Engineering Applications of Artificial Intelligence 25 (2012) 484–493

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

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

PID-type fuzzy logic controller tuning based on particle swarm optimization

S. Bouallègue, J. Haggège, M. Ayadi *, M. Benrejeb

Research Unit LA.R.A Automatique, National Engineering School of Tunis (ENIT), BP 37, Le Belvédère, 1002 Tunis, Tunisia

ARTICLE INFO

Article history:
Received 11 February 2011
Received in revised form
13 July 2011
Accepted 19 September 2011
Available online 19 October 2011

Keywords:
PID-type FLC
Systematic scaling factors tuning
Particle swarm optimization
Genetic algorithms
Real-time experimentation

ABSTRACT

In this paper, a new PID-type fuzzy logic controller (FLC) tuning strategy is proposed using a particle swarm optimization (PSO) approach. In order to improve further the performance and robustness properties of the proposed PID-fuzzy approach, two self-tuning mechanisms are introduced. The scaling factors tuning problem of these PID-type FLC structures is formulated and systematically resolved, using a proposed constrained PSO algorithm. The case of an electrical DC drive benchmark is investigated, within a developed real-time framework, to illustrate the efficiency and superiority of the proposed PSO-based fuzzy control approaches. Simulation and experimental results show the advantages of the designed PSO-tuned PID-type FLC structures in terms of efficiency and robustness.

1. Introduction

The complexity of dynamic system, especially when only qualitative knowledge about the process is available, makes it generally difficult to elaborate an analytic model, which is sufficiently precise enough for the control. Thus, it is interesting to use, for this kind of systems, non-conventional control techniques, such as fuzzy logic, in order to achieve high performances and robustness (Lee, 1990a, 1990b; Passino and Yurkovich, 1998). Fuzzy logic control approach has been widely used in many successful industrial applications, which demonstrated high robustness and effectiveness properties.

In the literature, various fuzzy logic controller (FLC) structures are proposed and extensively studied. The particular structure given in Qiao and Mizumoto (1996), namely PID-type FLC, is especially established and improved within the practical framework in Eker and Torun (2006), Güzelkaya et al. (2003) and Woo et al. (2000). Such a FLC structure, which retains the characteristics similar to the conventional PID controller, can be decomposed into the equivalent proportional, integral and derivative control components as shown in Qiao and Mizumoto (1996). In order to improve further the performance of the transient and steady state responses of this kind of fuzzy controller, various strategies and methods are proposed to tune the PID-type fuzzy controller parameters.

Indeed, Qiao and Mizumoto (1996) designed a parameter adaptive PID-type FLC based on a peak observer mechanism. This self-tuning mechanism decreases the equivalent integral control component of the fuzzy controller gradually with the system response process time. It allows also increasing the damping of the system when it is about to settle down, meanwhile keeps the proportional control component unchanged so as to guarantee fast reaction against the system's error. The oscillation of the system is strongly reduced and the settling time is shortened considerably. On the other hand, Woo et al. (2000) developed a method to tune the scaling factors related to integral and derivative components of the PID-type FLC structure via two empirical functions and based on the system's error information. To achieve some goals of the strategy originally proposed by Qiao and Mizumoto (1996), the used self-tuning mechanism decreases the equivalent integral control component and increases the damping of the system via the equivalent derivative component. Eksin et al. (2001) and Güzelkaya et al. (2003) proposed a technique that adjusts the scaling factors, corresponding to the derivative and integral components of the PID-type FLC, using a fuzzy inference mechanism. This fuzzy inference-based self-tuning mechanism has two inputs, namely the system's error and a new variable called "normalized acceleration". Such a variable gives relative rate information about the fastness or slowness of the system response. This method is more efficient since lesser number of parameters is to be tuned and it is more robust to the system parameter or structural changes compared to the other related methods, as shown in Güzelkaya et al. (2003).

However, the major drawback of all these PID-type FLC structures is the difficult choice of their relative scaling factors. Indeed, the fuzzy controller dynamic behavior depends on this

* Corresponding author. Tel.: +216 71 87 47 00; fax: +216 71 87 27 29.
E-mail addresses: soufiene.bouallegue@enit.rnu.tn (S. Bouallègue), joseph.haggege@enit.rnu.tn (J. Haggège), mounir.ayadi@enit.rnu.tn (M. Ayadi), mohamed.benrejeb@enit.rnu.tn (M. Benrejeb).

0952-1976/$ - see front matter Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2011.09.018

adequate choice. The tuning procedure depends on the control experience and knowledge of the human operator, and it is generally achieved based on a classical trials–errors procedure. Up to now there is no clear and systematic method to guide such a choice. So, this tuning problem becomes more delicate and hard as the complexity of the controlled plant increases. Hence, the proposition of a systematic approach to tune the scaling factors of these particular PID-type FLC structures is interesting.

In this paper, a new approach based on the particle swarm optimization (PSO) meta-heuristic technique is proposed for systematically tuning the scaling factors of the PID-type FLC, with and without self-tuning mechanisms. This work can be considered as an extension of the results given in Qiao and Mizumoto (1996), Eker and Torun (2006), Güzelkaya et al. (2003), Woo et al. (2000) and Eksin et al. (2001). The fuzzy control design is formulated as a constrained optimization problem, which is efficiently solved based on the developed PSO algorithm. In order to specify more robustness and performance control objectives of the proposed PSO-tuned PID-type FLC, different optimization criteria are considered and compared subjected to several various control constraints defined in the time-domain framework. The convergence conditions of the proposed and implemented PSO algorithm are analytically guaranteed and verified. The main contribution of this paper consists of proposing a systematic and hybrid control strategy to track reference trajectories using flatness property of linear systems. The proposed approach is based on a robust PID-type fuzzy control method using the PSO technique.

The remainder of this paper is organized as follows. In Section 2, the proposed fuzzy PID-type FLC structures, with and without self-tuning scaling factors mechanisms, are presented and discussed within the discrete-time framework. The optimization-based problems of the PID-type FLC scaling factors tuning are formulated in Section 3. A constrained PSO algorithm, used in solving the formulated problems, is also described. All PSO-based simulation results are compared with those obtained by the classical Genetic Algorithms Optimization (GAO)-based approach. Section 4 is dedicated to apply the proposed fuzzy control approaches on an electrical DC drive benchmark within an experimental real-time framework based on an Advantech PCI-1710 multifunctions board associated with a PC computer and MATLAB/Simulink environment.

2. PID-type fuzzy control design

In this section, the considered PID-type FLC structures are briefly described within the discrete-time framework based on Qiao and Mizumoto (1996), Eker and Torun (2006), Güzelkaya et al. (2003), Woo et al. (2000) and Eksin et al. (2001).

2.1. Discrete-time PID-type FLC

Proposed by Qiao and Mizumoto (1996) within continuous-time formalism, this particular fuzzy controller structure, called PID-type FLC, retains the characteristics similar to the conventional PID controller. This result remains valid while using a type of FLC with triangular uniformly distributed membership functions for the fuzzy inputs and a crisp output, a product–sum inference and center of gravity defuzzification methods.

Under these conditions, the equivalent proportional, integral and derivative control components of such a PID-type FLC are given by $\alpha K_e\mathcal{P}+\beta K_d\mathcal{D}$, $\beta K_e\mathcal{P}$ and $\alpha K_d\mathcal{D}$, respectively, as shown in Qiao and Mizumoto (1996), Güzelkaya et al. (2003) and Woo et al. (2000). In these expressions, $\mathcal{P}$ and $\mathcal{D}$ represent relative coefficients, K_e, K_d, α and β denoting the scaling factors associated to the inputs and output of the FLC, as shown in Fig. 1. The proof of this computation is shown with more details in Qiao and Mizumoto (1996).

When approximating the integral and derivative terms within the discrete-time framework, we can consider the closed-loop control structure for a discrete-time PID-type FLC, as shown in Fig. 1.

As shown in Qiao and Mizumoto (1996), Eker and Torun (2006), Güzelkaya et al. (2003) and Woo et al. (2000), the dynamic behavior of this PID-type FLC structure is strongly depending on the scaling factors K_e, K_d, α and β, difficult and delicate to tune.

2.2. PID-type FLC with self-tuning mechanisms

In order to improve the performances of the considered PID-type FLC structure, various self-tuning mechanisms for scaling factors have been proposed in the literature. Two methods are especially adopted in this paper.

2.2.1. Self-tuning via Empirical Functions Tuner Method (EFTM)

In this self-tuning method (Woo et al., 2000), the PID-type FLC integral and derivative components updating are achieved based on scaling factors β and K_d, using the information on system's error as follows:

$$\begin{aligned}\beta_k &= \beta_0\Phi(e_k)\\ K_{dk} &= K_{d0}\Psi(e_k)\end{aligned} \tag{1}$$

where β_0 and K_{d0} are the initial values of β and K_d, respectively; $\Phi(\cdot)$ and $\Psi(\cdot)$ are the empirical tuner functions defined, respectively, by

$$\begin{aligned}\Phi(e_k) &= \phi_1|e_k|+\phi_2\\ \Psi(e_k) &= \psi_1(1-|e_k|)+\psi_2\end{aligned} \tag{2}$$

In these equations, the parameters to be tuned φ_1, φ_2, ψ_1 and ψ_2 are all positives. The empirical function related to integral

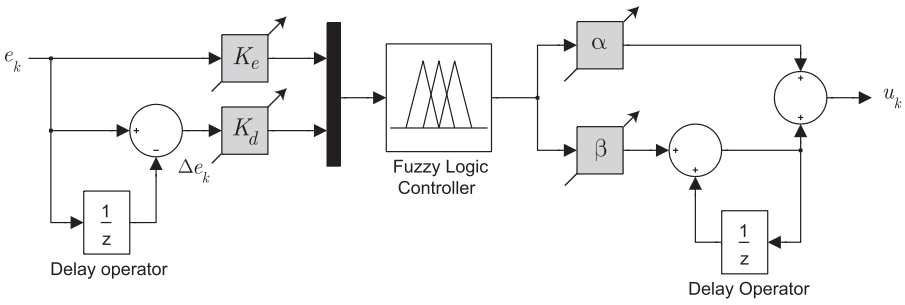

Fig. 1. The proposed discrete-time PID-type FLC structure.

component decreases as the error decreases while the function related to derivative factor increases. Indeed, the objective of the function $\Phi(.)$ is to decrease the β_k parameter with the change of error. However, the function $\Psi(.)$ has an inverse objective to make constant the proportional effect. Hence, the system may not always keep quick reaction against the error (Woo et al., 2000).

2.2.2. Self-tuning via Relative Rate Observer Method (RROM)

In this self-tuning method (Güzelkaya et al., 2003; Eksin et al., 2001), the PID-type FLC integral and derivative components updating is achieved as follows:

$$\begin{aligned}\beta_k &= \beta_0/K_f\delta_k\\ K_{dk} &= K_{d0}K_{fd}K_f\delta_k\end{aligned} \tag{3}$$

where δ_k is the output of the fuzzy Relative Rate Observer (RRO), K_f is the output scaling factor for δ_k and K_{fd} is the additional parameter that affects only the derivative factor of the FLC.

The rule-base for δ_k, as used in Eksin et al. (2001) and Güzelkaya et al. (2003), is considered for the fuzzy RRO. This fuzzy RRO block has as inputs the absolute values of error $|e_k|$ and the variable r_k, defined subsequently.

The variable r_k, defined in Güzelkaya et al. (2003) and Eksin et al. (2001) and called normalized acceleration, gives "relative rate" information about the fastness or slowness of the system response as showed in Güzelkaya et al. (2003). It is defined as follows:

$$r_k = \frac{\Delta e_k - \Delta e_{k-1}}{\Delta e^*} = \frac{\Delta(\Delta e_k)}{\Delta e^*} \tag{4}$$

where Δe_k and $\Delta(\Delta e_k)$ are the incremental change in error and the so-called acceleration in error given, respectively, by

$$\Delta e_k = e_k - e_{k-1} \tag{5}$$

$$\Delta(\Delta e_k) = \Delta e_k - \Delta e_{k-1} \tag{6}$$

In Eq. (4), the variable Δe^* is chosen as follows:

$$\Delta e^* = \begin{cases} \Delta e_k & \text{if} \quad |\Delta e_k| \geq |\Delta e_{k-1}| \\ \Delta e_{k-1} & \text{if} \quad |\Delta e_k| < |\Delta e_{k-1}| \end{cases} \tag{7}$$

For this RROM self-tuning approach, triangular uniformly distributed and symmetrical membership functions, as used in Güzelkaya et al. (2003), are assigned for the fuzzy inputs r_k and $|e_k|$ and fuzzy output variable δ_k.

3. The proposed PSO-based approach

In this section, the problem of scaling factors tuning, for all defined PID-type FLC structures, is formulated as a constrained optimization problem, which is solved using the proposed PSO-based approach.

3.1. PID-type FLC tuning problem formulation

The choice of the adequate values for the scaling factors of each PID-type FLC structure is often done by a trials–errors hard procedure. This tuning problem becomes difficult and delicate without a systematic design method. To deal with these difficulties, the optimization of these scaling factors is proposed like a promising solution. This tuning problem can be formulated as the following constrained optimization problem:

$$\begin{cases} \underset{\boldsymbol{x}\in\mathbf{D}}{\text{minimize}}\, f(\boldsymbol{x}) \\ \text{subject to} \\ g_l(\boldsymbol{x}) \leq 0; \quad \forall l = 1,\ldots,n_{con} \end{cases} \tag{8}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ the cost function, $\mathbb{D} = \{\boldsymbol{x} \in \mathbb{R}^m;\ \boldsymbol{x}_{\min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\max}\}$ the initial search space, which is supposed to contain the desired design parameters, and $g_l : \mathbb{R}^m \to \mathbb{R}$ the problem's constraints.

The optimization-based tuning problem consists of finding the optimal decision variables $\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_m^*)^T$, representing the scaling factors of a given PID-type FLC structure, which minimize the defined cost function, chosen as the Maximum Overshoot (MO) and the Integral of Absolute Error (IAE) performance criteria. These cost functions are minimized, using the proposed constrained PSO algorithm, under various time-domain control constraints such as overshoot D, steady state error E_{ss}, rise time t_r and settling time t_s of the system's step response, as shown in Eqs. (9)–(11).

Hence, in the case of the PID-type FLC structure without self-tuning mechanisms, the scaling factors to be optimized are K_e, K_d, α and β. The formulated optimization problem is defined as follows:

$$\begin{cases} \underset{x=(K_e,K_d,\alpha,\beta)^T \in \mathbb{R}_+^4}{\text{minimize}}\, f(\boldsymbol{x}) \\ \text{subject to} \\ D \leq D^{\max};\ \ t_s \leq t_s^{\max};\ \ t_r \leq t_r^{\max};\ \ E_{ss} \leq E_{ss}^{\max} \end{cases} \tag{9}$$

where $D^{\max}$, $E_{ss}^{\max}$ $t_r^{\max}$ and $t_s^{\max}$ are the specified overshoot, steady state, rise and settling times, respectively, that constraint the step response of the PSO-tuned PID-type FLC controlled system, and can define some time-domain templates.

In the case of the PID-type FLC structure with the EFTM self-tuning mechanism, the scaling factors to be optimized are ϕ_1, ϕ_2, ψ_1 and ψ_2. The formulated optimization problem is defined as follows:

$$\begin{cases} \underset{x=(\varphi_1,\varphi_2,\psi_1,\psi_2)^T \in \mathbb{R}_+^4}{\text{minimize}}\, f(\boldsymbol{x}) \\ \text{subject to} \\ D \leq D^{\max};\ \ t_s \leq t_s^{\max};\ \ t_r \leq t_r^{\max};\ \ E_{ss} \leq E_{ss}^{\max} \end{cases} \tag{10}$$

For the PID-type FLC structure with the RROM self-tuning mechanism, the scaling factors to be optimized are K_f and K_{fd}. The formulated optimization problem is defined as follows:

$$\begin{cases} \underset{x=(K_f,K_{fd})^T \in \mathbb{R}_+^2}{\text{minimiser}}\, f(\boldsymbol{x}) \\ \text{subject to} \\ D \leq D^{\max};\ \ t_s \leq t_s^{\max};\ \ t_r \leq t_r^{\max};\ \ E_{ss} \leq E_{ss}^{\max} \end{cases} \tag{11}$$

3.2. Particle swarm optimization technique

In this study, the proposed PSO approach is presented and a constrained PSO algorithm is also developed. The convergence conditions of such an algorithm are analyzed and established.

3.2.1. Overview

The PSO technique is an evolutionary computation method developed by Eberhart and Kennedy (1995). This recent meta-heuristic technique is inspired by the swarming or collaborative behavior of biological populations. The cooperation and the exchange of information between population individuals allow solving various complex optimization problems (Ruben and Kamran, 2007; Eberhart and Shi, 2001; Poli et al., 2007; Shi and Eberhart, 1999; Van den Bergh, 2006).

Without any regularity on the cost function to be optimized, the recourse to this stochastic and global optimization technique is justified by the empirical evidence of its superiority in solving a variety of non-linear, non-convex and non-smooth problems. In comparison with other meta-heuristics, this optimization technique

is a simple concept, easy to implement and a computationally efficient algorithm (Poli et al., 2007; Eberhart and Shi, 2001; Van den Bergh, 2006). The convergence and parameters selection of the PSO algorithm are proved using several advanced theoretical analysis (Ruben and Kamran, 2007; Van den Bergh, 2006). Its stochastic behavior allows overcoming the local minima problem.

Particle swarm optimization has been enormously successful in several and various industrial domains. It has been used across a wide range of engineering applications (Marinakis et al., 2010; Samanta and Nataraj, 2009; Liu et al., 2008; Venayagamoorthy et al., 2007). These applications can be summarized around domains of robotics, image and signal processing, electronic circuits design and communication networks, but more especially the domain of plant control design, as shown in Bouallègue et al. (2010a, 2010b, 2011).

3.2.2. BASIC PSO algorithm

The basic PSO algorithm uses a swarm consisting of n_p particles (i.e. $\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^{n_p}$), randomly distributed in the considered initial search space, to find an optimal solution $\boldsymbol{x}^*=\text{argmin}\, f(\boldsymbol{x})\in\mathbb{R}^m$ of a generic optimization problem (8). Each particle, that represents a potential solution, is characterized by a position and a velocity given by $\boldsymbol{x}_k^i:=(x_k^{i,1},x_k^{i,2},\ldots,x_k^{i,m})^T$ and $\boldsymbol{v}_k^i:=(v_k^{i,1},v_k^{i,2},\ldots,v_k^{i,m})^T$ where $(i,k)\in[1,n_p]\times[1,k_{\max}]$.

At each algorithm iteration, the ith particle position, $\boldsymbol{x}^i\in\mathbb{R}^m$, evolves based on the following update rules:

$$\boldsymbol{x}_{k+1}^i=\boldsymbol{x}_k^i+\boldsymbol{v}_{k+1}^i \tag{12}$$

$$\boldsymbol{v}_{k+1}^i=w_{k+1}\boldsymbol{v}_k^i+c_1r_{1,k}^i(\boldsymbol{p}_k^i-\boldsymbol{x}_k^i)+c_2r_{2,k}^i(\boldsymbol{p}_k^g-\boldsymbol{x}_k^i) \tag{13}$$

where w_{k+1} is the inertia factor; c_1 and c_2 the cognitive and the social scaling factors, respectively; $r_{1,k}^i$, $r_{2,k}^i$ the random numbers uniformly distributed in the interval [0,1]; $\boldsymbol{p}_k^i$ the best previously obtained position of the ith particle; $\boldsymbol{p}_k^g$ the best obtained position in the entire swarm at the current iteration k.

Hence, the principle of a particle displacement in the swarm is graphically shown in Fig. 2, for a two dimensional design space.

In order to improve the exploration and exploitation capacities of the proposed PSO algorithm, we choose for the inertia factor a linear evolution with respect to the algorithm iteration as given in Shi and Eberhart (1999):

$$w_{k+1}=w_{\max}-\left(\frac{w_{\max}-w_{\min}}{k_{\max}}\right)k \tag{14}$$

where $w_{\max}=0.9$ and $w_{\max}=0.4$ represent the maximum and minimum inertia factor values, respectively, and $k_{\max}$ is the maximum iteration number.

Similar to other meta-heuristic methods, the PSO algorithm is originally formulated as an unconstrained optimizer. Several techniques have been proposed to deal with constraints. One useful approach is by augmenting the cost function of problem (8) with penalties proportional to the degree of constraint infeasibility. In this paper, the following external static penalty technique is used:

$$\varphi(\boldsymbol{x})=f(\boldsymbol{x})+\sum_{l=1}^{n_{con}}\lambda_l\max[0,g_l(\boldsymbol{x})^2] \tag{15}$$

where λ_l is a prescribed scaling penalty parameters and n_{con} is the number of problem constraints $g_l(x)$.

Finally, the basic proposed PSO algorithm can be summarized by the following steps:

1. Define all PSO algorithm parameters such as swarm size n_p, maximum and minimum inertia factor values, cognitive c_1 and social c_2 scaling factors, etc.
2. Initialize the n_p particles with randomly chosen positions $\boldsymbol{x}_0^i$ and velocities $\boldsymbol{v}_0^i$ in the search space $\mathbb{D}$. Evaluate the initial population and determine $\boldsymbol{p}_0^i$ and $\boldsymbol{p}_0^g$.
3. Increment the iteration number k. For each particle apply the update Eqs. (12) and (13), and evaluate the corresponding fitness values $\varphi_k^i=\varphi(\boldsymbol{x}_k^i)$:
   (i) if $\varphi_k^i\le pbest_k^i$ then $pbest_k^i=\varphi_k^i$ and $\boldsymbol{p}_k^i=\boldsymbol{x}_k^i$;
   (ii) if $\varphi_k^i\le gbest_k$ then $gbest_k=\varphi_k^i$ and $\boldsymbol{p}_k^g=\boldsymbol{x}_k^i$

   where $pbest_k^i$ and $gbest_k$ represent the best previously fitness of the ith particle and the entire swarm, respectively.
4. If the termination criterion is satisfied, the algorithm terminates with the solution $\boldsymbol{x}^*=\arg\min_{\boldsymbol{x}_k^i}\{f(\boldsymbol{x}_k^i),\ \forall i,k\}$. Otherwise, go to step 3.

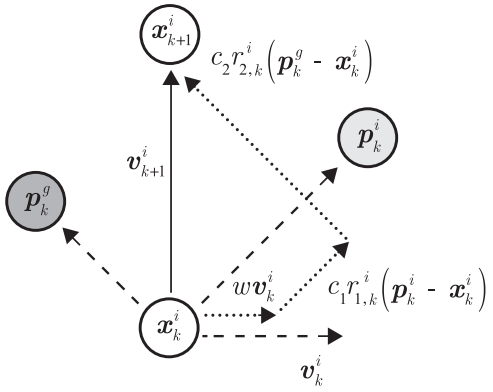

Fig. 2. Particle's position and velocity update.

3.2.3. The convergence of PSO algorithm analysis

In this part, the proposed PSO algorithm is analyzed based on the results of Ruben and Kamran (2007) and Van den Bergh (2006). Theoretical conditions for convergence algorithm and parameters choice are established.

Let us replace the velocity update Eq. (13) into the position update Eq. (12) to get the following expression:

$$\boldsymbol{x}_{k+1}^i=(1-c_1r_{1,k}^i-c_2r_{2,k}^i)\boldsymbol{x}_k^i+w\boldsymbol{v}_k^i+c_1r_{1,k}^i\boldsymbol{p}_k^i+c_2r_{2,k}^i\boldsymbol{p}_k^g \tag{16}$$

A similar re-arrangement of the velocity term (13) leads to

$$\boldsymbol{v}_{k+1}^i=-(c_1r_{1,k}^i+c_2r_{2,k}^i)\boldsymbol{x}_k^i+w\boldsymbol{v}_k^i+c_1r_{1,k}^i\boldsymbol{p}_k^i+c_2r_{2,k}^i\boldsymbol{p}_k^g \tag{17}$$

The obtained Eqs. (16) and (17) can be combined and written in matrix form as follows:

$$\begin{bmatrix}\boldsymbol{x}_{k+1}^i\\ \boldsymbol{v}_{k+1}^i\end{bmatrix}=\begin{bmatrix}1-(c_1r_{1,k}^i+c_2r_{2,k}^i) & w\\ -(c_1r_{1,k}^i+c_2r_{2,k}^i) & w\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_k^i\\ \boldsymbol{v}_k^i\end{bmatrix}+\begin{bmatrix}c_1r_{1,k}^i & c_2r_{2,k}^i\\ c_1r_{1,k}^i & c_2r_{2,k}^i\end{bmatrix}\begin{bmatrix}\boldsymbol{p}_k^i\\ \boldsymbol{p}_k^g\end{bmatrix} \tag{18}$$

This above expression can be considered as a state-space representation of a discrete-time dynamic linear system, given by

$$\hat{y}_{k+1}=\mathcal{M}\hat{y}_k+\mathcal{N}\hat{u}_k \tag{19}$$

where $\hat{y}_k$ is the state vector, $\hat{u}_k$ the external input system, $\mathcal{M}$ and $\mathcal{N}$ the dynamic and input matrices, respectively, defined as follows:

$$\hat{y}_k=\begin{bmatrix}\boldsymbol{x}_k^i\\ \boldsymbol{v}_k^i\end{bmatrix};\quad \hat{u}_k=\begin{bmatrix}\boldsymbol{p}_k^i\\ \boldsymbol{p}_k^g\end{bmatrix};\quad \mathcal{M}=\begin{bmatrix}1-\left(c_1r_{1,k}^i+c_2r_{2,k}^i\right) & w\\ -\left(c_1r_{1,k}^i+c_2r_{2,k}^i\right) & w\end{bmatrix};$$

$$\mathcal{N}=\begin{bmatrix}c_1r_{1,k}^i & c_2r_{2,k}^i\\ c_1r_{1,k}^i & c_2r_{2,k}^i\end{bmatrix} \tag{20}$$

For a given particle, the convergent behavior can be maintained while assuming that the external input is constant, as there is no external excitation in the dynamic system. In such a case, as the iterations go to infinity the updated positions and velocities become constants from the kth to the $(k+1)$th iteration, given the following equilibrium state:

$$\hat{y}_{k+1}-\hat{y}_k=\begin{bmatrix}-(c_1r^i_{1,k}+c_2r^i_{2,k}) & w\\ -(c_1r^i_{1,k}+c_2r^i_{2,k}) & w-1\end{bmatrix}\begin{bmatrix}\boldsymbol{x}^i_k\\ \boldsymbol{v}^i_k\end{bmatrix}+\begin{bmatrix}c_1r^i_{1,k} & c_2r^i_{2,k}\\ c_1r^i_{1,k} & c_2r^i_{2,k}\end{bmatrix}\begin{bmatrix}\boldsymbol{p}^i_k\\ \boldsymbol{p}^g_k\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix} \tag{21}$$

which is true only when

$$\boldsymbol{x}^i_k=\boldsymbol{p}^i_k=\boldsymbol{p}^g_k,\quad \boldsymbol{v}^i_k=0 \tag{22}$$

Therefore, we obtain an equilibrium point, for which all particles tend to converge as algorithm iteration progresses, given by

$$\hat{y}_{eq}=[\boldsymbol{p}^g_k,0]^T \tag{23}$$

So, the dynamic behavior of the ith particle can be analyzed using the eigenvalues derived from the dynamic matrix formulations (19) and (20), solutions of the following characteristic polynomial:

$$\lambda^2-(1+w-c_1r^i_{1,k}-c_2r^i_{2,k})\lambda+w=0 \tag{24}$$

The following necessary and sufficient conditions for stability of the considered discrete-time dynamic system (20) are obtained while applying the classical Jury criterion:

$$|w|<1,\quad c_1r^i_{1,k}+c_2r^i_{2,k}>0,\quad w+1-\frac{c_1r^i_{1,k}+c_2r^i_{2,k}}{2}>0 \tag{25}$$

Knowing that $r^i_{1,k}$, $r^i_{2,k}\in[0,1]$, the above stability conditions are equivalents to the following set of parameter selection heuristics, which guarantee convergence for the PSO algorithm:

$$0<c_1+c_2<4,\quad \frac{c_1+c_2}{2}-1<w<1 \tag{26}$$

While these heuristics provide useful selection parameter bounds, an analysis of the effect of the different parameter settings is achieved and verified by some numerical simulations to determine the effect of such parameters in the PSO algorithm convergence performances.

In order to illustrate the efficiency of the proposed PSO algorithm in the resolution of problems (9), (10) and (11), several comparisons with the Genetic Algorithms Optimization GAO-based method (Goldberg, 1989; The MathWorks Inc., 2009) are considered. The next section is dedicated to the application of the proposed PSO-tuned PID-FLC approaches to an electrical DC drive within a developed real-time framework.

4. Real-time control approach application

4.1. Plant model description

The considered benchmark is a 250 W electrical DC drive. The machine's speed rotation is 3000 rpm at 180 V DC armature voltage. The motor is supplied by an AC–DC power converter. The developed real-time application acquires input data (speed of the DC drive) and generates control signal for thyristors of AC–DC power converter (PWM signal). This is achieved using a data acquisition and control system based on a PC computer and a multi-functions data acquisition PCI-1710 board, which is compatible with MATLAB/Simulink.

The considered electrical DC drive can be described by the following model that is used in the design setup:

$$G(s)=\frac{k_m}{(1+\tau_m s)(1+\tau_e s)} \tag{27}$$

The model's parameters are obtained by an experimental identification procedure and they are summarized in Table 1 with their associated uncertainty bounds. Also, this model is sampled with 10 ms sampling time for simulation and experimental setups.

4.2. Simulation results

For all proposed PSO-tuned PID-type FLC structures, product–sum inference and center of gravity defuzzification methods are adopted for the FLC block. Uniformly distributed and symmetrical membership functions are assigned for the fuzzy input and output variables. The associated fuzzy rule-base is given in Table 2. The linguistic levels assigned to the input variables e_k and Δe_k and the output variable u_{fz} are given as follows: N (negative), Z (zero), P (positive), N (negative), NB (negative big) and PB (positive big). The view of this rule-base is illustrated in Fig. 3.

The swarm size algorithm's choice is generally a problem-dependent in PSO framework. However, Eberhart and Shi (2001) as well as Poli et al. (2007) show that this parameter is often set empirically in relation to the dimensionality and perceived

Table 1
Identified DC drive model parameters.

Parameters	Nominal values	Uncertainty bounds (%)
k_m	0.05	80
τ_m	300 ms	80
τ_e	14 ms	80

Table 2
Fuzzy rule-base for the output variable u_{fz}.

e_k	Δe_k,		
	N	Z	P
N	NB	N	Z
Z	N	Z	P
P	Z	P	PB

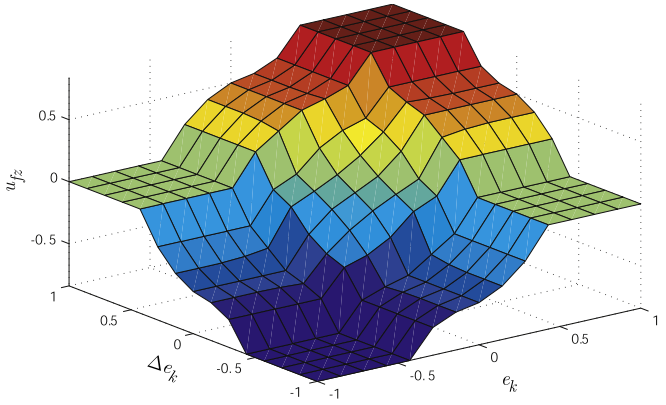

Fig. 3. View of fuzzy rule-base for the fuzzy output u_{fz}.

difficulty of a considered optimization problem. They suggest that swarm size values in the range 20–50 are quite common. For this purpose, we tested the proposed PSO algorithm with different values in this range for the case of PID-type FLC structure without self-tuning mechanisms. Globally, all the found results are close to each other. But, the best values of fitness are obtained while using a swarm size equal to 30. Henceforth, this value will be adopted for our following works.

In the PSO framework, it is necessary to run the algorithm several times in order to get some statistical data on the quality of results and so to validate the proposed approach. We run the proposed algorithm 20 times and feasible solutions are found in 100% of trials and within an acceptable CPU computation time for the IAE criterion case. However, feasible solutions are found in 80% for the MO criterion case. The obtained optimization results are summarized in Tables 3–5. Besides, the fact that the algorithm's convergence always takes place in the same region of the design space, whatever is the initial population, indicates that the algorithm succeeds in finding a region of the interesting research space to explore. The performances comparison of PSO- and GAO-based approaches is achieved in the same conditions.

Table 3
Optimization results from 20 trials of problem (9).

Cost function	Algorithm	Best	Mean	Worst	St. dev.
MO	PSO	5.3442	6.7501	9.0083	1.406
	GAO	10.6050	12.0005	14.7320	2.255
IAE	PSO	0.0162	0.0261	0.0497	0.016
	GAO	0.1892	0.2835	0.3227	0.066

Table 4
Optimization results from 20 trials of problem (10).

Cost function	Algorithm	Best	Mean	Worst	St. dev.
MO	PSO	3.7670	4.3001	6.4500	2.050
	GAO	6.4360	9.3881	12.1000	3.877
IAE	PSO	0.0659	0.0838	0.0946	0.014
	GAO	0.0718	0.0814	0.0973	0.013

Table 5
Optimization results from 20 trials of problem (11).

Cost function	Algorithm	Best	Mean	Worst	St. dev.
MO	PSO	1.5200	2.1283	4.5355	1.825
	GAO	4.3391	6.4786	8.0700	2.110
IAE	PSO	0.0673	0.0861	0.0993	0.016
	GAO	0.0855	0.0905	0.1009	0.008

Indeed, the population size, used in the GAO algorithm, is set as 30 individuals and the maximum generation number is 50. However, the GA parameters, used for MATLAB simulations, are chosen as the Stochastic Uniform selection and the Gaussian mutation methods. The Elite Count is set as 2 and the Crossover Fraction as 0.8. The algorithm stops when the number of generations reaches the specified value for the maximum generation.

According to the statistical analysis of Tables 3–5, we can conclude that the proposed PSO-based approach produces better results in comparison with the standard GAO-based one. Also, while using a Pentium IV, 1.73 GHz and MATLAB 7.7.0, the CPU computation times are about 325 and 364 s for MO and IAE criteria, respectively, for the considered PID-type FLC without self-tuning mechanisms structure.

On the other hand, performances on convergence properties of the proposed PSO and the used GAO algorithm, in term of iterations number's required to find the best solution, are compared for the IAE criterion case, as shown in Fig. 4. While using the proposed PSO-based method, we succeed to obtain the optimal solution within only about 28 iterations. However, the GAO-based method finds the same result after 40 iterations. All these observations can show the superiority of the proposed PSO-based method in comparison with the GAO-based one. Indeed, the quality of the obtained optimal solution, the fastness convergence as well as the simple software implementation is better than those of the GAO-based approach.

In a typical optimization procedure, the scaling parameters λ_I, given in Eq. (15), will be linearly increased at each iteration step so constraints are gradually enforced. Generally, the quality of the solution will directly depend on the value of the specified scaling parameters. In this paper and in order to make the proposed approach simple, great and constant scaling penalty parameters, equal to 10^3, are used for simulations. Indeed, simulation results show that with a great values of λ_I, the control system performances are weakly degraded and the effects on the tuning parameters are less meaningful. The PSO algorithm convergence is faster than the case with linearly variable scaling parameters.

The robustness of the proposed PSO algorithm convergence, under variation of the cognitive, social and inertia factor parameters, is analyzed based on numerical simulations as shown in Fig. 5. The PSO algorithm's convergence is guaranteed within the established domain given by Eq. (26).

The robust stability of the proposed PSO-tuned PID-type FLC approach is analyzed while considering external disturbances and

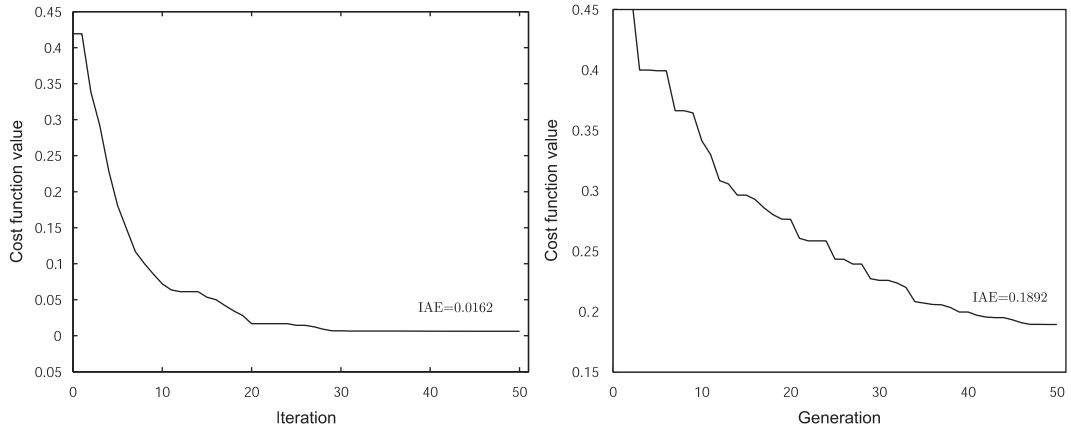

Fig. 4. Convergence properties comparison of the proposed PSO and GAO algorithms: IAE criterion case.

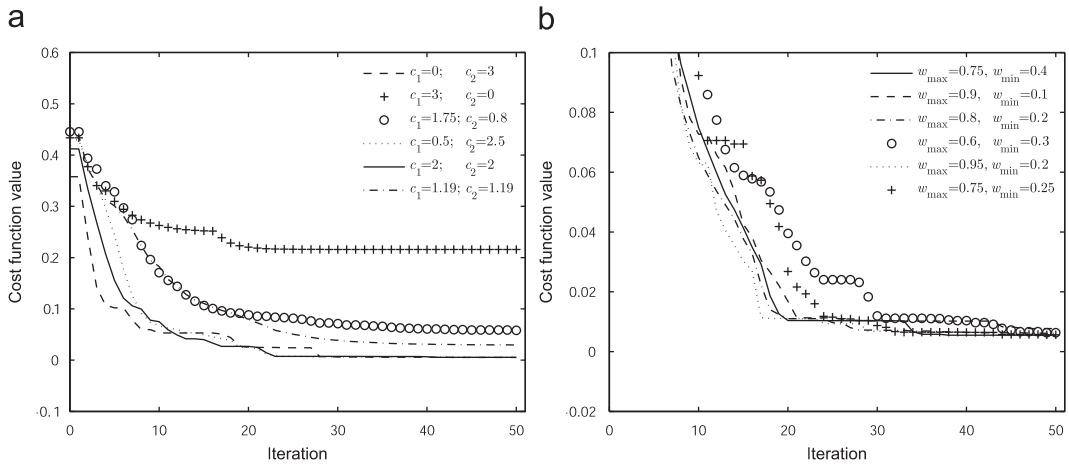

Fig. 5. Robustness of the proposed PSO algorithm under variations of cognitive and social coefficients (a), and inertia factor (b): IAE criterion case.

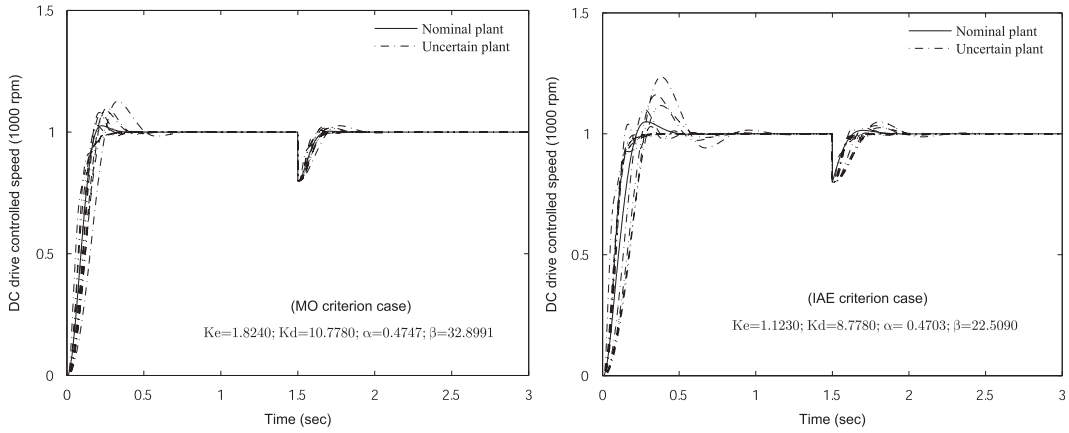

Fig. 6. Robustness of the closed-loop PSO-tuned PID-type FLC system stability under parameters uncertainties and external disturbances.

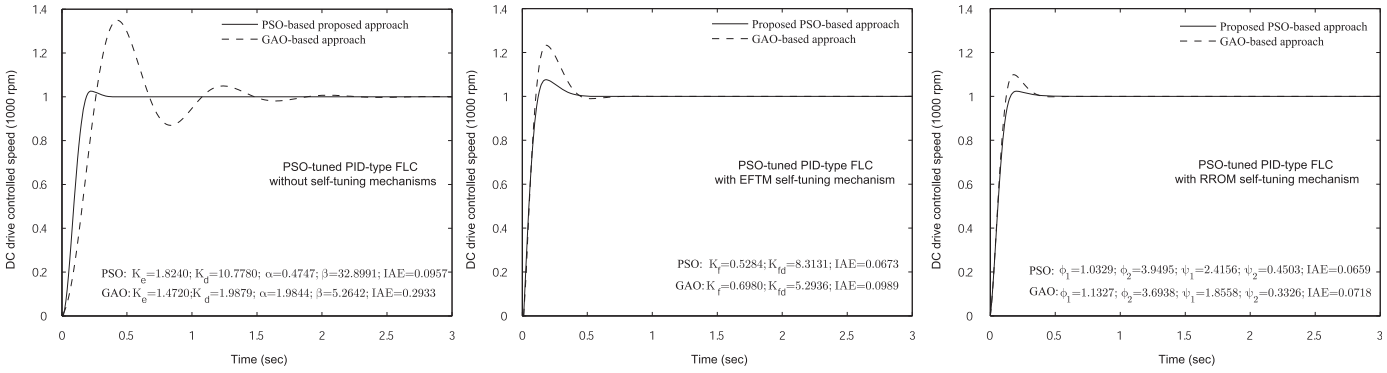

Fig. 7. Time-domain performances comparison of all designed PSO- and GAO-tuned PID-type FLC structures: IAE criterion case.

model uncertainties. According to uncertain bounds on nominal plant parameters, given in Table 1, we are going to consider the following family of continuous-time transfer functions supposed including the real studied plant:

$$G=\left\{\hat{G}(s)=\frac{k_m}{(1+\tau_m s)(1+\tau_e s)};\quad k_m\in[k_m^{\min},k_m^{\max}],\ \tau_e\in[\tau_e^{\min},\tau_e^{\max}],\right.$$

$$\left.\tau_m\in[\tau_m^{\min},\tau_m^{\max}]\right\} \tag{28}$$

Fig. 6 shows the step responses of a family of 10 random generated closed-loop uncertain models. The stability robustness of the uncertain plants, under the above considered uncertainty types, is guaranteed for all designed PID-type FLC structures.

Finally, the time-domain performances of all proposed PID-type FLC structures are compared for the PSO- and GAO-based design cases as shown in Fig. 7.

Besides, Table 6 shows the superiorities of the self-tuning EFTM and RROM PID-type FLC structures in relation to the one without self-tuning mechanisms as verified in Güzelkaya et al. (2003). Recall that the considered time-domain constraints for the PID-type FLC tuning problems (9), (10) and (11) problems, defined in terms of overshoot, steady state, rise and settling times, have been specified, respectively, as $D^{\max}=20\%$, $E_{ss}^{\max}=0$, $t_r^{\max}=0.25$ s and $t_s^{\max}=0.75$ s.

4.3. Experimental setup and results

In order to illustrate the efficiency of the proposed PSO-tuned fuzzy control structures within a real-time framework, the example of the PID-type FLC without self-tuning mechanisms is considered. The same principle of implementation remains valid for the other PID-type FLC structures.

The controlled process is constituted by the single-phase AC–DC power converter and the independent excitation DC motor. A schematic diagram of the experimental setup prepared for testing of the designed controller is shown in Fig. 8. The developed real-time application acquires input data (speed of the DC drive) and generates control signals for the AC–DC power converter through a thyristors gate drive circuit. This is achieved using a data acquisition and control system based on PC and a multi-function data acquisition PCI-1710 board with 12-bit resolution of A/D converter and up to 100 kHz sampling rate. A thyristors gate drive circuit, based on a multivibrator, is used to generate a triggering burst of high-frequency impulses. A pulse transformer is used to assure the galvanic insulation between the control and power circuits. The acquired speed measure, obtained from tachometer sensor, must be adapted to be applied to the used multi-function PCI-1710 board. The complete electronic circuit diagram of the designed control system is given in Haggège et al. (2010).

The multi-function data acquisition PCI-1710 board allows achieving measurement and controlling functions. This target is used to create a real-time application to let the implemented controller system run while synchronized to a real-time clock (Haggège et al., 2010).

The model of the plant was removed from the simulation model, and instead of it, the input device driver (Analog Input) and the output device driver (Analog Output) were introduced as shown in Fig. 9. These device drivers close the feedback loop when moving from simulations to experiments. Device driver's

Table 6
Performances of the PSO-tuned PID-type FLC structures: IAE criterion case.

PSO-tuned PID-type FLC structure	D (%)	t_r (s)	t_s (s)	E_{ss}	CPU computation time (s)
Without self-tuning mechanisms	17.5	0.23	0.49	0	364
With EFTM self-tuning mechanism	15	0.21	0.64	0	370
With RROM self-tuning mechanism	7	0.20	0.68	0	392

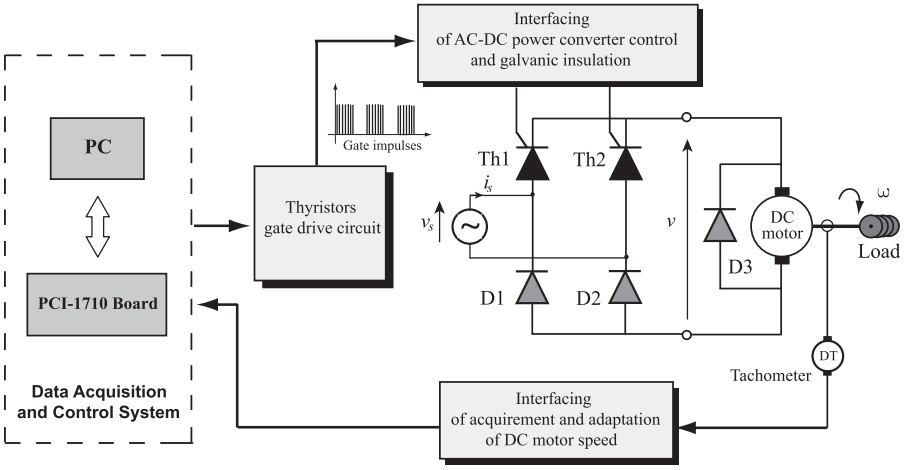

Fig. 8. Proposed experimental setup schematic.

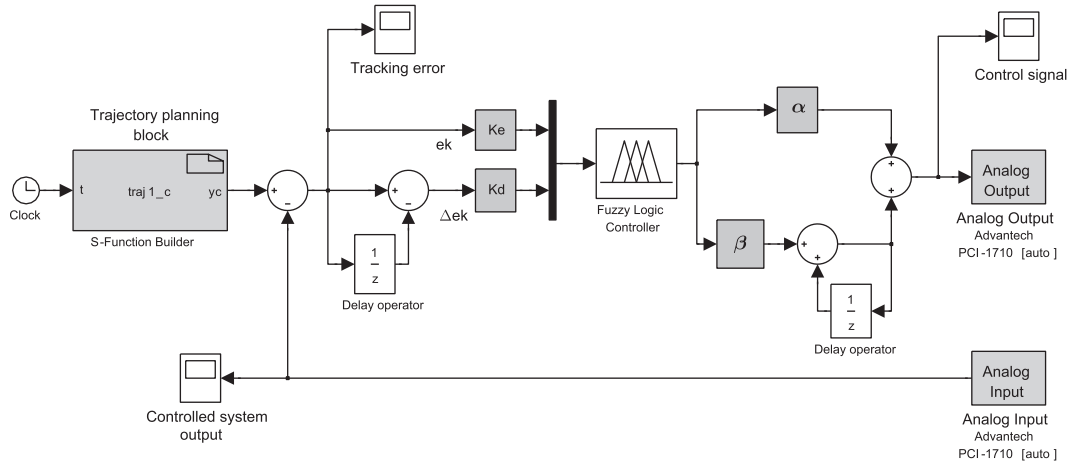

Fig. 9. PCI-1710 board based real-time implementation of the proposed PSO-tuned PID-type FLC structure.

blocks include procedures to access the inputs–outputs board. The real-time controller is developed through the compilation and linking stage, in a form of a Dynamic Link Library (DLL), which is then loaded in memory and started-up.

In this paper, the desired trajectory planning is achieved based on the flatness control technique as shown in Rotella et al. (2001) and applied for the some process benchmark in Haggège et al. (2010). Indeed, let us consider the discrete-time linear model obtained by sampling the continuous-time one, as given by Eq. (27):

$$A(q)y_k = B(q)u_k \tag{29}$$

where the polynomials $A(q)$ and $B(q)$ represent the denominator and the numerator of the discrete transfer function, respectively, and q is the forward operator.

The partial state of such a dynamic system can be considered as a discrete flat output, which can be expressed as a function of input and output signals as the following:

$$A(q)z_k = u_k, \quad B(q)z_k = y_k \tag{30}$$

In discrete-time framework, the discrete desired trajectory z_k^d is generally obtained by sampling a continuous trajectory $z^d(t)$. In order to plan the desired flat trajectory, the technique well known as polynomial interpolation can be used (Rotella et al., 2001; Haggège et al., 2010). Let consider the state vector Z^d, given by Eq. (31) and containing the desired continuous flat output with its successive derivatives, and two known constant state vectors $Z^d(t_0)$ and $Z^d(t_f)$:

$$Z^d(t) = (z^d(t), \dot{z}^d(t), \ldots, z^{d(r+1)}(t))^T \tag{31}$$

The expression of Z^d can be given as follows:

$$Z^d(t) = M_1(t-t_0)c_1 + M_2(t-t_0)c_2 \tag{32}$$

where M_1 and M_2 are two real matrices defined by

$$M_1(t) = \begin{pmatrix} 1 & t & \ldots & t^{n-1}/(n-1)! \\ 0 & 1 & \ldots & t^{n-2}/(n-2)! \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix},$$

$$M_2(t) = \begin{pmatrix} t^n/n! & t^{n+1}/(n+1)! & \ldots & \frac{t^{2n-1}}{(2n-1)!} \\ \frac{t^{n-1}}{(n-1)!} & \frac{t^n}{n!} & \ldots & \frac{t^{2n-2}}{(2n-2)!} \\ \vdots & \ddots & \ddots & \vdots \\ t & \ldots & t^{n-1}/(n-1)! & t^n/n! \end{pmatrix} \tag{33}$$

c_1 and c_2 are constant vectors defined as follows:

$$c_1 = Z^d(t_0), \quad c_2 = M_2^{-1}(t_f - t_0)(Z^d(t_f) - M_1(t_f - t_0)Z^d(t_0)) \tag{34}$$

After planning a desired flat output trajectory, the output desired trajectory y_k^d can be defined by considering Eq. (30). The system output signal y_k must asymptotically track the desired discrete-time trajectory given by Eq. (35).

$$y_k^d = B(q)z_k^d \tag{35}$$

The practical implementation of the PSO-tuned PID-type FLC approach leads to the experimental results of Figs. 10 and 11. The obtained results are satisfactory for a simple, systematic and non-conventional control approach and point out the controller's viability and validate the proposed control approach. The measured speed tracking error of the controlled DC drive is very small (less than 10% of set point) showing the high performances of the proposed control especially in terms of tracking.

On the other hand, Fig. 11 shows the robustness of the proposed PSO-tuned PID-type FLC in rejection of an external load

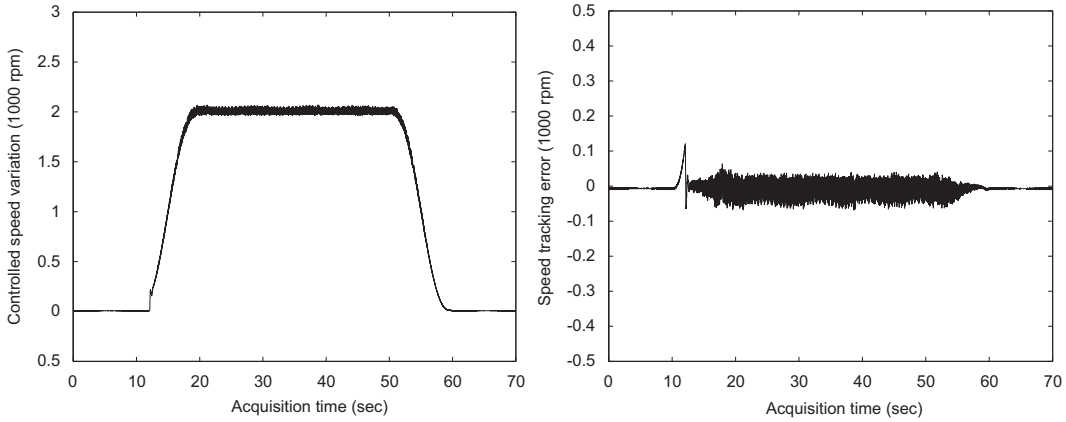

Fig. 10. Experimental results of PSO-tuned PID-type FLC implementation: controlled DC drive speed variation and speed tracking error.

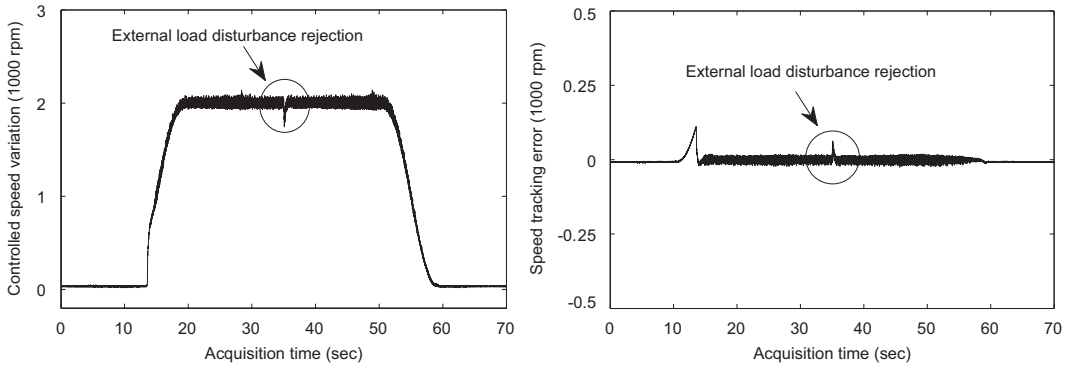

Fig. 11. Experimental results of PSO-tuned PID-type FLC implementation: robustness of the proposed method under external load disturbance.

disturbance applied on the controlled system. The dynamic of the disturbance rejection is fast and guaranteed.

5. Conclusion

In this paper, a new method for tuning PID-type FLC structures, using a PSO-based technique, is proposed and successfully applied to an electrical DC drive speed control within a real-time framework. This efficient tool leads to a robust and systematic fuzzy control design approach. The performances comparison, with the standard GAO-based method, shows the efficiency and superiority of the proposed PSO-based approach in terms of the obtained solution qualities, the convergence speed and the simple software implementation of its algorithm.

The PID-type FLC structures robustness, under external influences such as the output static disturbances and parametric uncertainties, are proved. The control design methodology is systematic, practical and simple without the need of exact analytic plant model description. The obtained simulation and experimental results show the efficiency in terms of performance and robustness of the proposed fuzzy control approach, which can be applied in industrial motor control field.

References

Bouallègue, S., Haggège, J., Benrejeb, M., 2010a. Structured loop-shaping H^{∞} controller design using particle swarm optimization. In: Proceedings of the 2010 IEEE International Conference on Systems, Man, and Cybernetics SMC'10, Istanbul.

Bouallègue, S., Haggège, J., Benrejeb, M., 2010b. Structured mixed-sensitivity H^{∞} design using particle swarm optimization. In: Proceedings of the 7th IEEE International Multi-Conference on Systems, Signals and Devices SSD'10, Amman.

Bouallègue, S., Haggège, J., Benrejeb, M., 2011. Particle swarm optimization-based fixed-structure H^{∞} control design. Int. J. Control Autom. Syst. 9 (2), 258–266.

Haggège, J., Ayadi, M., Bouallègue, S., Benrejeb, M., 2010. Design of fuzzy flatness-based controller for a DC drive. Control Intell. Syst. 38 (3), 164–172.

Eberhart, R.C., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, Nagoya, pp. 39–43.

Eberhart, R.C., Shi, Y., 2001. Particle swarm optimization: developments, applications and resources. In: Proceedings of the IEEE Congress on Evolutionary Computation, Seoul, Korea, pp. 81–86.

Eker, İ., Torun, Y., 2006. Fuzzy logic control to be conventional method. Energy Convers. Manage. 47, 377–394.

Eksin, İ., Güzelkaya, M., Gürleyen, F., 2001. A new methodology for deriving the rule-base of a fuzzy logic controller with a new internal structure. Eng. Appl. Artif. Intell. 14, 617–628.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Güzelkaya, M., Eksin, İ., Yeşil, E., 2003. Self-tuning of PID-type fuzzy logic controller coefficients via relative rate observer. Eng. Appl. Artif. Intell. 16, 227–236.

Lee, C.C., 1990a. Fuzzy logic in control systems: fuzzy logic controller—Part I. IEEE Trans. Syst. Man Cybern. 20 (2), 404–418.

Lee, C.C., 1990b. Fuzzy logic in control systems: fuzzy logic controller—Part II. IEEE Trans. Syst. Man Cybern. 20 (2), 419–435.

Liu, L., Liu, W., Cartes, D.A., 2008. Particle swarm optimization-based parameter identification applied to permanent magnet synchronous motors. Eng. Appl. Artif. Intell. 21, 1092–1100.

Marinakis, Y., Marinaki, M., Dounias, G., 2010. A hybrid particle swarm optimization algorithm for the vehicle routing problem. Eng. Appl. Artif. Intell. 23, 463–472.

Passino, K.M., Yurkovich, S., 1998. Fuzzy Control. Addison Wesley Longman, Menlo Park, CA.

Poli, R., Kennedy, J., Blackwell, T., 2007. Particle Swarm Optimization: An Overview. Swarm Intelligence, 1. Springer, pp. 33–57.

Qiao, W.Z., Mizumoto, M., 1996. PID type fuzzy controller and parameters adaptive method. Fuzzy Sets Syst. 78, 23–35.

Rotella, F., Carrillo, F.J., Ayadi, M., 2001. Digital flatness-based robust controller applied to a thermal process. In: IEEE Control Systems Society Conference on Control Applications, Mexico City, pp. 936–941.

Ruben, E.P., Kamran, B., 2007. Particle swarm optimization in structural design. In: Chan, F.T.S., Tiwari, M.K. (Eds.), Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publishing, Vienna, pp. 373–394.

Samanta, B., Nataraj, C., 2009. Use of particle swarm optimization for machinery fault detection. Eng. Appl. Artif. Intell. 22, 308–316.

Shi, Y., Eberhart, R., 1999. Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, Washington, pp. 1945–1950.

The MathWorks Inc., 2009. Genetic Algorithm and Direct Search Toolbox^{TM}: User's Guide. Natick.

Van den Bergh, F., 2006. An Analysis of Particle Swarm Optimizers. Ph.D. Thesis, University of Pretoria, Pretoria, South Africa.

Venayagamoorthy, G.K., Smith, S.C., Singhal, G., 2007. Particle swarm-based optimal partitioning algorithm for combinational CMOS circuits. Eng. Appl. Artif. Intell. 20, 177–184.

Woo, Z-W., Chung, H-Y., Lin, J-J., 2000. A PID type fuzzy controller with self-tuning scaling factors. Fuzzy Sets Syst. 115, 321–326.