Journal of the Brazilian Society of Mechanical Sciences and Engineering (2022) 44:212
https://doi.org/10.1007/s40430-022-03491-1

TECHNICAL PAPER

Comparative study of tuning techniques for fractional PID controllers for FOPDT systems

Rodrigo Teixeira Aguiar[1] · Bruno Silva de Lima[2] · Antônio Augusto Torres Maia[3]

Received: 29 October 2021 / Accepted: 22 March 2022 / Published online: 27 April 2022

Abstract
Traditionally, PID (proportional–integral–derivative) controllers are used in the process control field to keep the system output at a desirable value. Fractional-order PID controllers, however, tend to perform better than the traditional integer-order version due to characteristics like its nonlinear nature and the two extra tuning parameters that allow a better adjustment in the controller action. Similar to the integer controllers, there are different methods to tune fractional-order PID controllers. In this work, it is presented a comparative study of fractional PID controller tuning techniques applied to first-order plus dead-time plants (FOPDT). The equations are thoroughly detailed, and the algorithm used in the simulations is presented. Five tuning methods for fractional PIDs were found in the available literature and adjusted for being applicable to the problem under study. Two of these methods are analytical and the other three heuristics (genetic algorithm, differential evolution, and Nelder–Mead algorithm). The best methods found were the methods based on the use of genetic algorithm and Nelder–Mead simplex algorithm which presented values of the ITAE criterion at least two times lower when compared to the other methods. Also, robustness and disturbance rejection for each controller considered were analyzed, presenting adequate responses. Moreover, the step response curves represented overshoot values lower than 10% and a response time below 10 seconds, values considered satisfactory for the studied system.

Keywords Fractional Calculus · Fractional PID controller · Control systems · Fractional PID tuning methods

1 Introduction

Dead time is present in most processes of everyday life, being found in economic, biological, and industrial systems. It can be caused by several sources, for example one can quote sensor processing time, transport of mass, energy, among others. As stated by Saini and Sharma [20], most industrial processes with dead time can be approximated by a first-order plus dead-time processes (FOPDT). A limitation of this type of process is that the dead time decreases the stability margins of the system, making the task of developing a controller design more difficult. Likewise, tuning techniques for proportional, integral and derivative (PID) controllers, of FOPDT systems has been an attractive topic for research.

In the process control field, in most applications PID controllers are used. The controllers of this type are applied due to their simplicity and good performance, as written by Shahri et al. [23] and Padhee et al. [18]. Working with integer exponents tends to be simpler and more understandable for a large number of operators and engineers.

Despite this, the concept of fractional calculus has been changing this perception. One of its applications is performed in control theory because, since the modeling is based on differential equations, it takes only a small step to use the notion of fractional differential equations.

Fractional control was first proposed by Podlubny et al. [19]. As the fractional controllers have two parameters (α and β) more than the common PIDs to be tuned, they allow performances that cannot be achieved with conventional PID controllers and therefore better results can be obtained, as affirmed by Lachhab et al. [12]. For this reason, there is a growing interest in

Technical Editor: Adriano Almeida Gonçalves Siqueira.

✉ Rodrigo Teixeira Aguiar
rodrigotaguiar@gmail.com

[1] Universidade Estadual Paulista (UNESP), R. Dr. Bento Teobaldo Ferraz, 271 - Várzea da Barra Funda, São Paulo, SP 01140-070, Brasil

[2] Universidade Federal de Uberlândia (UFU), Av. João Naves de Ávila, 2121 - Santa Mônica, Uberlândia, MG 38408-100, Brasil

[3] Universidade Federal de Minas Gerais (UFMG), Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte, MG 31270-901, Brasil

the understanding of fractional control and in the methodology of tuning the gains and non-integer orders of these controllers.

Recent works on the field exemplify how this theory improves the effectiveness of control systems. Al-Saggaf et al. [1] used a fractional control to minimize the vibration of a rotary flexible joint system. By means of a comparison with an integer order state feedback control, it was concluded that the fractional controller presented better performance in relation of the parameters of interest. In order to asymptotically stabilize a damping system of a semi-active vehicle, Nguyen et al. [17] made use of a fractional-order derivative-based sliding mode controller (FD-SMC). Numerical simulations of the system were used to prove that the control system produced the expected result. To design an optimal controller for an air conditioning system, Nasirpour and Balochian [15] presented a new methodology that uses Particle Swarm Optimization (PSO) to search for the optimal parameters for a fractional-order PID (FOPID) controller. The parameters were tuned based on the minimization of a nonlinear objective function, which considered overshoot, ITSE, raising time, and settling time. The PSO-FOPID controller performed well with respect to the reference input.

For the control of the temperature on a surface of a thin plate, Jarrah [10] used the fractional control approach. The comparison of the proposed fractional controlling system with integer-order presented a better performance, mainly on the overshoot answer of the system. Several other authors have been using fractional calculus theory to improve systems control , such as Chekari et al. [6], Treesatayapun and Muñoz-Vázquez [25], Jahanshahi et al. [9], Birs et al. [4] and Homaeinezhad and Shahhosseini [8], and good results have been achieved. Several sources are available and summarized in Machado et al. [13], which is a report of works using fractional calculus theory from 1974 to 2010, and more recently by Kochubei [11] and Baleanu and Lopes [3], covering more recent works.

Thus the main objective of this work is to perform a comparison of some of the different tuning methods of fractional order controllers for first-order plus dead-time systems (FOPDT). In addition, the objective is to identify the best of these methods according to pre-established evaluation criteria, moreover, contribute to making the use of fractional PID controllers more accessible and understandable.

2 First-order plus dead-time systems (FOPDT)

A first-order system is a system that contains only one pole and no zeros. The modeling of these systems can be simple and capable of representing a large part of the control systems plants. For this reason, it is commonly used to approach systems with higher orders.

To approximate an unknown order system in open-loop as a FOPDT, its response to the unit step function must present the so-called "S-shaped response". With that, the main parameters of the plant are graphically identified. These systems are represented by the transfer function given by Equation 1, as in Ziegler et al. [28].

$$\frac{C_s}{R_s} = \frac{Ke^{-Ls}}{Ts+1}, \tag{1}$$

in which 'K' is the gain, 'T' is the time constant and 'L' is the delay time (dead time) of the system.

3 Fractional control

The traditional PID controllers are generalized by $PI^{\alpha}D^{\beta}$ type fractional order controllers, in which the exponents α and β represent the fractional portion of Differintegral as a function of time. These controllers were presented for the first time in the late 1990s by Podlubny et al. [19], evidencing that studies on this type of control are very recent. According to Lachhab et al. [12], when well-tuned, these controllers perform better than controllers with integer orders. Shah and Agashe [22] present the advantages of fractional control:

- Ability to achieve five different specifications, while classic PID reaches only three;
- The ease of achieving robustness to variations;
- The ease of continuous damping;
- Increased ability to tune to large orders or with large delays;
- Possibility of better responses for systems without phase minimum.
- Ease of dealing with nonlinearities with only one controller, without the need to break the system at different points of operation.
- Good results for nonlinearities.

The typical formulation of this controller in the time domain is given by Eq. 2 and in the frequency domain by Eq. 3, as in Valério and Da Costa [26].

$$u(t) = Pe(t) + I_0 D_t^{-\alpha} e(t) + D_0 D_t^{\beta} e(t), \tag{2}$$

$$G_c(s) = P + \frac{I}{s^{\alpha}} + Ds^{\beta}, \tag{3}$$

4 Tuning methods

The methods of tuning the parameters of the controller are divided into two parts: analytical and heuristic, as presented by Shahri et al. [23].

As the name implies, the analytic method is based on the analysis of characteristics and specifications of the system, which can be measured and placed in mathematical formulas, in order to find the parameters that will lead to the desired response. The heuristic method is based on the use of optimization algorithms, such as the genetic algorithm (GA), the electromagnetic (EM) algorithm, the differential evolution (DE), among others. These algorithms make several iterations to achieve the best result of a certain function using concepts of trial and error, or evolutionary concepts, in which the best result of one "generation" continues in the next "generation".

According to Monje et al. [14], Valério and Da Costa [27] and Lachhab et al. [12], the analytical tuning methods for fractional controllers are based on the specifications given to the system. These specifications include:

- Minimum α value to reset the error in steady state;
- Phase margin (ϕ_m) and gain cut-off frequency $(j\omega_{cg})$ specified to complete the conditions as represented in Eqs. 4 and 5.

$$\angle\left[C(j\omega_{cg})G(j\omega_{cg})\right] = -\pi + \phi_m, \tag{4}$$

$$|C(j\omega_{cg})G(j\omega_{cg})|_{dB} = 0dB, \tag{5}$$

- Gain Margin (g_m) and phase cutoff frequency $(j\omega_{cf})$ specified to complete the conditions as represented in Eqs. 6 and 7.

$$\frac{1}{|C(j\omega_{cf})G(j\omega_{cf})|} = g_m, \tag{6}$$

$$\angle\left[C(j\omega_{cf})G(j\omega_{cf})\right] = -\pi, \tag{7}$$

- Robustness to gain variations in Eq. 8.

$$\frac{d}{d\omega}\angle\left[C(j\omega_{cg})G(j\omega_{cg})\right] = 0, \tag{8}$$

- Rejection of high frequency noises (frequencies above ω_t) in Eqs. 9 and 10.

$$|T(j\omega)| = \left|\frac{C(j\omega)G(j\omega)}{1+C(j\omega)G(j\omega)}\right|_{dB} \leq AdB, \tag{9}$$

$$\forall\omega \geq \omega_t \frac{rad}{s} \rightarrow |T(j\omega_t)|_{dB} = AdB, \tag{10}$$

- Rejection of low frequency disturbances (frequencies below ω_s) in Eqs. 11 and 12.

$$|S(j\omega)| = |\frac{1}{1+C(j\omega)G(j\omega)}|_{dB} \leq BdB, \tag{11}$$

$$\forall\omega \leq \omega_s \frac{rad}{s} \rightarrow |S(j\omega_s)|_{dB} = BdB, \tag{12}$$

- Being $|f(x)|$ the magnitude in absolute value, $|f(x)|_{dB}$ the magnitude in decibels and $\angle|f(x)|$ the phase angle of the function f (x). These functions are formulated according to Eqs. 13, 14 and 15.

$$|f(x)| = \sqrt{R\left[f(x)\right]^2 + I\left[f(x)\right]^2}, \tag{13}$$

$$|f(x)|_{dB} = 20\log_{10}|f(x)|, \tag{14}$$

$$\angle\left[f(x)\right] = arctan\left(\frac{I[f(x)]}{R[f(x)]}\right), \tag{15}$$

In the available literature, five tuning methods for fractional PIDs were found. Methods 1 and 2, presented by Valério and Da Costa [26], are analytical methods that have similarities with the methods of Ziegler et al. [28]. Methods 3, 4, and 5 are heuristic methods that use different forms of optimization. Method 3 is demonstrated in the work of Shahri et al. [23], which uses the algorithm of differential evolution (DE). Method 4 uses the genetic algorithm (GA) in two distinct ways, described by Cao et al. [5], and by Padhee et al. [18]. Method 5 is based on the use of the Nelder–Mead algorithm, found in the Toolbox Fractional-Order Modeling and Control, as in Tepljakov et al. [24]. The subsequent topics explain the main features of these methods.

4.1 First method: method KTL

The KTL method resembles the first tuning method by Ziegler et al. [28] for using the open-loop system response due to a step input to tune the controller. Considering that the system response has the "S-Shape" format, the plant can be approximated as a FOPDT system. With this, the controller parameters can be tuned according to the values of the performed approximation.

Since this is an analytical method, the system specifications are needed to fine-tune the parameters. To do this, test specification values for FOPDT type plants must be created. Thus, two sets of test specifications are created, and it is possible to solve Eqs. 4, 5, 8, 9 and 11.

For the first group of specifications, the values used are represented in Eqs. 16, 17, 18, 19, 20 and 21.

$$\omega_{cg} = 0.5rad/s, \tag{16}$$

$$\phi_m = \frac{2}{3}rad \approx 38°, \tag{17}$$

$$\omega_t = 10rad/s, \tag{18}$$

$$A = -10dB, \tag{19}$$

$$\omega_s = 0.01 rad/s, \tag{20}$$

$$B = -20 dB, \tag{21}$$

The P, I, α, D, and β parameters are dependent on K, L, and T plant constants. Then, using the least-squares method, it is possible to adjust polynomial curves for each parameter that depend on plant constants. By transferring each of the curves to matrices, as in Eqs. 22 and 24, it is possible to tune the fractional controller. It is worth noting that the polynomial approximation works well only in a range of K, T, and L values, and such value constraints are represented in Eqs. 23 and 25, as in the work of Valério and Da Costa [27].

$$\begin{pmatrix} -0.0048 & 0.2664 & 0.4982 & 0.0232 & -0.0720 & -0.0348 \\ 0.3254 & 0.2478 & 0.1429 & -0.1331 & 0.0258 & -0.0171 \\ 1.5766 & -0.2098 & -0.1313 & 0.0713 & 0.0016 & 0.0114 \\ 0.0662 & -0.2528 & 0.1081 & 0.0702 & 0.0328 & 0.2202 \\ 0.8736 & 0.2746 & 0.1489 & -0.1557 & -0.0250 & 0.0323 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ L \\ T/K \\ L^2 \\ (T/K)^2 \\ TL/K \end{pmatrix} = \begin{pmatrix} P \\ I \\ \alpha \\ D \\ \beta \end{pmatrix} \tag{22}$$

$$0.1 \leq \frac{T}{K} \leq 5 \;\; e \; L \leq 2, \tag{23}$$

$$\begin{pmatrix} 2.1187 & -3.5207 & -0.1563 & 1.5827 & 0.0025 & 0.1824 \\ -0.5201 & 2.6643 & 0.3453 & -1.0944 & 0.0002 & -0.1054 \\ 1.0645 & -0.3268 & -0.0229 & 0.2018 & 0.0003 & 0.0028 \\ 1.1421 & -1.3707 & 0.0357 & 0.5552 & -0.0002 & 0.2630 \\ 1.2902 & -0.5371 & -0.0381 & 0.2208 & 0.0007 & -0.0014 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ L \\ T/K \\ L^2 \\ (T/K)^2 \\ TL/K \end{pmatrix} = \begin{pmatrix} P \\ I \\ \alpha \\ D \\ \beta \end{pmatrix} \tag{24}$$

$$5 \leq \frac{T}{K} \leq 50 \;\; e \; L \leq 2, \tag{25}$$

For the second set of test specifications, a new polynomial approximation is found, represented in Eq. 32, with its proper constraint on Eq. 33. The values of this second group of variables are presented in Eqs. 26, 27, 28, 29, 30 and 31.

$$\omega_{cg} = 0.5 rad/s, \tag{26}$$

$$\phi_m = 1 rad \approx 57°, \tag{27}$$

$$\omega_t = 10 rad/s, \tag{28}$$

$$A = -20 dB, \tag{29}$$

$$\omega_s = 0.01 rad/s, \tag{30}$$

$$B = -20 dB, \tag{31}$$

$$\begin{pmatrix} -1.0574 & 24.5420 & 0.3544 & -46.7325 & -0.0021 & -0.3106 \\ 0.6014 & 0.4025 & 0.7921 & -0.4508 & 0.0018 & -1.2050 \\ 1.1851 & -0.3464 & -0.0492 & 1.7317 & 0.0006 & 0.0380 \\ 0.8793 & -15.0846 & -0.0771 & 28.0388 & -0.0000 & 1.6711 \\ 0.2778 & -2.1552 & 0.0675 & 2.4387 & -0.0013 & 0.0021 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ L \\ T/K \\ L^2 \\ (T/K)^2 \\ TL/K \end{pmatrix} = \begin{pmatrix} P \\ I \\ \alpha \\ D \\ \beta \end{pmatrix} \tag{32}$$

$$0.1 \leq \frac{T}{K} \leq 50 \;\; e \; L \leq 0.5, \tag{33}$$

4.2 Second method: method Kcr

The Kcr method applies to systems that, if inserted in a closed-loop control with proportional gain, exhibit constant oscillation for a given value of this gain. Therefore, it resembles the second tuning method of Ziegler et al. [28]. The value of gain that generates the constant oscillation is called Critical Gain (Kcr), and the oscillation period with this same gain is called the Critical Period (Pcr). As in Method 1 the control parameters are found analytically using Eqs. 4, 5, 8, 9 and 11. However, in this case, the parameters of the control are observed varying according to the constants Kcr and Pcr. It is then possible to adapt matrices of polynomial approximation dependent on these constants. Using the first test group of Method 1 and Eqs. 16 to 21, the matrices presented in Eqs. 34 and 36 are found with their proper constraints on Eqs. 35 and 37, as presented by Valério and Da Costa [27].

$$\begin{pmatrix} 0.4139 & 0.0145 & 0.1584 & -0.4384 & -0.0855 \\ 0.7067 & 0.0101 & -0.0049 & -0.2951 & -0.1001 \\ 1.3240 & -0.0081 & -0.0163 & 0.1393 & 0.0791 \\ 0.2293 & 0.0153 & 0.0936 & -0.5293 & -0.0440 \\ 0.8804 & -0.0048 & 0.0061 & 0.0749 & 0.0810 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ Kcr \\ Pcr \\ 1/Kcr \\ 1/Pcr \end{pmatrix} = \begin{pmatrix} P \\ I \\ \alpha \\ D \\ \beta \end{pmatrix} \tag{34}$$

$$K_{cr} * P_{cr} \leq 64, \tag{35}$$

$$\begin{pmatrix} -1.4405 & 0.0000 & 0.4795 & 32.2516 & 0.6893 \\ 5.7800 & 0.0238 & 0.2783 & -56.2373 & -2.5917 \\ 0.4712 & -0.0003 & -0.0029 & 7.0519 & 0.1355 \\ 1.3190 & -0.0024 & 2.6251 & -138.9333 & 0.1941 \\ 0.5425 & -0.0023 & -0.0281 & 5.0073 & 0.2873 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ Kcr \\ Pcr \\ 1/Kcr \\ 1/Pcr \end{pmatrix} = \begin{pmatrix} P \\ I \\ \alpha \\ D \\ \beta \end{pmatrix} \tag{36}$$

$$64 \leq K_{cr} * P_{cr} \leq 640, \tag{37}$$

For the second test group used in Method 1 Eqs. 26 to 31, the matrix on Eq. 38 with its proper restriction on Eq. 39 is found.

$$\begin{pmatrix} -25.8467 & -0.0119 & 40.4266 & -14.5136 & 0.0147 & 1.6841 & 4.7503 & 0.0144 & -7.0200 \\ 10.5528 & 0.2352 & -17.0426 & 6.3144 & -0.0617 & -0.9399 & -1.5547 & -0.0687 & 3.4357 \\ 0.6213 & -0.0034 & 0.2257 & 0.1069 & 0.0008 & 1.1809 & 0.0904 & 0.0010 & -0.8139 \\ 15.7620 & -0.1771 & -23.0396 & 8.2724 & 0.1987 & -0.8892 & -2.9981 & 0.0389 & 2.8619 \\ 1.0101 & 0.0024 & -0.8606 & 0.1991 & -0.0005 & -0.9300 & -0.1609 & -0.0009 & 0.5846 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ Kcr \\ Pcr \\ Pcr^2 \\ Pcr * Kcr \\ 1/Kcr \\ 1/Pcr \\ Kcr/Pcr \\ Pcr/Pcr \end{pmatrix} = \begin{pmatrix} P \\ I \\ \alpha \\ D \\ \beta \end{pmatrix} \tag{38}$$

$$P_{cr} \leq 2, \tag{39}$$

4.3 Third method: DE algorithm

A heuristic method based on the algorithm called DE (differential evolution) was proposed by Shahri et al. [23]. In order to find one or more poles of the system and perform an optimization, it is necessary to know at least two specifications of the control. In this case, these specifications are the Maximum overshoot, Eq. 40, and the time to establishment in 2%, Eq. 41. Using these two specifications, the desired dominant poles ($P_{1,2}$) can be found as in Eq. 42.

$$M_p = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}, \tag{40}$$

$$T_s = \frac{4}{\zeta * \omega_n}, \tag{41}$$

$$P_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -x \pm jy, \tag{42}$$

Thus, taking into account the equation of characteristic equation of the closed-loop transfer function, Eq. 43, the transfer function for first-order systems with delay, Eqs. 44, 3 and 42, and considering H (s) = 1, Eq. 45 is obtained.

$$1 + G_c(s)G_p(s)H(s) = 0, \tag{43}$$

$$G_p(s) = \frac{K}{Ts+1}e^{-Ls}, \tag{44}$$

$$1 + \left[P + I(-x \pm jy)^{-\alpha} + D(-x \pm jy)^{\beta}\right] * G_p(-x \pm jy) = 0, \tag{45}$$

From Eq. 45 it is possible to define the objective function (J), which is minimized by the algorithm and is represented in Eq. 46. The modification of Eq. 45 to 46 is intended to ensure that the equation is always positive and it is as close as possible to the real number axis. Therefore, this minimized equation provides the poles of the system.

$$J(P, I, \alpha, D, \beta) = |R|^2 + |I|^2 + |\phi|^2, \tag{46}$$

in which R is the real part, I is the imaginary part and ϕ is the phase angle of Eq 45.

The first step of the DE algorithm is to define the number of variables (V), the number of the population (NP), which, according to Ardia et al. [2] can correspond to ten times of 'V' and, then the crossing frequency (CR),a value between 0.1 and 0. The mutation factor (F), which, according to Salehinejad et al. [21], can be a constant or a random value between 0 and 2. Thus, Eq. 47 can be used to calculate this factor Shahri et al. [23].

$$F = 0.4 + [rand(0.1)] * 0.6, \tag{47}$$

The next step is to define the maximum and minimum for each parameter (P, I, D, α, and β) in the form of Eqs. 48 and 49.

$$\overline{X}_{\min} = \{P_{min,} I_{min,} \alpha_{min,} D_{min,} \beta_{min,}\}, \tag{48}$$

$$\overline{X}_{\max} = \{P_{max,} I_{max,} \alpha_{max,} D_{max,} \beta_{max,}\}, \tag{49}$$

The number of maximum iterations used in this type of algorithm is 200, a standard value according to Ardia et al. [2].

Then, it is necessary to generate an initial 'population' of 'NP' vectors, named $\overline{X}_{iG}$ according to Eq. 50, and then choose one of these to play the initial vector role.

$$\overline{X}_{iG} = \left\{P_{iG,} I_{iG,} \alpha_{iG,} D_{iG,} \beta_{iG,}\right\}, \tag{50}$$

in which G is the generation (starts G = 0), i is an integer between 1 and NP and each of the variables ($P_{iG}, I_{iG}, \alpha_{iG}, D_{iG}, \beta_{iG}$) are presented as possible solutions of the system, generated randomly between the maximum and minimum limits.

In the third part, the "mutation" of the vectors is carried out. It is possible to find in the literature, several different ways of performing this mutation, as represented in Ganbavale [7]. Some examples are represented in Eqs. 51, 52 and 53.

$$'DE/rand/1' : \overline{V}_{iG} = \overline{X}_{i1G} + F\left(\overline{X}_{i2G} - \overline{X}_{i3G}\right)), \tag{51}$$

$$'DE/current - to - rest/1' : \overline{V}_{iG} = \overline{X}_{i1G} + F\left(\overline{X}_{bestG} - \overline{X}_{i1G}\right) + F\left(\overline{X}_{i2G} - \overline{X}_{i3G}\right), \tag{52}$$

$$'DE/rand/1' : \overline{V}_{iG} = \overline{X}_{bestG} + F\left(\overline{X}_{i1G} - \overline{X}_{i2G}\right), \tag{53}$$

in which *i1*, *i2* and *i3* are random integers with values between 1 and NP, different from each other and different from the initial 'i' chosen. $\overline{X}_{bestG}$ is the $\overline{X}$ that presents the best result in the optimization of the objective function among the generation G population.

The fourth step of the algorithm is the crossing of the vectors. In this step, a vector $\overline{U}_{ig}$ is created from the intersection of the vectors $\overline{V}_{iG}$ and $\overline{X}_{iG}$, as represented in Eq. 54.

$$\overline{U}_{ig} = \begin{cases} \overline{V}_{ijG} & \text{if rand}(0.1) \le CR \text{ or } j = j_{rand} \\ \overline{X}_{ijG} & \text{for other cases} \end{cases}, \tag{54}$$

so, 'j' is an integer with a value between 1 and 'V', which represents the variable to be selected for crossover. 'j_{rand}' is an arbitrary 'j' chosen in the same interval to ensure that at least one variable passes through the crossing process, that is, to ensure that $\overline{U}_{ig}$ is different from $\overline{X}_{ijG}$.

The last step is to choose the best option that minimizes the objective function and start a new generation with a new $\overline{X}_{i(G+1)}$, according to Eq. 55. Once this step is completed, the algorithm restarts until it reaches the maximum number of iterations defined.

$$\overline{X}_{i(G+1)} = \begin{cases} \overline{U}_{ig} & \text{if } f\left(\overline{U}_{ig}\right) \le f\left(\overline{X}_{iG}\right) \\ \overline{X}_{iG} & \text{if } f\left(\overline{U}_{ig}\right) \ge f\left(\overline{X}_{iG}\right) \end{cases}, \tag{55}$$

4.4 Fourth method: GA algorithm

This section describes each of the special features of the GA algorithm application.

4.4.1 GA optimizing the five parameters of the controller

This method, presented by J. Cao et al. [5], consists of the optimization of all five FOPID (first-order proportional–integral–derivative) parameters. In this case, the function to be minimized (J) is the weighted average of ISE and IAE, which are two control evaluation criteria that are explained in Sect. 4.2: Evaluation Criteria. This function has the format of Eq. 56.

$$J = w_1 * \int_0^\infty |e(t)|dt + w_2 * \int_0^\infty e^2(t)dt, \tag{56}$$

The values of w_1 and w_2 are constants which when added have a value equal to 1 and may vary depending on what is required to obtain from the function 'J'.

4.4.2 GA optimizing the fractional exponents of the controller

In this variation of the GA method, there is only the optimization of the fractional exponents of the plant, the other parameters being tuned from the second method of Ziegler et al. [28]. This application of GA is presented by Padhee et al. [18] and the function to be minimized, in this case, is the ISE (J_{new}) according to Eq. 57.

$$J_{\text{new}} = \int_0^\infty e^2(t)dt, \tag{57}$$

4.4.3 Fifth method: Nelder–Mead simplex algorithm

Nelder and Mead [16] created a method of optimizing functions in order to find the local minimum in a given value constraint. The algorithm begins with the construction of a Simplex (a triangle of 'N' dimensions) and aims to find the minimum local 'N' variables. This algorithm was then used

by the FOMCOM Library to find the optimum values for the 5 variables (5 dimensions) of the FOPID.

As in the genetic algorithm, the function chosen to be optimized by the Nelder–Mead algorithm is usually a criterion of control evaluation. Therefore, none of the control specifications are required. The ITAE, IAE, ISE, among others, are generally the ones chosen to find an optimal solution of the controller parameters

5 Methodology

The tests were performed in simulations in MATLAB software (R2020a). For the simulation of fractional transfer functions, a specific toolbox called FOMCON was found in the literature in Tepljakov et al. [24]. The choice of software is due to the good capacity to compute the fractional differential equations, as well as to accurately simulate the responses of the control systems.

To perform the tests of the methods found in the literature, a test FOPDT was used, represented in Eq. 58. The values of 'K', 'T' and 'L' of Eq. 58 have been chosen so that they are within the limits of values in which all described methods can be applied.

$$C(s) = \frac{1}{s+1}e^{-2s}, \tag{58}$$

In order to model the responses of this plant, a change in the way of writing the delay is necessary. Using Padé function, which makes the polynomial expansion of an exponential function truncating the function in n terms, the delay (e^{-2s}) can be converted into a polynomial approximation.

To find the satisfactory truncation of the polynomial expansion, the step input response was tested for one, four, and eight truncation terms, and the results can be visualized in Fig. 1. In the permanent regime, the approximation of all three is satisfactory, but in the transient regime, the approximation with truncation in one term presented a significant difference. It is also observed that the higher the number of terms, the smaller the difference in the steady-state. When performing the disturbance test however, bigger Padé approximations resulted in oscillatory behaviour, so the truncation was chosen at four terms, resulting in a plant shown in Eq. 59.

$$C(S)_{pade4} = \frac{s^4 - 10s^3 + 45s^2 - 105s + 105}{s^5 + 11s^4 + 55s^3 + 150s^2 + 210s + 105} \tag{59}$$

From there, the function that will represent FOPID is created using FOMCON libraries.

The code used in all simulations is represented in Appendix 1, where the values of the parameters P, I, α, D and β found in each of the methods were replaced in lines 24 to 28.

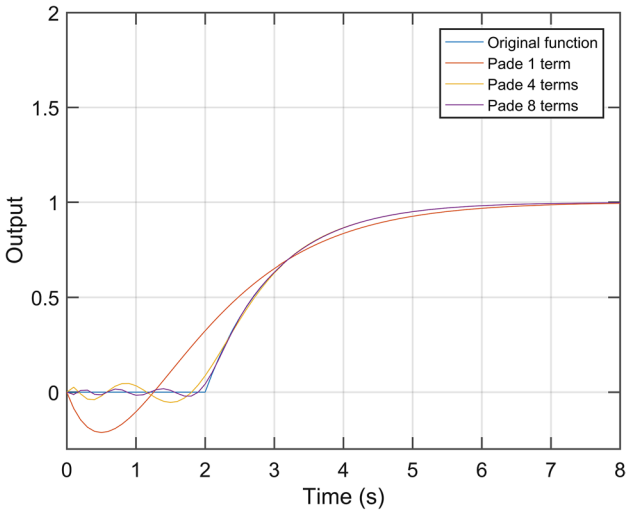

Fig. 1 Comparison between the Padé approximations and the real delay

Table 1 Equations for the second tuning method of Ziegler and Nichols(ZIEGLER; NICHOLS, 1942)

Controller	Kp	Ti	Td
P	0.5 Kcr	∞	0
PI	0.45 Kcr	Pcr / 1.2	0
PID	0.6 Kcr	Pcr / 2	0.125 Pcr

To evaluate if the FOPID controller has advantages over the conventional PID, the second tuning method of Ziegler and Nichols was used. This method uses the values of Critical Gain and Critical Period, applying them in simple equations, as represented in Table 1.

The value of the critical gain was found by the Root locus method, applied to the modified FOPDT plant (Eq. 59), which can be seen in Fig. 2. At the point where the dominant pole crosses the imaginary axis with the lowest k possible, both the K_{cr} value and the critical frequency value (ω_{cr}), represented in Eq. 60 and in Fig. 2, can be associated. The critical period can then be calculated in the critical frequency equation, as represented in Eq. 61.

$$K_{\text{cr}} = 1.55022;\ \omega_{\text{cr}} = 1.21200, \tag{60}$$

$$P_{\text{cr}} = \frac{2\pi}{1.14} \cong 5.6, \tag{61}$$

The conventional PID was then tuned, replacing the values of Kcr and Pcr in Table 1.

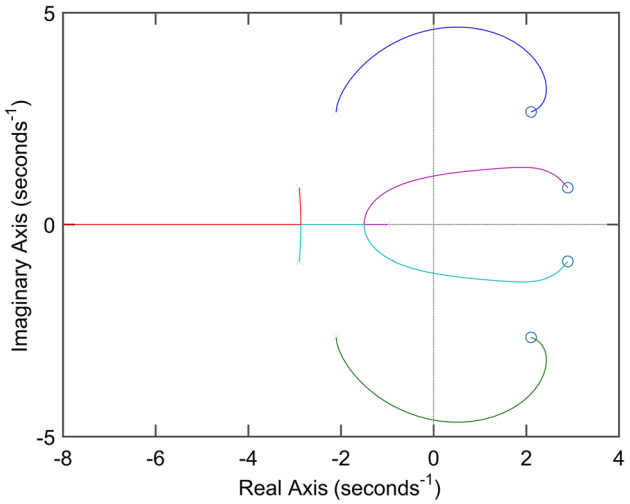

Fig. 2 Geometric Site of Roots of plant C(s) pade8 in sisotool

5.1 Evaluation criteria

The most used criteria were the integral of error (IE), integral of the absolute error (IAE), integral of square error (ISE) and integral of the time multiplied by the absolute error (ITAE), represented respectively in Eqs. 62, 63, 64 and 65.

$$IE = \int_0^\infty (r(t) - y(t))dt, \tag{62}$$

$$IAE = \int_0^\infty |r(t) - y(t)|dt, \tag{63}$$

$$ISE = \int_0^\infty |r(t) - y(t)|^2 dt, \tag{64}$$

$$ITAE = \int_0^\infty t * |r(t) - y(t)|dt, \tag{65}$$

The IE and IAE criteria consider only the error area. ISE, by having the error term squared, maximizes the areas where the error is large, making delays and slowness of response more significant than small steady-state errors. ITAE, by multiplying the error by time, makes steady-state errors more significant than delay or slowness of response. Due to these characteristics of ISE and ITAE, both were chosen as the criteria for the analysis of the methods.

5.2 Robustness test

In order to test whether the controllers tuned by the methods studied are robust, a variation of ± 20% in all parameters of the plant (the gain, the time constant, and the time delay) was made and the parameters of the FOPID controller were kept constant. The plants, after the modification of ± 20% , are presented in Eqs. 66 and 67 respectively.

$$C_{+20\%}(s) = \frac{1.2}{1.2s+1}e^{-2.4s}, \tag{66}$$

$$C_{-20\%}(s) = \frac{0.8}{0.8s+1}e^{-1.6s}, \tag{67}$$

The robustness was evaluated using the ITAE and ISE criteria.

5.3 Disturbance rejection test and controller signal saturation

For the disturbance rejection test a step disturbance is added in the system at t=50s. This value of time was chosen because all systems were already in a steady state at this time. All other parameters were kept the same. To compare the controller methods, the ISE criterion was chosen over the ITAE criterion because the latest is sensitive to long simulation times. To see the contribution of the disturbance separately, the original (without the disturbance) value of ISE was subtracted to the full ISE taken from the signal with disturbance.

$$ISE_{\text{dist}} = ISE_{\text{Total}} - ISE_{\text{Original}} \tag{68}$$

The control signals were also evaluated, presenting a maximal signal of 11.42 for both GA-5 and N-M algorithms. This signal is not a concern since it can represent physical signals without surpassing the saturation of actuators, as in the example by Zhang et al. (2008) of the control of a F8 airship. For the sake of brevity, the resulting signals will not be presented in this work.

5.4 Comparison of methods

Firstly, the reliability, robustness and results obtained for each method were examined.

A comparative study of all the methods was carried out, including the common PID tuning with the method of Ziegler and Nichols. For this comparison, robustness analysis, optimization time of each tool, and ISE and ITAE criteria were included.

6 Results and discussion

First, the parameters Kp, Ti, and Td were calculated by the method of Ziegler and Nichols. The values found are presented in Eq. 69 and the PID tuned in Eq. 70.

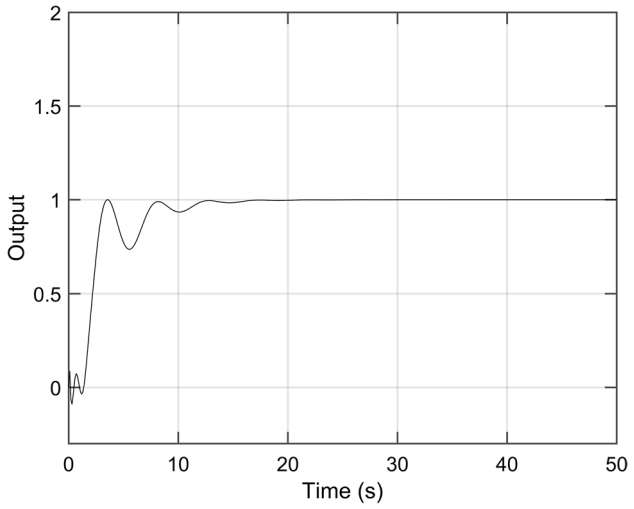

Fig. 3 System response to a step input. Controller tuned with Ziegler and Nichols method

$$P = 0.930132; I = 0.33219; D = 0.6510924, \tag{69}$$

$$G_{\text{c-ZN}}(s) = \left(0.930132 + \frac{0.33219}{s} + 0.6510924 * s\right), \tag{70}$$

Then, the response to the step input (Fig. 3), as well as the result of ISE and ITAE presented in Eqs. 71 and 72 were found. It can be observed that the ITAE and ISE values are not high, evidencing a good tuning. However, the output oscillates quite frequently, which can be disadvantageous in the event of plant disturbances or variations.

$$ISE_{\text{ZN}} = 2.1117, \tag{71}$$

$$ITAE_{\text{ZN}} = 8.7623, \tag{72}$$

6.1 First method

The values of K, T and L, are taken from Eq. 58 and are presented in Eq. 73. With these values, the matrix described in Eq. 22 was applied and the controller parameters calculated, as presented in Eqs. 74 and 75.

$$K = 1;\ T = 1;\ L = 2, \tag{73}$$

$$P = 0.9774;\ I = 0.4235;\ D = 0.4227, \tag{74}$$

$$\alpha = 1.3353;\ \beta = 0.8593, \tag{75}$$

Figure 4 illustrates the response to a step input (gray line) and the related ISE and ITAE values are presented in Eqs. 76 and 77. In Fig. 4, it is possible to observe high overshoot and a considerable delay in the time response, justifying the high value of ITAE. However, the output had a fast rising time and low oscillation, which explains the low ISE.

$$ISE_{\text{KTL}} = 2.5646, \tag{76}$$

$$ITAE_{\text{KTL}} = 31.2117, \tag{77}$$

The step response for the robustness test can also be seen in Fig. 4 and their respective ISE and ITAE with the percentage difference for the original plant, are found in Table 2. It can be observed that this method is stable, fulfilling the first requirement of the control system. In addition, the step responses were similar in rising time and response time. However, due to the increase in the overshoot value for the + 20% plant, ISE and ITAE increased by almost 50%. In addition, the fact that the -20% plant presented significantly lower values than the original plant indicates that the initial optimization process can be improved to provide better results.

6.2 Second method

After applying the values of Kcr and Pcr (found in Eqs. 60 and 61) in the matrix presented in Eq. 34, the controller

Table 2 ISE and ITAE values for the robustness test of the KTL method

	ITAE	ISE	% ITAE	% ISE
Original plant	31.21	2.56	–	–
+20%	44.73	3.73	43.32%	45.70%
−20%	28.29	2.09	−9.46%	−44.00%

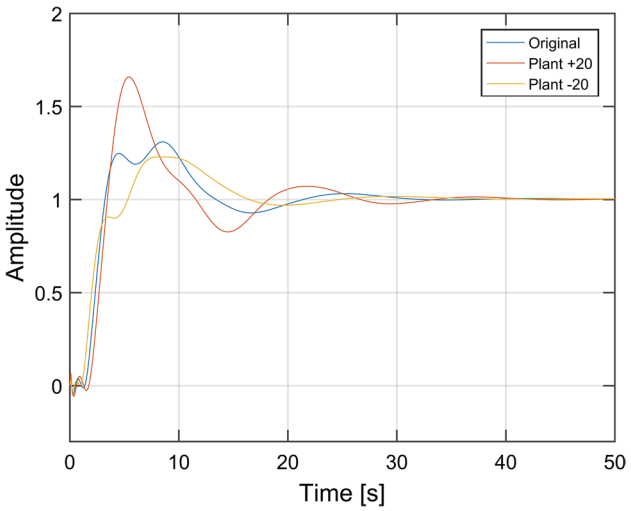

Fig. 4 System response to a step input. Controller tuned with KTL method

Table 3 ISE and ITAE of plants for robustness test of the Kcr method

	ITAE	ISE	% ITAE	% ISE
Original plant	32.57	2.69	–	–
+20%	70.48	4.56	116.40%	69.30%
−20%	25.66	2.07	−21.22%	−22.94%

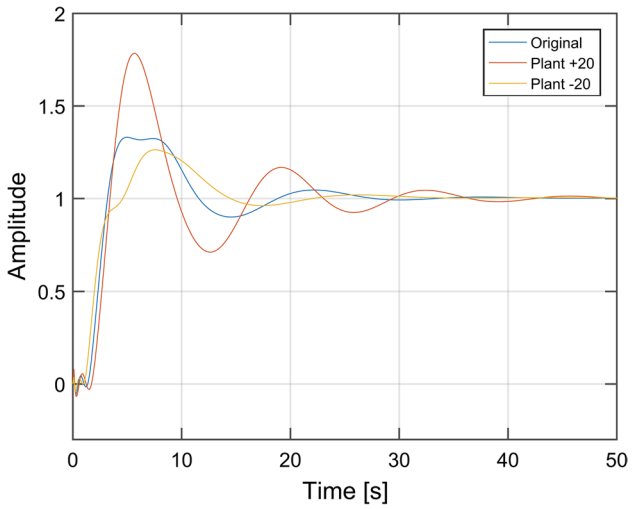

Fig. 5 System response to a step input. Controller tuned with the Kcr method

parameters and the transfer function were obtained. These parameters and equations are presented in Eqs. 78, 79 and 80.

$$P = 1.0050;\ I = 0.4827;\ D = 0.4122, \tag{78}$$

$$\alpha = 1.3279;\ \beta = 0.9707, \tag{79}$$

$$G_{\text{c-Kcr}}(s) = (1.0050 + \frac{0.4827}{s^{1.3279}} + 0.4122 * s^{0.9707}), \tag{80}$$

Figure 5 illustrates the system response to a step input for the Kcr controller (gray line). As in the previous method, there was also a high overshoot and a relative delay in response, so both methods present similarities. This fact was expected since both follow the same specifications. Therefore, the ITAE and ISE values are also similar, as represented in Eqs. 81 and 82.

$$ISE_{\text{Kcr}} = 2.6944, \tag{81}$$

$$ITAE_{\text{Kcr}} = 32.5687, \tag{82}$$

In the robustness test, the values obtained from ITAE and ISE are represented in Table 3 and the step system responses is presented in Fig. 5. As mentioned, the system response

Table 4 ISE and ITAE results for the robustness test. Controller tuned with the DE method

	ITAE	ISE	% ITAE	% ISE
Original plant	15.43	1.99	–	–
+20%	32.18	2.85	108.55%	43.22%
−20%	12.73	1.62	−17.50%	−18.59%

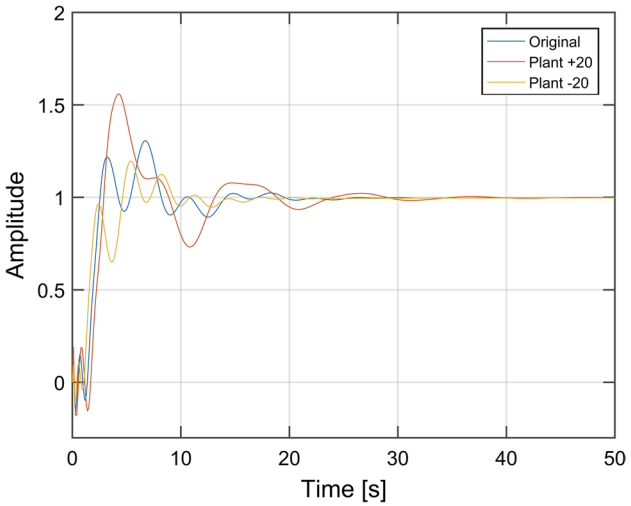

Fig. 6 System response to a step input. Controller tuned with the DE Algorithm

for this method was similar to that of the first method, and therefore, the same observations regarding the overshoot for the plant of + 20% and the reduction of the ITAE and ISE of the plant of −20% . However, in this case, a much greater oscillation can be observed for the plant of + 20%. Such instability can be reinforced by the percentage increase of ITAE, with a value greater than 100%. With this analysis, it can be said that this system presents less robustness to increase the values of the plant when compared to Method 1.

6.3 Third method

By executing the code for the DE algorithm, as in Appendices 2 and 3, the values represented in Eqs. 83 and 84 for the controller parameters were found and, the FOPID transfer function is presented as in Eq. 85.

$$P = 0.6016;\ I = 0.8831;\ D = 1.2743, \tag{83}$$

$$\alpha = 0.9244;\ \beta = 0.9964, \tag{84}$$

$$G_{\text{c-Kcr}}(s) = (0,6016 + \frac{0,8831}{s^{0,9244}} + 1,2743 * s^{0,9964}), \tag{85}$$

Figure 6 illustrates the system response to a step input (gray line). It can be observed that the output presents high oscillation, justifying a moderately high ITAE value, as represented in Eq. 87. At the same time, a low amplitude of oscillation is observed, which in turn, justifies the low ISE, as observed in Eq. 86.

$$ISE_{\mathrm{DE}} = 1.9937, \tag{86}$$

$$ITAE_{\mathrm{DE}} = 15.4282, \tag{87}$$

After the robustness test, Fig. 6 was obtained with the outputs for each of the test plants and for the original plant. The ITAE and ISE values for this test are given in Table 4. It can be noticed that for both variations the plant remained stable, fulfilling the first requirement of robustness. In addition, for the plant modification of −20% the values are consistent with expected values. However, the plant modification of + 20%, besides obtaining a very oscillatory response, presented a large increase of both ITAE and ISE. With this, it can be said that this tuning did not present good results in the test of robustness of + 20% of the plant.

6.4 Fourth method

This section discusses the results for the GA Algorithm method.

6.4.1 Method 4.1

After optimizing the 'J' function with the optimtool tool (Eq. 56 and Appendix 4), the values found for the controller parameters are presented in Eqs. 88, 89 and 90.

$$P = 1.04;\ I = 0.494D = 0.723, \tag{88}$$

$$\alpha = 1.003;\ \beta = 1.229, \tag{89}$$

$$G_{\text{c-GA5}}(s) = 1.04 + \frac{0.494}{s^{1.003}} + 0,723 * s^{1.229}, \tag{90}$$

In Fig. 7 the system response to a step input (gray line) is illustrated. It can be observed very adequate control parameters, presenting low overshoot, low oscillation, fast response, as well as low ISE and ITAE values, represented in Eqs. 91 and 92.

$$ISE_{\mathrm{GA5}} = 1.8886, \tag{91}$$

$$ITAE_{\mathrm{GA5}} = 3.5109, \tag{92}$$

Finally, the robustness test was performed, obtaining the remaining curves for Fig. 7 containing the system responses to the step input, and Table 5, comparing the ITAE and ISE values. From these data, it was observed that the original plant had values already well optimized since the value of ITAE increases even with the decrease of the constants of time and delay. Moreover, the variation values were within the expected range and, even though the ITAE has grown more than 200%, its absolute value is still relatively low.

Table 5 Comparison of ISE and ITAE values for robustness test using GA for five parameters method

	ITAE	ISE	% ITAE	% ISE
Original plant	3.51	1.89	–	–
+20%	11.44	2.51	225.92%	32.80%
−20%	5.72	1.68	62.96%	−11.11%

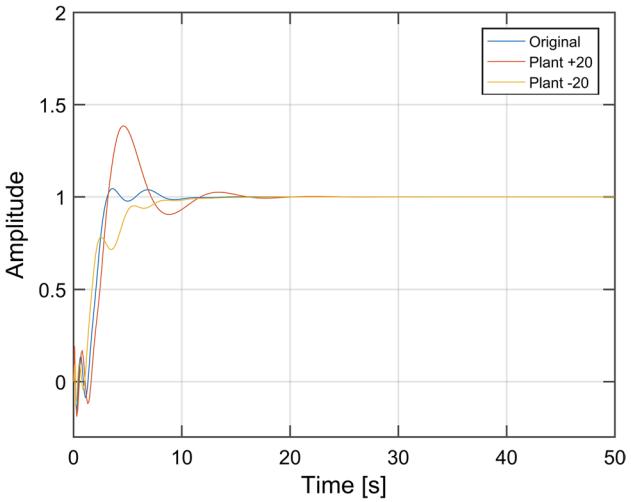

Fig. 7 System response to a step input. Controller tuned with the GA-5 Algorithm

6.4.2 Method 4.2

After optimizing the new function with the MATLAB optimtool tool, the values found for the controller parameters are presented in Eqs. 93 and 94, and the values for Eq. 95 from the Ziegler and Nichols tuning are kept constant.

$$P = 0.9119;\ I = 0.3309;\ D = 0.6282, \tag{93}$$

$$\alpha = 1.06;\ \beta = 1.29, \tag{94}$$

$$G_{\text{c-GA2}}(s) = 0.9119 + \frac{0.3309}{s^{1.06}} + 0.6282 * s^{1.293}, \tag{95}$$

The ISE and ITAE values were calculated and the system response to the step input is presented in Fig. 8 (gray line). It can be seen that the parameters are also quite adequate. However, there is a relatively higher response delay compared to previous GA results, which explains the relatively

Table 6 ISE and ITAE of plants for GA robustness test for two parameters

	ITAE	ISE	% ITAE	% ISE
Original plant	12.26	2.04	–	–
+20%	11.62	2.33	−5.38%	14.21%
−20%	17.52	1.99	42.87%	−2.45%

higher ISE and ITAE, represented in Eqs. 96 and 97, respectively. It is important to note that the ISE value was lower in this method than in the Ziegler and Nichols method, highlighting the effectiveness of the fractional exponents.

$$ISE_{\mathrm{GA2}} = 2.0414, \tag{96}$$

$$ITAE_{\mathrm{GA2}} = 12.2562, \tag{97}$$

In the test of robustness performed in this variation of the GA algorithm, the values of Table 6 and the responses to the step response are illustrated in Fig. 8. It can be observed that this method presented less than 50% variation in all the criteria, as well as responses of similar aspects in the graph. The ISE evaluation remained almost constant, even with the variation of + 20% or -20% , and considering that this is the criterion minimized by the algorithm, it can be inferred that the robustness to the variation of the parameters of the plant was satisfactory.

6.5 Fifth method

Using FOMCOM libraries optimization tool (Nelder–Mead algorithm) the new controller parameters and transfer function are found and represented in Eqs. 98, 99 and 100.

$$P = 0.74009;\ I = 0.39926;\ D = 0.39784, \tag{98}$$

$$\alpha = 1.0011;\ \beta = 1.0746, \tag{99}$$

$$G_{\text{c-NM}}(s) = 0.74009 + \frac{0.39926}{s^{1.0011}} + 0.39784 * s^{1.0746}, \tag{100}$$

As for the other methods, the ISE and ITAE values were calculated and the graph of the response to the input step of the system was plotted in Fig. 9 (gray line). It can be observed that the parameters of this controller were very adequate, with relatively low ISE and ITAE values, as represented in Eqs. 101 and 102. In addition, the graph presented a fast response, with very low overshoot and oscillation.

$$ISE_{\mathrm{NM}} = 2.1802, \tag{101}$$

$$ITAE_{\mathrm{NM}} = 3.7237, \tag{102}$$

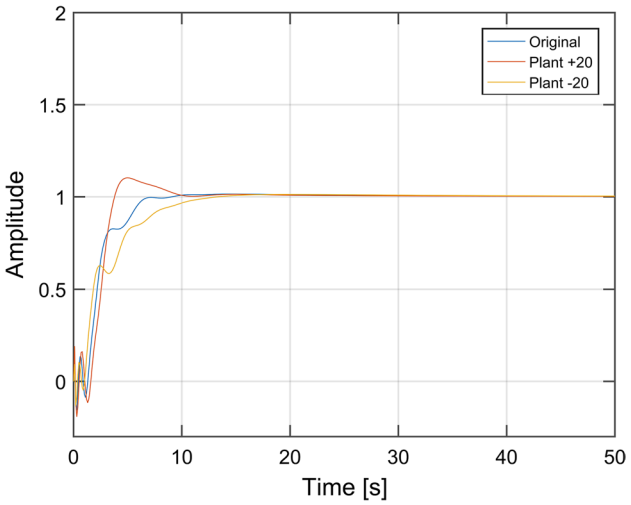

Fig. 8 System response to a step input. Controller tuned with the GA-2 Algorithm

Table 7 Comparison of ISE and ITAE values for robustness test using N-M algorithm

	ITAE	ISE	% ITAE	% ISE
Original plant	3.72	2.18	–	–
+20%	12.50	2.72	235.86%	24.81%
−20%	8.30	2.04	122.81%	−6.53%

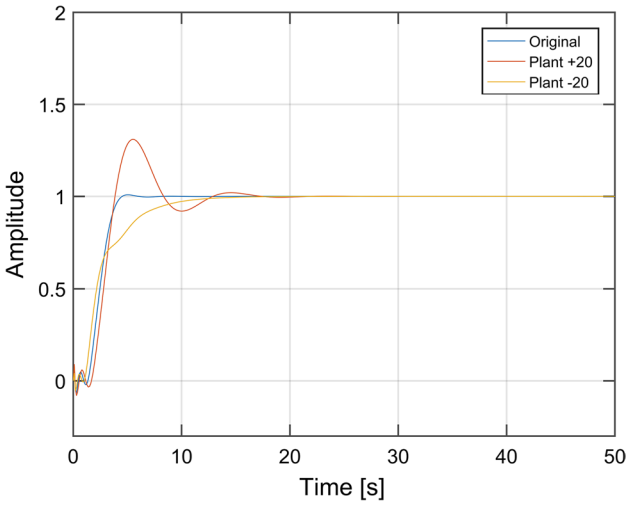

Fig. 9 System response to a step input. Controller tuned with the Nelder–Mead Algorithm

The results of the robustness test of this method are represented in Table 7 and in Fig. 9. As in Method 4.2 it is observed that the ITAE value grows even with the reduction of time and delay constants, indicating that the original

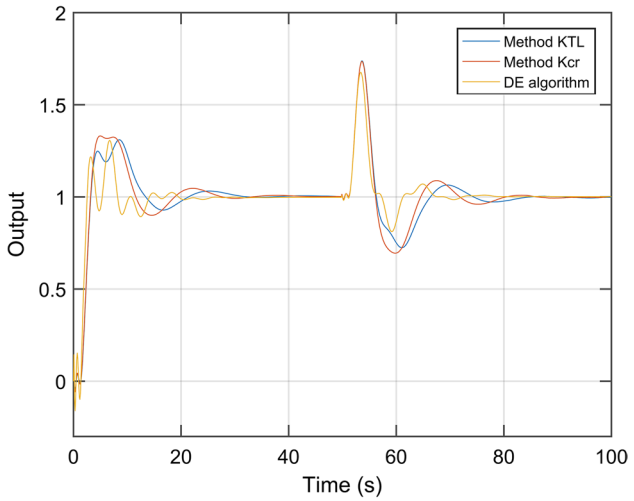

Fig. 10 System response to a step input and a disturbance at 50s. Controller tuned with KTL, Kcr, and DE algorithm methods

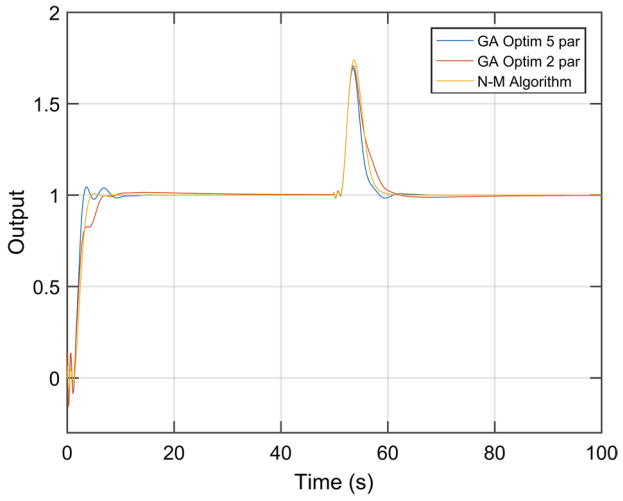

Fig. 11 System response to a step input and a disturbance at 50s. Controller tuned with the GA Optim 5 par, GA Optim 2 par, and N-M Algorithm methods

plant is well-tuned. Moreover, even though the ITAE value also increased by more than 200% , the absolute value still remained relatively low.

The disturbance analysis are shown in Figs. 13 and 14 for all methods combined, and in the next section, the values for the ISE will be shown and compared.

7 Comparative analysis of methods

Fractional controller gains can be more easily determined using analytical methods. However, the results obtained were worse and more sensitive to variations in plant coefficients when compared to heuristic methods. With the heuristic methods, better results could be obtained, especially using the method "GA optimizing the five parameters of the controller". On the other hand, the computational implementation of the methods is more complicated and requires more time to obtain the solution.

In Figs. 10 and 11 are presented the system response to a unit step disturbance applied in t=50s. Figure 10 contains the system response for the methods Kcr, KTL, and DE algorithm, and Fig. 11 for the methods GA-2, GA-5, and N-M algorithm. The values of the ISE criterion chosen are shown in Table 8. The DE algorithm was the most effective for disturbance rejection but also the GA-5 presented a satisfying performance.

For the comparative analysis, the values for the comparison criteria found for each method are presented in Table 8 and the responses to the step of these methods in Fig. 12. It is possible to see, analyzing the table, a certain advantage in using heuristic methods for the FOPID tuning when considering the ITAE criterion.

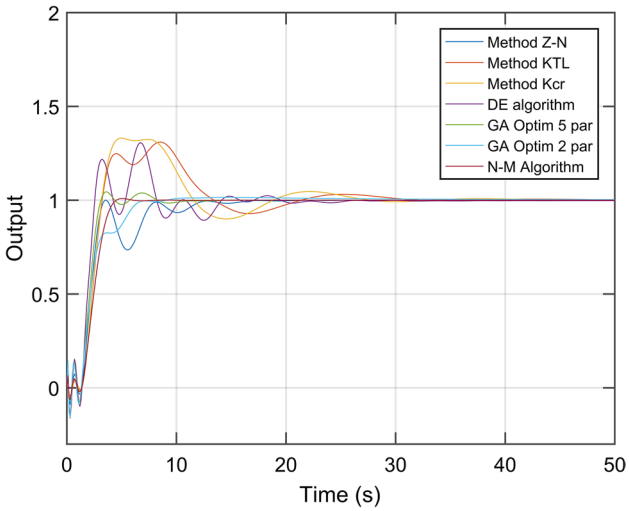

Fig. 12 System response to a step input. All the methods employed

Table 8 Comparative analysis for tuning methods

Method	ITAE	ISE	ISE_{dist}	Run time
Ziegler- Nichols	8.7623	2.1117	1.2094	–
KTL method	31.2117	2.5646	1.5052	–
Kcr Method	32.5687	2.6944	1.5795	–
DE algorithm	15.4282	1.9937	0.8803	06:30
GA-5 algorithm	3.5109	1.8886	0.9762	01:27
GA-2 algorithm	12.2562	2.0414	1.2345	00:43
N-M algorithm	3.7237	2.1802	1.3052	02:12

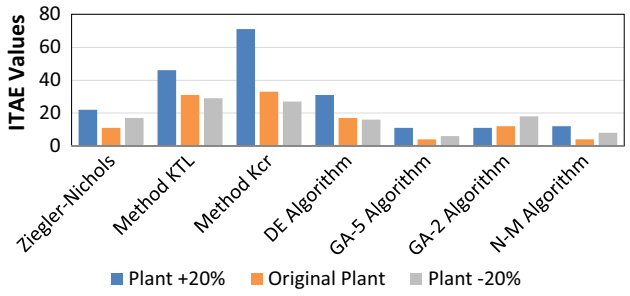

Fig. 13 Graph in ITAE's vertical bars of the robustness test

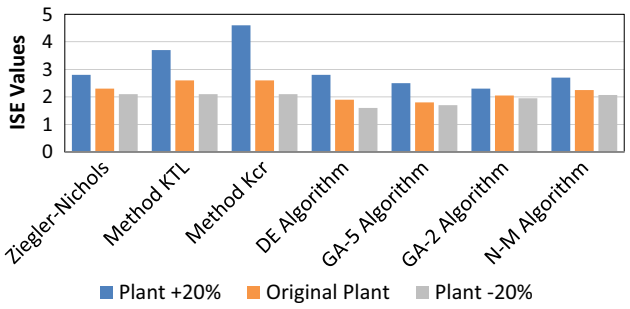

Fig. 14 Graph in vertical bars of ISE's of robustness test

The methods that stood out for this criterion are the GA for five parameters and the Nelder–Mead algorithm, GA presenting a slight advantage. In the ISE criterion, the GA algorithm for five parameters was also slightly better. Considering all parameters presented in Table 8, the GA-5 method presented the best performance.

It was possible to observe that the values of the fractional exponents coincidentally did not differ significantly from 1. Because of this, the use of the common PID can present results close to those of the FOPID for this plant specifically, justifying the fact that the Ziegler and Nichols tuning generated better results than some of the FOPID tunings.

Concerning the execution times of the algorithms, it is evident that the GA algorithm is the one that optimizes the functions more quickly, and, in contrast, the algorithm DE is the one that presents more slowness. These times may be more significant when the function of the plant is more complex or with greater orders. Therefore, it can be said that, in this requirement, the GA algorithm also had an advantage over the other methods.

In Figs. 13 and 14 are represented the comparison of the robustness tests for the ITAE and ISE criteria, respectively. Since the first important requisite in the analysis of robustness is stability, it can be seen in the graphs that all the plants have met this requirement.

The graph of the ITAE (Fig. 13) illustrates a good indication of the robustness of the systems, considering that it did not change much in this criterion after the modification of the plant. With this in mind, it can be concluded that the KTL, GA for five and two parameters , and the Nelder–Mead algorithm presented low changes, demonstrating good robustness to the variation of the plant.

It can be seen that the ISE analysis (Fig. 14) presents little efficiency for the robustness evaluation, since in all the methods the same pattern appeared (higher ISE for the plant of + 20% and lower for the plant of -20 %). This was an expected behaviour, considering that the variation in the plant parameters varies linearly with ISE.

It is important to note that when comparing the tuning rules with the genetic algorithm and the simplex method, the former requires less information for tuning the desired system. Moreover, it is computationally simpler than the others. On the other hand, the tuning rules techniques were surpassed by the latest in terms of performance. Thus, because it is simpler, computationally cheaper, and more practical, the first tuning technique was used as a reference for judging the heuristic results.

8 Conclusions

In this work, five different tuning techniques for a fractional-order PID were compared. The controllers were employed to regulate the output of a first-order plus dead-time plant, and its effectiveness was evaluated regarding robustness, disturbance rejection, ITAE, and ISE. Details of the algorithm used for the methods are presented in the annex, making the use of the methods considered in this work more accessible.

The results presented demonstrated that heuristic methods tend to tune the controller better. The analytical methods presented (methods 1 and 2) are simpler and faster to apply, and influenced the choice of the value ranges used for the variables of the heuristic methods. If there is no first reference result, the notion about this range would not be appropriate.

Another important point observed in the case of heuristic methods is that it is desirable to minimize the evaluation criteria, as seen after the first attempts to use method 3 (DE algorithm). Minimizing the evaluation criteria ensured stability and good system response, even when there was no knowledge of the system specifications.

In general, the comparative study indicated that the use of the GA method to tune all five parameters of the controller (Method 4.1) may be the best option for FOPID tuning. However, the method using the Nelder–Mead algorithm (Method 5) also presented adequate values and response curves with good combination of overshoot and stabilization time. In addition, the application tool of this method allows great flexibility regarding changes in the system requirements. Therefore, when it is intended to tune the fractional

controller, it may be said that it is advisable to apply one of these two methods, considering that it is advantageous to apply Methods 1 or 2 to know the approximate range of values to be used as constraints.

The fractional controllers can be more efficient in controlling a system than the common controllers since it is possible to improve an evaluation criterion by the variation of the Differintegral exponents. However, the relations employed for the analytical methods and the programming and computational requirements for the heuristic methods are more complex compared to the techniques commonly employed in integers controllers for the same plant considered in this work. In this case, the relations available for integer PID controllers can be more practical and simpler.

Appendix 1

```
%To run this matlab script you need to fill the empty [] with the desired values
clear all
final=[20]; %Simulation time (in seconds)
dt=.1; %time increment
t=0:dt:final; % Create the ?time? vector
n=size(t);
degrau=ones(n); %create a step at 0 seconds
%code to create the disturbance
for i=1:(final/dt)+1
    if(i≥500) %define when the disturbance step will happens in unit of (1s/10)
        disturbance(i)=1;
    else
        disturbance(i)=0;
    end
end
%Plant Definitions
k=[1]; %Plant gain
tal=[1]; %Plant Time Constant
teta=[2]; %Plant Delay
G=tf(k,[tal 1]); %Plant transfer function without delay
Greal=tf(k,[tal 1],  inputdelay ,teta); % Delayed Plant Transfer Function
Gs=pade(Greal,4); %Pade approximation
Gsfo=fotf([1 11 55 150 210 105],[5 4 3 2 1 0],[1 −10 45 −105 105],[4 3 2 1 0]); %plade aproximation rewriten in FONCOM language
%Controller definitions
P=[0.91188]; %Controller Proportional Gain
I=[0.330895]; % Integral gain of the controller
Alfa=[1]; %Integral Controller Expo
D=[0.628185]; % Derivative Gain of Controller
Beta=[1]; % Exponent Derivative Controller
b=[D P I]; % Create vector b
nb=[Alfa+Beta Alfa 0]; % Create the vector nb
a=[1]; %Create the vector a
na=[Alfa]; % Create the vector an
Gc=fotf(a,na,b,nb,0); % Creates the controller transfer function with FOMCON
S=Gc/(1+Gc*Gs); %control loop
C=Gc*Gs/(1+Gc*Gs); %system loop
W=Gsfo/(1+Gc*Gsfo); %disturbance loop
F1= lsim(Gc,degrau,t); %control signal before disturbance
F = F1. +disturbance; %sum of disturbanc to the control signal
figure(1);lsim(S,degrau,t) %Plot control signal
figure(2);lsim(C,degrau,t) %Plot system signal without disturbance
figure(3);lsim(W,F,t) %Plot system signal with disturbance
%Calculation of comparison criteria
Cise=lsim(C,degrau,t); %system signal without disturbance
Fise=lsim(W,F,t); %system signal with disturbance
ISEdiferenca = 0.1*(dot(degrau. −Fise,degrau. −Fise) − dot(degrau. −Cise,degrau. −Cise)) %relative disturbance ISE
```

Appendix 2

```
%To run this script you first need to run Appendix A
D=5; %  Number of Variables
objf=inline( abs((x1+(x2/(-0.4+0.5457i)^x3)+ ...
    x4*(-0.4+0.5457i)^x5)*(1/(0.6+0.5457i))+1)^2 ...
    +angle((x1+(x2/(-0.4+0.5457i)^x3)+ ...
    x4*(-0.4+0.5457i)^x5)*(1/(0.6+0.5457i))+1)^2  ...
    , x1 , x2 , x3 , x4 , x5 );objf = vectorize (objf); %  ...
    Function to be minimized vectorized
objf=vectorize(objf); %  Function to be minimized vectorized ...
    vectorized
N = 50; %  population
itmax = 200; %  number of maximum iterations
CR = 0.5; %  Frequency of cross-over
a(1:N,1)=[0] ; b(1:N,1)=[50]; %  Limits of the variable P (x1)
a(1:N,2)=[0] ; b(1:N,2)=[25]; %  Limits of the variable I (x2)
a(1:N,3)=[0] ; b(1:N,3)=[2]; %  Limits of the alpha variable (x3)
a(1:N,4)=[0] ; b(1:N,4)=[25]; %  Limits of the variable D (x4)
a(1:N,5)=[0] ; b(1:N,5)=[2]; %  Limits of the beta variable (x5)
d=(b-a); %  Limits of the problem
basemat=repmat(int16(linspace(1,N,N)),N,1);
basej=repmat(int16(linspace(1,D,D)),N,1);  %  Used for ...
    parameters j
x=a+d.*rand(N,D); %  Random initialization of positions from ...
    mutation type 1
fx=objf(x(:,1),x(:,2),x(:,3),x(:,4),x(:,5)); %$  Evaluation ...
    objective for all parameters
[fxbest, ixbest] = min (fx); %  Selecting the best
xbest = x (ixbest, 1: D);
for it = 1: itmax;  %  Initiation of iterations
    permat=bsxfun(@(x,y) ...
        x(randperm(y(1))),basemat ,N(ones(N,1))) ;
    F=0.4+rand(1,1)*0.6;  %  Variable frequency of mutation
    v(1:N,1:D)=abs(repmat(xbest,N,1)+F * (x(permat(1:N,1), ...
        1:D)-x(permat(1:N,2), 1:D)));  %  Donor vector ...
        generator for mutation
    r = repmat (randi ([1 D], N, 1), 1, D);  %$  Recombination
    muv = ((rand(N,D)<CR) + (basej==r)) ≠ 0;  %$  Crossing
    mux = 1-muv;  %$  Crossing
    u(1:N,1:D)=x(1:N,1:D).*mux(1:N,1:D)+ ...
        v(1:N,1:D).*muv(1:N,1:D);  %$  Creation of vector u
    fu = objf (u (:, 1), u (:, 2), u (:, 3), u (:, 4), u (:, ...
        5)); %  Function for selection
    idx = fu <fx;  %  Selection
    fx (idx) = fu (idx); %  Selection
    x (idx, 1: D) = u (idx, 1: D); %  Selection
    [fxbest, ixbest] = min (fx); %  Choosing the Best
    xbest = x (ixbest, 1: D); %  Choosing the Best
end %  End of iterations
[xbest fxbest]; %  Reveals the best values for the variables ...
    and for the objective function.
```

Appendix 3

```
%To run this matlab script you need to fill the empty [] with the desired values

clear all %  Initialization
clc %  Initialization

final=[]; %Simulation time (in seconds)
dt=.1; %time increment
t=0:dt:final; % Create the ?time? vector
%Plant Definitions
k=[]; %Plant gain
tal=[]; %Plant Time Constant
teta=[]; %Plant Delay
G=tf(k,[tal 1]); %Plant transfer function without delay
Greal=tf(k,[tal 1],  inputdelay ,teta); % Delayed Plant Transfer Function
Gs=pade(Greal,4); %Pade approximation

iteramax = 1000;  %  Maximum number of times to execute the DE algorithm
VALORINI = 50;  %  Random value to start the algorithm, well above the expected value of "ITAE"
ITAEBEST = VALORINI;  %  Initial value of ITAE
for itera = 1: iteramax %  Beginning of the runs of the DE algorithm
    D = 5;  %$  Number of Variables
    objf=inline( abs((x1+(x2/(-0.4 + 0.5457i)^x3)+x4* (-0.4+0.5457i)^x5)* (1/(0.6+0.5457i))+1)^2 + angle((x1+(x2/(-0.4+0.5457i)^x3)+x4* (-0.4+0.5457i)^x5)*(1/(0.6+0.5457i))+1)^2 , x1 , x2 , x3 , x4 , x5 );%  Function to be minimized
    objf = vectorize (objf);  %  Function to be minimized vectorized
    %  starting the DE parameters
    N = 100;  %  Population
    itmax = 100;  %  Maximum number of iterations
    CR = 0.6;  % Cross-Over Frequency
    %  F = 0.5;  % Fixed mutation frequency
    % Limits of the problem
    a(1: N, 1) = [0]; b (1: N, 1) = [50];  %  Limits of the variable P (x1)
    a(1: N, 2) = [0]; b (1: N, 2) = [25];  %  Limits of variable I (x2)
    a(1: N, 3) = [0]; b (1: N, 3) = [2];  %  Limits of the alpha variable (x3)
    a(1: N, 4) = [0]; b (1: N, 4) = [25];  %  Limits of variable D (x4)
    a(1: N, 5) = [0]; b (1: N, 5) = [2];  %  Limits of the beta variable (x5)
    d = (b-a);
    basemat=repmat(int16(linspace(1,N,N)),N,1);
    basej = repmat (int16 (linspace (1, D, D)), N, 1);  %$  Used for parameters j
    x=a+d.*rand(N,D);  %$  Random initialization of positions from mutation type 1
    fx=objf(x(:,1),x(:,2),x(:,3),x(:,4),x(:,5)); %$  Evaluation objective for all parameters
    [fxbest, ixbest] = min (fx);  %$  Selecting the best
    xbest = x (ixbest, 1: D);
    for it = 1: itmax;  %  Initiation of iterations
        permat=bsxfun(@(x,y) x(randperm(y(1))),basemat ,N(ones(N,1))) ;
        F=0.4+rand(1,1)*0.6;  %  Variable frequency of mutation
        v(1:N,1:D)=abs(repmat(xbest,N,1)+F * (x(permat(1:N,1), 1:D)-x(permat(1:N,2), 1:D)));  %  Donor vector generator for mutation
        r = repmat (randi ([1 D], N, 1), 1, D);  %$  Recombination
        muv = ((rand(N,D)<CR) + (basej==r)) ≠ 0;  %$  Crossing
```

```
48          mux = 1-muv;  %$  Crossing
49          u(1:N,1:D)=x(1:N,1:D).*mux(1:N,1:D)+ ...
                v(1:N,1:D).*muv(1:N,1:D);  %$  Creation of vector u
50          fu=objf(u(:,1),u(:,2),u(:,3),u(:,4),u(:,5));  %$  ...
                Function for selection
51          idx = fu <fx;  %$  Selection
52          fx(idx)=fu(idx);  %$  Selection
53          x (idx, 1: D) = u (idx, 1: D);  %$  Selection
54          [fxbest, ixbest] = min (fx);  %$  Choosing the Best
55          xbest = x (ixbest, 1: D);  %$  Choosing the best vector
56      end %  End of iterations
57      [xbest fxbest];  %  best values for the variables and for ...
            the function
58      % Calculation of the values to find the ITAE
59      kp4=xbest(1);
60      ki4=xbest(2);
61      alpha4=xbest(3);
62      kd4=xbest(4);
63      beta4=xbest(5);
64      b4=[kd4 kp4 ki4];
65      nb4=[alpha4+beta4 alpha4 0];
66      a4=[1];
67      na4=[alpha4];
68      Gc4=fotf(a4,na4,b4,nb4);
69      Ft4=Gc4*Gs;
70      h4=feedback(Ft4,1);
71      output4 = step(h4, t);
72      ITAEDE=t*abs(1-output4)*0.1;  %$  ITAE calculated with the ...
            best values
73      if  (ITAEDE <ITAEBEST) %$  Comparison of ITAE s
74          ITAEBEST = ITAEDE %$  Definition of the new ITAE
75          xbesto = xbest %$  Definition of the new parameters
76      end
77  end
78  [ITAEBEST, xbesto] %$  Best values found
```

Appendix 4

```
%To run this script you first need to run "optimtool" from ...
    "Global optimization toolbox"
%Select ga—Genetic Algorithm and define number of variables as ...
    the number of unknwon variables to be optimized, in this ...
    case 5
%Fill the bounds with [0 0 0 0 0] and [100 50 2 50 2]
%Set options as "Default" or "Constraint dependent"
%In Fitness function add the name of the function. Example ...
    "AppendixD"

function J = funcao(x)
    t=0:0.1:50;   %Create the Vector Time
    v = zeros(length(t), 1);
    v(:)=0.1; %Create vector t  for calculation of evaluation ...
        criteria
    k=1;   %Gain of plant
    tal=1;   %Plant time constant
    theta=2;   %Delay of plant
    Greal=tf(k,[tal 1], inputdelay ,theta); %Plant Transfer ...
        Function
    Gs=pade(Greal,4);   %Pade approximation
    b=[x(4) x(1) x(2)];   %Create the  b
    nb=[x(5)+x(3) x(3) 0];   %Create the  nb
    a=1;   %Create the  a
    na=x(3);   %Create the  na
    Gc=fotf(a,na,b,nb);   %Controller transfer function
    Ft=Gc*Gs;   %System transfer function (plant + controller)
    h=feedback(Ft, 1);   %Closed loop system
    output=step(h, t);   %Applies a step input to the system ...
        in closed loop
    J=0.001*dot((1—output), ...
        (1—output))*0.1+0.999*dot(v,abs(1—output))   %J function
```

Acknowledgements We would like to express gratitude to FOMCON for their assistance and for the use of their software. Acknowledgement is given to Universidade Federal de Minas Gerais, Universidade Federal de Uberlandia, and for Universidade Estadual Paulista for providing the necessary resources used for the simulations presented. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. We thank the reviewers whose critical reading helped improve and clarify this manuscript.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Al-Saggaf UM, Mehedi IM, Mansouri R, Bettayeb M (2017) Rotary flexible joint control by fractional order controllers. Int J Control Automation Syst 15(6):2561–2569
2. Ardia D, Boudt K, Carl P, Mullen K, Peterson BG (2011) Differential evolution with deoptim: an application to non-convex portfolio optimization. R J 3(1):27–34
3. Baleanu D, Mendes Lopes A (eds) (2019) Handbook of fractional calculus with applications. Walter de Gruyter GmbH, Berlin/Boston
4. Birs I, Nascu I, Ionescu C, Muresan C (2020) Event-based fractional order control J Adv Res
5. Cao JY, Liang J, Cao BG (2005) Optimization of fractional order pid controllers based on genetic algorithms In: 2005 international conference on machine learning and cybernetics, IEEE, vol 9, pp 5686–5689
6. Chekari T, Mansouri R, Bettayeb M (2018) Imc-pid fractional order filter multi-loop controller design for multivariable systems based on two degrees of freedom control scheme. Int J Control Automation Syst 16(2):689–701
7. Ganbavale M (2014) Differential evolution algorithm for structural optimization using matlab. https://doi.org/10.13140/2.1.3239.5843
8. Homaeinezhad MR, Shahhosseini A (2020) Fractional order actuation systems: theoretical foundation and application in feedback control of mechanical systems. Applied Mathematical Modelling 87:625–639
9. Jahanshahi H, Yousefpour A, Munoz-Pacheco JM, Moroz I, Wei Z, Castillo O (2020) A new multi-stable fractional-order four-dimensional system with self-excited and hidden chaotic attractors: Dynamic analysis and adaptive synchronization using a novel

fuzzy adaptive sliding mode control method. Appl Soft Comput 87:105943
10. Jarrah B (2020) Fractional order and inverse problem solutions for plate temperature control PhD thesis, Université d'Ottawa/ University of Ottawa
11. Kochubei A, Luchko Y, Tarasov VE, Petráš I (eds) (2019) Handbook of fractional calculus with applications (Vol. 1). Walter de Gruyter GmbH, Berlin/Boston
12. Lachhab N, Svaricek F, Wobbe F, Rabba H (2013) Fractional order pid controller (fopid)-toolbox In: 2013 European control conference (ECC), IEEE, pp 3694–3699
13. Machado JT, Kiryakova V, Mainardi F (2011) Recent history of fractional calculus. Commun Nonlinear Sci Numer Simul 16(3):1140–1153
14. Monje CA, Chen Y, Vinagre BM, Xue D, Feliu-Batlle V (2010) Fractional-order systems and controls: fundamentals and applications. Springer, New York
15. Nasirpour N, Balochian S (2017) Optimal design of fractional-order pid controllers for multi-input multi-output (variable air volume) air-conditioning system using particle swarm optimization. Intell Build Int 9(2):107–119
16. Nelder JA, Mead R (1965) A simplex method for function minimization. Computer J 7(4):308–313
17. Nguyen SD, Lam BD, Ngo VH (2020) Fractional-order sliding-mode controller for semi-active vehicle MRD suspensions. Springer. Nonlinear Dyn 101(2):795–821
18. Padhee S, Gautam A, Singh Y, Kaur G (2011) A novel evolutionary tuning method for fractional order pid controller. Int J Soft Comput Eng 1(3):1–9
19. Podlubny I, Dorcak L, Kostial I (1997) On fractional derivatives, fractional-order dynamic systems and pi/sup/spl lambda//d/sup/spl mu//-controllers In: Proceedings of the 36th IEEE Conference on Decision and Control, IEEE, vol 5, pp 4985–4990
20. Saini P, Sharma C (2019) Comparative analysis of controller tuning techniques for dead time processes
21. Salehinejad H, Rahnamayan S, Tizhoosh HR (2017) Micro-differential evolution: diversity enhancement and a comparative study. Appl Soft Comput 52:812–833
22. Shah P, Agashe S (2016) Review of fractional pid controller. Mechatronics 38:29–41
23. Shahri ME, Balochian S, Balochian H, Zhang Y (2014) Design of fractional-order pid controllers for time delay systems using differential evolution algorithm. Indian J Sci Technol 7(9):1307–1315
24. Tepljakov A, Petlenkov E, Belikov J, Finajev J (2013) Fractional-order controller design and digital implementation using fomcon toolbox for matlab In: 2013 IEEE conference on computer aided control system design (CACSD), IEEE, pp 340–345
25. Treesatayapun C, Jonathan Munoz-Vazquez A (2020) Discrete-time fractional-order control based on data-driven equivalent model. Elsevier Appl Soft Comput 96:106633
26. Valério D, Da Costa JS (2006) Tuning-rules for fractional pid controllers In: Proceedings of the Second IFAC Symposium on Fract Differ Appl (FDA06), IFAC
27. Valério D, Da Costa JS (2013) An introduction to fractional control, Vol. 91, p 357, Institution of Engineering and Technology
28. Ziegler JG, Nichols NB (1942) Optimum settings for automatic controllers. J Dyn Syst Meas Control-Trans, ASME 64(11)