluated in terms of required mathematical multiplication and addition operations. The proposed control scheme in this paper consists of two components: a feedforward torque compensation system and a fuzzy PD feedback controller. For the first component, i.e., the feedforward fuzzy system, the calculation has three stages: computation of the membership functions, computation of the contribution of each rule, and computation of the final output of the fuzzy system. The results are given in Table 7, where $n$ is the DOF of the manipulator. With respect to the standard fuzzy system followed in this paper, each

**Table 7** Computational complexity of the fuzzy feedforward system.

| | Inverse dynamics | Standard fuzzy system | Optimal fuzzy system (average) |
|---|---|---|---|
| Addition | $117n - 24$ | $56n$ | $33n$ |
| Multiplication | $103n - 21$ | $33n$ | $25n$ |

**Table 8** Computational complexity of the fuzzy feedback controller.

| | Self-tuning fuzzy controller [29] | The proposed fuzzy controller |
|---|---|---|
| Addition | $97n$ | $6n$ |
| Multiplication | $113n$ | $15n$ |

variable is supposed to have at most four subsets and therefore there are eight fuzzy membership functions involved for each joint. For the optimized fuzzy system, we list the total number of the addition and multiplication operation of the two fuzzy systems obtained in Section 4 divided by two. This manipulation has been adapted because the two fuzzy systems have different number of rules and membership functions in each rule base. So that, the average number of arithmetic operations is presented in the last column. It is obvious that the number of arithmetic calculations of the optimized fuzzy system is the lowest compared with that of the inverse dynamic model and the standard fuzzy system. In comparison with the conventional torque computing method, Table 7 demonstrates that the computation burden of the proposed fuzzy torque computing system is significantly low.

The computation of the fuzzy feedback controller can also be divided into two parts: computation of (19) and computation of the adaptive gain; $k_i$ (21). For the sake of comparison, Table 8 demonstrates the computational complexity of our scheme with the self-tuning fuzzy controller proposed in [29]. The comparison is fair since the feedback controller in [29] is essentially a PD fuzzy controller with self-tuning mechanism. In [29], the rule base has been transformed to a decision table and is used by a back-propagation algorithm to adjust the scaling factors of the fuzzy system. The difference resides in the fact that the rule base in [29] consists of 49 rules for one DOF system and the mapped elements ($e$ and $\dot{e}$) are obtained by interpolation. Furthermore, the tuning procedure is composed of two stages and some learning steps are needed by the second stage, while the tuning system using (21) is much simpler. Simulation results, in the coming Section, show that it is also efficient.

## 7. Simulation results

The purpose of the simulation is to investigate the robustness of the proposed control scheme. The robot system considered in the simulation is the two-link robot presented in Section 2. Through the simulations, the physical insight of the behavior is revealed. In the coming results, it is assumed that initial positions of joints $\theta_1(0) = \theta_2(0) = 0°$ *rad* and the robot is at rest, i.e., the initial velocities of joints $\dot{\theta}_1(0) = \dot{\theta}_2(0) = 0°$ *rad*/*s*. This initialization imposes a large initial velocity error since $\dot{e}_1(0) = -\pi/2$, $\dot{e}_2(0) = -\pi$ *rad*/*s*. One can expect uneasy transient stage.

The input torque shown in Figs. 10 and 11 shows the evolution of the tracking errors. They show that the errors have converged to zero. Note that the transient period is less than 0.5 s. Otherwise, it is interesting to notice how the control gains evolve with time. Fig. 12 depicts the evolution of these parameters with time. They have been initialized as $k_1(0) = k_2(0) = 100$ N m.

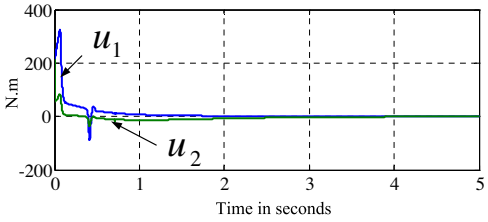


**Figure 10** The control effort.

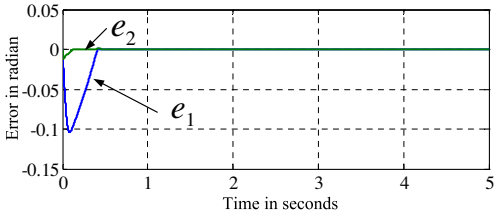


**Figure 11** The tracking errors.

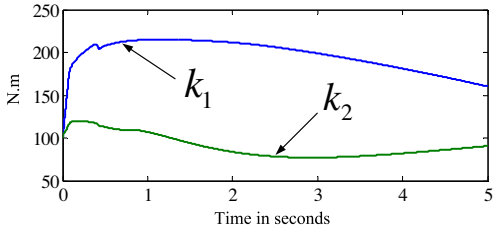


**Figure 12** Record of the adaptive control gains during motion.

In order to observe how the controller behaves in the presence of various uncertainties, three types of uncertainties are considered, such as, parameter variations, unmodeled nonlinear friction, and unknown payloads.

### 7.1. Parameter variations

By parameter variations, we mean here the masses of the links. It is assumed that they vary randomly with time every 0.3 s. The mass of the base link varies in the range of $5 \rightarrow 7$ kg (the nominal mass is 5 kg) and the mass of the elbow link $2 \rightarrow 5$ kg (the nominal mass is 2.5 kg). Fig. 13 depicts their variation with respect to time and Fig. 14 shows the corresponding tracking errors. It can be noticed that with respect to previous results, there is little or no change has taken place during the transient and the steady state periods. However, it has been noticed that increasing the range of variations of the masses has resulted in unstable system. Results of these tests are not presented here. However, this can be explained that under such situations, the torque computed from the trained feedforward fuzzy systems is no longer near the nominal torque.

### 7.2. Unmodeled friction

At the off-line training stage of our simulation, we obtain the training samples from the robot model in (1), which does not

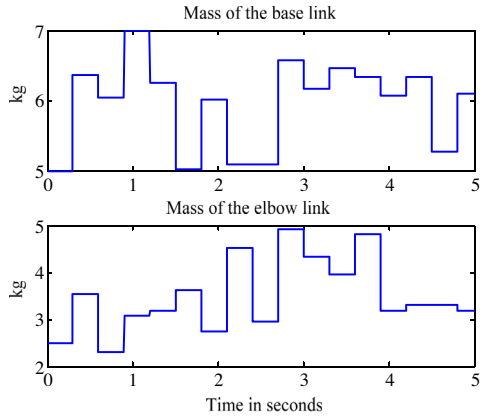


**Figure 13** Mass variations during motion of links.

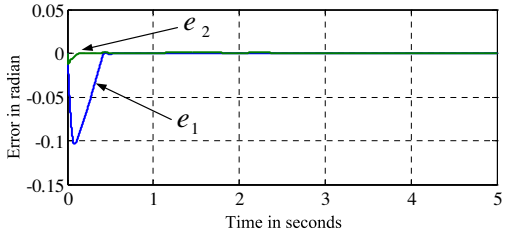


**Figure 14** The tracking errors when the mass of links varies randomly within specified ranges.

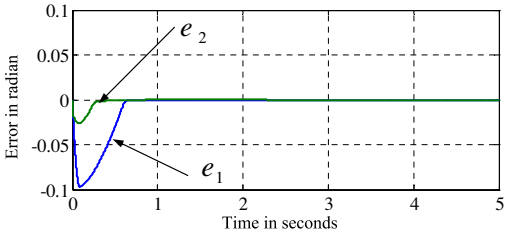


**Figure 15** The tracking errors in the presence of unmodeled friction.

consider the nonlinear friction. In order to examine the performance of the controller in the presence of unmodeled nonlinear friction, the following unmodeled nonlinear friction is added at the control stage:

$$F = F_d + F_s$$

where $F_d$ and $F_s$ are the dynamic and static friction torque, respectively. They can be expressed by:

$$F_d = \begin{bmatrix} d_1 \cos(x_1) & 0 \\ 0 & d_2 \cos(x_2) \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \text{ and } F_s = \begin{bmatrix} c_1 \text{sgn}(\dot{x}_1) \\ c_2 \text{sgn}(\dot{x}_2) \end{bmatrix}$$

We use $d_1 = 50$, $d_2 = 30$ and $c_1 = 18$, $c_2 = 12$. Results are shown in Figs. 15 and 16. It can be noticed that the transient period has increased relative to the cases when the friction was not considered. Also, the input torque is relatively higher during this period. Nevertheless, convergence of the tracking errors has been achieved.

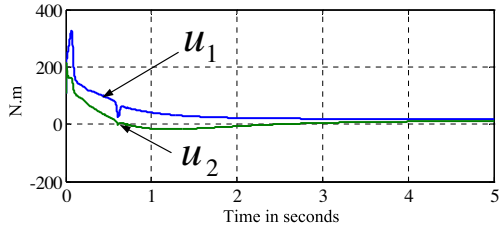


**Figure 16** The control input in the presence of unmodeled friction.

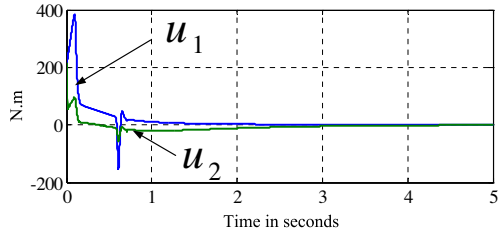


**Figure 17** The input torque when the payload increases to 150%.

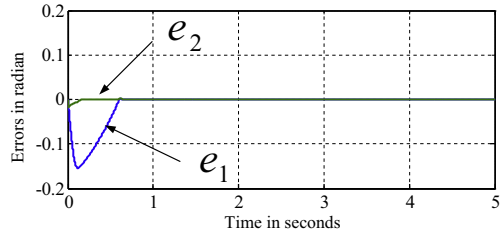


**Figure 18** The tracking errors in the presence of 150% increase in the payload.

### *7.3. Unknown payload*

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. The coming results are obtained when the mass and inertia of the base and elbow links (carrying the payload) have been increased to 150%. This increase in the mass and inertia of the two links is supposed to be unknown. Fig. 17 shows that input torque is relatively high. Also, the tracking errors exhibit larger overshoot during the transient period, Fig. 18. However, convergence of errors to a narrow region close to zero has taken place.

## 8. Conclusions

In this paper, a decentralized fuzzy control scheme for robot manipulator is developed. The controller for each joint has a feedforward fuzzy torque computing system and a feedback fuzzy PD controller. The online computational burden for nonlinear feedforward compensation is greatly relaxed due to the simple structure of the fuzzy systems. GAs are applied to

fuzzy system training because they are fully data-driven and are able to optimize both structure and parameters of the fuzzy system simultaneously. The training samples can be collected by doing experiments or by establishing an ideal model. Simulation results show that the proposed control scheme works well, even if the ideal model is not in concordance with the real inverse dynamics.

An important feature of this study is that it has transferred the proposed fuzzy feedback controller to a closed form relation between the inputs and the output, leading to a computationally efficient fuzzy logic controller. The rule base consists of only four rules and has a PD-like structure. The gains are tuned online based on the gradient method. This feedback controller is inherently bounded; the upper and lower bounds can be arbitrary selected by suitably adjust its parameters. Various simulation results prove that the proposed controller is effective. Finally, it can be concluded that using the proposed control approach presents a convenient option for controlling a large class of nonlinear MIMO second order systems.

## References


[1] Knut Graichen, Michael Zeitz, Feedforward control design for nonlinear systems under input constraints, Control and Observer Design, vol. 322, Springer-Verlag, Berlin Heidelberg, 2005, pp. 235–252 (LNCIS).

[2] Stefano Piccagli, Antonio Visioli, An optimal feedforward control design for set-point following of MIMO processes, Journal of Process Control 19 (2009) 978–984.

[3] J.L. Guzman, T. Hagglund, Simple tuning rules for feedforward compensators, Journal of Process Control 21 (2011) 92–102.

[4] Saverio Messineo, Andrea Serrani, Adaptive feedforward disturbance rejection in nonlinear systems, Systems & Control Letters 58 (2009) 576–583.

[5] B. Brogliato, D. Rey, A. Pastore, J. Barnier, Experimental comparison of nonlinear controllers for flexible joint manipulators, The International Journal of Robotics Research 17 (3) (1998) 260–281.

[6] M. Margaloit, G. Langholz, Fuzzy control of a benchmark problem: computing with words approach, IEEE Transactions on Fuzzy Systems 12 (2) (2004) 230–235.

[7] Yong Ho Kim, Sang Chul Ahn, Wook Hyun Kwon, Computational complexity of general fuzzy logic control and its simplification for a loop controller, Fuzzy Sets and Systems 111 (2000) 215–224.

[8] Hala Bezine, Nabil Derbel, Adel M. Alimi, Fuzzy control of robot manipulators: some issues on design and rule base size reduction, Engineering Applications of Artificial Intelligence 15 (2002) 401–416.

[9] Abdel Badie Sharkawy, A computationally efficient fuzzy logic controller for robotic systems, in: The Proceedings of the Ninth International Conference on Production Engineering, Design and Control (PEDAC'9), Alexandria, Egypt, February 10–12, 2009.

[10] M. Sun, S.S. Ge, I.M.Y. Mareels, Adaptive repetitive learning control of robotic manipulators without the requirement for initial repositioning, IEEE Transactions on Robotics 22 (3) (2006) 563–568.

[11] Tzuu-Hseng S. Li, Yun-Cheng Huang, MIMO adaptive fuzzy terminal sliding-mode controller for robotic manipulators, Information Sciences 180 (2010) 4641–4660.

[12] Stefano Liuzzo, Patrizio Tomei, A global adaptive learning control for robotic manipulators, Automatica 44 (2008) 1379–1384.

[13] Serhan Yamacli, Huseyin Canbolat, Simulation of a SCARA robot with PD and learning controllers, Simulation Modelling Practice and Theory 16 (2008) 1477–1487.

[14] T. Das, C. Dülger, Mathematical modeling, simulation and experimental verification of a SCARA robot, Simulation Modelling Practice and Theory 13 (2005) 257–271.

[15] George P. Moustris, Spyros G. Tzafestas, Switching fuzzy tracking control for mobile robots under curvature constraints, Control Engineering Practice 19 (2011) 45–53.

[16] Jyh-Shing Roger Jang, Chuen-Tsai Sun, Eiji Mizutani, Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall International, Inc., 1997.

[17] T. Lee, H. Lam, F. Leung, P. Tam, A practical fuzzy logic controller for the path tracking of wheeled mobile robots, IEEE Control Systems Magazine 23 (2) (2003) 60–65.

[18] Yaochu Jin, Decentralized adaptive fuzzy control of robot manipulators, IEEE Transaction on System, Man, and Cybernetics – Part B; Cybernetics 28 (1) (1988).

[19] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1999.

[20] Chih-Jer Lin, Shu-Yin Chen, Evolutionary algorithm based feedforward control for contouring of a biaxial piezo-actuated stage, Mechatronics 19 (2009) 829–839.

[21] Abdel Badie Sharkawy, Genetic fuzzy self-tuning PID controllers for antilock braking systems, Engineering Applications of Artificial Intelligence 23 (7) (2010) 1041–1052.

[22] Aytekin. Bagis, Dervis. Karaboga, Evolutionary algorithm-based fuzzy PD control of spillway gates of dams, Journal of the Franklin Institute 344 (2007) 1039–1055.

[23] L.X. Wang, A Course in Fuzzy Systems and Control, Prentice-Hall, Upper Saddle River, NJ, 1997.

[24] M. Margaloit, G. Langholz, Fuzzy control of a benchmark problem: computing with words approach, IEEE Transaction on Fuzzy Systems 12 (2) (April 2004) 230–235.

[25] M. Margaloit, G. Langholz, Fuzzy Lyapunov-based approach to the design of fuzzy controllers, Fuzzy Sets and Systems 106 (1999) 49–59.

[26] L.A. Zadeh, Fuzzy logic = computing with words, IEEE Transaction on Fuzzy Systems 4 (2) (1996) 103–111.

[27] Radu-Emil Precup, Hans Hellendoorn, A survey on industrial applications of fuzzy control, Computers in Industry 62 (2011) 213–226.

[28] T.L. Seng, M. Khalid, R. Yusof, Tuning of a neuro-fuzzy controller by genetic algorithm, IEEE Transaction on Systems, Man, and Cybernetics, Part B: Cybernetics 29 (2) (April 1999) 226–239.

[29] Chun-Tang Chao, Ching-Cheng Teng, A PD-like self-tuning fuzzy controller without steady-state error, Fuzzy Sets and Systems 87 (1997) 141–154.