SHORT COMMUNICATION

Tuning of optimal fractional-order PID controller using an artificial bee colony algorithm

Ameya Anil Kesarkar and N. Selvaganesan∗

Department of Avionics, Indian Institute of Space Science and Technology (IIST), Thiruvananthapuram 695547, India

(Received 20 November 2013; accepted 10 November 2014)

In this paper, the design of fractional-order PID controller is considered in order to minimize certain performance indices such as integral absolute error, integral square error and integral time absolute error. The design-construction leads to a high-dimensional, multi-modal, complex optimization problem which is difficult to solve analytically. We show that it can be solved heuristically using an artificial bee colony (ABC) algorithm, which is a recently emerged ‘stochastic’ technique inspired from the intelligent foraging behavior of honey bee swarm. For the numerical examples under consideration, we further compare the performance of ABC with a ‘deterministic’ Nelder–Mead simplex algorithm.

Keywords: fractional-order controller; $PI^\lambda D^\mu$; artificial bee colony (ABC); Nelder–Mead simplex algorithm

1. Introduction

Fractional-order calculus (Oldham & Spanier, 1974) is a more than 300 years old branch of mathematics which generalizes the conventional integer-order calculus to arbitrary orders. The application of fractional-order calculus in control engineering is a recent area of interest, popularly known as fractional-order control (FOC) (Chen, Petras, & Xue, 2009; Xue & Chen, 2007, chap. 8). It is a powerful tool for designing high-performance robust controllers with a less number of tuning parameters (Monje et al., 2004; Podlubny, 1999; Valerio, 2005).

Several attempts are found in the literature for the tuning of fractional-order controllers for a specific class of plants (Merrikh-Bayat, 2012; Valerio & Costa, 2006). Frequency domain-based tuning of $PD^\mu$ and $[PD]^\mu$ controllers for position servo plants is proposed in Valerio and Costa (2006), Li and Chen (2008), Li, Luo, and Chen (2010) and Luo and Chen (2009). Similar works for $PI^\lambda$ and $[PI]^\lambda$ controllers in the context of first-order plus dead time (FOPDT) plants (Wang, Luo, & Chen, 2009) and velocity servo plants (Wang, Luo, & Chen, 2009) are also proposed in the literature. An attempt to generalize this approach is presented in Kesarkar and Selvaganesan (2011) and Kesarkar and Selvaganesan (2011).

Tuning of fractional-order PID controller ($PI^\lambda D^\mu$) to meet certain performance indices such as integral square error (ISE), integral absolute error (IAE) and integral time absolute error (ITAE) are also gaining interest among the researchers working in the area of FOC. This becomes an unconstrained, high-dimensional and multimodal optimization problem, which is complex to solve analytically.

There are several heuristic algorithms in the literature for solving such complex optimization problems. These algorithms are mainly classified into two broad categories, stochastic and deterministic (Karaboga & Basturk, 2007). In deterministic methods, the given initial feasible guess proceeds towards the final solution based on certain deterministic guidelines. In stochastic methods, a search is made for the better solution in a probabilistic manner.

Stochastic algorithms such as genetic algorithm (GA) (Buanovi, Lazarevi, & Batalov, 2014; Cao, Liang, & Cao, 2005), particle swarm optimization (PSO) (Biplab, Koley, & Datta, 2014; Cao & Cao, 2006), adaptive GA (Chang & Chen, 2009), improved electromagnetic-like algorithm with GA (IEMGA) (Lee & Chang, 2010) and differential harmony search algorithm (Roy, Chakraborty, & Das, 2010) have been used to tune fractional-order controllers. Among deterministic methods, the simplex method of Nelder and Mead (1965) is the most popular one used for tuning the fractional $PI^\lambda D^\mu$ controller (Monje et al., 2004; Valerio & Costa, 2006).

The artificial bee colony (ABC) algorithm (Karaboga, 2005; Karaboga & Akay, 2009; Karaboga & Basturk, 2007, 2008) is a newly emerged stochastic technique which is gaining popularity among the researchers due to its efficiency. In Karaboga and Akay (2009), several multidimensional optimization problems are presented to prove the potential superiority of the ABC algorithm over existing population-based algorithms such as GA, PSO, evolution strategy and differential evolution algorithm.

The usefulness of ABC algorithm for tuning the integer-order PID controller has been attempted in Karaboga and Akay (2010). This motivates us to explore...

© 2015 The Author(s). Published by Taylor & Francis. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The moral rights of the named author(s) have been asserted.

∗Corresponding author. Email: n_selvag@iist.ac.in
the ABC algorithm is an optimization tool based on the intelligent foraging behavior of honey bee swarm, proposed by Karaboga (2005).

The paper is organized as follows: Section 2 presents the preliminaries of ABC as well as the Nelder–Mead simplex algorithm. Section 3 presents the basics of fractional-order calculus and FOC. In Section 4, the PI$^\lambda$D$^\mu$ controller design problem is formulated for minimizing IAE, ISE and ITAE. Section 5 discusses numerical examples which illustrate the application of ABC algorithm for such designs. The results are also compared with those obtained using the Nelder–Mead algorithm. Finally, Section 6 presents the concluding remarks.

## 2. Preliminaries of ABC and Nelder–Mead simplex algorithms

### 2.1. ABC algorithm

In computer science and operations research, the ABC algorithm is an optimization tool based on the intelligent foraging behavior of honey bee swarm, proposed by Karaboga (2005).

The minimal ABC model consists of three kinds of bees: employed bees, onlooker bees and scout bees. Half of the colony comprises employed bees and the other half includes the onlooker bees. It is assumed that there is only one artificial employed bee for each food source. In other words, the number of employed bees in the colony is equal to the number of food sources around the hive. Employed bees go to their food source and come back to hive and dance on this area. The employed bee whose food source has been abandoned becomes a scout and starts to search for finding a new food source. Scouts randomly search the environment in order to find a new food source depending on an internal motivation or possible external clues or randomly. Onlooker bees wait in the hive and decide a food source to exploit depending on the information shared by the employed bees. The main steps of the algorithm are as follows:

- Initial food sources are produced for all employed bees
- REPEAT
  - Each employed bee goes to a food source in her memory and determines a neighbor source, then evaluates its nectar amount and dances in the hive.
  - Each onlooker watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source. After choosing a neighbor around that, she evaluates its nectar amount.
  - Abandoned food sources are determined and are replaced with the new food sources discovered by scouts.
  - The best food source found so far is registered.
- UNTIL (requirements are met)

The mathematical steps involved in the ABC algorithm are as follows (Karaboga and Basturk, 2007, 2008):

In the first step of the algorithm, $x_i^j (i = 1, \ldots, SN)$ solutions are randomly produced by Equation (1) in the range of parameters, where $SN$ is the number of the food sources

$$ x_{ij} = x_{ij}^{\min} + (x_{ij}^{\max} - x_{ij}^{\min}) \times \text{rand}, \quad (1) $$

$j = 1, \ldots, D$, where $D$ is the dimension of problem.

In the second step, for each employed bee, a new source is produced by the following equation:

$$ v_{ij} = x_{ij} + \phi_{ij} (x_{ij} - x_{ij}), \quad (2) $$

where $\phi_{ij}$ is a uniformly distributed real random number within the range $[-1, 1]$, $k$ is the index of the solution chosen randomly from the colony ($k = \text{int} (\text{rand} \times SN) + 1$). New solution $v_i$ is compared with old solution $x_i$ and the best one is exploited by the employed bee.

In the third step of the algorithm, each onlooker bee chooses a food source site with the probability calculated by Equation (3) and produces a modification on the position of the food source based on the probability value

$$ p_i = \frac{\text{fit}_i}{\sum_{j=1}^{SN} \text{fit}_j}, \quad (3) $$

where $\text{fit}_i$ is the fitness of the solution $x_i$.

If the number of cycles through which a source cannot be improved is greater than a predetermined value, then the source is considered to be exhausted. The employed bee associated with this exhausted source becomes a scout and makes a random search in the problem domain to discover a new solution using Equation (1).

### 2.2. Nelder–Mead simplex algorithm

The Nelder–Mead simplex algorithm was proposed by Nelder and Mead (1965). It is an enormously popular direct search method for multidimensional unconstrained minimization. The method is deterministic in its formulation. It attempts to minimize a scalar-valued nonlinear function of real variables using only function values, without any derivative information (explicit or implicit).

The method maintains at each step a non-degenerate simplex, a geometric figure in $n$ dimensions of nonzero volume that is the convex hull of $n + 1$ vertices. Each iteration begins with a simplex, specified by its $n + 1$ vertices and the associated function values. One or more test points
are computed along with their function values and the iteration terminates with bounded level sets. In MATLAB, a standard function \texttt{fminsearch()} has been defined which simulates the Nelder–Mead algorithm.

3. Basics of fractional-order calculus and FOC

3.1. Fractional-order calculus

In fractional-order calculus, the fundamental differ-integration operator $aD_t^\lambda$ (where $a$ and $t$ are the limits of the operation) is defined as (Chen et al., 2009)

$$ aD_t^\lambda = \begin{cases} \frac{d^\lambda}{dt^\lambda}, & \lambda > 0, \\ 1, & \lambda = 0, \\ \int_a^t (dr)^{-\lambda}, & \lambda < 0, \end{cases} $$

where $\lambda$ is the order of the operation, generally $\lambda \in \mathbb{R}$ but $\lambda$ could also be a complex number.

3.1.1. Definitions

Some popular definitions of fractional differ-integration in fractional-order calculus are (Xue & Chen, 2007, chap. 8):

1. Grunwald–Letnikov definition

$$ aD_t^\lambda f(t) = \lim_{h \to 0} \frac{1}{h^n} \sum_{j=0}^{[(t-a)/h]} (-1)^j \binom{\lambda}{j} f(t - jh) $$

$[(t-a)/h]$ truncates $(t-a)/h$ to an integer.

2. Riemann–Liouville definition

$$ aD_t^\lambda f(t) = \frac{1}{\Gamma(n-\lambda)} \left( \frac{d}{dt} \right)^n \int_a^t \frac{f(\tau)}{(t-\tau)^{\lambda-n+1}} d\tau $$

$(n-1) \leq \lambda < n$, where $n$ is an integer and $a$ is a real number.

3. Caputo definition

$$ aD_t^\lambda f(t) = \frac{1}{\Gamma(n-\lambda)} \int_a^t (\tau)^{n-1} f^{(n)}(\tau) (t-\tau)^{-\lambda-n+1} d\tau. $$

3.1.2. Fractional-order transfer function

A linear time-invariant fractional model of a system with input $u$ and output $y$ takes the following form (Xue & Chen, 2007, chap. 8):

$$ a_n D_t^{\lambda_n} y(t) + a_{n-1} D_t^{\lambda_{n-1}} y(t) + \cdots + a_0 D_t^{\lambda_0} y(t) = b_m D_t^{\mu_m} u(t) + b_{m-1} D_t^{\mu_{m-1}} u(t) + \cdots + b_0 D_t^{\mu_0} u(t), $$

(4)

where $a_i, \lambda_i$ ($i = 0, 1, \ldots, n$), $b_k, \mu_k$ ($k = 0, 1, \ldots, m$) are real constants. $n$ and $m$ are positive integers.

The Laplace transform of the fractional-order operator (Xue & Chen, 2007) for zero initial conditions is

$$ L(aD_t^\lambda f(t)) = s^\lambda F(s). $$

Therefore, the Laplace transform on both sides of Equation (4) for zero initial conditions leads to the following transfer function:

$$ \frac{Y(s)}{U(s)} = \frac{b_m s^{\lambda_m} + b_{m-1} s^{\lambda_{m-1}} + \cdots + b_0 s^{\lambda_0}}{a_n s^{\lambda_n} + a_{n-1} s^{\lambda_{n-1}} + \cdots + a_0 s^{\lambda_0}}. $$

3.2. Fractional-order controllers

From control engineering point of view, the application of fractional-order calculus can be in either system modeling or controller design. The typical fractional-order controllers ($C(s)$) found in the literature (Chen et al., 2009) are as follows:

1. Fractional-order proportional–integral controllers

   - $PI^\lambda$

   $$ C(s) = K_p \left(1 + \frac{K_i}{s^{\lambda}}\right) $$

   [For integer $PI$, $\lambda = 1$.]

2. Fractional-order proportional–derivative controllers

   - $PD^\mu$

   $$ C(s) = K_p (1 + K_d s^\mu) $$

   [For integer $PD$, $\mu = 1$.]

3. Fractional-order proportional–integral–derivative controller

   - $PI^\lambda D^\mu$

   $$ C(s) = K_p + \frac{K_i}{s^{\lambda}} + K_d s^\mu $$

   (5)

   [For integer $PID$, $\lambda = \mu = 1$.]

4. Problem formulation

A typical unity feedback closed loop system is shown in Figure 1.

For our exploration, we select $PI^\lambda D^\mu$ controller as given in Equation (5). The various performance indices and their significance are as follows:

- IAE index

$$ J = \int_0^\infty |e(t)| \, dt. $$

(6)
The following cost functions are defined corresponding to the ISE case.

- **IAE cost function**
  \[ J_c = \sum_{k=0}^{N} |e(kT)|. \]  

- **ISE cost function**
  \[ J_c = \sum_{k=0}^{N} |e(kT)|^2. \]  

- **ITAE cost function**
  \[ J_c = \sum_{k=0}^{N} kT|e(kT)|. \]

Each performance index emphasizes different aspects of the system response (Umez-Eronini, 2002, chap. 11). Large errors contribute more in ISE than IAE. Consequently, the controller tuned for minimizing ISE ensures lower overshoot in the transient response than the IAE minimizing controller. The ISE, however, tends to give larger settling time. This is because, smaller errors \((e(t) < 1)\) are less quantified in ISE than IAE. The ITAE is the most sensitive of the three criteria. Due to the presence of the \(t\) (time) product term, ITAE weighs more heavily errors that occur later in the time (i.e. near to steady state). Therefore, the settling time is shortest. Also, as compared to ISE, large errors occurring in the transient part (i.e. for smaller \(r\) values) are less quantified. This leads to larger overshoots than the ISE case.

The controller parameters \((K_p, K_i, K_d, \lambda, \mu)\) are tuned by minimizing the selected performance index.

### 5. Results and discussion

For illustration purposes, we consider the following two plant examples:

**Example 1** First-order plus dead time (FOPDT) plant

\[
\text{Transferfunction, } G(s) = \frac{40}{s + 35}e^{-0.3s}. 
\]

**Example 2** Second-order plus dead time (SOPDT) plant

\[
\text{Transferfunction, } G(s) = \frac{400}{s^2 + 50s}e^{-0.5s}. 
\]

For the given plant, it is desired to tune \(PI^2D^\mu\) controller for IAE, ISE and ITAE minimization. The following numerical details are maintained common for these examples:

- For reducing the time of optimization, the search space is limited by considering the following bounds: \(K_p \in (0, 1], K_i \in (0, 1], K_d \in (0, 1], \lambda \in (0, 1], \mu \in (0, 1]\). A unit step input is given to the closed loop system for 5 s. The sampling period is chosen as 0.001 s. For the plant under consideration, the \(PI^2D^\mu\) controller is tuned to minimize IAE, ISE and ITAE. A MATLAB code is written to simulate ABC and Nelder–Mead simplex algorithms for solving the optimization problem.

- For the calculation of performance index, Oustaloup et al. (2000) approximated a transfer function model of the \(PI^2D^\mu\) controller considered. The order of Oustaloup approximation is taken as 3 and is valid over the frequency range \([0.01, 100]\) rad/s.

- In the ABC algorithm, colony size (employed bees + onlooker bees) is taken as 20. The number of trials after which a food source is to be abandoned in case of no improvement is taken as 30. The algorithm is iterated through 1000 evaluations and such a process is run 30 times with different random seeds. The Nelder–Mead algorithm is run 30 times with random initial guess. For each of these algorithms, the best among the converged solutions is taken as the final solution.

The simulation results for various performance criteria using ABC and Nelder–Mead simplex algorithms for Example 1 are shown in Table 1. The similar results for Example 2 are presented in Table 2.

As seen in Tables 1 and 2, the ABC algorithm produces quite comparable results to the classic Nelder–Mead optimization algorithm. It is interesting to note that for the ISE case in Example 2, ABC outperforms the Nelder–Mead algorithm. This is due to the in-built randomness which helps ABC to come out of local minima. Such a solution is not possible with the deterministic Nelder–Mead algorithm.
Table 1. Results for Example 1.

| Index | Algorithm | Best controller | Cost ($J_c$) |
|-------|-----------|----------------|-------------|
| IAE   | ABC       | $0.7515 + \frac{1}{s} + 0.01s$ | 851.0708    |
|       | Nelder–Mead | $0.7513 + \frac{1}{s} + 0.01s$ | 851.0696    |
| ISE   | ABC       | $0.5679 + \frac{1}{s} + 0.01s^{0.9468}$ | 437.6184    |
|       | Nelder–Mead | $0.5681 + \frac{1}{s} + 0.01s^{0.9464}$ | 437.6183    |
| ITAE  | ABC       | $0.01 + \frac{1}{s^{0.9997}} + 0.01s$ | 491.9330    |
|       | Nelder–Mead | $0.01 + \frac{1}{s^{0.9997}} + 0.01s$ | 491.9330    |

Table 2. Results for Example 2.

| Index | Algorithm | Best controller | Cost ($J_c$) |
|-------|-----------|----------------|-------------|
| IAE   | ABC       | $0.2842 + \frac{0.01}{s^{0.0864}} + 0.0756s$ | 433.566     |
|       | Nelder–Mead | $0.2842 + \frac{0.01}{s^{0.0866}} + 0.0756s$ | 433.566     |
| ISE   | ABC       | $0.1226 + \frac{0.1051}{s^{0.0766}} + 0.0661s$ | 530.3219    |
|       | Nelder–Mead | $0.1 + \frac{0.1265}{s^{0.3792}} + 0.1s^{0.8989}$ | 541.3945    |
| ITAE  | ABC       | $0.2862 + \frac{0.01}{s^{0.01}} + 0.0738s$ | 72.4956     |
|       | Nelder–Mead | $0.2854 + \frac{0.01}{s^{0.01}} + 0.07354s$ | 72.4853     |

Table 3. Statistical measures for ABC (Example 1).

| Measure | IAE   | ISE   | ITAE  |
|---------|-------|-------|-------|
| Mean    | 851.4792 | 437.6378 | 491.9330 |
| Standard deviation | 0.3758 | 0.0427 | 0     |

Table 4. Statistical measures for ABC (Example 2).

| Measure | IAE   | ISE   | ITAE  |
|---------|-------|-------|-------|
| Mean    | 433.6268 | 530.3473 | 72.7576 |
| Standard deviation | 0.0888 | 0.0265 | 0.3984 |

Thus, the stochastic ABC algorithm can serve as a valid candidate for solving complex multi-modal optimization problems.

For Example 1, the statistical data obtained from 30 runs of ABC algorithm for various cost functions are summarized in Table 3. The similar results for Example 2 are presented in Table 4. Since standard deviation values are small, one can go for lesser number of runs.

Average performance value versus iteration number is plotted for 30 runs of ABC algorithm. For Example 1, Figures 2, 3 and 4 show such plots for IAE, ISE and ITAE.
minimization cases, respectively. The corresponding plots for Example 2 are shown in Figures 5, 6 and 7. It is observed that the average performance value converges as the iteration number progresses.

6. Conclusion
Application of stochastic ABC algorithm was proposed for solving high-dimensional and multi-modal optimization problem while tuning the \( PI^\lambda D^\mu \) controller. The results were compared with those obtained using the deterministic Nelder–Mead algorithm. The results were promising, though regressive in certain cases. The study emphasized on the potential of ABC as an alternate candidate algorithm to seek for the possible better solution. Future scope includes the extension of ABC algorithm for solving the constrained optimization problem to tune \( PI^\lambda D^\mu \) controller satisfying desired specifications.

Disclosure statement
No potential conflict of interest was reported by the authors.

References
Biplab, S., Koley, C., & Datta, S. (2014). Robust PID controller design using PSO enabled automated QFT approach for first order lag system with minimal dead time. Systems Science and Control Engineering, 2(1) 502–511.
Buanovi, L. J., Lazarevi, M. P., & Batalov, S. N. (2014). Fractional PID controllers tuned by genetic algorithms for expansion turbine in the cryogenic air separation process. Hemijska industrija, 68(5) 519–528.
Cao, J. Y., & Cao, B. G (2006). Design of fractional order controller based on particle swarm optimization. International Journal of Control, Automation and Systems, 4(6), 775–781.
Cao, J. Y., Liang, J., & Cao, B. G. (2005, August 18–21). Optimization of fractional order PID controllers based on genetic algorithms. Proceedings of international conference on machine learning and cybernetics, Guangzhou, China (Vol. 9, pp. 5686–5689).
Chang, L. Y., & Chen, H. C. (2009). Tuning of fractional PID controllers using adaptive genetic algorithm for active magnetic bearing system. WSEAS Transactions on Systems, 8(1), 158–167.
Chen, Y. Q., Petras, I., & Xue, D. U. (2009, June 10–12). Fractional order control—A tutorial. American control conference, WeC02 Tutorial Session, Grand Ballroom B (east), St. Louis, MO (pp. 1397–1411).
Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization (Technical Report-TR-06). Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey.
Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation, 214, 108–132.
Karaboga, D., & Akay, B. (2010). Proportional integral derivative controller design by using artificial bee colony, harmony search and the bees algorithms. Proceedings of the IMechE, 224(7), 869–883.
Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39(3), 459–471.
Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing, 8(1), 687–697.

Kesarkar, A. A., & Selvaganesan, N. (2011, December 8–10). Design of fractional order robust controller for universal plant structure. The second international conference on current trends in technology (NUiCONe), Ahmedabad, Gujarat, India (pp. 1–4).

Kesarkar, A. A., & Selvaganesan, N. (2011, December 15–17). Tuning of robust $PI^\lambda D^\mu$ controller for generalized plant structure. International conference on recent advancements in electrical, electronics and control engineering (ICON-RAeCE), Sivakasi, Tamilnadu, India (pp. 104–108).

Lee, C. H., & Chang, F. K. (2010). Fractional-order $PID$ controller optimization via improved electromagnetism-like algorithm. Expert Systems with Applications, 37(12), 8871–8878.

Li, H. S., & Chen, Y. Q. (2008, July 2–4). A fractional order proportional and derivative (FOPD) controller tuning algorithm. Proceedings of the Chinese control and decision conference, Yantai, Shandong (pp. 4059–4063).

Li, H. S., Luo, Y., & Chen, Y. Q. (2010). A fractional order proportional and derivative (FOPD) motion controller: Tuning rule and experiments. IEEE Transactions on Control Systems Technology, 18(2), 516–520.

Luo, Y., & Chen, Y. Q. (2009, June 10–12). Fractional-order [proportional derivative] controller for robust motion control: Tuning procedure and validation. Proceedings of the American control conference, St. Louis, MO (pp. 1412–1417).

Merrikh-Bayat, F. (2012). General rules for optimal tuning the $PI^\lambda D^\mu$ controllers with application to first-order plus time delay processes. Canadian Journal of Chemical Engineering, 90(6), 1400–1410.

Monje, C. A., Vinagre, B. M., Chen, Y. Q., Feliu, V., lanusse, P., & Sabatier, J. (2004). Proposals for fractional $PI^\lambda D^\mu$ tuning. Proceedings of the first IFAC symposium on fractional differentiation and its applications, Bordeaux, France.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. Computer Journal, 7, 308–313.

Oldham, K. B., & Spanier, J. (1974). The fractional calculus. New York: Academic Press.

Oustaloup, A., Levron, F., Mathieu, B., & Nanot, F. M. (2000). Frequency-band complex noninteger differentiator: Characterization and synthesis. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 47(1), 25–40.

Podlubny, I. (1999). Fractional-order systems and $PI^\lambda D^\mu$-controllers. IEEE Transactions on Automatic Control, 44(1), 208–214.

Roy, G. G., Chakraborty, P., & Das, S. (2010). Designing fractional-order $PI^\lambda D^\mu$ controller using differential harmony search algorithm. International Journal of Bio-Inspired Computation, 2(5), 303–309.

Umez-Eronini, E. (2002). System dynamics and control. Introducing automatic control systems design. Singapore: Thomson Asia.

Valerio, D. (2005). Fractional robust system control (Dissertation). Instituto Superior Tencico, Lisboa.

Valerio, D., & Costa, J.S.D. (2006). Tuning-rules for fractional $PID$ controllers. Preprints IFAC workshop on fractional differentiation and its applications, Porto, Portugal.

Wang, C. Y., Luo, Y., & Chen, Y. Q. (2009, June 17–19). Fractional order proportional integral (FOPI) and [proportional integral] (FO[PI]) controller designs for first order plus time delay (FOPTD) systems. Chinese control and decision conference, Guilin (pp. 329–334).

Wang, C. Y., Luo, Y., & Chen, Y. Q. (2009, May 25–27). An analytical design of fractional order proportional integral and [proportional integral] controllers for robust velocity servo. Proceedings of the fourth IEEE conference on industry electronics and applications, Xian, China (pp. 3448–3453).

Xue, D., Chen, Y. Q., & Atherton, D. (2007). Fractional-order controller: an introduction. In Linear feedback control: analysis and design with Matlab (pp. 283–306). Philadelphia, PA: SIAM Press.