Expert Systems with Applications 38 (2011) 1017–1031

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A Fuzzy Logic Controller tuned with PSO for 2 DOF robot trajectory control

Zafer Bingül *, Oğuzhan Karahan

Department of Mechatronics Engineering, Kocaeli University, Kocaeli, Turkey

ARTICLE INFO

Keywords:
Fuzzy Logic Controller
PSO
PID
Robot trajectory control

ABSTRACT

In this paper, a 2 DOF planar robot was controlled by Fuzzy Logic Controller tuned with a particle swarm optimization. For a given trajectory, the parameters of Mamdani-type-Fuzzy Logic Controller (the centers and the widths of the Gaussian membership functions in inputs and output) were optimized by the particle swarm optimization with three different cost functions. In order to compare the optimized Fuzzy Logic Controller with different controller, the PID controller was also tuned with particle swarm optimization. In order to test the robustness of the tuned controllers, the model parameters and the given trajectory were changed and the white noise was added to the system. The simulation results show that Fuzzy Logic Controller tuned by particle swarm optimization is better and more robust than the PID tuned by particle swarm optimization for robot trajectory control.

1. Introduction

For linear systems and some of non-severe non-linear systems, the PID controller has been widely used in industrial control processes because of its simple structure and robust performance in a wide range of operating conditions. However, it is quite difficult to determine optimum PID parameters as the system's parameters are coupled, non-linear and time-dependent. In the tuning process of a PID controller, three constants must be selected in such a way that the closed loop system has to give the desired response. The desired response should have minimal settling time with a small or no overshoot in the step response of the closed loop system. Several numerical approaches such as Fuzzy Logic Controller (FLC) algorithm (Visioli, 2001) and evolutionary algorithms (Bingul, 2004; Krohling & Rey, 2001; Mitsukura, Yamamoto, & Kaneda, 1999; Varol & Bingul, 2004) have been used for the optimum design of PID controller.

FLC is popular technique that has seen increasing interest in the past decades since it has a linguistic based structure and its performance are quite robust for non-linear systems. However, FLC including some parameters such as linguistic control rules and limits and type of membership functions have to be tuned for a given system. A major drawback of FLC is that the tuning process becomes more difficult and very time consuming when the number of the system inputs and outputs are increased.

The particle swarm optimization (PSO) is a relatively new evolutionary algorithm that may be used to find optimal or near optimal solutions in big search space. PSO algorithm is especially useful for parameter optimization in continuous, multi-dimensional search spaces. The PSO method can generate a high-quality solution within shorter calculation time and it tends to converge very fast compared to other stochastic methods. Moreover, it is implemented easily in most of the programming languages since the core of the program can be written in a single line of code. The PSO has proven both very effective and quick in diverse set of benchmark optimization problems (Aliyari, Teshnehlab, & Sedigh, 2006; Berenji & Khedkar, 1992; Fuller, 2000; Hoffmann, 2001; Kennedy, Eberhart, & Shi, 2001).

Evolutionary algorithms regarding tuning the membership function parameters of FLC have been studied extensively in the literature. These studies can be divided into three groups as genetic algorithm (Herrera, Lozano, & Verdegay, 1998; Leng, McGinnity, & Prasad, 2006; Seng, Khalid, & Yusof, 1999), PSO (Aliyari, Teshnehlab, & Sedigh, 2007; Chatterjee & Watanabe, 2006; Daniel & Xiaodong, 2006; Eberhart & Kennedy, 1995; Mukherjee & Ghoshal, 2007; Pulasinghe, Chatterjee, & Watanabe, 2005; Wong, Wang, & Li, 2008) and ant algorithm (Juang & Lo, 2007; Juang & Lo, 2008). Wong et al. (2008) proposed a motion control structure with a distance fuzzy controller and an angle fuzzy controller for the two-wheeled mobile robot. They used PSO algorithm to determine automatically appropriate membership functions of the fuzzy systems. Pulasinghe et al. (2005) developed fuzzy–neural networks (FNNs) for navigation of a mobile robot and motion control of a redundant manipulator. They employed PSO to train the FNNs that can accurately output the crisp control signals for the robot systems. Mukherjee and Ghoshal (2007) studied regarding the determination of optimal PID gains for automatic voltage regulator (AVR). In their study, Craziness based particle swarm optimization (CRPSO) and binary coded genetic algorithm (GA) was used to

* Corresponding author.
E-mail addresses: zaferb@kocaeli.edu.tr (Z. Bingül), sebnemkn@hotmail.com (O. Karahan).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.07.131

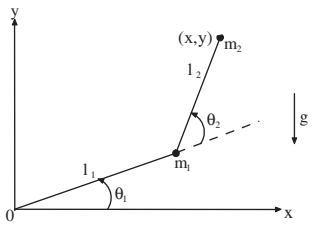

Fig. 1. Model of a two axis robot.

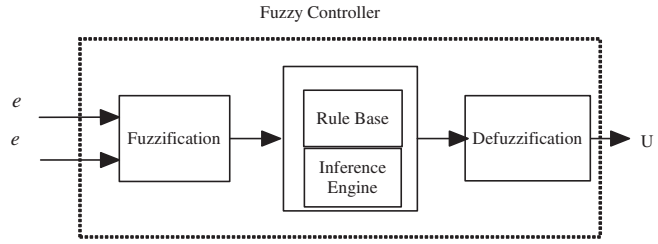

Fig. 2. The structure of a Fuzzy Logic Controller.

obtain the optimal PID gains. Aliyari et al. (2007) introduced a new hybrid approach for training the adaptive network based fuzzy inference system (ANFIS) and used PSO for training procedure parameters in antecedent part. Chatterjee and Watanabe (2006) described a Takagi–Sugeno (TS)-type Neuro-fuzzy system (NFS) trained by PSO for dynamic modeling of two-link and three-link model robot manipulators.

In this work, PSO algorithm was used to tune the Mamdani-type-fuzzy controller's antecedent and consequent parameters for the robot trajectory control. The rest of this article is organized as follows. Dynamic model of the robot arm is described in Section 2. A fuzzy PD control algorithm is proposed in Section 3. The PSO-tuning method for the fuzzy controller and the PID controller is described in Section 4. The experimental results and conclusion are given in Sections 5 and 6, respectively.

2. Dynamic model of the planar robot

The dynamical analysis of the robot investigates a relation between the joint torques/forces applied by the actuators and the position, velocity and acceleration of the robot arm with respect to the time. Robot manipulators have complex non-linear dynamics that might make accurate and robust control difficult. Therefore, they are good examples to test performance of the controllers.

To validate the proposed FLC tuned with PSO, the 2DOF robot shown in Fig. 1 was selected as an example problem. The dynamic equations of the serial robot are usually represented by the following coupled non-linear differential equations:

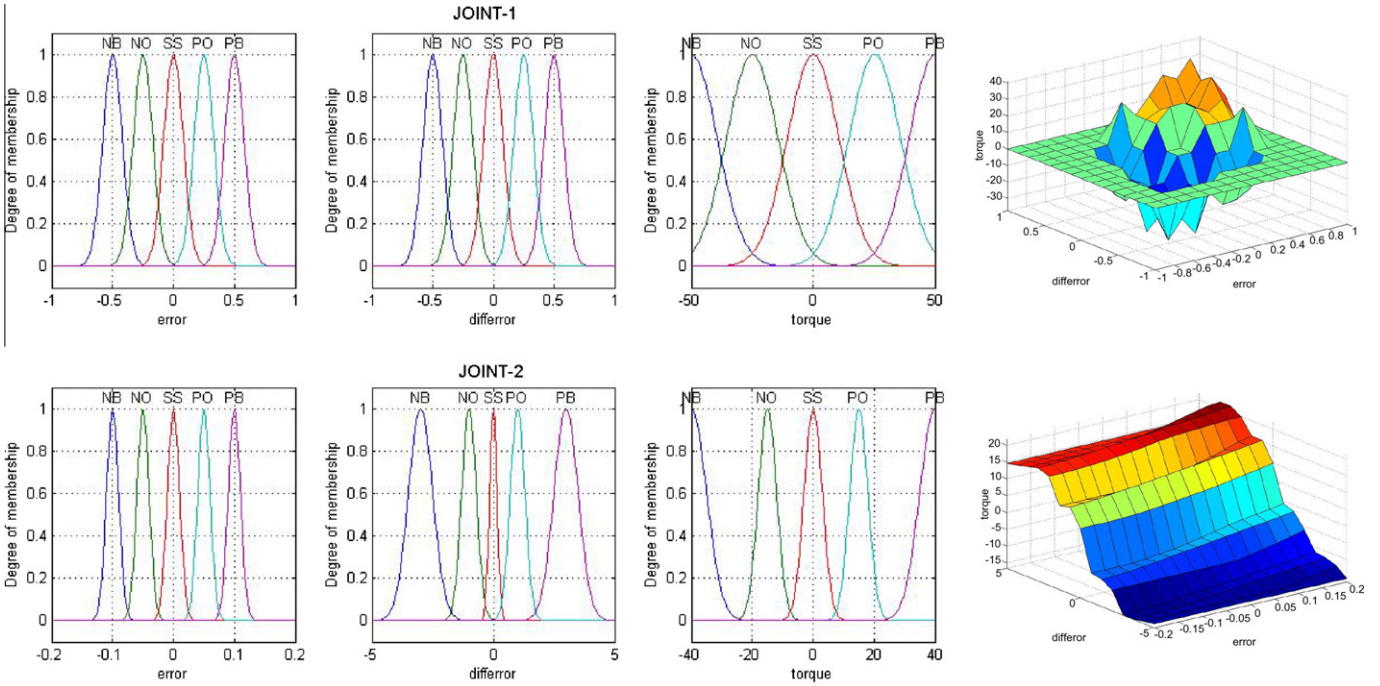

Fig. 3. Membership functions of the fuzzy sets and fuzzy control surfaces.

Table 1
Fuzzy logic rule base.

		Derivative of error				
		NB	NO	SS	PO	PB
Error	NB	NB	NB	NO	NO	SS
	NO	NB	NO	NO	SS	PO
	SS	NO	NO	SS	PO	PO
	PO	NO	SS	PO	PO	PB
	PB	SS	PO	PO	PB	PB

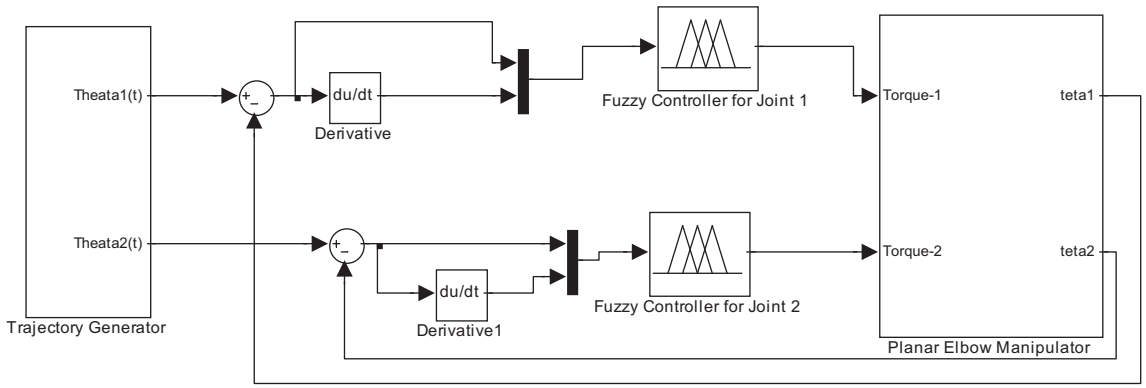

Fig. 4. Simulink model of the FLC for robot trajectory control.

```
For each particle
        Initialize particle with feasible random number
END
Do
   For each particle
      Calculate the fitness value
      If the fitness value is better than the best fitness value (pbest) in history
            Set current value as the new pbest
   End
   Choose the particle with the best fitness value of all the particles as the gbest
   For each particle
         Calculate particle velocity according to velocity update equation
         Update particle position according to position update equation
   End
While maximum iterations or minimum cost criteria is not attained
```

Fig. 5. Pesudo code for PSO.

$$\tau = D(q)\ddot{q} + C(q,\dot{q}) + G(q) \tag{1}$$

where $D(q)$ is the inertia matrix, $C(q,\dot{q})$ is the coriolis/centripetal matrix, $G(q)$ is the gravity vector, and τ is the control input torque. The joint variable q is an n-vector containing the joint angles for revolute joints. The dynamic equation of the 2 DOF planar robot can be computed by:

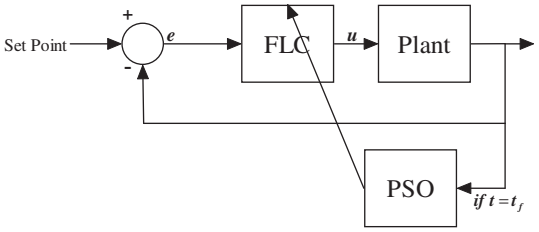

Fig. 6. Tuning process of FLC parameters with PSO.

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{pmatrix}(m_1+m_2)l_1^2 + m_2l_2^2 + 2m_2l_1l_2\cos\theta_2 & m_2l_2^2 + m_2l_1l_2\cos\theta_2\\ m_2l_2^2 + m_2l_1l_2\cos\theta_2 & m_2l_2^2\end{pmatrix}\begin{pmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{pmatrix} + \begin{pmatrix}-m_2l_1l_2(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2)\sin\theta_2\\ m_2l_1l_2\dot{\theta}_1^2\sin\theta_2\end{pmatrix} + \begin{pmatrix}(m_1+m_2)gl_1\cos\theta_1 + m_2gl_2\cos(\theta_1+\theta_2)\\ m_2gl_2\cos(\theta_1+\theta_2)\end{pmatrix} \tag{2}$$

where m_i is link mass, l_i is link length, g is the gravity and θ, $\dot{\theta}$ and $\dot{\theta}$, respectively are the joint positions, velocities and accelerations.

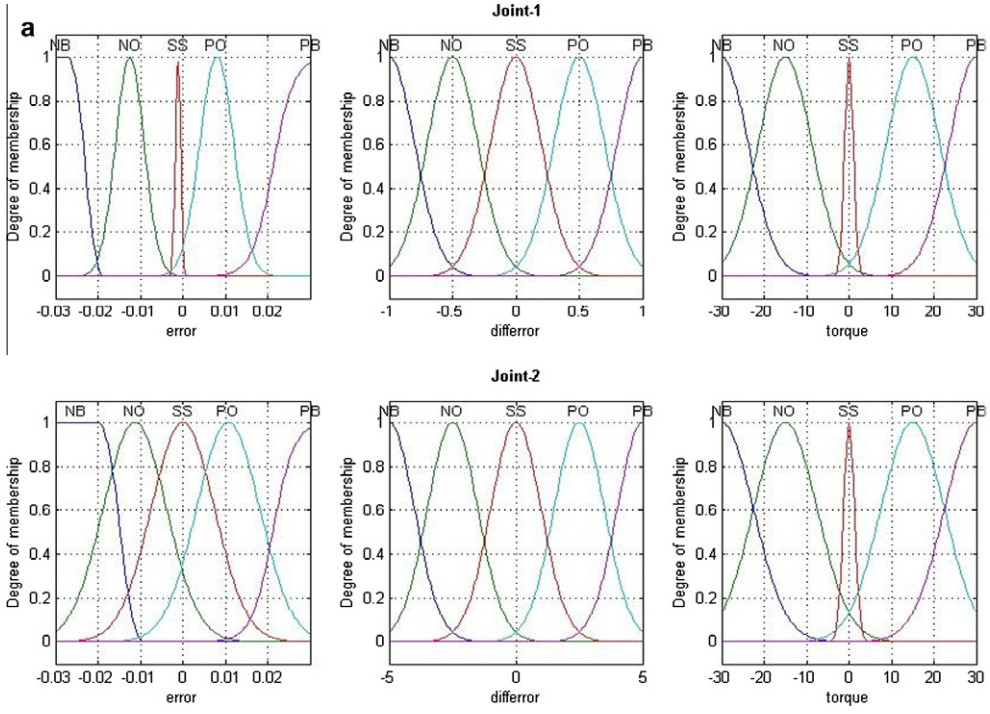

Fig. 7. Membership functions for joint 1 after training with MRSE (a), MAE (b) and (c) RBECE.

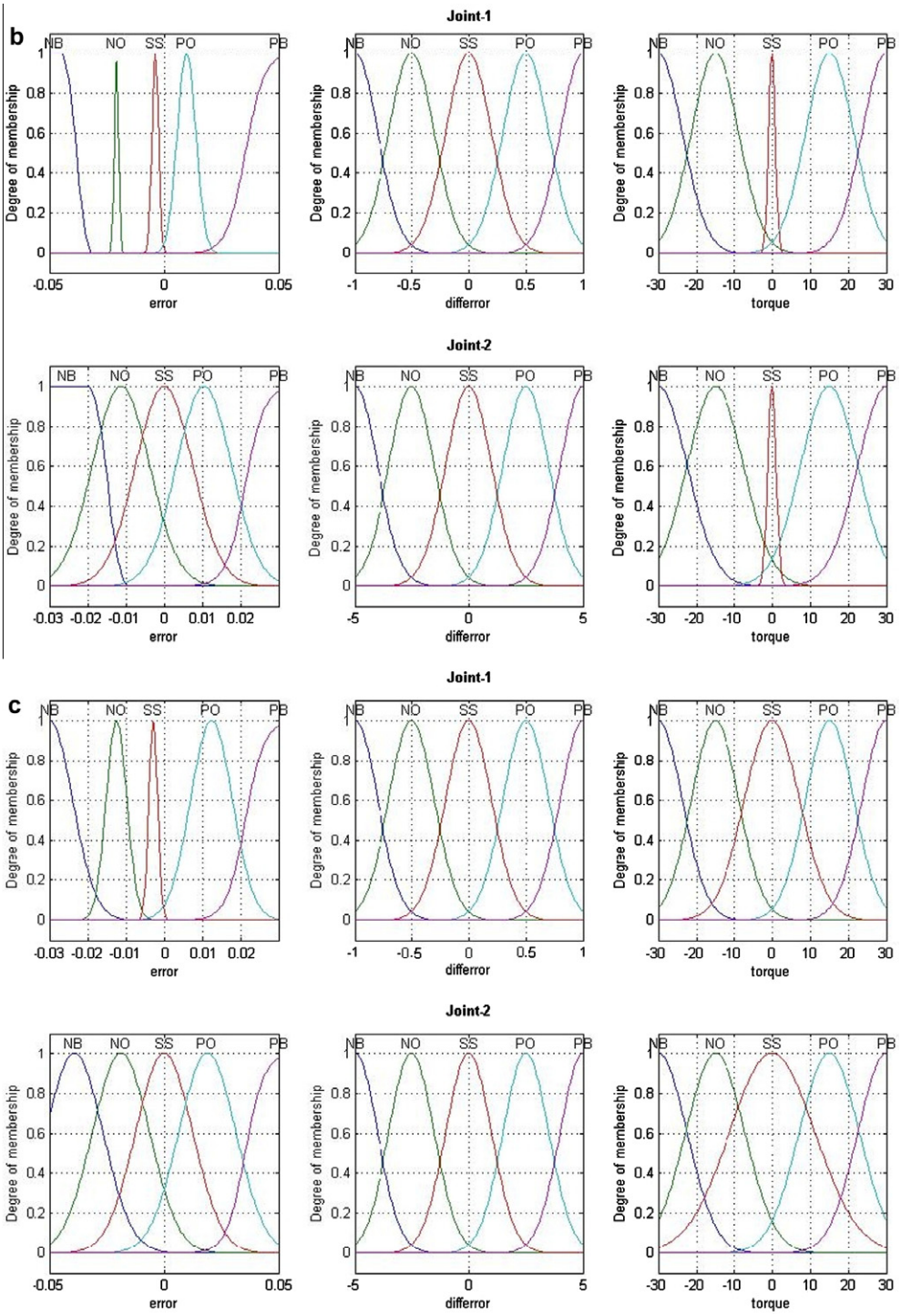

Fig. 7 (*continued*)

3. Fuzzy Logic Controller

The FLC has three main components such as fuzzification, fuzzy inference engine (decision logic), and defuzzification stages. The block diagram of FLC is shown in Fig. 2. The first block in the figure is *fuzzification* which converts each element of input data to degrees of membership by a lookup in one or several membership functions. The rule base and inference base have the capability of simulating human decision-making based on fuzzy concepts and the capability of inferring fuzzy control actions employing fuzzy implication and the rules of inference in fuzzy logic. The membership functions of the fuzzy sets and the fuzzy control rules have a big effect on control performance (Deskur, Muszynski, & Sarnowski, 1998; Simon, 2002; Wang, 1994). The third operation is called as *defuzzification.* The resulting fuzzy set is defuzzified into a crisp control signal. There are five defuzzification methods: centroid, bisector, middle of maximum, largest of maximum, and smallest of maximum (Wang, 1994).

In the robot trajectory control here, the input variables of the FLC are error, e (deg) and error derivation, $\dot{e}$ (deg/s) of joint posi-

tion. The output variable of the fuzzy controller is the joint control input u (*torque*). The values of the e and $\dot{e}$ (Fig. 3) are scaled to the interval of [−1 1] and [−1 1] for the first joint and the interval of [−0.2 0.2] and [−5 5] for the second joint. e and $\dot{e}$ are mapped to linguistic variables *error* and *differror* by the *fuzzification* operator. The FLC inputs are composed of the five linguistic terms *NB* (*Negative Big*), *NO* (*Negative Medium*), *SS* (*Zero*), *PO* (*Positive Medium*) and *PB* (*Positive Big*). Also, the torques for the joints (FLC outputs) is partitioned into the same five fuzzy set. This set of linguistic terms forms a fuzzy partition of input and output spaces. Fig. 3 shows the initial membership functions of the fuzzy sets for the position errors, changes in the position errors, and control inputs and their fuzzy control surfaces.

The Gaussian membership functions were used for both input and output of FLC. Gaussian membership function is defined as,

$$\mu(x) = \exp\left[\frac{-(x-c)^2}{\sigma^2}\right] \tag{3}$$

where c and σ are respectively, the mean and the deviation of a Gaussian membership function. The fuzzy if-then rules for robot trajectory control are given in Table 1. The total number of rules is 25 in Table 1.

The resulting fuzzy set must be converted to a signal that can be sent to the process as a control input. Centroid of area was used here for defuzzification schema. The simulink model of FLC for the robot arm is shown in Fig. 4.

4. Particle swarm optimisation

In this section, PSO algorithm is applied to FLC design for a two axis robot arm. The objective in here is to tune all parameters (the

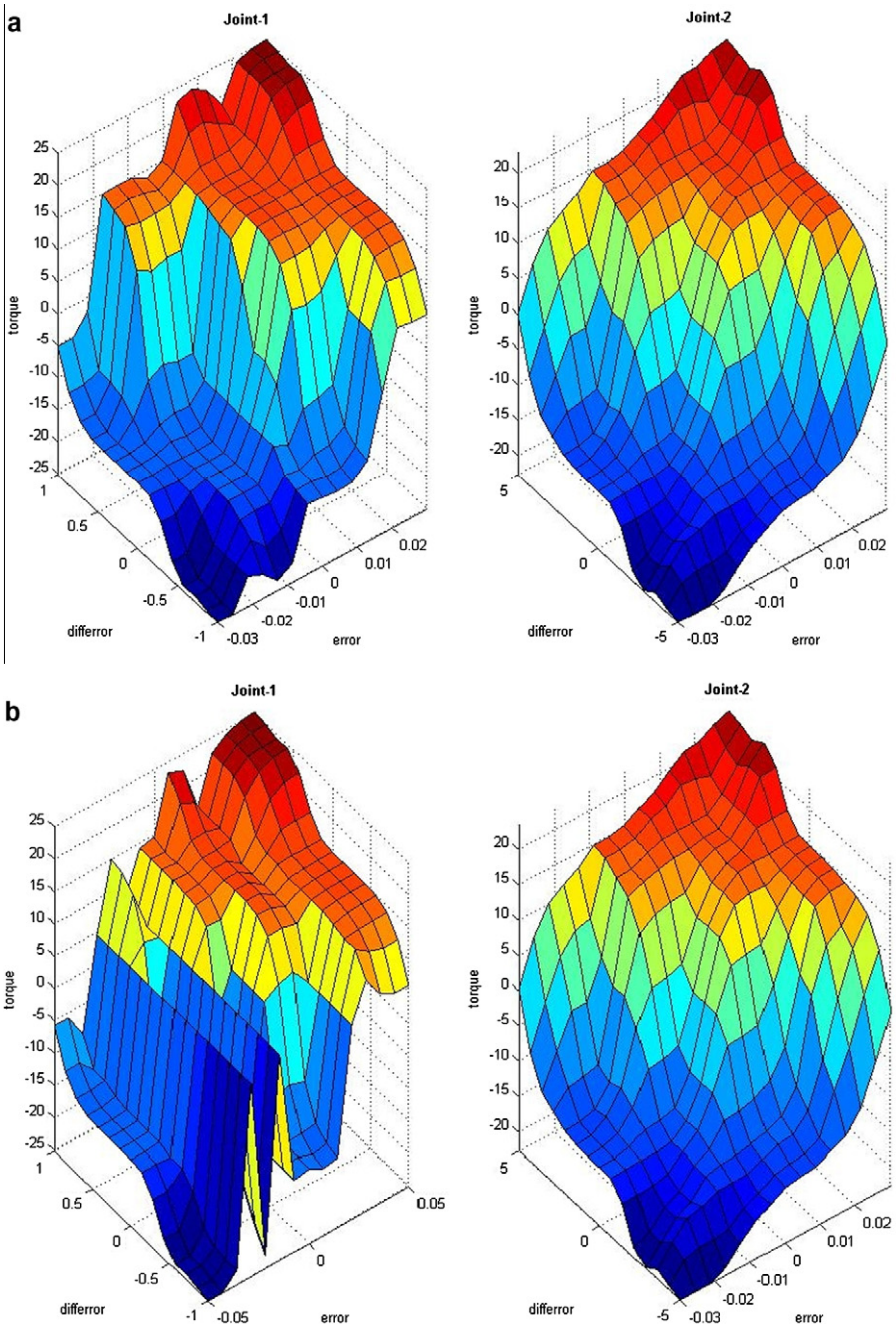

Fig. 8. Fuzzy control surface after training with respect to MRSE (a), MAE (b) and (c) RBECE.

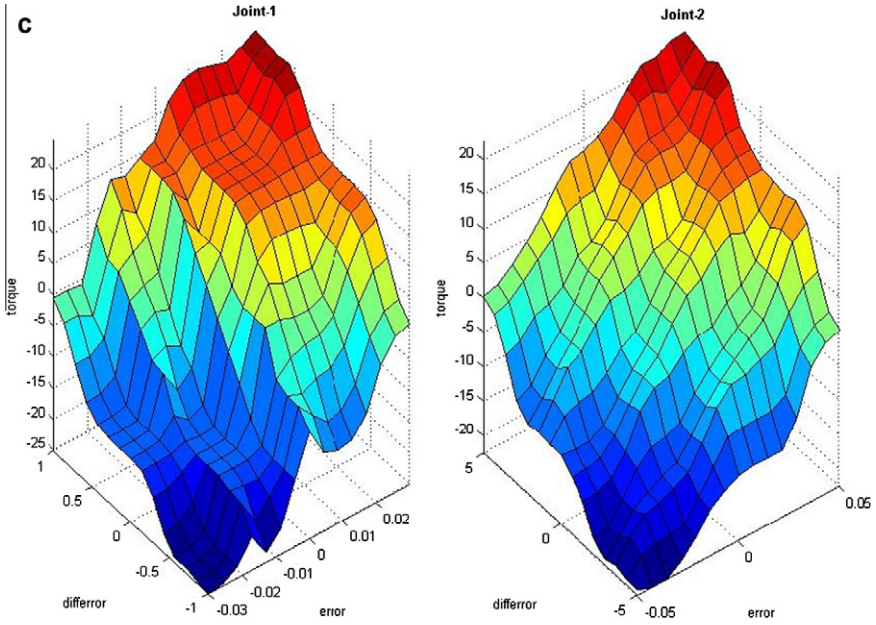

Fig. 8 (*continued*)

fuzzy controller's antecedent and consequent parameters) of the Mamdani type FLC through a given algorithm (Fig. 5) to control the given trajectory.

4.1. PSO algorithm

Evolutionary computational technique based on the movement and intelligence of swarms looking for the most fertile feeding location. A "swarm" is an apparently disorganized collection (population) of moving individuals that tend to cluster together while each individual seems to be moving in a random direction. It uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution (Eberhart & Kennedy, 1995; Kennedy et al., 2001).

Table 2
PID constants for the three different cost functions.

PID parameters	PSO-MRSE		PSO-MAE		PSO-RBECE	
	Joint-1	Joint-2	Joint-1	Joint-2	Joint-1	Joint-2
Kp	973	1064	756	1526	358	465
Ki	591	26	586	96	259	220
Kd	30	9	25	15	84	11

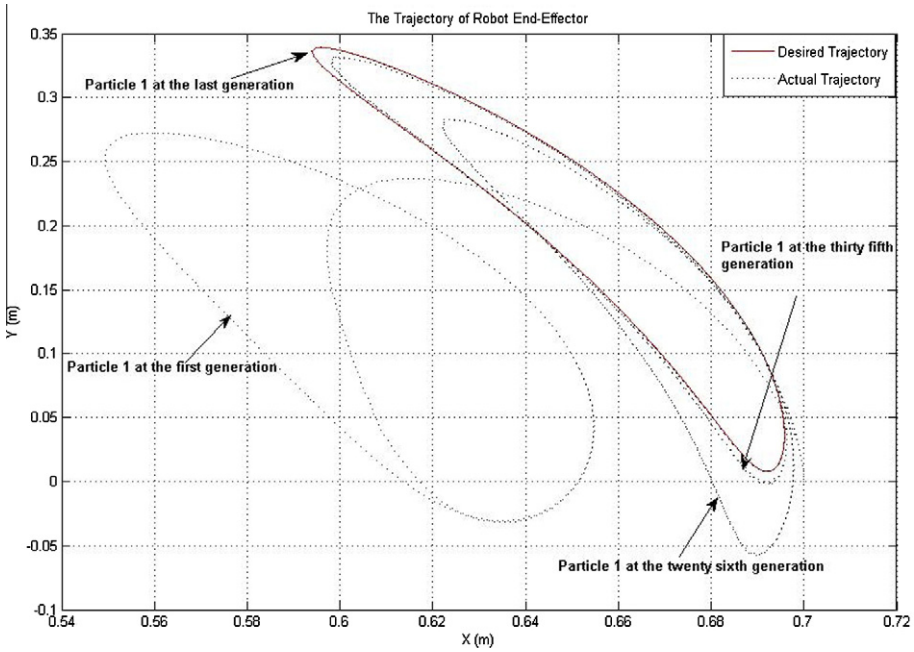

Fig. 9. One particle movement during tuning process.

Table 3
Cost function values of the PID–PSO and the Fuzzy–PSO without any disturbance.

Cost functions	PID–PSO	Fuzzy–PSO
MRSE	0.0029	0.0041
MAE	0.0032	0.0040
RBECE	7.2339	7.2920

Each particle is treated as a point in an *n*-dimensional space which adjusts its "flying" according to its own flying experience as well as the flying experience of other particles. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) that has achieved so far. This value is called *pbest*. Another best value that is tracked by the PSO is the best value obtained so far by any particle in the neighbors of

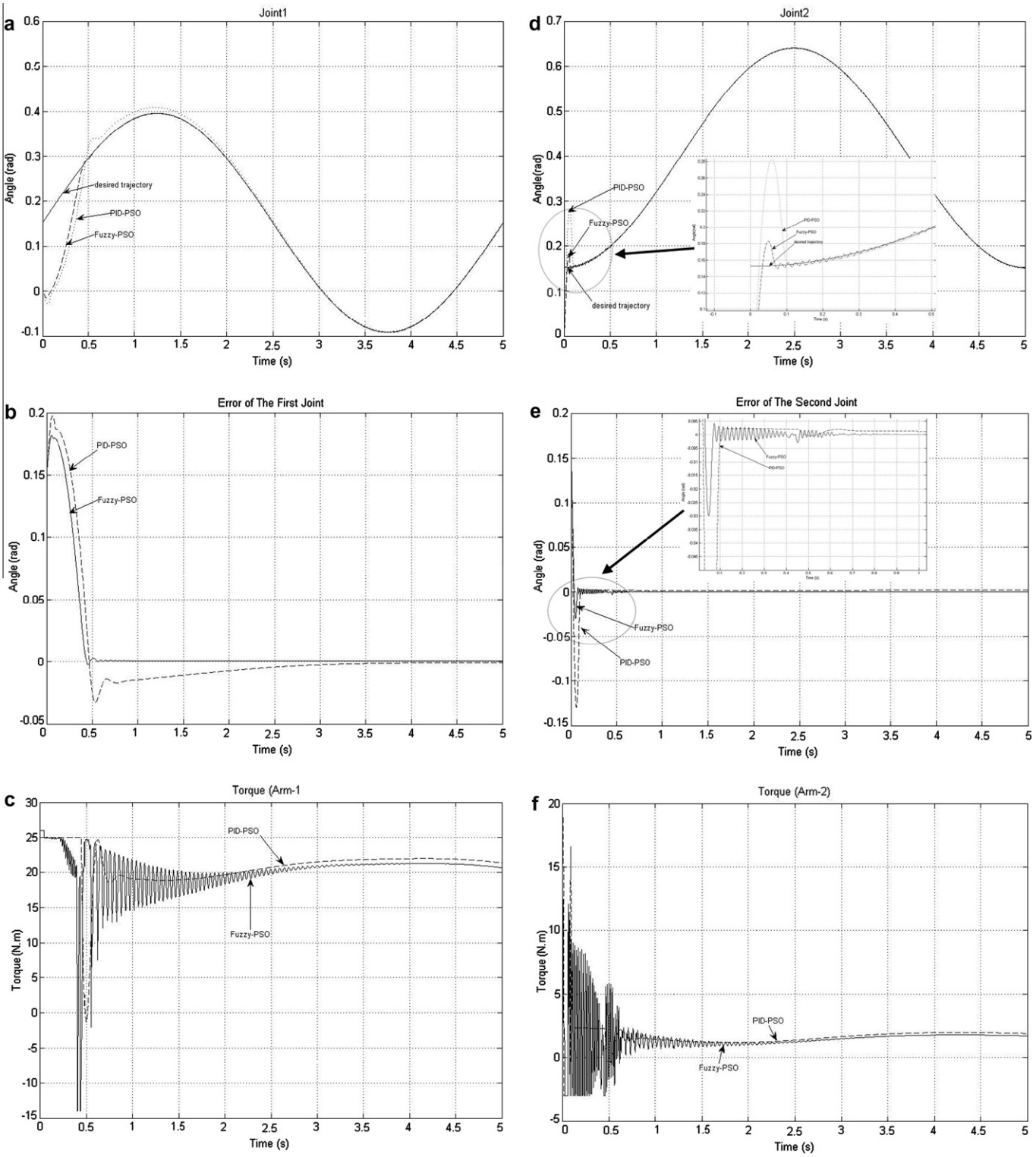

Fig. 10. The performance of *Fuzzy–PSO* and *PID–PSO* controllers with cost function *MRSE* under mass changes for trajectory tracking (a, d), position errors (b, e) and control signal (c, f).

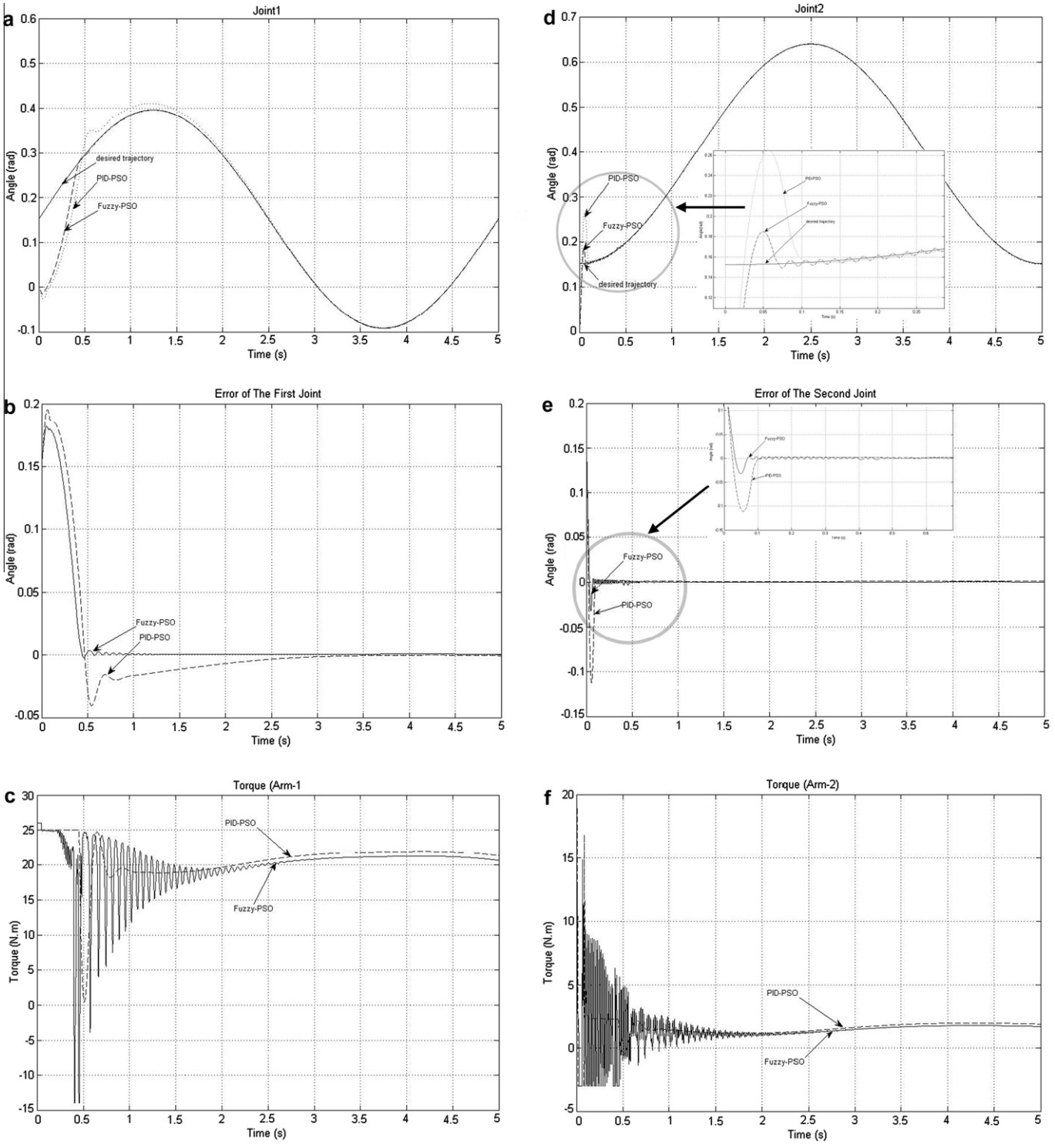

Fig. 11. The performance of *Fuzzy–PSO* and *PID–PSO* controllers with cost function *MAE* under mass changes for trajectory tracking (a, d), position errors (b, e) and control signal (c, f).

the particle. This value is called *gbest* (Daniel and Xiaodong, 2006; Mukherjee & Ghoshal, 2007).

The PSO concept consists of changing the velocity (or accelerating) of each particle toward its *pbest* and the *gbest* position at each time step. Each particle tries to modify its current position and velocity according to the distance between its current position and *pbest*, and the distance between its current position and *gbest* as shown the following. At each step n, by using the individual best position, *pbest*, and global best position, *gbest*, a new velocity for the ith particle is updated by

$$V_i(n) = \chi(V_i(n-1) + \varphi_1 r_1(pbest_i - P_i(n-1)) + \varphi_2 r_2(gbest - P_i(n-1))) \quad (4)$$

where each particle represents a potential solution and has a position represented by a position vector P_i, r_1 and r_2 are random num-

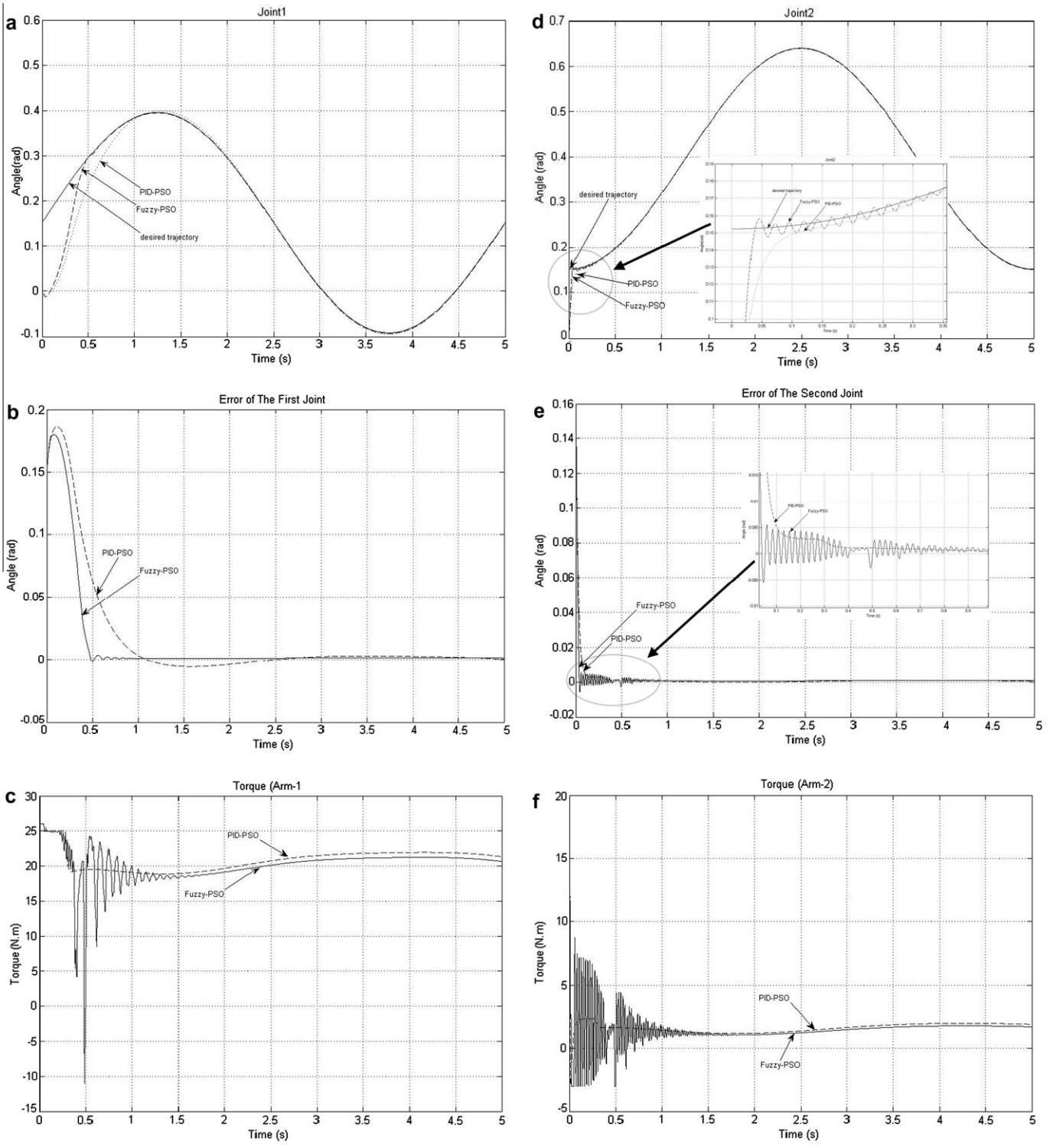

Fig. 12. The performance of *Fuzzy–PSO* and *PID–PSO* controllers with cost function *RBECE* under mass changes for trajectory tracking (a, d), position errors (b, e) and control signal (c, f).

bers between 0 and 1; ϕ_1 and ϕ_2 are positive constant learning rates; χ is called the constriction factor and is defined by (5).

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad \varphi = \varphi_1 + \varphi_2, \quad \varphi > 4 \tag{5}$$

The velocity is confined within the range of $[-v_{\max}, +v_{\max}]$. If the velocity violates these limits, it is forced to its proper values. Changing velocity by this way enables the *i*th particle to search around its local best position, *pbest*, and global best position, *gbest*. Based on the updated velocity, each particle changes its position as following:

$$P_i(n) = P_i(n-1) + V_i(n) \tag{6}$$

4.2. Optimization of FLC with PSO

A basic code structure for PSO is shown in Fig. 5. Since each FLC has 15 MFs and 25 rules, there are total 60 parameters to be opti-

mized in this study. PSO searches all of the antecedent and consequent parameters in 60 dimensional spaces.

The order of a particle is shown as the following

$$P_i = [\sigma_{11}c_{11}\cdots\sigma_{15}c_{15}\sigma_{21}c_{21}\cdots\sigma_{25}c_{25}\sigma_{31}c_{31}\cdots\sigma_{35}c_{35}\sigma_{41}c_{41}\cdots \sigma_{45}c_{45}c_{51}\sigma_{52}\cdots sigma_{55}c_{55}\sigma_{61}c_{61}\cdots\sigma_{65}c_{65}] \quad (7)$$

where the parameters σ_{ij} and c_{ij} represent the center and deviation of the Gaussian MFs. In the above equation, the first line and the second line constitute the parameters of the first controller and the second controller, respectively. The initial values of particles are randomly generated in the first generation.

The most crucial step in applying PSO is to choose the best cost function which are used to evaluate fitness of each particle. In during tuning process with PSO, one of three different cost functions are used as *Mean of Root of Squared Error* (*MRSE*), *Mean of Absolute Magnitude of the Error* (*MAE*) and *Reference Based Error with Control Effort* (*RBECE*). The cost functions for *i*th particle are computed as follows:

$$MRSE = E(k) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(e_1^2(i)+e_2^2(i))} \quad (8)$$

$$MAE = E(k) = \frac{1}{N}\sum_{i=1}^{N}|e_1(i)| + |e_2(i)| \quad (9)$$

$$RBECE = E(k) = \frac{1}{N}\sum_{i=1}^{N}|e_1(i)| + |e_2(i)| + |u_1(i)| + |u_2(i)| \quad (10)$$

where $e_1(i)$ is the trajectory error of *i*th sample for the first joint, $e_2(i)$ is the trajectory error of *i*th sample for the second joint, N is the number of sample, k is the iteration number, $u_1(i)$ and $u_2(i)$ are the control signals for the first joint and the second joint, respectively.

Fig. 6 illustrates the block structure of the FLC optimizing process using PSO. All parameters of the FLC are updated at every final time (t_f).

In here, the population size is set to 10 particles for considering excessive computational load. As each particle P_i has 60 elements, swarm size is 10×60. In PSO block (Fig. 5), the cost function is calculated for each particle and *pbest* and *gbest* are computed for every final time (t_f). A particle velocity (Eq. (4)) is calculated for each particle and a particle position (Eq. (6)) is updated. This algorithm is run until maximum iteration or minimum cost criteria are attainable.

In PSO algorithm, the parameters χ, ϕ_1 and ϕ_2 are set to 0.76, 2.05 and 2.05 (Eberhart & Kennedy, 1995). It should be noted that the values of an element in the particle may exceed its reasonable range. In this case, the centers c_{ij} and widths σ_{ij} of MFs are confined within the ranges of $[c_{\min}, c_{\max}]$ and $[\sigma_{\min}, \sigma_{\max}]$, respectively.

For the first FLC, $c_{\min}$ and $c_{\max}$ are -1 and +1 for the MFs of *error* and *differror* inputs. $\sigma_{\min}$ and $\sigma_{\max}$ are 0.02 and 0.1 for the MFs of *error* and *differror* inputs. $c_{\min}$ and $c_{\max}$ are -50 and +50 for the

Table 4
Cost function values of the controllers under the mass changes.

Cost functions	PID–PSO	Fuzzy–PSO
MRSE	0.0194	0.0115
MAE	0.0211	0.0124
RBECE	22.5424	21.7528

Table 5
MRSE cost function values in the case of adding noise with different variance.

Noise power	Joint 1		Joint 2		Joints 1 and 2	
	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO
0.01	0.0115	0.0183	0.0124	0.0188	0.0122	0.0183
0.1	0.0115	0.0174	0.0134	0.0191	0.013	0.0176
0.5	0.0127	0.0166	0.0167	0.0208	0.0178	0.0186
0.7	0.014	0.0169	0.0194	0.0223	0.0198	0.0205
0.8	0.0147	0.0172	0.0209	0.0232	0.02	0.0218

Table 6
MAE cost function values in the case of adding noise with different variance.

Noise power	Joint 1		Joint 2		Joints 1 and 2	
	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO
0.01	0.0124	0.0192	0.0134	0.0197	0.0134	0.0192
0.1	0.0129	0.0184	0.0146	0.0201	0.0144	0.0188
0.5	0.0146	0.018	0.0185	0.0227	0.0197	0.0206
0.7	0.0155	0.0184	0.0222	0.0248	0.0232	0.0232
0.8	0.0159	0.0187	0.0244	0.0261	0.0239	0.0248

Table 7
RBECE cost function values in the case of adding noise with different variance.

Noise power	Joint 1		Joint 2		Joints 1 and 2	
	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO
0.01	21.6959	22.4923	22.4423	22.6614	22.3042	22.5636
0.1	21.6143	22.3863	23.7178	24.7061	23.3599	24.1921
0.5	21.8345	22.2701	26.9391	29.2854	26.0566	28.0575
0.7	21.9327	22.2752	28.0276	30.7149	27.0011	29.321
0.8	21.972	22.2839	28.4338	31.2986	27.4521	29.8447

MFs of *torque* control input. σ_{min} and σ_{max} are used as 1 and 12 for the MFs of *torque* control input.

For the second FLC, c_{min} and c_{max} are −0.19 and + 0.19 for the MFs of *error* input, −5 and + 5 for the MFs of *differror* input. σ_{min} and σ_{max} are 0.006 and 0.014 for the MFs of *error* input, 0.2 and 0.6 for the MFs of *differror* input. c_{min} and c_{max} are −40 and + 40 for the MFs of *torque* control input. σ_{min} and σ_{max} are 2 and 10 for the MFs of *torque* control input, respectively.

For the given trajectory below, the PSO algorithm with one of three different cost functions is processed for 60 generations and repeated for 10 runs. All the simulations are carried out using a program writing in Matlab 7.0.1.

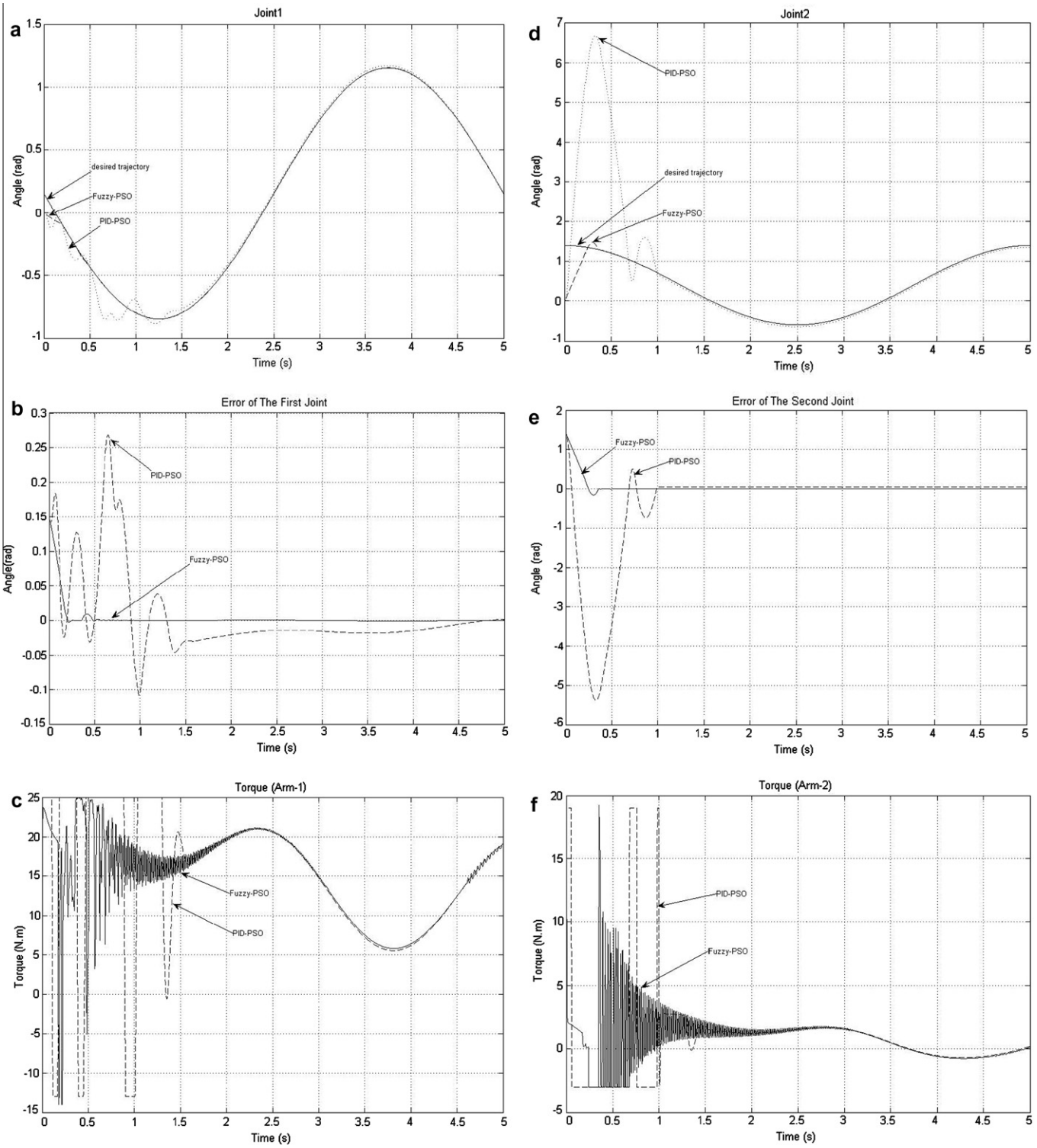

Fig. 13. The performance of *Fuzzy–PSO* and *PID–PSO* controllers with cost function MRSE under changes in both the amplitude and the phase for trajectory tracking (a, d), position errors (b, e) and control signal (c, f).

$$\theta_1(t) = 0.1524 + 0.24384\cos\left(\frac{2\pi t}{5} - \frac{\pi}{2}\right) \quad (11)$$

$$\theta_2(t) = 0.39624 + 0.24384\sin\left(\frac{2\pi t}{5} - \frac{\pi}{2}\right) \quad (12)$$

For the each of three different cost functions (*MRSE*, *MAE* and *RBECE*), the tuned membership functions of FLC and their control surfaces for joint 1 and 2 are shown in Figs. 7 and 8, respectively.

As can be seen from Fig. 7a–c, the initial membership functions are changed based on the given robot trajectory. The changes at the end of training can be summarized as: the membership functions of *error* input for *NB* (*Negative Big*) and *PB* (*Positive Big*) are changed to *spline-based function* and *sigmoidal function*, and the limits of the final membership functions are reduced for position errors (*error*) and control inputs (*torque*) for all cost functions. In order to compare the results obtained from the cost functions *MRSE* and *MAE*,

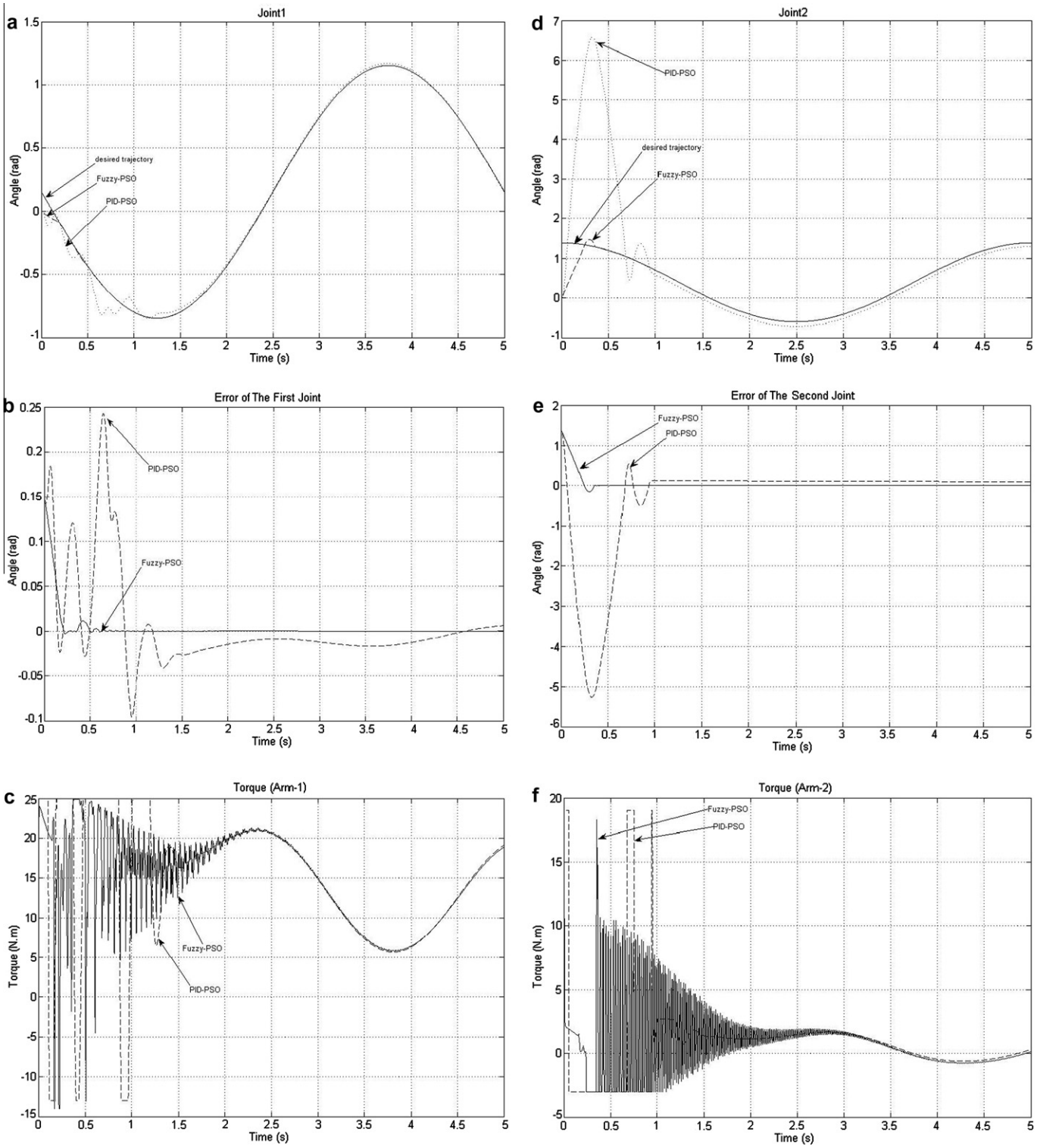

Fig. 14. The performance of *Fuzzy–PSO* and *PID–PSO* controllers with MAE under changes in both the amplitude and the phase for trajectory tracking (a, d), position errors (b, e) and control signal (c, f).

the center and variance parameters of MFs for both joint 1 and 2 have almost the same values. On the other hand, PSO using *RBECE* produces different values of MFs.

In order to compare the tuned control surfaces with the three different cost functions, they are seen to be different to each other. The differences between the tuned surfaces are small for the joint 2 but not for joint 1. Fig. 9 shows movement of the *Particle 1* in the swarm from the first generation to the final generation along the given trajectory.

4.3. Optimization of PID with PSO

Tuning process with PSO using one of the three different cost functions (*MRSE*, *MAE* and *RBECE*) is applied to the PID controller

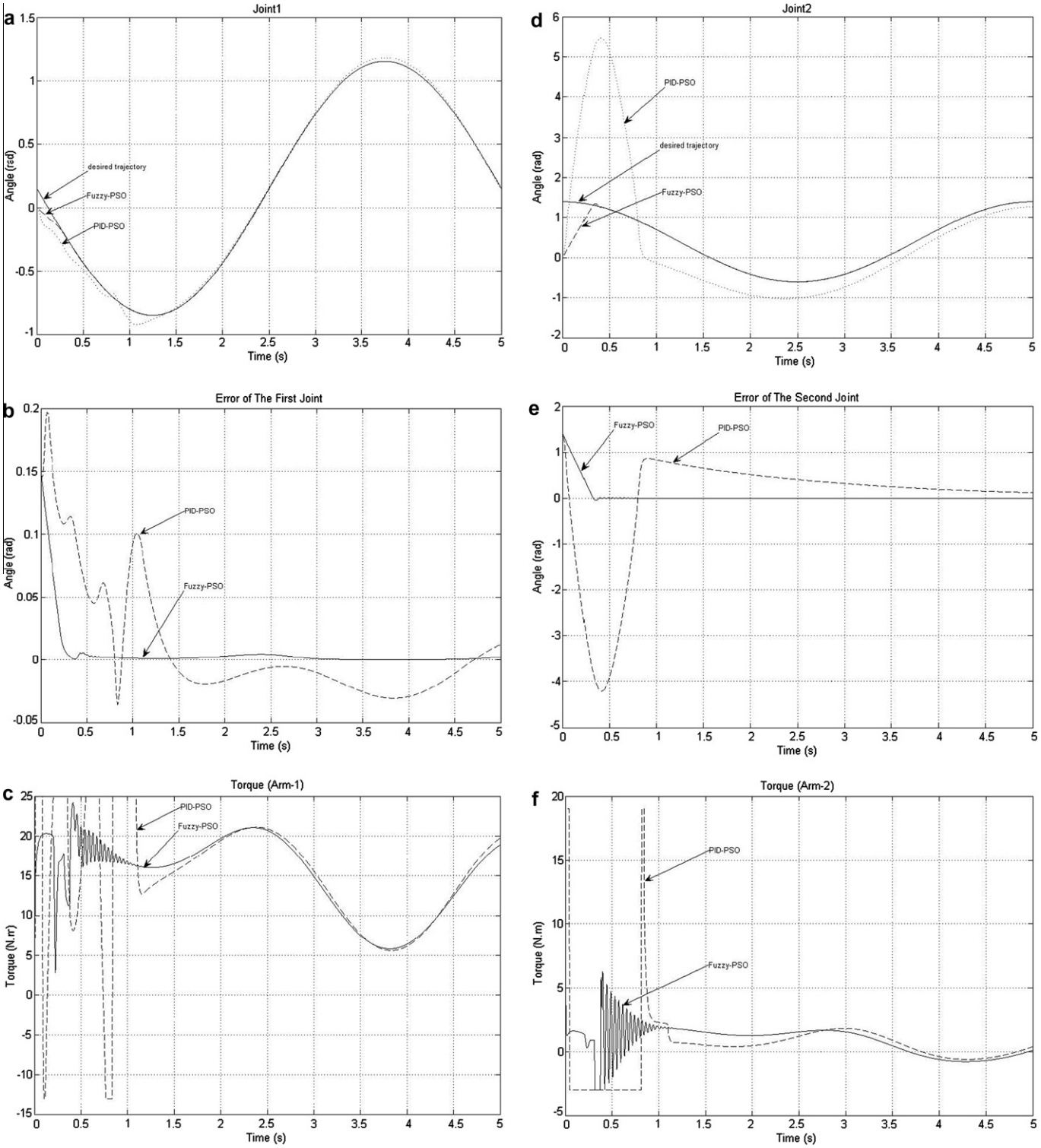

Fig. 15. The performance of *Fuzzy–PSO* and *PID–PSO* controllers with *RBECE* under changes in both the amplitude and the phase for trajectory tracking (a, d), position errors (b, e) and control signal (c, f).

in order to show the differences between the *Fuzzy–PSO* controller and the *PID–PSO* controller and compare their performances. The optimization conditions for both the FLC and the PID controller should be same for accurate comparison. Therefore, the range of the possible values of control signal in optimization process is restricted to upper bound of 26 N m and 20 N m and lower bound of -14 N m and -3 N m for the first joint and the second joint, respectively. The PID constants optimized with the three different cost functions are summarized in Table 2.

It is seen from the Table that the optimized PID constants are different based on the cost function and joint. The PID constants with the cost functions, *MRSE* and *MAE* have almost the same values for joint 1. For joint 2, the differences among PID constants are big and proportional constants are higher than others.

5. Experimental results

The objective of this section is to test the robustness of the *Fuzzy–PSO* controller and the *PID–PSO* controller and show the performances of these controllers. In case of no disturbance, Table 3 gives the values of the three cost functions for these controllers. As can be seen in Table 3, the PID–PSO is better than the Fuzzy–PSO.

In order to test the robustness of the controllers, the mass of the robot arm for each joint is increased to three times. As the controllers' parameters tuned by PSO are kept unchanged, the performance of the fuzzy and the PID controllers for trajectory control are compared. Fig. 10 shows desired, actual trajectory, position error and control signal for each joint. The *Fuzzy–PSO* controller performs better than the *PID–PSO* controller in terms of the convergence and steady state error. On the other, the *Fuzzy–PSO* controller yields fluctuations in the trajectory. Since these fluctuations are getting decreased and become very small, they can be neglected.

The results obtained using cost function *MAE* is shown in Fig. 11. As can be seen, the performance of the Fuzzy–PSO is better compared to the *PID–PSO* controllers. The *Fuzzy–PSO* controller has shorter settling time and fast response time.

The simulation results with cost function *RBECE* are shown in Fig. 12. As can be seen, response time and convergence tendency of the FLC–PSO has faster than that of the *PID–PSO* controller. The values of the cost functions obtained from mass changes were summarized in Table 4. Under the internal disturbance, the FLC–PSO is better than the PID–PSO.

In the second case for the robustness evaluation of the controllers, external disturbances such as the white noise with different variance are added to the robot system. The noise is applied to order of the first joint, the second joint and both of them, respectively. All values of the cost functions for different noise variance are given in Tables 5–7. When the variance of the noise in the tables is increased, it is seen that the deviations from the given tra-

Table 8
MRSE cost function values in the case of changes in the given trajectory.

	Joint 1		Joint 2		Joints 1 and 2	
	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO
A						
0.1	0.0092	0.0112	0.013	0.0223	0.0106	0.0177
0.5	0.0171	0.0284	0.0098	0.0118	0.0147	0.0233
1	0.0369	0.0723	0.0125	0.0121	0.031	0.0543
φ						
0.3927	0.0304	0.0451	0.0135	0.0239	0.0327	0.0518
0.7854	0.0265	0.0401	0.0125	0.0191	0.0277	0.0432
4.7124	0.0059	0.008	0.019	0.178	0.0132	0.1819
A and φ						
0.1	0.0154	0.0213	0.014	0.028	0.0178	0.0326
0.3927						
0.5	0.058	0.0879	0.0112	0.015	0.0576	0.0883
0.7854						
1	0.0037	0.0086	0.0455	0.5935	0.0395	0.4839
4.7124						

Table 9
MAE cost function values in the case of changes in the given trajectory.

	Joint 1		Joint 2		Joint 1 and 2	
	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO
A						
0.1	0.0102	0.0122	0.0146	0.0266	0.0122	0.0216
0.5	0.018	0.0301	0.0105	0.0127	0.0154	0.0244
1	0.0375	0.0751	0.0151	0.0154	0.0341	0.0589
φ						
0.3927	0.0313	0.0477	0.0152	0.0281	0.0345	0.059
0.7854	0.0274	0.0425	0.0138	0.0217	0.029	0.0475
4.7124	0.0067	0.0095	0.0224	0.1871	0.0163	0.1803
A and φ						
0.1	0.0164	0.0232	0.0158	0.0342	0.0199	0.0404
0.3927						
0.5	0.0585	0.0914	0.0118	0.0155	0.0578	0.0907
0.7854						
1	0.0045	0.011	0.0551	0.6366	0.0426	0.5347
4.7124						

Table 10
RBECE cost function values in the case of changes in the given trajectory.

	Joint 1		Joint 2		Joint 1 and 2	
	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO	Fuzzy–PSO	PID–PSO
A						
0.1	21.9532	22.7667	21.7993	22.6194	22.006	22.8431
0.5	20.8613	21.5558	21.5747	22.3917	20.6806	21.4049
1	17.8119	17.9512	21.0969	21.8584	17.1554	17.4827
φ						
0.3927	21.9717	22.6542	21.698	22.5033	21.9437	22.6299
0.7854	21.9352	22.6326	21.7074	22.4983	21.9025	22.5944
4.7124	21.549	22.3137	21.8267	22.714	21.6177	22.5074
A and φ						
0.1	22.0155	22.7905	21.7836	22.6095	22.054	22.8659
0.3927						
0.5	21.5242	21.885	21.4784	22.2566	21.2774	21.5963
0.7854						
1	16.5288	16.9958	21.5076	23.6514	16.4247	17.711
4.7124						

jectory occur and the Fuzzy–PSO produces better results than the PID–PSO.

In the third case of robustness test, the values of the amplitude, *A* and phase, φ in the given trajectory are changed for joint 1, joint 2 and both of them, respectively. Fig. 13 shows the desired and actual positions, errors and control signals of the joints in this case. As can be seen, the *Fuzzy–PSO* controller with *MRSE* has a quick convergence rate and more efficient steady-state response than that of the *PID–PSO* with *MRSE*.

In using of cost functions, *MAE* and *RBECE*, desired and actual trajectories, position errors and control signals of the joints is shown in Figs. 14 and 15, respectively.

Tables 8–10 give the cost functions' values of the controllers with three different cost functions. From the tables, the *Fuzzy–PSO* controller is better than the *PID–PSO* controller in all of the cost functions and all of the changes.

6. Conclusions

This paper introduced the PSO based tuning method for FLC and PID controller to control the given robot trajectory. The all parameters concerning the fuzzy controller and the PID controller were determined using PSO algorithm. In order to examine effects of the cost functions on the controller parameter optimisation, the three different cost functions (*MRSE, MAE* and *RBECE)* was employed in PSO. Also, robustness of the controllers was tested in the case of the mass changes, the presence of noise with different variance and the changes in trajectories. As a result, the *Fuzzy–PSO* controller can achieve better accuracy and has less or no deviation from the trajectory compared to the *PID–PSO* controller. It is verified that the *Fuzzy–PSO* controller has better control performance in robot trajectory control. Furthermore, implementation of the FLC tuning with PSO is much easier than the traditional methods because there is need neither derivative knowledge nor complex mathematical equations.

References

Aliyari, M. Sh., Teshnehlab, M., & Sedigh, A. K. (2006). A novel training algorithm in ANFIS structure. In *Proceedings of American control conference, Minneapolis, Minnesota*.

Aliyari, M. Sh., Teshnehlab, M., & Sedigh, A. K. (2007). A novel hybrid learning algorithm for tuning ANFIS parameters using adaptive weighted PSO. In *Proceedings of IEEE International Fuzzy System Conference, London, UK*.

Berenji, H. R., & Khedkar, P. (1992). Learning and tuning fuzzy-logic controllers through reinforcements. *IEEE Transactions on Neural Networks, 3*(5), 724–740.

Bingul, Z. (2004). *A new PID tuning technique using differential evolution for unstable and integrating processes with time delay*. *LNCS* (Vol. 3316, pp. 254–260) Springer.

Chatterjee, A., & Watanabe, K. (2006). An optimized Takagi–Sugeno type neuro-fuzzy system for modeling robot manipulators. *Neural Computing & Applications, 15*(1), 55–61.

Daniel, P., & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation, 10*(4), 440–458.

Deskur, J., Muszynski, R., & Sarnowski, D. (1998). Tuning and investigation of combined fuzzy controller. In *Proceeding of international conference on power electronics and variable speed drives*.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of sixth international symposium on micromachine human science, Nagoya, Japan*.

Fuller, R. (2000). *Introduction to neuro-fuzzy systems, advances in soft computing series*. Berlin: Springer-Verlag.

Herrera, F., Lozano, M., & Verdegay, J. L. (1998). A learning process for fuzzy control rules using genetic algorithms. *Fuzzy Sets and Systems, 100*(1–3), 143–158.

Hoffmann, F. (2001). Evolutionary algorithms for fuzzy control system design. *Proceedings of the IEEE, 89*(9), 1318–1333.

Juang, C. F., & Lo, C. (2007). Fuzzy systems design by clustering-aided ant colony optimization for plant control. *International Journal of General Systems, 36*, 623–641.

Juang, C. F., & Lo, C. (2008). Zero-order TSK-type fuzzy system learning using a two-phase swarm intelligence algorithm. *Fuzzy Sets and Systems, 159*(21), 2910–2926.

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence*. New York: Morgan Kaufmann.

Krohling, R. A., & Rey, J. P. (2001). Design of optimal disturbance rejection PID controllers using genetic algorithm. *IEEE Transactions on Evolutionary Computation, 5*, 78–82.

Leng, G., McGinnity, T. M., & Prasad, G. (2006). Design for self-organizing fuzzy neural networks based on genetic algorithms. *IEEE Transactions on Fuzzy Systems, 14*, 755–766.

Mitsukura, Y., Yamamoto, T., & Kaneda, M. (1999). A design of self-tuning PID controllers using a genetic algorithm. In *Proceedings of American Control Conference, San Diego, CA* (pp. 1361–1365).

Mukherjee, V., & Ghoshal, S. P. (2007). Intelligent particle swarm optimized fuzzy PID controller for AVR system. *Electric Power Systems Research, 77*, 1689–1698.

Pulasinghe, K., Chatterjee, A., & Watanabe, K. (2005). A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems. *IEEE Transactions on Industrial Electronics, 52*(6), 1478–1489.

Seng, T. L., Khalid, M. B., & Yusof, R. (1999). Tuning of a neuro-fuzzy controller by genetic algorithm. *IEEE Transactions on Systems Man and Cybernetics B, 29*, 226–236.

Simon, D. (2002). Training fuzzy systems with the extended Kalman filter. *Fuzzy Sets and Systems, 132*, 189–199.

Varol, H. A., & Bingul, Z. (2004). A new PID tuning technique using ant algorithm. In *Proceedings of American Control Conference, Boston, Massachusetts*.

Visioli, A. (2001). Tuning of PID controllers with fuzzy logic. *IEE Proceedings of Control Theory and Applications, 148*(1), 1–8.

Wang, L. (1994). *Adaptive fuzzy systems and control: Design and stability analysis*. Englewood Cliffs, NJ: Prentice-Hall.

Wong, C. C., Wang, H. Y., & Li, S. A. (2008). PSO-based motion fuzzy controller design for mobile robots. *International Journal of Fuzzy Systems, 10*(1), 24–32.