Martín, Fernando; Monje, Concepción A.; Moreno, Luis; Balaguer, Carlos (2015) DE-based tuning of $PI^{\lambda}D^{\mu}$ controllers. *ISA Transactions*, v. 59, pp.: 398-407.

DOI: https://doi.org/10.1016/j.isatra.2015.10.002

DE-based Tuning of $PI^{\lambda}D^{\mu}$ Controllers

Fernando Martín[a], Concepción A. Monje[a], Luis Moreno[a], Carlos Balaguer[a]

[a]*Robotics Lab, Department of Systems Engineering and Automation, Carlos III University, Madrid, Spain*

Abstract

A new method that relies on evolutionary computation concepts is proposed in this paper to tune the parameters of fractional order $PI^{\lambda}D^{\mu}$ controllers, in which the orders of the integral and derivative parts, λ and μ, respectively, are fractional. The main advantage of the fractional order controllers is that the increase in the number of parameters in the controller allows an increase in the number of control specifications that can be met. A Differential Evolution (DE) algorithm is proposed to make the controlled system fulfill different design specifications in time and frequency domains. This method is based on the minimization of a fitness function, and one of its advantages is that any requirement could be satisfied if it is properly incorporated into the fitness function. The tuning method has been implemented to control the position output of a DC motor. Experiments have been carried out in simulated and real conditions and they illustrate the effectiveness of this method. The same tuning technique has been applied to obtain the parameters of integer order PID controllers for comparison. The results indicate that the specifications can also be satisfied with integer order controllers for the system studied in this work.

Keywords: Fractional order $PI^{\lambda}D^{\mu}$ Controllers, Differential Evolution, Evolutionary Algorithms, Robust Control

1. Introduction

Fractional calculus can be defined as a natural extension of the classical mathematics. Since the earliest theoretical contributions on fractional

Email address: fmmonar@ing.uc3m.es (Fernando Martín)

derivatives and integrals made by Euler, Liouville and Abel, fractional order control has drawn the attention of many researchers [1].

For convenience, Laplace domain notion is commonly used to describe the fractional integro-differential operation. The Laplace transform of the fractional derivative/integral under zero initial conditions for order α ($0 < \alpha < 1$) is given by

$$\mathcal{L}\{{}_a\mathrm{D}_t^{\pm\alpha} f(t)\} = s^{\pm\alpha} F(s). \tag{1}$$

In [2], Podlubny proposed a generalization of the classical PI and PID controllers defined as PI^λ and $PI^\lambda D^\mu$, where the integrator order λ and the differentiator order μ assume real non-integer values. He also proved that these types of fractional order controllers have better control performances [3].

This new type of controllers encouraged a lot of works dedicated to fractional order control systems. Many researchers have been working on the development of new effective tuning techniques for non-integer order controllers by an extension of the classical control theory. A classification of these tuning methods was presented in [4]. In summary, these methods can be classified into analytical, rule-based and numerical ones.

In [5], by setting λ=μ, all parameters can be analytically derived by solving four nonlinear equations based on the gain crossover frequency, phase margin, phase crossover frequency and gain margin specifications. Based on the same specifications, the authors further developed this method in [6, 7]. Another widely used specification is the robustness to loop gain variations, which was proposed by Chen in [8] for fractional PI controllers. In [9, 10, 11], this specification along with the gain crossover frequency and phase margin was also used to design fractional-order PD^μ and PI^λ controllers, respectively. Obviously, the analytical methods are available only when the equations are simple and their number is as few as possible.

As for the rule-based method, it can easily calculate the controller parameters based on some tuning rules [12, 13, 14]. However, the plant should usually be a system having an S-shaped step response. This will limit the applications of this method.

Besides, the numerical method is another option for tuning the $PI^\lambda D^\mu$ controller, which is an optimization-based method. In [15], an **F-MIGO** algorithm for tuning a fractional PI controller was proposed, in which the parameters of the controller can be obtained by solving an optimization problem

with the load disturbance constraint. In one of our previous works [16], we proposed a method for tuning the fractional order $PI^{\lambda}D^{\mu}$ controller by solving a nonlinear optimization problem with five constraints. Considering the complexity of the equations, the Matlab optimization toolbox was adopted to solve this problem. However, the success of this method still mainly depends on the initial starting values. **In [17], Tepljakov *et al.* have applied their tuning method, named FOMCON ("Fractional-order Modeling and Control") [18], to fix the parameters of a fractional-order PD controller for position control of a laboratory modular servo system. Their numerical method permits the consideration of specifications in time and frequency domains.**

The objective of this work is to develop a novel numerical tuning method for $PI^{\lambda}D^{\mu}$ controllers based on evolutionary optimization techniques. In particular, we will exploit our previous knowledge about the Differential Evolution (DE) algorithm [19] to apply this method to the cited purpose.

DE is an evolutionary algorithm that solves an optimization problem by iteratively trying to improve a candidate solution according to a fitness value. In other words, the DE technique can be viewed as a particle-based method that evolves in time to the solution that yields the lowest value of the fitness function. In this case, each member of the population will be a possible fractional controller with five parameters to tune. If the system specifications are properly implemented in the fitness function, it is possible to obtain a controller that makes the system meet the requirements. This is an interesting feature, because the tuning method is not limited to a number of conditions like in traditional approaches [1]. In our previous work, this method has been successfully applied to several tasks for mobile robots [20, 21]. **The good behavior shown by the DE-based techniques encouraged us to implement the method presented in this paper. DE uses a stochastic gradient search that can be applied to find the solution of multiple optimization problems. Moreover, it has other interesting characteristics: it can deal with nonlinear state space dynamics and noise distributions, it does not require any assumptions on the shape of the posterior density, and the computational resources focus on the most relevant areas.**

The DE-based tuning method is applied here to estimate the parameters of a $PI^{\lambda}D^{\mu}$ controller for a DC motor with a known transfer function. Two different fitness functions have been modeled to meet specifications in time or frequency. The algorithm performance has been tested in simulation and

in the real plant. The same method and the same motor transfer function have been used to obtain the parameters of a traditional PID controller for comparison, showing that both types of controllers can be employed to meet the requirements for the given plant.

Many authors have suggested to use evolutionary optimization techniques to control processes. Some examples are given here. Chang and Chen [22] have proposed an adaptive Genetic Algorithm (GA) to tune the parameters of a fractional PID controller applied to the control of an active magnetic bearing system. Zhang *et al.* [23] have implemented a self-organized GA with "good global search properties and a high convergence speed" to optimize the parameters of PID controllers, showing simulated results for different plants. A tuning method based on a GA was designed by Thomas and Poongodi [24] to select the parameters of a PID controller for a third-order DC motor. Altinen *et al.* [25] have applied a tuning method based on GA to tune a PID controller for a jacketed batch polymerization reactor. Cao and Cao [26] have developed a tuning method for fractional order PID controllers that relies on Particle Swarm Optimization (PSO). Korani *et al.* [27] have presented "a new algorithm for PID controller tuning based on a combination of the foraging behavior of E coli bacteria foraging and PSO". To the best of our knowledge, most of these algorithms try to minimize variables in the time-domain, such as the quadratic error of the output. An interesting aspect of our approach is that it also allows the user to specify multiple constraints in the frequency-domain.

This paper is organized as follows. The evolutionary technique developed in this paper, which is applied to the tuning of fractional order $PI^{\lambda}D^{\mu}$ controllers, is detailed in Section 2, introducing two different cost functions to deal with the design specifications both in time and frequency domains. The experimental results are presented in Section 3, discussing the results obtained from the implementation of both cost functions. Finally, the most important conclusions are summarized in Section 4.

2. DE-based Tuning of $PI^{\lambda}D^{\mu}$ Controllers

A new strategy to estimate the parameters of a fractional $PI^{\lambda}D^{\mu}$ controller that meets some specifications for a given plant is proposed here. It relies on the DE algorithm, which is an evolutionary optimization technique based on the minimization of a fitness function. The block diagram of the control system is shown in Figure 1. Given a process with a known transfer

function $G(s)$, the DE-based algorithm calculates the parameters of the fractional controller $C(s)$ in order to satisfy several specifications in closed-loop.

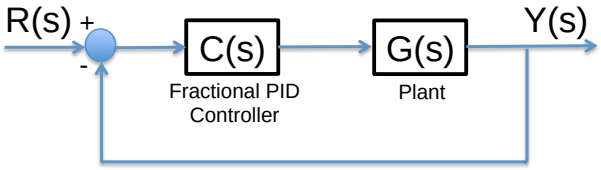

Figure 1: Block diagram of the closed-loop system.

The generalized formula of a $PI^{\lambda}D^{\mu}$ controller is

$$C(s) = k_p + \frac{k_i}{s^{\lambda}} + k_d s^{\mu}, \tag{2}$$

where λ and μ are the fractional orders of the integral and derivative parts of the controller, respectively. The other variables are the proportional (k_p), integral (k_i) and derivative (k_d) gains of a classic PID controller. It can be observed that there are two more parameters to tune in this type of controllers (λ, μ) when compared to the traditional PID. This means that more specifications can be met. It is crucial to study which are the most interesting specifications with regard to the robustness and the performance of the controlled system.

It is the purpose that the tuning method can be used to fulfill control specifications both in time and frequency domains. For this reason, two different cost functions will be used, depending on the objectives to be satisfied: the first one is a cost function defined in the time-domain that minimizes the error between the output and the reference input; the second one has been defined to satisfy some specifications in the frequency-domain, including robustness to gain variations of the plant.

The DE algorithm and cost functions used in this work are described next. **The code has been implemented as a user-defined function (.m) in Matlab according to the concepts explained in Agorithm 1. One advantage of the DE method is that it is very simple to implement, just a few lines of code are needed (about 50).**

2.1. Differential Evolution Algorithm

The performance of the DE algorithm implemented here is displayed in Algorithm 1. A more detailed explanation can be found in our previous works

Algorithm 1 DE-based Tuning of $PI^{\lambda}D^{\mu}$ Controllers

1: **for** $i = 1 : N_P$ **do**
2: $\quad \mathbf{pop}_i^1 \leftarrow init\_pop(pop\_limits)$ ▷ First population generation
3: $\quad e_i^0 \leftarrow fitness(plant, \mathbf{pop}_i^1)$ ▷ Cost function calculation
4: **end for**
5: **for** $k = 1 : max$ **do**
6: $\quad$ **for** $i = 1 : N_P$ **do**
7: $\quad\quad \mathbf{v}_i^k = \mathbf{pop}_a^k + F(\mathbf{pop}_b^k - \mathbf{pop}_c^k)$ ▷ Mutation
8: $\quad\quad$ **for** $j = 1 : D$ **do**
9: $\quad\quad\quad u_{i,j}^k = v_{i,j}^k, \forall p_{i,j}^k < \delta$ ▷ Crossover
10: $\quad\quad\quad u_{i,j}^k = pop_{i,j}^k, \forall p_{i,j}^k \geq \delta$
11: $\quad\quad$ **end for**
12: $\quad\quad e_i^k \leftarrow fitness(plant, \mathbf{pop}_i^k)$ ▷ New cost function calculation
13: $\quad\quad$ **if** $e_i^k < e_i^{k-1}$ **then** ▷ Selection
14: $\quad\quad\quad \mathbf{pop}_i^{k+1} = \mathbf{u}_i^k$
15: $\quad\quad$ **else**
16: $\quad\quad\quad \mathbf{pop}_i^{k+1} = \mathbf{pop}_i^k$
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: $\quad ind\_best \leftarrow min(e^k)$
20: $\quad bestmem \leftarrow \mathbf{pop}^k(ind\_best)$
21: $\quad$ **if** $convergence = true$ **then** ▷ Execution stops after convergence
22: $\quad\quad exit(bestmem)$
23: $\quad$ **end if**
24: **end for** ▷ Return best estimation

[20, 28].

The population set is composed of N_P candidates representing possible solutions. The fractional controller has five parameters to be fixed:

$$\mathbf{pop}_i^k = (k_p, k_i, k_d, \lambda, \mu), \tag{3}$$

where $\mathbf{pop}_i^k$ represents element i at iteration k. The first three parameters are the proportional, integral, and derivative gains of the controller, respectively. The other two are the exponents that are introduced to give the controller its fractional behavior. The initial population will be generated randomly in predefined intervals (line 2 of Algorithm 1). These intervals are necessary to limit the search space. A not-limited search space would imply the requirement of

huge populations, which is not practical from a computational point of view.

The fitness value has to be evaluated for each possible controller (line 3). The fitness function is a crucial aspect of the evolutionary technique because it includes the specifications that will be optimized.

The optimization process begins in line 5 with a loop that ends when the upper limit of iterations is reached or one of the convergence conditions is met. An inner loop is executed (line 6) to generate the new population for the next generation, evolving in time to the controller parameters that minimize the fitness function.

The evolutionary mechanism is divided into three main steps: mutation, crossover, and selection. First, each candidate of the current population is perturbed (line 7) to generate a mutated vector:

$$\mathbf{v}_i^k = \mathbf{pop}_a^k + F(\mathbf{pop}_b^k - \mathbf{pop}_c^k), \tag{4}$$

where $\mathbf{pop}_a^k$, $\mathbf{pop}_b^k$ and $\mathbf{pop}_c^k$ are three randomly chosen elements at iteration k and a, b and c are different from running index i. F is a real and constant factor which controls the amplification of the differential variations. The adequate value for this parameter is in the interval $[0.4, 1]$, as demonstrated in [19, 29].

After that, the crossover step (lines $8-11$) increases the diversity of the new generation. A trial vector $\mathbf{u}_i^k = (u_{i,1}^k, u_{i,2}^k, \ldots, u_{i,D}^k)^T$ is created from $\mathbf{v}_i^k$ and $\mathbf{pop}_i^k$ depending on the crossover probability:

$$u_{i,j}^k = \begin{cases} v_{i,j}^k; & \text{if } p_{i,j}^k < \delta, \\ pop_{i,j}^k; & \text{otherwise,} \end{cases} \tag{5}$$

where $p_{i,j}^k$ is a randomly chosen value from the interval $[0, 1]$ for each parameter j of the population member i at iteration k, and δ is the crossover probability that constitutes the crossover control variable. D is usually referred to as the number of chromosomes. In this case, it is equal to five because the controller has five parameters.

The population set of the next generation $(k + 1)$ is determined by a selection mechanism (lines $12-17$) that compares $\mathbf{u}_i^k$ to $\mathbf{pop}_i^k$. If the fitness value of the proposal $\mathbf{u}_i^k$ is better than the fitness value of the current member $\mathbf{pop}_i^k$, then $\mathbf{pop}_i^k$ is replaced by $\mathbf{u}_i^k$; otherwise, the current member $\mathbf{pop}_i^k$ is maintained for the next generation. The selection mechanism is described by the next expression:

$$\mathbf{pop}_i^{k+1} = \begin{cases} \mathbf{u}_i^k, & \text{if } e_{\mathbf{u}_i}^k < e_{\mathbf{pop}_i}^k, \\ \mathbf{pop}_i^k, & \text{otherwise,} \end{cases} \tag{6}$$

where $e^k_{\mathbf{pop}_i}$ is the cost value of the current candidate and $e^k_{\mathbf{u}_i}$ represents the fitness value of the trial vector.

The previous process (mutation, crossover, and selection) is applied to the whole population, obtaining the next generation population $(k+1)$.

After convergence or when a maximum number of iterations is reached, the algorithm returns the best member of the population (lines $19-20$). This solution corresponds to the fractional order controller that minimizes the fitness function.

To define a convergence criterion for the evolutionary algorithm is not an easy task. In [20], it is possible to determine the form of the probability distribution defined by the fitness function when the optimum value is found. The stopping condition is established depending on the expected value of the cost function. The same ideas cannot be applied in this work. The cost value will be (ideally) zero when the output exactly follows the reference (explained below in Section 2.2.1) or the phase is exactly flat around the crossover frequency (Section 2.2.2), but these cases are ideal situations that never happen in practical applications. In the current method, the convergence criterion is based on the empirical conditions that were also proposed in [20]. These requirements do not ensure convergence but may lead to good results in less time. If one of the following conditions is satisfied, the tuning process finishes:

- **Number of iterations without changes in the cost value of the best estimate is bigger than a constant (C_1).**
- **Number of iterations without changes in the cost value of the worst estimate is bigger than a constant (C_2).**
- **Number of iterations without changes in the difference between the cost value of the best estimate and the cost value of the worst estimate is bigger than a constant (C_3).**

The fitness function has to be designed according to the requirements that must be satisfied: the first one minimizes the error of the step response in the time-domain; the second one allows the introduction of multiple specifications in the frequency-domain.

2.2. Fitness functions

The DE method is capable of obtaining an adequate solution for a given problem if the fitness function is modeled in an adequate way. In this case, the objective is to estimate the parameters of a fractional PID controller for a given plant. This means that the process to be controlled needs to be analyzed. After that, a cost function has to be proposed to satisfy some design specifications. The evolutionary algorithm introduces a great flexibility in the tuning method, and multiple functions could be implemented. The requirements are not restricted to a limited number like in classic approaches. In this case, two different cost functions will be defined based on specifications in time and frequency, respectively. Each option has advantages and disadvantages, as discussed later.

An important limitation that has to be taken into account in this optimization problem is the control action that can be sent to the plant. When controlling a real device, the input that it can receive according to its components is usually limited to an interval. The control strategy will not be adequate if the control action is outside the admissible limits. The limitation of the control action has been included as an additional restriction in the cost function for both approaches: time and frequency. The possible solution will be discarded if the control action is not inside the desired interval. In the experiments, the controller will be applied to a DC motor which admits an input voltage in the interval $[-5, 5]$.

2.2.1. Time-Domain Tuning

The time response is a simple and illustrative variable that can be considered when designing a controller. One possibility consists of measuring the error between the output and the reference:

$$fitness(i) = \sum_{t=t_i}^{t_f} e_{TD}^2(t) = \sum_{t=t_i}^{t_f} [r(t) - y(t)]^2, \tag{7}$$

where $e_{TD}(t)$ is the signal error in the time domain. This error is computed in a finite interval in discrete-time between an initial and a final time. $r(t)$ is the reference signal (ideal output that is desirable to obtain) and $y(t)$ is the output of the plant. The optimum case from a control point of view will be a scenario where the output signal follows exactly the reference one.

The minimization of this fitness function implies that the time response will be the closest possible to the ideal one (taking into account the limitation

of the control action). Since the signal is controlled in the time-domain, if the optimization method succeeds, it is expected that the most typical characteristics of this control approach (peak time, stabilization time, overshoot) will present good values.

Nevertheless, there are many other factors that are not included in this type of control. For example, the robustness to variations in the gain of the plant is ignored, condition that occurs in many occasions in real applications. This factor can be taken into account when designing in the frequency domain, second type of cost function proposed here.

2.2.2. Frequency-Domain Tuning

There are many different conditions that can be met when designing in the frequency domain. Since the fractional order $PI^{\lambda}D^{\mu}$ controller has five parameters to be tuned, the traditional approaches consider that five different conditions could be satisfied [16]. As said before, an advantage of this tuning method is that more than five specifications could be satisfied if they are included in the fitness function.

One of the most common specifications for the PID controllers is the robustness to variations in the gain of the plant [30], which can be obtained if the following condition is satisfied:

$$\left.\frac{d(arg(F(s)))}{d\omega}\right|_{\omega=\omega_{cg}} = 0, \tag{8}$$

where ω_{cg} is the gain crossover frequency and $F(s) = C(s)G(s)$ is the open-loop transfer function. $G(s)$ is the transfer function of the plant. It means that the phase of the open-loop system will be flat at ω_{cg} and almost constant within an interval around ω_{cg}. The system will be more robust to gain changes and the overshoot of the response is almost constant within a gain range (iso-damping property of the time response).

For each possible solution, the cost function is calculated according to the slope of the phase of the open-loop response around the gain crossover frequency:

$$fitness(i) = \sum_{\omega=\omega_i}^{\omega_f} e_{FD}^2(\omega) = (\varphi(\omega) - (\varphi_m - \pi))^2, \tag{9}$$

where the error is calculated in a predefined interval between ω_i and ω_f around the gain crossover frequency. $\varphi(\omega)$ is the phase of the system at a frequency ω.

An advantage of this technique is that the interval where the response is robust to variations in the gain can be fixed. In our previous work [1, 16], only the slope of the argument were forced to be zero, and it was not possible to fix the interval of gains with a flat response.

More requirements have been included in this fitness function to satisfy the same specifications described in [16] (the solution is discarded if the following conditions are not met):

- Intervals for gain crossover frequency (ω_{cg}) and phase margin (φ_m). These variables have an important influence on the robustness of the system. For example, the phase margin is related to the damping of the system [31]. These parameters are computed via the following equations:

$$|C(j\omega_{cg})G(j\omega_{cg})|_{dB} = 0\ dB, \tag{10}$$
$$\arg\left(C(j\omega_{cg})G(j\omega_{cg})\right) = -\pi + \varphi_m. \tag{11}$$

  An acceptance interval for ω_{cg} and a minimum φ_m have been fixed.

- High frequency noise rejection: the objective is to attenuate the noise at high frequencies. The complementary sensitivity function is computed to satisfy this constraint:

$$\left|T(j\omega) = \frac{C(j\omega)G(j\omega)}{1 + C(j\omega)G(j\omega)}\right|_{dB} = A\ dB, \tag{12}$$
$$\forall\ \omega \geq \omega_t\ rad/s \Rightarrow |T(j\omega_t)|_{dB} \leq A\ dB, \tag{13}$$

  The possible solution is discarded if Equation 13 is not satisfied. The noise attenuation will be equal to A for frequencies $\omega \geq \omega_t$ rad/s.

- To ensure a good output disturbance rejection: this requirement is given by the sensitivity function:

$$\left|S(j\omega) = \frac{1}{1 + C(j\omega)G(j\omega)}\right|_{dB} = B\ dB, \tag{14}$$
$$\forall\ \omega \leq \omega_s\ rad/s \Rightarrow |S(j\omega_s)|_{dB} \leq B\ dB, \tag{15}$$

  where B is the desired value of the sensitivity for low frequencies ($\omega \leq \omega_s$). Equation 15 is applied to meet this requirement.

- Steady-state error cancellation: the fractional integrator $1/s^{\lambda}$ is as efficient as an integer order integrator [16], which means that this condition is always fulfilled with this type of controller.

As can be observed, it is possible to include many additional conditions if they can be incorporated into the fitness formula. The fitness value of the frequency-domain approach is computed by Equation 9, but many different specifications are included by restricting the fitness function to specified values that meet the requirements. If the controlled system does not meet one of this requirements, the candidate solution is discarded.

In order to prevent the control system to reach the stability border, which may happen when using optimization algorithm-based methods, the stability of the controlled system has to be checked after the tuning process, and so we did for the two proposals presented in this work, even though stability requirements have been also considered in the fitness formula for our approaches.

3. Experimental Results

The method performance will be studied in experiments in both simulated and real environments. The objective will be to control the position of a DC motor whose transfer function is known. It has to be said that this method can also work when the transfer function of the plant is unknown if the system output in closed-loop is available (black box).

The open-loop transfer function of a DC motor with position output can be deduced from the next equation:

$$G(s) = \frac{\theta(s)}{V(s)} = \frac{k_1}{1+Ts}\frac{k_2}{s}, \tag{16}$$

where k_1, k_2 and T are the motor gain, the encoder gain, and the motor constant time, respectively. The identification of these parameters can be done by analyzing the open-loop time response. $\theta(s)$ is the position output and $V(s)$ is the input voltage, which will be typically a unit step in these experiments.

The resulting second order transfer function that has been experimentally identified for the motor in Figure 2 is the following one:

$$G(s) = \frac{3.666}{s(0.2193s+1)}. \tag{17}$$

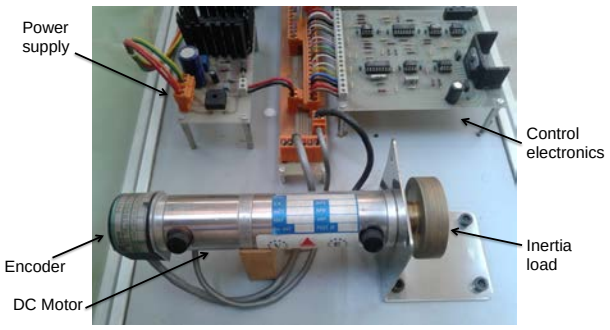

Figure 2: Real DC motor used in the experiments

In Section 2.2, two types of tuning methods were proposed. The first one was based on a fitness function obtained from parameters in the time-domain. The second one was designed in the frequency-domain. Both approaches will be considered in these experiments.

An additional restriction to be taken into account is the limitation of the control action that could be given to the real DC motor used for the experiments. The device works with a control action in the interval $[-5, 5]$ volts. A saturation effect appears when the input to the motor is out of the interval. This restriction has been included in all the experiments, causing a limitation in the speed of the control system.

First, the performance of our tuning method has been tested in simulation. However, the design of an optimum controller in a simulated environment is not enough because real conditions usually deteriorate the actual performance of the system. Therefore, the controllers have been tested later in the real motor platform.

Since the current method permits to use the same concepts and equations for different types of controllers, it has been applied to estimate the parameters of classical PID controllers for comparison. The only difference is that each population member will be formed by three parameters:

$$\mathbf{pop}_i^k = (k_p, k_i, k_d), \tag{18}$$

since the PID controller is given by the transfer function

$$C(s) = k_p + \frac{k_i}{s} + k_d s. \tag{19}$$

The configuration parameters of the optimization filter are: upper limit of iterations: 100; $\mathbf{C_1 = C_2 = C_3 = 20}$; population size: 100; initial interval for the controller gains: $[0, 5]$; initial interval for the controller exponents: $[0, 1]$; DE internal parameters: $F = 0.7$, $\delta = 0.5$.

The Ninteger Matlab toolbox, developed by Valério and Sá da Costa [32], has been applied to implement the fractional-order controller. The approximation algorithm is the Crone method (with N = 10) proposed by Oustaloup [33, 34], which uses a recursive distribution of N poles and N zeros.

3.1. Time-Domain Results

When the fitness function is defined in the time-domain according to Equation 7, the following fractional order controller is obtained:

$$C_{ft}(s) = 0.375 + \frac{1.119}{s^{0.226}} + 0.202s^{0.669}, \tag{20}$$

where subindex f indicates fractional order (i for integer order) and t means cost function in time-domain (ω for frequency-domain). This notation will be used from now on.

Regarding the computational time, the time per iteration is 7.00 **s and the number of iterations to converge is 23. The computational time is basically dependent on the cost function (**97.5% **of the total execution time), which calls the *nipid* function of the Ninteger toolbox (**80.8%**) and Matlab native functions (*feedback*:** 11.1%**, *step*:** 7.9%**)**.

The same method has been applied to estimate the parameters of an integer order PID controller for comparison. In this case and with the same configuration parameters, the next PID controller has been computed:

$$C_{it}(s) = 0.891 + \frac{0.001}{s} + 0.0402s. \tag{21}$$

The step response of the system when these controllers are included can be observed in Figure 3. For the fractional order controller, the overshoot is approximately equal to 17% and the peak time is 0.74 s. The settling time (with a 2% error band, value that will be taken from now on to compute this parameter) is 1.36 s. For the traditional PID, the overshoot is equal to 6%, the peak time is 1.06 s, and the settling time is 1.51 s. The fractional controller is a little bit faster but it presents a higher overshoot. However,

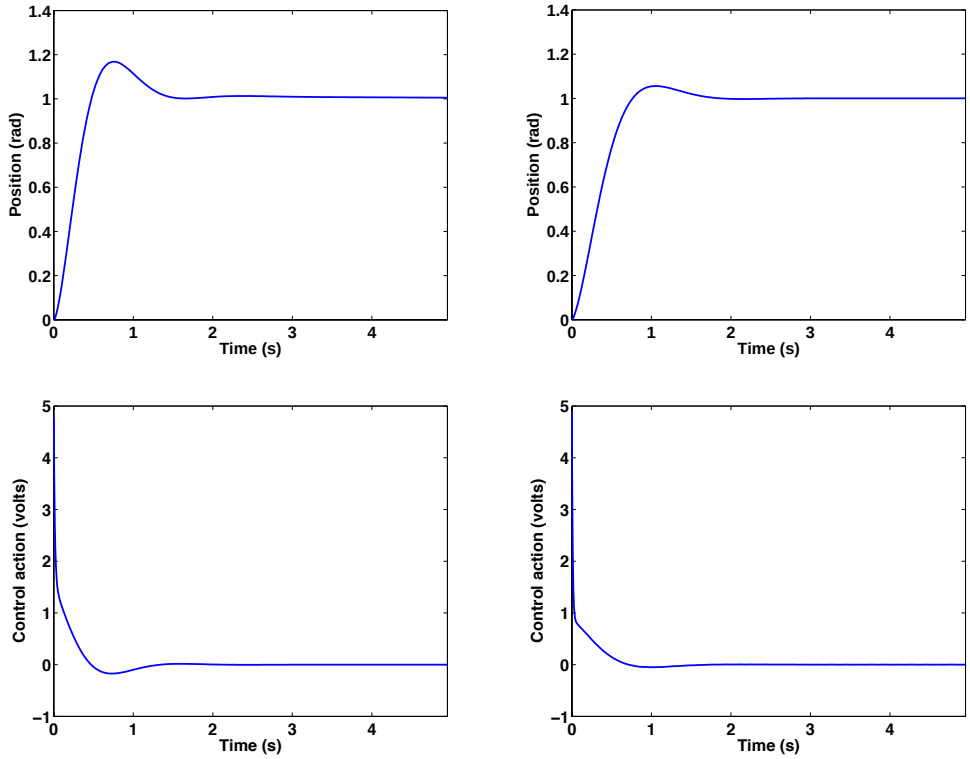

Figure 3: Step response (closed-loop) when designing in the time-domain. Simulation. Top left: fractional order $PI^{\lambda}D^{\mu}$ step response. Top right: traditional PID step response. Bottom left: fractional order $PI^{\lambda}D^{\mu}$ control action. Bottom right: traditional PID control action.

the differences are not important and both controllers present similar performances. The steady-state error is zero in both cases, which is a logical result because the plant transfer function presents a pole at the origin. The control action is inside the tolerance interval in both cases.

It has to be remarked that the limitation of the plant control action is specially important in this case because faster systems imply more abrupt control actions. Much better results could be obtained with the current method for a different plant with a wider control action interval. However, this limitation has been included to make this experiment more realistic.

The controllers presented in Equations 20 and 21 have been implemented in the control loop of the real system. The step responses and the control

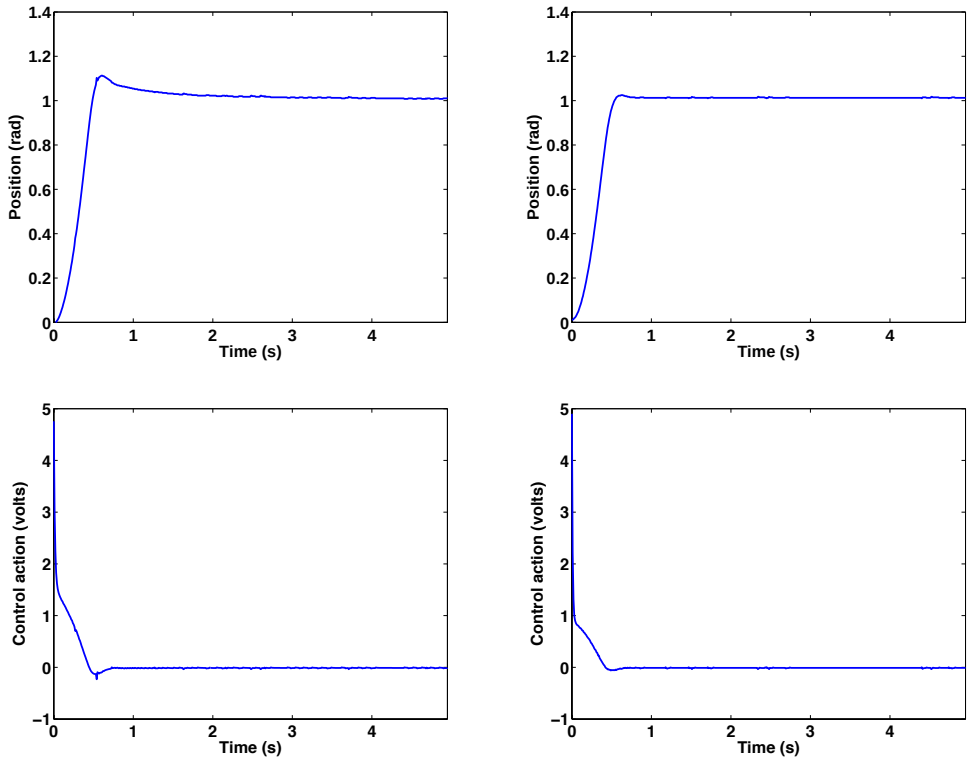

Figure 4: Step response (closed-loop) of the real system when designing in the time-domain. Top left: fractional order $PI^{\lambda}D^{\mu}$ step response. Top right: traditional PID step response. Bottom left: fractional order $PI^{\lambda}D^{\mu}$ control action. Bottom right: traditional PID control action.

actions are displayed in Figure 4. For the fractional order $PI^{\lambda}D^{\mu}$ controller, the overshoot is 11%, the peak time is 0.60 s and the settling time is 2.08 s. For the traditional PID, the overshoot is 2.5%, the peak time is 0.63 s and the settling time is 0.67 s. Only slightly better results are obtained for the overshoot and the peak response when compared to the simulated ones.

The differences between the real and the simulated results are sufficiently small, which implies that the linear motor model has been estimated within a reasonable error and that the nonlinear characteristics of the motor are not very significant. However, the existing mismatches between the theoretical and real models will be used later for the frequency-domain approach to test the

robustness of the system to these uncertainties, reason why we are not interested in a very accurate model identification.

To sum this section up, the DE-based method has estimated a controller that meets the time-domain requirements. Therefore, it can be concluded that it is an appropriate tuning method for this type of controller. No significant differences were found between both control approaches (fractional order and integer order), so it is not possible to conclude that one control scheme was better than the other for this plant. Although the DE-based algorithm estimates adequate controllers, the control action restriction due to the real system saturation constraint limits the control performance.

3.2. Frequency-Domain Results

Interesting specifications such as robustness to changes in the gain plant cannot be defined in the time-domain. This requirement must be specified in the frequency-domain. Therefore, the second approach proposed in this paper was to design a controller that follows specifications in the frequency-domain. Several conditions can be satisfied if the controller is estimated according to the fitness function presented in Equation 9 and the restrictions described in Section 2.2.2. The following conditions have been included as configuration parameters of the fitness function:

- Minimum phase margin φ_m equal to 15°.
- Interval when the phase is intended to be flat $(\omega_f - \omega_i)$ equal to 0.25 decades.
- Gain crossover frequency ω_{cg} in the interval $[1, 30]$ rad/s.
- High frequency noise rejection: $\omega_t = 100\ rad/s \Rightarrow |T(j\omega_t)|_{dB} \leq -15\ dB$.
- Sensitivity: $\omega_s = 0.01\ rad/s \Rightarrow |S(j\omega_s)|_{dB} \leq -15\ dB$

An important property can be deduced when this control approach is compared to the classical control methods. With the classical methods, the fractional order $PI^{\lambda}D^{\mu}$ controller with five different parameters can be tuned to meet five different specifications. With the evolutionary-based technique, the number of specifications that could be satisfied is not that restrictive. More specifications could be met if they can be implemented in the cost function. When an integer order PID controller is tuned with a classical

method, three different requirements could be met. With the DE-based tuning method, it will be possible to meet the previous five specifications even with this integer order controller. Maybe the solution is worse or cannot be found because the controller is not the most adequate one to satisfy the conditions, but the tuning method will try to find the best solution.

The solution of the optimization filter for the fractional order controller is now

$$C_{f\omega}(s) = 0.2725 + \frac{0.0577}{s^{0.8291}} + 0.0007s^{0.9499}. \tag{22}$$

Regarding the computational time, the time per iteration is 20.45 **s and the number of iterations to converge is** 43**. The computational time is basically dependent on the cost function (**98.6% **of the total execution time), which calls the *nipid* function of the Ninteger toolbox (**37.5%**) and Matlab native functions (*bode*:** 27.7%**, *margin*:** 19.6%**, *step*:** 6.0%**, *feedback*:** 3.9%**). The difference with respect to the time-domain approach is that Bode diagrams are needed to compute the fitness value of each particle.**

The Bode diagrams of the system with this controller are drawn in Figure 5. It can be observed that the system follows the specifications, the phase is the flattest possible around the gain crossover frequency. The phase margin is $\varphi_m = 65.71^\circ$ and the crossover frequency is $\omega_{cg} = 1.00$ rad/s. The high frequency noise rejection and the sensitivity conditions are also satisfied.

When an integer order PID controller is designed according to the frequency-domain specifications, the following solution is obtained:

$$C_{i\omega}(s) = 0.3178 + \frac{0.1131}{s} + 0.0001s. \tag{23}$$

The Bode diagrams can be seen in Figure 6. The phase is the flattest possible around the gain crossover frequency. The phase margin is $\varphi_m = 58.70^\circ$ and the crossover frequency is $\omega_{cg} = 1.18$ rad/s. The high frequency noise rejection and the sensitivity conditions are also met.

The step responses of the system with both controllers are shown in Figure 7. For the fractional controller, the overshoot is 11.2%, the peak time is 3.34 s and the settling time is 12.32 s. For the traditional controller, the overshoot is 20.4%, the peak time is 2.76 s and the settling time is 7.54 s. However, no conclusions can be drawn from these data because the controllers have been tuned to meet other specifications.

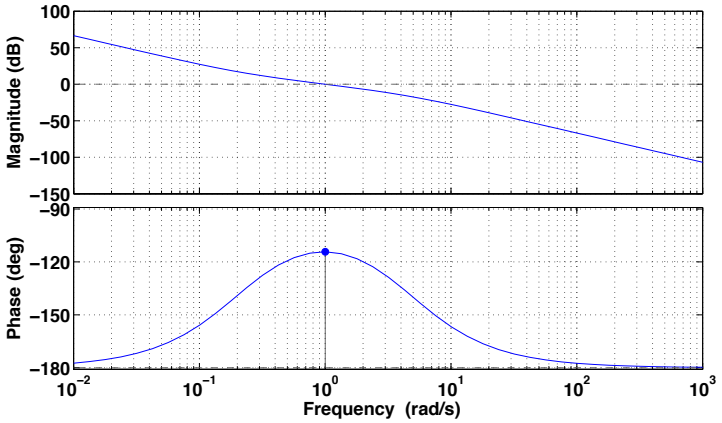

Figure 5: Bode diagrams when designing a fractional order controller in the frequency-domain. $\varphi_m = 65.71°$, $\omega_{cg} = 1.00$ rad/s.

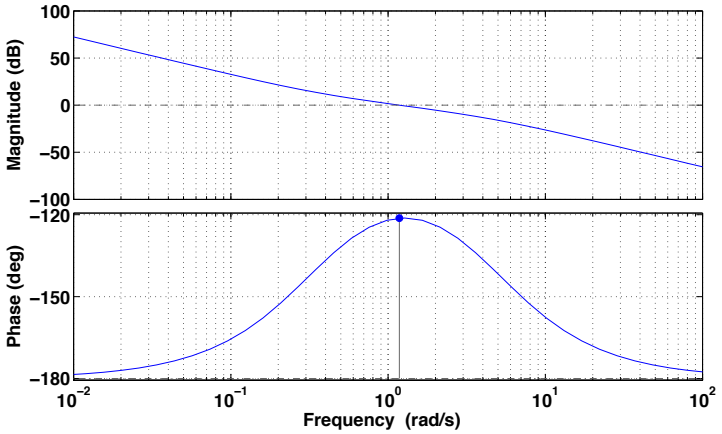

Figure 6: Bode diagrams when designing an integer order controller in the frequency-domain. $\varphi_m = 58.70°$, $\omega_{cg} = 1.18$ rad/s.

Since the controllers have been designed to be robust under changes in the plant gain, small changes in this parameter should not cause poorer performances. This condition means that the overshoot should remain constant (or within a narrow interval) when the gain is shifted. To check this property, the

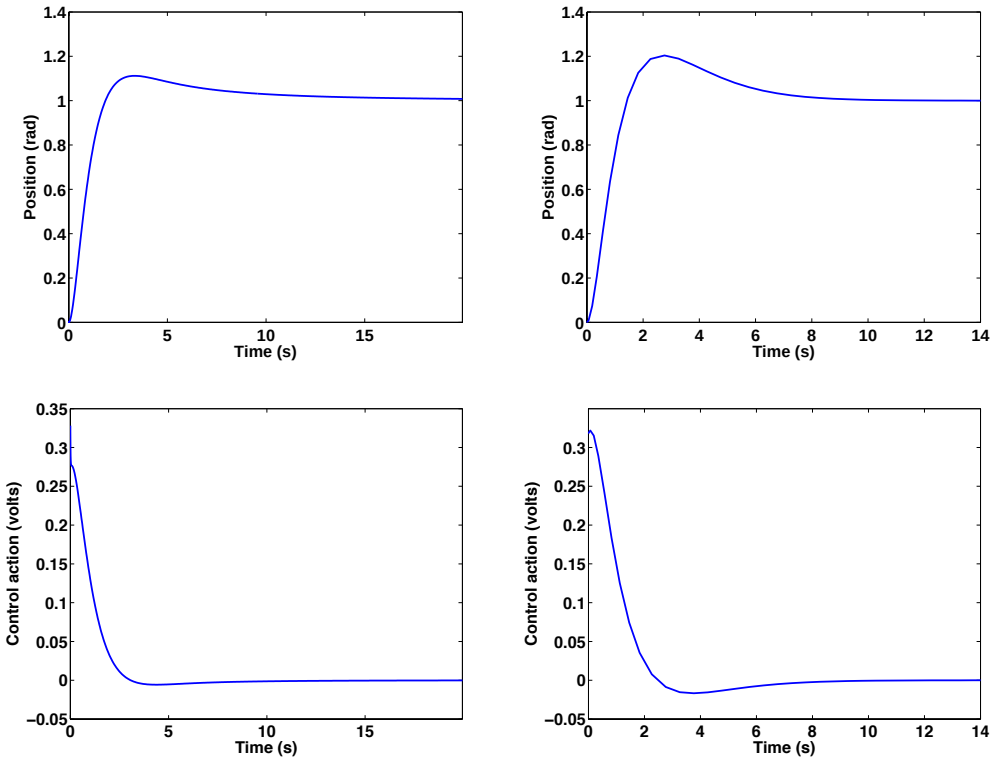

Figure 7: Step response (closed-loop) when designing in the frequency-domain. Top left: fractional order $PI^{\lambda}D^{\mu}$ step response. Top right: traditional PID step response. Bottom left: fractional order $PI^{\lambda}D^{\mu}$ control action. Bottom right: traditional PID control action.

step response is computed when the gain of the plant is changed and the controller is not modified. The nominal gain of the plant is $K = k_1 k_2/T = 16.72$. The step response for this gain is represented in blue in Figure 8, together with the responses for three more gain values: $1.5K$ (25.08, magenta), $1.2K$ (20.06, red) and $0.8K$ (13.37, green).

As can be seen, the system is faster for higher gains and slower for lower ones. If the overshoot is analyzed, it is in a narrow band for the interval of gains studied in this experiment ($[0.8K, 1.5K]$). It varies within the interval $[10.6, 12.1]\%$ for the fractional order controller and the interval $[18.5, 22.3]\%$ for the traditional one. This is a promising result that leads us to conclude that the system presents a good behavior in this aspect, which is one of our main objectives when designing in the frequency-domain. Although the

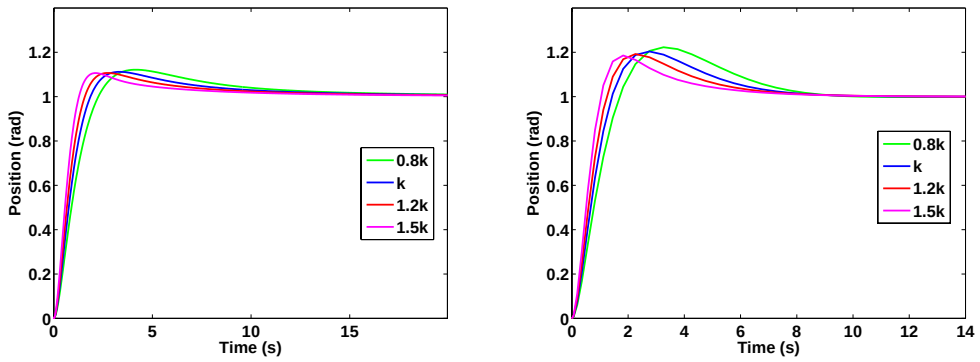

Figure 8: Step response for variable gains when designing in the frequency-domain. Simulation. Left: fractional order $PI^{\lambda}D^{\mu}$. Right: traditional PID.

results are better for the fractional order controller (the interval is narrower), it is also possible to obtain an adequate performance with the integer order one.

It can be concluded that both approaches (fractional and integer) can be used to perform an adequate control for the plant described in this paper. In a future work, it will be interesting to implement the current method for the control of more complex processes, comparing the fractional and the traditional approaches to highlight the advantages and disadvantages of both types of controllers.

The step responses and the control actions when implementing these controllers (Equations 22 and 23) in the real motor platform are shown in Figure 9.

For the fractional controller, the overshoot is 19.1%, the peak time is 0.72 s and the settling time is 6.11 s. For the traditional controller, the overshoot is 25.9%, the peak time is 0.73 s and the settling time is 4.81 s.

The step response of the real system has been measured when the gain of the plant (K) is changed (Figure 10). In order to do that, the gain of the plant has been multiplied by different values: nominal gain (K, blue), $1.5K$ (magenta), $1.2K$ (red) and $0.8K$ (green).

It can be seen that the speed of the output with both controllers slightly varies for this interval of gains. Observing the enlarged images, the overshoot varies within similar intervals for both controllers.

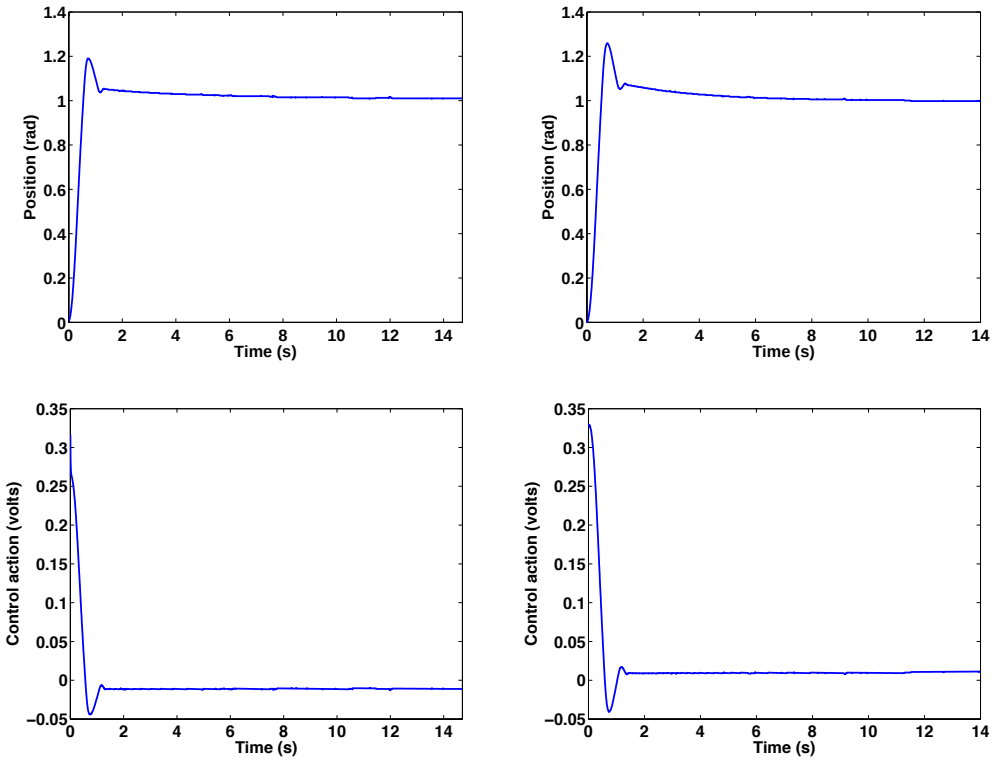

Figure 9: Step response (closed-loop) of the real system when designing in the frequency-domain. Top left: fractional order $PI^{\lambda}D^{\mu}$ step response. Top right: traditional PID step response. Bottom left: fractional order $PI^{\lambda}D^{\mu}$ control action. Bottom right: traditional PID control action.

Beyond the slight differences between the simulated and experimental results, our interest lays on the robustness of the experimental system to gain variations, which is the main design constraint to achieve with the tuning method proposed for the frequency domain. In fact, the mismatches between the theoretical and real models are needed to test how robust the control system performs to those model uncertainties, this way validating our control approach.

If the frequency-domain approach is compared to the time-domain one, the overshoots are similar for both cases but the settling times have increased for the frequency-based controllers, which is a logical result because temporal

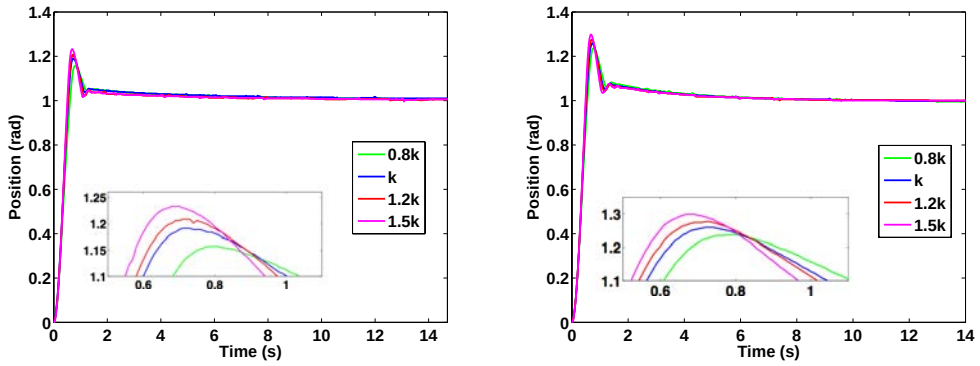

Figure 10: Step response for variable gains when designing in the frequency-domain. Real motor. Left: fractional order $PI^{\lambda}D^{\mu}$. Right: traditional PID. Bottom: zoom around the peak value.

constraints have not been included in the frequency-based fitness function. These increments have to be taken into account in the control process (for example, if the system is fed by a square signal, the frequency of the input must be restricted by the settling time). An interesting development to accomplish in a future work could be to combine both time and frequency fitness functions to obtain a controller that inherits the advantages of both approaches.

4. Conclusions

A new method based on evolutionary computation concepts is proposed in this work to estimate the parameters of fractional order $PI^{\lambda}D^{\mu}$ controllers and traditional PID ones. This new tuning method has multiple advantages: it allows more flexibility in the design process because different control specifications can be chosen by simply changing the cost function; less knowledge about the plant is needed and successful results can be obtained without knowing the plant transfer function (if the output is available); if an adequate controller is in the space of parameters where the algorithm searches, the optimization method will success.

The algorithm capabilities have been examined when controlling the position output of a DC motor in both simulated and real environments. Two different strategies have been followed to design the controllers: the first one

optimizes the step response and the second one relies on some properties in the frequency-domain. To the best of our knowledge, there are no other previous research works dealing with the design of fractional order controllers with evolutionary techniques in the frequency-domain. In all cases, an additional restriction is included to limit the maximum control action that could be sent to the plant (according to the real system saturation limits).

The proposed method has computed two controllers (fractional and integer ones) that meet the time-domain requirements in a satisfactory way, both in simulation and experimental environments. No significant differences were found between both controlled systems for the real plant studied here when the time-domain is analyzed.

When designing in the frequency-domain, the variation of the overshoot of the response is the smallest possible when the gain of the plant is varied within an interval. This is an interesting feature because it makes the system more robust to changes in the gain of the plant. Slightly better results are obtained for the fractional order controller than for the integer order one, but both approaches fulfill the frequency specifications both in simulation and real environments.

A challenging work to do is to apply the current method for the control of more complex systems (higher order or nonlinear ones) to take the most of this technique and compare fractional and integer order performances under more restrictive situations.

Another interesting expansion of this work could be to combine both time and frequency fitness functions to tune a controller that inherits the advantages of both approaches.

Besides, other objectives could be pursued in the optimization process. For example, the goal of the control strategy could be to minimize the energy or the control action. A deeper analysis of the evolutionary method (convergence conditions, initialization parameters) when it is applied for these purposes would also be helpful to improve this technique. **Other tuning methods based on different evolutionary strategies, such as PSO, are being developed in order to make comparisons with the current method.**

References

[1] C. A. Monje, Y. Q. Chen, D. Xue, B. M. Vinagre, V. Feliu, Fractional-order Systems and Controls. Fundamentals and Applications (Advances

in Industrial Control), Springer-Verlag London, 2010.

[2] I. Podlubny, Fractional-order systems and $PI^{\lambda}D^{\mu}$ controllers, IEEE Transactions on Automatic Control 44 (1999) 208–214.

[3] I. Podlubny, I. Petráš, B. M. Vinagre, P. O'Leary, L. Dorčák, Analogue realizations of fractional-order controllers, Nonlinear Dynamics 29 (2002) 281–296.

[4] D. Valério, J. Sá da Costa, A review of tuning methods for fractional PIDs, in: Proceedings of the 4th IFAC Workshop on Fractional Differentiation and Its Applications, Badajoz, Spain (2010).

[5] B. M. Vinagre, Modelling and control of dynamic systems characterized by integro-differential equations of fractional order (in Spanish), Ph.D. thesis, UNED, Madrid, Spain, 2001.

[6] R. Caponetto, L. Fortuna, D. Porto, A new tuning strategy for a non integer PID controllers, in: Proceedings of the 1st IFAC Workshop on Fractional Differentiation and Its Applications, Niagara Falls (2004).

[7] C. Zhao, D. Xue, Y. Q. Chen, A fractional order PID tuning algorithm for a class of fractional order plants, in: Proceedings of the International Conference on Mechatronics and Automation, Bordeaux, France (2005), pp. 216–221.

[8] Y. Q. Chen, H. Dou, B. M. Vinagre, C. A. Monje, A robust tuning method for fractional orderPI controllers, in: Proceedings of the 2nd IFAC Workshop on Fractional Differentiation and Its Applications, Porto, Portugal (2006), pp. 22–27.

[9] H. Li, Y. Luo, Y. C. Chen, A fractional order proportional and derivative (FOPD) motion controller: tuning rule and experiments, IEEE Transactions on Control System Technology 18 (2010) 516–520.

[10] Y. Luo, Y. Q. Chen, C. Wang, Y. Pi, Tuning fractional order proportional integral controllers for fractional order systems, Journal of Process Control 20 (2010) 823–831.

[11] Y. L. H. Malek, Y. Q. Chen, Identification and tuning fractional order proportional integral controllers for time delayed systems with a fractional pole, Mechatronics 23 (2013) 746–754.

[12] D. Valério, J. S. da Costa, Tuning of fractional PID controllers with Ziegler-Nichols type rules, Signal Processing 86 (2006) 2771–2784.

[13] D. Valério, J. S. da Costa, Tuning Rules for Fractional PIDs. Fractional Calculus: Theoretical Developments and Applications in Physics and Engineering, Springer Berlin, 2007.

[14] Y. Q. Chen, H. Dou, B. M. Vinagre, C. A. Monje, Optimal fractional order proportional integral controller for varying time-delay systems, in: Proceedings of the 17th IFAC World Congress (2008), pp. 4910–4915.

[15] Y. Q. Chen, T. Bhaskaran, D. Xue, Practical tuning rule development for fractional order proportional and integral controllers, Journal of Computational and Nonlinear Dynamics 3 (2008) 021403(1)–021403–(8).

[16] C. A. Monje, B. M. Vinagre, V. Feliu, Y. Chen, Tuning and auto-tuning of fractional order controllers for industry applications, Control Engineering Practice 16 (2008) 798–812.

[17] A. Tepljakov, E. Petlenkov, J. Belikov, Tuning and digital implementation of a fractional-order PD controller for a position servo, International Journal of Microelectronics and Computer Science 4 (2013) 116–123.

[18] A. Tepljakov, E. Petlenkov, J. Belikov, A flexible matlab tool for optimal fractional-order PID controller design subject to specifications, in: W. L. Li, Q. Zhao (Eds.), Proceedings of the 31st Chinese Control Conference, Hefei, Anhui, China (2012), pp. 4698–4703.

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

[20] F. Martín, L. Moreno, S. Garrido, D. Blanco, High-accuracy global localization filter for three-dimensional environments, Robotica 30 (2011) 363–378.

[21] F. Martín, L. Moreno, R. Triebel, R. Siegwart, Two different tools for three-dimensional mapping: DE-based Scan Matching and Feature-based Loop Detection, Robotica 32 (2014) 19–41.

[22] L. Y. Chang, H. C. Chen, Tuning of fractional PID controllers using adaptive genetic algorithm for active magnetic bearing system, WSEAS Transactions on Systems 8 (2009) 158–167.

[23] J. Zhang, J. Zhuang, H. Du, S. Wang, Self-organizing genetic algorithm based tuning of PID controllers, Information Sciences 179 (2009) 1007–1018.

[24] N. Thomas, P. Poongodi, Position control of DC motor using genetic algorithm based PID controller, in: World Congress on Engineering (WCE), volume 2, London, U.K. (2009).

[25] A. Altinten, F. Ketevanlioğlua, S. Erdoğana, H. Hapoğlub, M. Alpbaz, Self-tuning PID control of jacketed batch polystyrene reactor using genetic algorithm, Chemical Engineering Journal 138 (2008) 490–497.

[26] J. Y. Cao, B. G. Cao, Design of fractional order controllers based on particle swarm optimization, in: 1st IEEE Conference on Industrial Electronics and Applications (2006), pp. 1–6.

[27] W. M. Korani, T. H. Dorrah, H. M. Emara, Bacterial foraging oriented by particle swarm optimization strategy for PID tuning, in: IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) (2009), pp. 445–450.

[28] L. Moreno, S. Garrido, M. L. Muñoz, Evolutionary filter for robust mobile robot localization, Robotics and Autonomous Systems 54 (2006) 590–600.

[29] D. Zaharie, Critical values for the control parameters of differential evolution algorithms, in: Proceedings of MENDEL 2002, 8th International Conference on Soft Computing, Czech Republic (2002), pp. 62–67.

[30] Y. Q. Chen, K. L. Moore, Relay feedback tuning of robust PID controllers with iso-damping property, IEEE Transactions on Systems, Man, and Cybernetics, Part B 35 (2005) 23–31.

[31] G. Franklin, J. Powell, A. Naeini, Feedback Control of Dynamic Systems, Reading, MA, 1986.

[32] D. Valério, J. Sá da Costa, Ninteger: a non-integer control toolbox for MatLab, in: Proceedings of Fractional Differentiation and its Applications, Citeseer, Bordeaux, 2004.

[33] A. Oustaloup, CRONE control: Robust control of non-integer order, Paris, Hermes (1991).

[34] A. Oustaloup, F. Levron, F. Nanot, B. Mathieu, Frequency-band complex noninteger differentiator: Characterization and synthesis, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 47 (2000) 25–40.