Engineering Applications of Artificial Intelligence 22 (2009) 343–350

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Design of fractional-order $PI^{\lambda}D^{\mu}$ controllers with an improved differential evolution

Arijit Biswas [a], Swagatam Das [a,*], Ajith Abraham [b], Sambarta Dasgupta [a]

[a] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India
[b] Norwegian University of Science and Technology, Norway

ARTICLE INFO

Article history:
Received 21 April 2008
Accepted 4 June 2008
Available online 2 September 2008

Keywords:
Differential evolution
Fractional calculus
PID controllers
Fractional-order controllers
Evolutionary algorithms

ABSTRACT

Differential evolution (DE) has recently emerged as a simple yet very powerful technique for real parameter optimization. This article describes an application of DE to the design of fractional-order proportional–integral–derivative (FOPID) controllers involving fractional-order integrator and fractional-order differentiator. FOPID controllers' parameters are composed of the proportionality constant, integral constant, derivative constant, derivative order and integral order, and its design is more complex than that of conventional integer-order proportional–integral–derivative (PID) controller. Here the controller synthesis is based on user-specified peak overshoot and rise time and has been formulated as a single objective optimization problem. In order to digitally realize the fractional-order closed-loop transfer function of the designed plant, Tustin operator-based continuous fraction expansion (CFE) scheme was used in this work. Several simulation examples as well as comparisons of DE with two other state-of-the-art optimization techniques (Particle Swarm Optimization and binary Genetic Algorithm) over the same problems demonstrate the superiority of the proposed approach especially for actuating fractional-order plants. The proposed technique may serve as an efficient alternative for the design of next-generation fractional-order controllers.

1. Introduction

Fractional-order dynamic systems and controllers, which are based on fractional-order calculus (Oldham and Spanier, 1974; Lubich, 1986; Miller and Ross, 1993), have been gaining attention in several research communities since the last few years (Oustaloup, 1981; Chengbin and Hori, 2004). In Podlubny (1999b), it was advocated that fractional-order calculus would play a major role in a smart mechatronic system. Podlubny proposed the concept of the fractional-order $PI^{\lambda}D^{\mu}$ controllers and demonstrated the effectiveness of such controllers for actuating the responses of fractional-order systems in 1999. A few recent works in this direction as well as schemes for digital and hardware realizations of such systems can be traced in Chen et al. (2004), Nakagawa and Sorimachi (1992) and Chen et al. (2005). Vinagre et al. (2000) proposed a frequency domain approach based on expected crossover frequency and phase margin for the same controller design. Petras came up with a method based on the pole distribution of the characteristic equation in the complex plane (Petras, 1999). Dorcak et al. (2001) proposed a state-space design approach based on feedback pole placement. The fractional controller can also be synthesized by cascading a proper fractional unit to an integer-order controller (Chengbin and Hori, 2004).

Proportional–integral–derivative (PID) controllers have been used for several decades in industries for process control applications. The reason for their wide popularity lies in the simplicity of design and good performance including low percentage overshoot and small settling time for slow process plants (Astrom and Hagglund, 1995). In fractional-order proportional–integral–derivative (FOPID) controller, I and D operations are usually of fractional order; therefore, besides setting the proportional, derivative and integral constants K_p, T_d, T_i we have two more parameters: the order of fractional integration λ and that of fractional derivative μ. Finding an optimal set of values for K_p, T_i, T_d, λ and μ to meet the user specifications for a given process plant calls for real parameter optimization in five-dimensional hyperspace.

Differential evolution (DE) (Price et al., 2005; Storn and Price, 1997) has recently become quite popular as a simple and efficient scheme for global optimization over continuous spaces. It has reportedly outperformed many types of evolutionary algorithms

* Corresponding author. Tel.: +91 33 2528 2717.
E-mail addresses: arijitbiswas87@gmail.com (A. Biswas), swagatamdas19@yahoo.co.in (S. Das), ajith.abraham@ieee.org (A. Abraham), sambartadg@gmail.com (S. Dasgupta).

0952-1976/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2008.06.003

and search heuristics like PSO when tested over both benchmarks and real-world problems (Vesterstrøm and Thomson, 2004). In this work, a state-of-the-art version of DE has been used for finding the optimal values of K_p, T_i, T_d, λ and μ. The design method focuses on optimum placing of the dominant closed-loop poles and incorporate the constraints thus obtained using DE algorithm. The optimization-based design process has been tested for actuating the response of four process plants of which two are of integer order and two are of fractional order. The performance of the DE-based PI$^{\lambda}$D$^{\mu}$ controller has been compared with two other fractional-order controllers designed with the state-of-the-art versions of two recent swarm intelligence-based techniques well known as the Hierarchical Particle Swarm Optimizer with Time Varying Acceleration Coefficients (HPSO-TVAC) (Ratnaweera and Halgamuge, 2004) and the genetic algorithm (Holland, 1975; Cao et al., 2005). Such comparison reflects the superiority of the proposed method in terms of quality of the final solution, convergence speed and robustness.

The rest of the paper is organized as follows. Section 2 describes the rudiments of fractional calculus and fractional-order control systems. Section 3 provides a brief overview of the DE family of algorithms and describes a recent state-of-the-art version of DE called DE/rand/either–or, which was used, in this specific task. Section 4 demonstrates how the DE can be applied to the PI$^{\lambda}$D$^{\mu}$ controller design problem when formulated as an optimization task. Simulation strategies and experimental results have been presented and discussed in Section 5 and finally the paper is concluded with a discussion on future research issues in Section 6.

2. Fractional-order systems: a brief overview

Fractional calculus is a branch of mathematical analysis that studies the possibility of taking real number power of the differential operator and integration operator. From a purely mathematical point of view, there are several ways to define fractional-order derivatives and integrals. The generalized differ-integrator operator may be put forward as

$$ {}_aD_t^q f(t) = \frac{\mathrm{d}^q f(t)}{[\mathrm{d}(t-a)]^q} \tag{1}$$

where q represents the real order of the differintegral (an n is used in some literature to denote an integer order), t is the parameter for which the differintegral is taken and a is the lower limit. Unless otherwise stated, the lower limit will be 0 and left out of the notation. Caputo used a popular definition used to compute differintegral in 1960s. The definition for Caputo's fractional derivative of order λ with respect to the variable t and with the starting point $t=0$ goes as follows (Caputo, 1967, 1969):

$$ {}_0D_t^{\lambda} y(t) = \frac{1}{\Gamma(1-\delta)} \int_0^t \frac{y^{(m+1)}(\tau)\,\mathrm{d}\tau}{(t-\tau)^{\delta}} \quad (\gamma = m+\delta,\ m \in Z,\ 0<\delta\leqslant 1) \tag{2}$$

where $\Gamma(Z)$ is Euler's gamma function. If $\gamma<0$, then we have a fractional integral of order $-\gamma$ given as

$$ {}_0I_t^{-\gamma} y(t) = {}_0D_t^{\gamma} y(t) = \frac{1}{\Gamma(-\gamma)} \int_0^t \frac{y(\tau)\mathrm{d}\tau}{(t-\tau)^{1+\gamma}} \quad (\gamma<0) \tag{3}$$

One distinct advantage of using Caputo's definition is that it only allows for consideration of easily interpretable initial conditions but it is also bounded, which means the derivative of a constant is equal to zero. In time domain, a fractional-order system is governed by an n-term inhomogeneous fractional-order differential equation (FDE):

$$ a_n D^{\beta_n} y(t) + a_{n-1} D^{\beta_{n-1}} y(t) + \cdots + a_1 D^{\beta_1} y(t) + a_0 D^{\beta_0} y(t) = u(t) \tag{4}$$

where $D^{\lambda} \equiv {}_0D_t^{\lambda}$ is the Caputo's fractional derivative of order λ. Converting to frequency domain, the fractional-order transfer function of such a system may be obtained through the Laplace transform function as follows:

$$ G_n(s) = \frac{1}{a_n s^{\beta_n} + a_{n-1} s^{\beta_{n-1}} + \cdots + a_1 s^{\beta_1} + a_0 s^{\beta_0}} \tag{5}$$

where β_k $(k=0,\ 1,\ \ldots,\ n)$ is an arbitrary real number, $\beta_n > \beta_{n-1} > \cdots > \beta_1 > \beta_0 > 0$ and a_k $(k=0,\ 1,\ \ldots,\ n)$ is an arbitrary constant. Finally, we would like to mention here that the Laplace transform of the fractional derivative might be given as

$$ \int_0^{\infty} \mathrm{e}^{-st} D^{\gamma} y(t)\,\mathrm{d}t = s^{\gamma} Y(s) - \sum_{k=0}^{m} s^{\gamma-k-1} y^{(k)}(\theta) \tag{6}$$

For $\gamma<0$ (i.e., for the case of a fractional integral) the sum in the right-hand side must be omitted.

3. The DE algorithm and its modification

Like any other evolutionary algorithm, DE starts with a population of NP D-dimensional parameter vectors representing the candidate solutions. We shall denote subsequent generations in DE by $G=0,\ 1,\ \ldots,\ G_{\max}$. Since the parameter vectors are likely to be changed over different generations, we may adopt the following notation for representing the ith vector of the population at the current generation as

$$ \vec{X}_{i,G} = [x_{1,i,G},\ x_{2,i,G},\ x_{3,i,G},\ \ldots,\ x_{D,i,G}] \tag{7}$$

The initial population (at $G=0$) should better cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds: $\vec{X}_{\min} = \{x_{1,\min},\ x_{2,\min},\ \ldots,\ x_{D,\min}\}$ and $\vec{X}_{\max} = \{x_{1,\max},\ x_{2,\max},\ \ldots,\ x_{D,\max}\}$. Hence we may initialize the jth component of the ith vector as

$$ x_{j,i,0} = x_{j,\min} + \mathrm{rand}_j(0,1)(x_{j,\max} - x_{j,\min}) \tag{8}$$

where $\mathrm{rand}_j(0,1)$ is the jth instantiation of a uniformly distributed random number lying between 0 and 1. Following steps are taken next: mutation, crossover and selection, which are explained below.

3.1. Mutation

After initialization, DE creates a *donor* vector $\vec{V}_{i,G}$ corresponding to each population member or *target* vector $\vec{X}_{i,G}$ in the current generation through mutation. It is the method of creating this donor vector, which differentiates between the various DE schemes. For example, five most frequently referred mutation strategies implemented in the public-domain DE codes available online at http://www.icsi.berkeley.edu/~storn/code.html are listed below:

$$ \text{"DE/rand/1"}: \ \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) \tag{9}$$

$$ \text{"DE/best/1"}: \ \vec{V}_{i,G} = \vec{X}_{\mathrm{best},G} + F(\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \tag{10}$$

$$ \text{"DE/target-to-best/1"}: \ \vec{V}_{i,G} = \vec{X}_{i,G} + F(\vec{X}_{\mathrm{best},G} - \vec{X}_{i,G}) + F(\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \tag{11}$$

$$\text{“DE/best/2”}: \vec{V}_{i,G} = \vec{X}_{\text{best},G} + F(\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) + F(\vec{X}_{r_3^i,G} - \vec{X}_{r_4^i,G}) \quad (12)$$

$$\text{“DE/rand/2”}: \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) + F(\vec{X}_{r_4^i,G} - \vec{X}_{r_5^i,G}) \quad (13)$$

The indices $r_1^i, r_2^i, r_3^i, r_4^i$ and r_5^i are mutually exclusive integers randomly chosen from the range (Oldham and Spanier, 1974), NP, which are also different from the index i. These indices are randomly generated once for each mutant vector. The scaling factor F is a positive control parameter for scaling the difference vectors. $\vec{X}_{\text{best},G}$ is the best individual vector with the best fitness function value in the population at generation G. The general convention used for naming the various mutation strategies is DE/x/y/z, where DE stands for differential evolution, x represents a string denoting the vector to be perturbed and y is the number of difference vectors considered for perturbation of x. z stands for the type of crossover being used (exp: exponential; bin: binomial).

3.2. Crossover

To increase the potential diversity of the population, a crossover operation comes into play after generating the donor vector through mutation. The classical DE family of algorithms generally uses two kinds of crossover schemes—*exponential* and *binomial* (Price et al., 2005). The donor vector exchanges its components with the target vector $\vec{X}_{i,G}$ under this operation to form the *trial* vector $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, \ldots, u_{D,i,G}]$. Here we briefly discuss the binomial crossover and the arithmetic crossover, which has recently been introduced in the DE community in order to circumvent the problem of rotational variance. The binomial crossover is performed on each of the D variables whenever a randomly picked number between 0 and 1 is less than or equal to the Cr value. In this case, the number of parameters inherited from the mutant has a (nearly) binomial distribution. The scheme may be outlined as

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } (\text{rand}_j(0,1) \leqslant \text{Cr or } j = j_{\text{rand}}) \\ x_{j,i,G} & \text{otherwise} \end{cases} \quad (14)$$

where $\text{rand}_j(0,1) \in [0,1]$ is the jth evaluation of a uniform random number generator. $j_{\text{rand}} \in [1, 2, \ldots, D]$ is a randomly chosen index, which ensures that $\vec{U}_{i,G}$ gets at least one component from $\vec{V}_{i,G}$.

The crossover scheme described in Eq. (14) is in spirit a discrete recombination (Price et al., 2005). The discrete recombination is a rotationally variant operation. A rotation of the coordinate systems moves the location of the potential trial solutions. To overcome this limitation, a new trial vector generation strategy ‘DE/current-to-rand/1’ is proposed in Price (1999), which replaces the crossover operator prescribed in Eq. (14) with the rotationally invariant arithmetic crossover operator to generate the trial vector $\vec{U}_{i,G}$ by linearly combining the target vector $\vec{X}_{i,G}$ and the corresponding donor vector $\vec{V}_{i,G}$ as follows:

$$\vec{U}_{i,G} = \vec{X}_{i,G} + K(\vec{V}_{i,G} - \vec{X}_{i,G}) \quad (15)$$

Now incorporating Eq. (9) in (15) we have

$$\vec{U}_{i,G} = \vec{X}_{i,G} + K(\vec{X}_{r_1,G} + F(\vec{X}_{r_2,G} - \vec{X}_{r_3,G}) - \vec{X}_{i,G})$$

which further simplifies to

$$\vec{U}_{i,G} = \vec{X}_{i,G} + K(\vec{X}_{r_1,G} - \vec{X}_{i,G}) + F'(\vec{X}_{r_2,G} - \vec{X}_{r_3,G}) \quad (16)$$

where K is the combination coefficient, which has been proven (Price, 1999) to be effective when it is chosen with a uniform random distribution from [0, 1] and $F' = KF$ is a new constant here.

3.3. Selection

To keep the population size constant over subsequent generations, the next step of the algorithm calls for *selection* to determine whether the target or the trial vector survives to the next generation, i.e., at $G = G+1$. The selection operation may be outlined as

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \text{if } f(\vec{U}_{i,G}) \leqslant f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} & \text{if } f(\vec{U}_{i,G}) > f(\vec{X}_{i,G}) \end{cases} \quad (17)$$

where $f(\vec{X})$ is the function to be minimized. So if the new trial vector yields a lower value of the objective function, then it replaces the corresponding target vector in the next generation; otherwise, the target is retained in the population. Hence the population either gets better (w.r.t. the minimization of the objective function) or remains constant, but never deteriorates. The complete pseudo-code has been provided below:

Pseudo-code for the DE algorithm family

Step 1: Set the generation number $G = 0$, and randomly initialize a population of NP individuals $P_G = \{\vec{X}_{1,G}, \ldots, \vec{X}_{\text{NP},G}\}$ with $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}]$ and each individual uniformly distributed in the range $[\vec{X}_{\min}, \vec{X}_{\max}]$,
where $\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}\}$ and $\vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}\}$ with $i = [1, 2, \ldots, \text{NP}]$.
Step 2: WHILE stopping criterion is not satisfied
DO
FOR $i = 1$ to NP //do for each individual sequentially
Step 2.1: *Mutation step*
Generate a donor vector $\vec{V}_{i,G} = \{v_{i,G}^1, \ldots, v_{i,G}^D\}$ corresponding to the ith target vector $\vec{X}_{i,G}$ via one of the different mutation schemes of DE (Eqs. (9)–(13)).
Step 2.2: *Crossover step*
Generate a trial vector $\vec{U}_{i,G} = \{u_{i,G}^1, \ldots, u_{i,G}^D\}$ for the ith target vector $\vec{X}_{i,G}$ through binomial crossover (Eq. (9)) or exponential crossover (Eq. (14)) or through the arithmetic crossover (Eq. (16)).
Step 2.3: *Selection step*
Evaluate the trial vector $\vec{U}_{i,G}$
IF $f(\vec{U}_{i,G}) \leqslant f(\vec{X}_{i,G})$, THEN $\vec{X}_{i,G+1} = \vec{U}_{i,G}$,
$f(\vec{X}_{i,G+1}) = f(\vec{U}_{i,G})$IF $f(\vec{U}_{i,G}) < f(\vec{X}_{\text{best},G})$,
THEN $\vec{X}_{\text{best},G} = \vec{U}_{i,G}$,

In the original DE mutation scheme, the difference vector $(\vec{X}_i(t) - \vec{X}_j(t))$ is scaled by a constant factor ‘F’. The usual choice for this control parameter is a number between 0.4 and 1. We propose to vary this scale factor in a random manner in the range (0.5, 1) by using the relation

$$F = 0.5(1 + \text{rand}(0,1)) \quad (18)$$

where rand (0, 1) is a uniformly distributed random number within the range [0, 1]. The mean value of the scale factor is 0.75. This allows for stochastic variations in the amplification of the difference vector and thus helps retain population diversity as the search progresses. In Das et al. (2005), it has already been shown that the DERANDSF (DE with Random Scale Factor) can meet or beat the classical DE and also some versions of PSO in a statistically significant manner. In addition to that, here we also decrease the crossover rate CR linearly with time from $\text{CR}_{\max} = 1.0$ to $\text{CR}_{\min} = 0.5$. If CR = 1.0, it means that all components of the parent vector are replaced by the difference vector operator according to (14). But at the later stages of the optimizing process, if CR be decreased, more components of the parent vector are then inherited by the offspring. Such a tuning of CR helps to explore the search space exhaustively at the beginning, but adjust the

movements of trial solutions finely during the later stages of search, so that they can explore the interior of a relatively small space in which the suspected global optimum lies. The time variation of CR may be expressed in the form of the following equation:

$$\mathrm{CR} = (\mathrm{CR}_{\max} - \mathrm{CR}_{\min})\left(\frac{G_{\max} - G}{G_{\max}}\right) + \mathrm{CR}_{\min} \tag{19}$$

where $\mathrm{CR}_{\max}$ and $\mathrm{CR}_{\min}$ are the maximum and minimum values of crossover rate CR, G is the current generation number and $G_{\max}$ is the maximum number of allowable generations. After performing a series of experiments we find that the DE/rand/1/bin scheme (Eq. (9)) equipped with these modifications can outperform all other classical DE variants for the controller design problem investigated here. We exclude the detailed comparison results in order to save space.

4. The DE-based design of fractional PI$^{\lambda}$D$^{\mu}$ controllers

4.1. The FOPID controller

A PID controller is a generic control loop feedback mechanism widely used in industrial control systems. The PID controller attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly. An integer-order PID controller has the following transfer function:

$$G_c(s) = K_p + K_i s^{-1} + K_d s \tag{20}$$

The PID controller calculation (algorithm) involves three separate parameters: the proportional (K_p), the integral (K_i) and derivative (K_d) time constants. The proportional gain determines the reaction to the current error, the integral determines the reaction based on the sum of recent errors and the derivative determines the reaction to the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element. The block diagram of a generic closed-loop control system involving the PID controller has been shown in Fig. 1.

The real objects or processes that we want to control are generally fractional (for example, the voltage–current relation of a semi-infinite lossy RC line). However, for many of them the fractionality is very low. In general, the integer-order approximation of the fractional systems can cause significant differences between mathematical model and real system. The main reason for using integer-order models was the absence of solution methods for FDEs. PID controllers belong to dominating industrial controllers and therefore are objects of steady effort for improvements of their quality and robustness. One of the possibilities to improve PID controllers is to use fractional-order controllers with non-integer derivation and integration parts.

Following the works of Podlubny (1999b), we may go for a generalization of the PID controller, which can be called the PI$^{\lambda}$D$^{\mu}$ controller because of involving an integrator of order λ and a differentiator of order μ. The continuous transfer function of such a controller has the form

$$G_c(s) = K_p + T_i s^{-\lambda} + T_d s^{\mu} \quad (\lambda, \mu > 0) \tag{21}$$

The output response of the PI$^{\lambda}$D$^{\mu}$ controller in time domain may be given as

$$u(t) = K_p e(t) + K_i D^{-\lambda} e(t) + K_d D^{\mu} e(t) \tag{22}$$

where $\lambda = +1$, $\mu = +1$ implies normal PID controller, for $\lambda = 0$, $\mu = +1$, we get a normal PD controller, $\lambda = +1$, $\mu = 0$ implies normal PI controller and $\lambda = 0$, $\mu = 0$ implies a proportional gain. All these classical types of PID controllers are the special cases of the fractional PI$^{\lambda}$D$^{\mu}$ controller. As can be perceived from Fig. 2, the FOPID controller generalizes the integer-order PID controller and expands it from point to plane. This expansion adds more flexibility to controller design and we can control our real-world processes more accurately.

4.2. Formulation of the objective function

The design approach presented here is based on the root locus method (dominant roots method) of synthesizing integral PID controllers (Astrom and Hagglund, 1995). As in the traditional root locus method, based on the user specifications of peak overshoot M_p and rise time t_{rise} (or requirements of stability and damping levels), we find out the damping ratio ξ and the un-damped natural frequency ω_0 of the closed-loop system to be designed. Then dominant poles will be

$$p_{1,2} = -\xi\omega_0 \pm \mathrm{j}\omega_0\sqrt{1-\xi^2} = -x \pm \mathrm{j}y \quad \text{(say)} \tag{23}$$

Let the closed-loop transfer function be

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \tag{24}$$

where the transfer function of the process to be controlled is $G_p(s)$ and that of the controller is $G_c(s) = U(s)/E(s)$ and $G(s) = G_c(s)G_p(s)$. We assume unity feedback gain, i.e., $H(s) = 1$. From Eq. (24) the

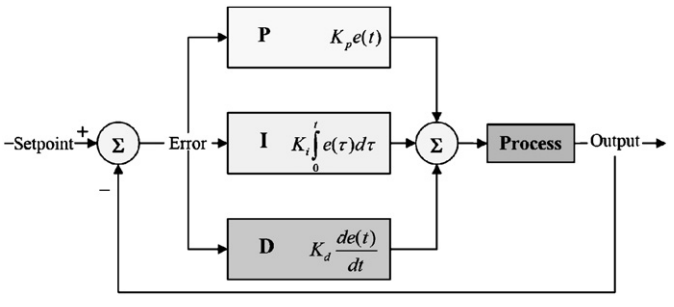

Fig. 1. A generic closed-loop process-control system with PID controller.

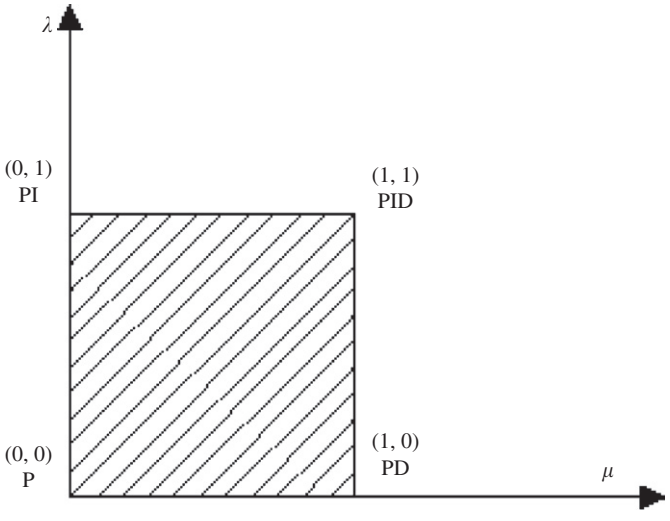

Fig. 2. Generalization of the FOPID controller: from point to plane.

characteristics equation of the closed-loop system is

$$1 + G(s)H(s) = 0 \Rightarrow 1 + G_p(s)G_c(s) \cdot 1 = 0 \tag{25}$$

Now the dominant poles of the system are the zeros of this characteristics equation, so they will obviously satisfy the equation. Thus from (25) we get

$$1 + [K_p + K_i(-x + jy)^{-\lambda} + K_d(-x + jy)^{\mu}]G_p(-x + jy) = 0 \tag{26}$$

This equation has total five unknowns, K_p, K_i, K_d, λ and μ. Let R be the real part of the complex expression (26), I the imaginary part of the complex expression (26) and ψ the phase angle $= \tan^{-1}(I/R)$.

Now we define the following objective function:

$$J(K_p, K_i, K_d, \lambda, \mu) = |I|^2 + |R|^2 + |\psi|^2 \tag{27}$$

Our goal is to find out an optimal solution set $\{K_p, K_i, K_d, \lambda, \mu\}$ for which $J = 0$. Here the above function has been minimized with modified DE/rand/1/bin algorithm.

4.3. Vector representation in DE

The solution space of Eq. (27) is five-dimensional, the five dimensions being $\{K_p, K_i, K_d, \lambda, \mu\}$. So each parameter vector in DE has five components, i.e., the jth population member at Gth generation may be given as

$$\vec{X}_{j,G} = (K_p, K_i, K_d, \lambda, \mu)^{\mathrm{T}} \tag{28}$$

From the practical consideration of the PID controller design (Astrom and Hagglund, 1995), we fixed the following numerical ranges for each parameter:

$$\begin{aligned} &1 \leqslant K_p \leqslant 1000 \\ &0 \leqslant \lambda,\ \delta \leqslant 1 \\ &1 \leqslant T_i,\ T_d \leqslant 500 \end{aligned} \tag{29}$$

5. Experimental results

5.1. Problem instances

We have tested the proposed method on three specific instances of the design problem. All the design examples follow the basic framework detailed in Section 4. The first problem involves the speed control of a DC motor. First, the uncompensated motor can only rotate at 0.1 rad/s with an input voltage of 1 V (this was obtained when the open-loop response is simulated). Since the most basic requirement of a motor is that it should rotate at the desired speed, the steady-state error of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time of 2 s. Since a speed faster than the reference may damage the equipment, we want to have an overshoot of less than 5%. The second problem also involves a second-order (integer) plant, which is to be controlled for obtaining a peak overshoot $M_P = 30\%$ and rise time $t_{\mathrm{rise}} = 0.3$ s in the closed-loop response.

The third and fourth problem instances involve fractional-order plants. The third one is taken from Podlubny's seminal paper on fractional controllers (Podlubny, 1999b). In some cases a real system is better described by such FDEs (Podlubny, 1999a) and from this consideration, it is important to investigate the controlling mechanism of such systems through FOPID-type controllers. Table 1 summarizes all the test problems along with the corresponding user specifications.

5.2. Digital realization of the FOPID controller

For a fractional-order differentiator/integrator s^r, where r is a real number, its discretization is a key step in digital implementation. Furthermore for control applications, the obtained approximate discrete time rational transfer function should be stable and of minimum phase. Continuous fraction expansion (CFE) by Tustin rule enjoys all those desirable properties. By using this method, the discrete transfer function approximating fractional-order operators can be expressed as

$$\begin{aligned} D^{\pm r}(z) = (w(z^{-1}))^{\pm r} &= \left(\frac{2}{T}\right)^{\pm r} \mathrm{CFE}\left(\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)^{\pm r}\right)_{p,q} \\ &= \left(\frac{2}{T}\right)^{\pm r} \frac{P_p(z^{-1})}{Q_q(z^{-1})} \end{aligned} \tag{30}$$

where T is the sampling period and P_p and Q_q are polynomials of degree p and q, respectively, in the variable z^{-1}. The general expression for numerator $P_p(z^{-1})$ and denominator $Q_q(z^{-1})$ of $D^{\pm r}(z)$ is given below for $p = q = 1,3,5$.

In this work we have used the Tustin-rule-based CFE where the sampling time is $T = 0.001$ s and the order of the approximate model is 5 (Table 2).

5.3. Competitor algorithms and parametric set-up

The proposed design method has been extensively compared with two state-of-the-art design methods for FOPID controllers based on the binary GA (Holland, 1975; Cao et al., 2005) and a recently proposed extension of the canonical PSO namely self-organizing HPSO-TVAC (Ratnaweera and Halgamuge, 2004). The GA-based scheme was proposed by Cao et al. (2005) and uses a 50-bit binary string to encode five parameters of the FOPID controller. The fitness functions in Cao's method employ the integral of the squared error and absolute error signal value and

Table 1
Description of the problem instances considered

Problem number	Process plant transfer function $G_p(s)$	Users specification		
		Maximum overshoot (%)	Rise time (s)	Steady-state error (%)
I	$\frac{k}{(Js+b)(Ls+R)+k^2}$, $J = 0.01$, $b = 0.1$, $k = 0.01$, $R = 1$, $L = 0.5$	5	0.5	4
II	$\frac{s^2}{50s + 400}$	20	0.1	Not specified
III	$\frac{1}{0.8s^{2.2} + 0.5s^{0.9} + 1}$	10	0.2	3
IV	$\frac{1}{0.9s^{0.3} + 0.6s^{0.8} + 1}$ (hypothetical plant)	5	0.3	3

Table 2
Expressions for numerator and denominator polynomials in the CFE

$p = q$	$P_p(Z^{-1})$ $(k = 1)$ and $Q_q(Z^{-1})$ $(k = 0)$
1	$(-1)^k z^{-1} r + 1$
3	$(-1)^k (r^3 - 4r) z^{-3} + (6r^2 - 9) z^{-2} + (-1)^k 15 z^{-1} r + 15$
5	$(-1)^k (r^5 - 20r^3 + 64r) z^{-5} + (-195r^2 + 15r^4 + 225) z^{-4}$ $+ (-1)^k (105r^3 - 735r) z^{-3} + (420r^2 - 1050) z^{-2} + (-1)^k 945 z^{-1} r + 945$

are typically borrowed from the realm of optimal control (Stengel, 1994). The HPSO-TVAC algorithm on the other hand uses the same particle representation scheme as well as objective function as that used for the modified DE. Table 3 shows the parametric set-up for these algorithms. We choose the standard set of parameters, equipped with which the algorithms have been shown to be at the peak of their performance (over benchmark functions) in the existing literature (Ratnaweera and Halgamuge, 2004; Cao et al., 2005; Das et al., 2005). No hand tuning of parameters has been allowed in any case to make the comparison fair enough.

Table 3
Parameter settings for the different algorithms

HPSO-TVAC		Modified DE		Binary GA (Vesterstrøm and Thomson, 2004)	
Parameter	Value	Parameter	Value	Parameter	Value
Pop_size	40	Pop_size	10*D*	Initial Pop_size	50
Inertia weight	0.794	CR_{max}	1.0	No. of bits per gene	50
C_1	Linearly varying 0.35→2.4	CR_{min}	0.5	Mutation probability	0.01
C_2	Linearly varying 2.4→0.35	Scale factor *F*	Uniformly distributed random number between 0.5 and 1.0 with mean value 0.75	Uniform crossover probability	0.6
V_{max}	3.00				
Re-initialization velocity	Linearly decaying from V_{max} to $0.1V_{max}$				

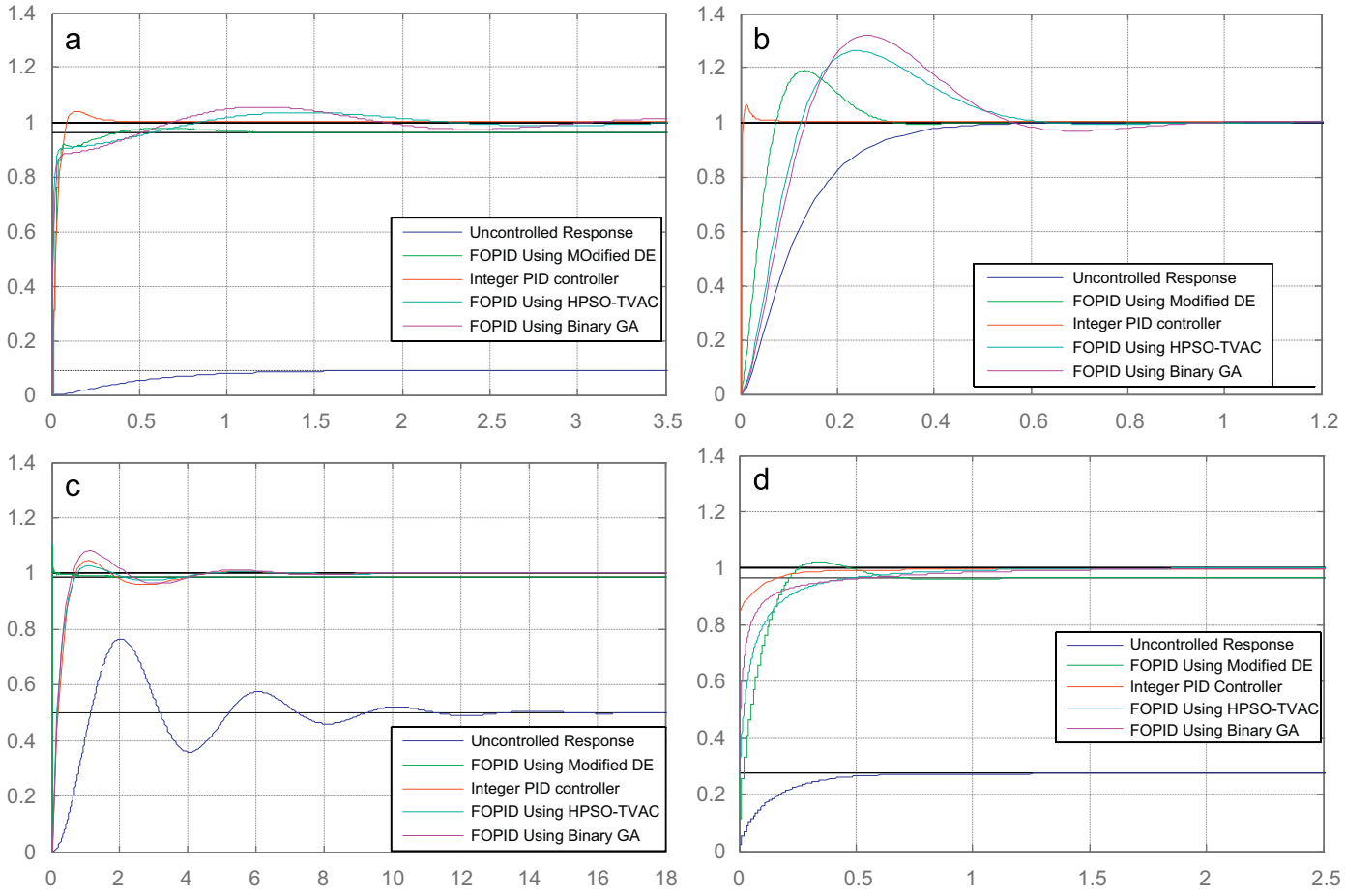

Fig. 3. Unit step response of the closed-loop systems for the test problems: (a) Design Problem I (integer-order plant), (b) Design Problem II (integer-order plant), (c) Design Problem III (fractional-order plant), (d) Design Problem IV (fractional-order plant).

5.4. Simulation strategy

We run three population-based optimization algorithms namely HPSO-TVAC, a modified DE and the binary encoded GA suggested in Holland (1975) and Cao et al. (2005) over four design problems according to the user specifications summarized in Table 1. All the algorithms have been developed from scratch in Visual C++ platform on a Pentium IV, 2.2 GHz PC, with 512 KB cache and 2 GB of main memory in Windows Server 2003 environment. The graphs and figures have been obtained using MATLAB 6.5. Twenty-five independent runs (with different seeds for the random number generator) were carried out for each of the algorithms and each run was continued up to 10^5 function evaluations (FEs). In case of DE since $D = 5$, NP = 50 and this approximately corresponds to a $G_{max} = 2000$ for 10^5 FEs. In what follows, we report the results for the median run of each algorithm (when the runs for a single algorithm have been ranked according to their final accuracy).

Table 4
The FOPID controller transfer functions as found with modified DE for four test problems

Process plant transfer function $G_p(s)$	Controller transfer function $G_c(s)$
$\frac{k}{(Js+b)(Ls+R)+k^2}$, $J = 0.01$, $b = 0.1$, $k = 0.01$, $R = 1$, $L = 0.5$	$36.762 + 221.852s^{-0.668} + 40.719s^{0.824}$
$\frac{s^2}{50s+400}$	$0.349 + 19.287s^{-0.949} + 1.009s^{0.322}$
$\frac{1}{0.8s^{2.2}+0.5s^{0.9}+1}$	$21.22 + 1.37s^{-0.92} + 12.05s^{0.93}$
$\frac{1}{0.9s^{0.3}+0.6s^{0.8}+1}$ (Hypothetical plant)	$1.72 + 41.524s^{-0.668} + 1.59s^{0.824}$

5.5. Results

Fig. 3 shows the dynamic response characteristics of the closed-loop systems for design problems I–IV as specified in Table 1. The integer-order PID controller as marked in Fig. 3 was obtained by minimizing the same objective function (in Eq. (27)) in three dimensions, taking $\lambda = \mu = 1$. Table 4 provides the FOPID controller transfer functions for four test problems as found with

Table 5
Summary of the performance of closed-loop system under different PID controllers against the unit step

Process plant	Different controllers used	Unit step response obtained			Final objective function values obtained
		Maximum overshoot (%)±standard deviation (%)	Rise time (s)±standard deviation (s)	Steady-state error (%)±standard deviation (%)	
I	Fractional controller using DE	3.11±(0.31)	0.395±(0.051)	3.5±(0.010)	0.00±(0.0000)
	Integer PID controller using DE	4.23±(0.34)	0.101±(0.007)	0.1±(0.001)	0.00±(0.0000)
	Fractional controller using PSO	3.91±(0.41)	0.822±(0.091)	1.9±(0.021)	0.0001±(0.0000)
	Fractional controller using GA	6.31±(0.87)	0.695±(0.088)	2.1±(0.056)	0.0312±(0.0025)
II	Fractional controller using DE	18.15±(0.77)	0.0727±(0.001)	0±(0.000)	0.00±(0.0000)
	Integer PID controller using DE	6.11±(0.45)	0.00667±(0.001)	0.1±(0.001)	0.00±(0.0000)
	Fractional controller using PSO	26±(0.95)	0.125±(0.009)	1.0±(0.012)	0.0001±(0.0000)
	Fractional controller using GA	32±(1.02)	0.138±(0.011)	1.3±(0.034)	0.0632±(0.0041)
III	Fractional controller using DE	7.69±(0.23)	0.023±(0.001)	0.9±(0.023)	0.00±(0.0000)
	Integer PID controller using DE	6.12±(0.39)	0.651±(0.034)	0.4±(0.021)	0.00±(0.0000)
	Fractional controller using PSO	5.78±(0.45)	0.646±(0.056)	1.1±(0.071)	0.0001±(0.0000)
	Fractional controller using GA	8.29±(0.89)	0.625±(0.077)	1.3±(0.081)	0.0711±(0.0021)
IV	Fractional controller using DE	1.93±(0.089)	0.218±(0.015)	1.6±(0.034)	0.00±(0.0000)
	Integer PID controller using DE	0.21±(0.011)	0.435±(0.078)	0.2±(0.010)	0.00±(0.0000)
	Fractional controller using PSO	0.12±(0.012)	0.982±(0.101)	0.1±(0.009)	0.0001±(0.0000)
	Fractional controller using GA	0.27±(0.017)	1.312±(0.313)	0.3±(0.013)	0.0522±(0.0037)

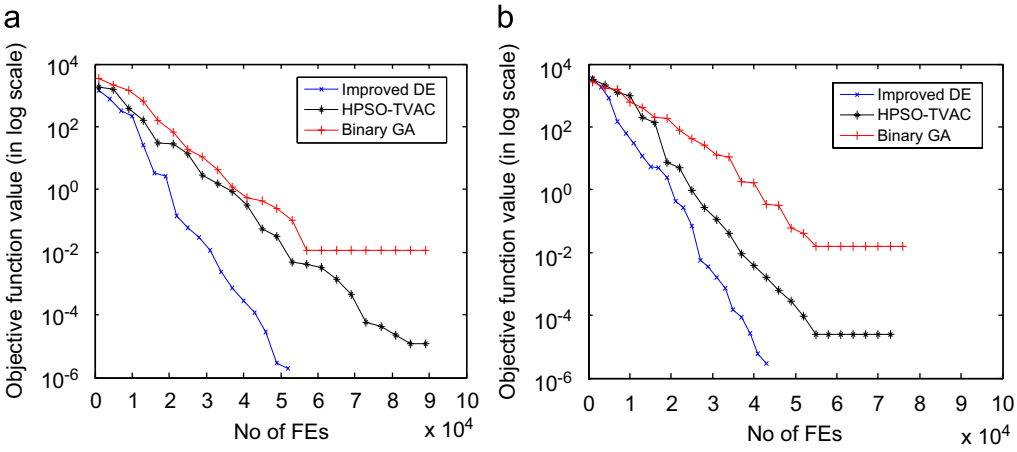

Fig. 4. Convergence characteristics for an integer and a fractional plant: (a) Integer plant I, (b) Fractional plant I.

modified DE. Table 5 reports the maximum overshoot (in %), rise time (in s) and steady-state error (in %) for the unit step response of each closed-loop system under the competitor PID controllers considered here. All entries in this table are the mean of the 25 independent runs of the modified DE, the HPSO-TVAC and the binary GA algorithm and come with the respective standard deviations as well. Convergence characteristics, i.e., how the objective function value decreases with the number of FEs, are shown in Fig. 4 for one integer-order plant and one fractional-order plant. The graphs indicate that the DE-based method could find better solutions consuming lesser amount of computational time.

It is noted that for the given common performance criteria on peak overshoot M_P, rise time t_{rise} (s), and steady-state error e_s, the fractional-order controller achieves better results than its integer counterpart in general for the fractional-order plants III and IV. The DE-based FOPID controller provides results closest to the three user specifications as listed in Table 1 in each case.

6. Discussion and conclusions

An intelligent optimization method for designing FOPID controllers based on the DE is presented in this paper. Fractional calculus can provide novel and higher performance extension for FOPID controllers. However, the difficulties of designing FOPID controllers increase, because FOPID controllers also take into account the derivative order and integral order in comparison with traditional PID controllers. To design the parameters of the FOPID controllers efficiently, the DE/rand/1/bin algorithm is modified with respect to its scale factor F and crossover rate CR. The proposed method has been shown to outperform a state-of-the-art version of the PSO algorithm and a binary GA-based method especially for the fractional-order plants. The proposed scheme of fractional PID controller design will thus find extensive commercial application in the next-generation controller design.

References

Astrom, K., Hagglund, T., 1995. PID Controllers; Theory, Design and Tuning. Instrument Society of America, Research Triangle Park.

Cao, J., Liang, J., Cao, B., 2005. Optimization of fractional order PID controllers based on genetic algorithms. In: Proceedings of the International Conference on Machine Learning and Cybernetics, Guangzhou, 18–21 August.

Caputo, M., 1967. Linear model of dissipation whose Q is almost frequency independent—II. Geophysical Journal of the Royal Astronomical Society 13, 529–539.

Caputo, M., 1969. Elasticita e Dissipacione. Zanichelli, Bologna.

Chengbin, Ma., Hori, Y., 2004. The application of fractional order PID controller for robust two-inertia speed control. In: Proceedings of the 4th International Power Electronics and Motion Control Conference, Xi'an, August.

Chen, Y.Q., Xue, D., Dou, H., 2004. Fractional calculus and biomimetic control. In: Proceedings of the First IEEE International Conference on Robotics and Biomimetics (RoBio04), Shengyang, China, August. IEEE.

Chen, Y.Q., Ahn, H., Xue, D., 2005. Robust controllability of interval fractional order linear time invariant systems. In: Proceedings of the ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Paper # DETC2005-84744, Long Beach, CA, September 24–28, pp. 1–9.

Das, S., Konar, A., Chakraborty, U.K., 2005. Two improved differential evolution schemes for faster global search. In: ACM-SIGEVO Proceedings of GECCO, Washington, DC, June, pp. 991–998.

Dorcak, L., Petras, I., Kostial, I., Terpak, J., 2001. State-space controller design for the fractional-order regulated system. In: Proceedings of the ICCC'2001, Krynica, pp. 15–20.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor.

Lubich, C.H., 1986. Discretized fractional calculus. SIAM Journal on Mathematical Analysis 17 (3), 704–719.

Miller, K.S., Ross, B., 1993. An Introduction to the Fractional Calculus and Fractional Differential Equations. Wiley, New York.

Nakagawa, M., Sorimachi, K., 1992. Basic characteristics of a fractance device. IEICE Transactions on Fundamentals E75-A (12), 1814–1819.

Oldham, K.B., Spanier, J., 1974. The Fractional Calculus. Academic Press, New York.

Oustaloup, A., 1981. Fractional order sinusoidal oscillators: optimization and their use in highly linear FM modulators. IEEE Transactions on Circuits and Systems 28 (10), 1007–1009.

Petras, I., 1999. The fractional order controllers: methods for their synthesis and application. Journal of Electrical Engineering 50 (9–10), 284–288.

Podlubny, I., 1999a. Fractional Differential Equations. Academic Press, San Diego, Boston, New York, London, Tokyo, Toronto, p. 368.

Podlubny, I., 1999b. Fractional-order systems and $PI^{\lambda}D^{\mu}$ controllers. IEEE Transactions on Automatic Control 44 (1), 208–213.

Price, K., 1999. An introduction to differential evolution. In: Corne, D., Dorigo, M., Glover, V. (Eds.), New Ideas in Optimization. McGraw-Hill, UK, pp. 79–108.

Price, K., Storn, R., Lampinen, J., 2005. Differential Evolution—A Practical Approach to Global Optimization. Springer, New York.

Ratnaweera, A., Halgamuge, K.S., 2004. Self organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation 8 (3), 240–254.

Stengel, R.F., 1994. Optimal Control and Estimation. Dover Publications, New York.

Storn, R., Price, K., 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (4), 341–359.

Vesterstrøm, J., Thomson, R., 2004. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Proceedings of the Sixth Congress on Evolutionary Computation (CEC-2004). IEEE Press, New York, July 6–9.

Vinagre, B.M., Podlubny, I., Dorcak, L., Feliu, V., 2000. On fractional PID controllers: a frequency domain approach. In: Proceedings of IFAC Workshop on Digital Control-PID'00, Terrassa, Spain.