Nonlinear Dyn (2019) 97:1735–1746
https://doi.org/10.1007/s11071-019-05103-5

ORIGINAL PAPER

Fractional-order flight control of quadrotor UAS on vision-based precision hovering with larger sampling period

Bo Shang · Jianxin Liu · Yunzhou Zhang · Chengdong Wu · YangQuan Chen

Received: 29 August 2018 / Accepted: 28 June 2019 / Published online: 9 July 2019

Abstract Fractional-order flight control has a history of nearly 10 years. Fractional-order controllers (FOCs) have been proved to be better in rising time, overshoot and robustness against plant variation. However, there are still not many real applications of FOC in industry. More case studies need to be carried out to accelerate the maturation of FOC. Quadrotor vision-based control often has a large and time-varying sampling period due to drone's resource limitation. Therefore, our research has been focused on a specific case for drone vision-based control to investigate the benefits of FOC. In this paper, FOC has been first discovered to be able to tolerate larger sampling period than integer-order controllers. This fact has been proved both theoretically and numerically. First of all, the speed model was identified from real flight tests. Then an integral-order proportional, integral and derivative (IOPID) controller and a fractional-order proportional–derivative (FOPD) controller were designed. After that, a stability criteria, optimization method, graphic method and parallel computing techniques were employed to theoretically prove that the largest sampling period of the designed FOC (0.933 s) is much larger than that of the designed integral-order controller (0.546 s). Later, Simulink simulation with identified linear model proved that the FOC can tolerate larger sampling period. Finally, flight tests showed that the designed FOC has a nearly 20% better precision on drone vision-based hovering than the IOC.

Keywords Fractional-order flight control · Robust control · Sampling period · Vision-based control

This work was supported in part by the research project granted by Xihua University and China Scholarship Council (CSC), National Key R&D Program of China, No. 2017YFB1300900 and Shenyang City Foundation #17-87-0-00.

B. Shang
College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
e-mail: cnpcshangbo@gmail.com

J. Liu
School of Mechanical Engineering, Xihua University, Chengdu, Sichuan, China
e-mail: jianxin2013liu@outlook.com

Y. Zhang · C. Wu (✉)
Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China
e-mail: wuchengdong_neu@outlook.com

Y. Zhang
e-mail: zhangyunzhou@mail.neu.edu.cn

Y. Chen
Mechatronics, Embedded Systems and Automation (MESA) Lab, School of Engineering, University of California, Merced, CA, USA
e-mail: ychen53@ucmerced.edu

1 Introduction

Fractional-order flight control (FOFC) was introduced by Dr. Chen about 10 years ago [1]. Over these years, FOFC has been proved to have a better performance on step response rising time, overshoot and robustness

against plant gain variation [2]. Intuitively, fractional-order controller (FOC) can fit a plant better than integer-order controller (IOC). Therefore, it should have a boost of usage over these years. However, FOFC is still not widely adopted by the industry.

The FOC has been proved from mathematics point of view and tested for some applications. However, the road to industry still needs to be improved. More researchers and case studies are needed to get it mature. Therefore, we need to draw more attention from control researchers by proving that FOC have enough benefits that we are worth spending more time and effort to make it more accessible to the industry.

This paper is based on two conference papers [2,3]. A case study on drone vision-based precision hovering is investigated. Plant model has been identified for position control. An IOC and an FOC have been designed. A stability criteria, two optimization methods and parallel computing techniques have been used to estimate the largest sampling period for the two controllers which proves the FOC can tolerate larger sampling period (our code: [4]). Then a linear model simulation has been done with different sampling periods. Simulation results show that FOC can tolerate larger sampling period, too. Flight tests show that the designed FOC hovers the drone more precisely.

The contributions of this paper are:

- A fractional-order controller (FOC) has been designed for vision-based control of a custom-made quadrotor.
- FOC has been first proved to be able to tolerate larger sampling period theoretically and numerically.
- Optimization method, graphic method and parallel computing techniques have been used to finish the estimation of the system's largest sampling period.
- A fractional-order proportional–derivative (FOPD) controller tuner has been provided to the research community.

2 Preliminaries

2.1 Fractional-order differentiation definition

There are several commonly used fractional differentiation definitions [5–7]. For FOC designing, we use Caputo definition:

$$ {}_aD_t^{\alpha} = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^n(\tau)}{(t-\tau)^{(\alpha-n+1)}} \mathrm{d}\tau. \tag{1} $$

2.2 Proportional and fractional-order derivative (PD$^{\lambda}$) controller designing specifications

This paper uses the phase-flat rule for FOC designing. To clarify, $P(s)$ represents the plant's transfer function (TF) and $C(s)$ represents the controller's TF. Therefore, $G(s) = C(s)P(s)$ is the open-loop transfer function. If we have expected gain crossover frequency ω_c and phase margin ϕ_m, the three specifications for phase-flat rule are as described in the literature [8,9].

1. Phase margin specification

$$ \begin{aligned} Arg[G(j\omega_c)] &= Arg[C(j\omega_c)P(j\omega_c)] \\ &= -\pi + \phi_m. \end{aligned} \tag{2} $$

2. Robustness to variation in the gain of plant

$$ \frac{\mathrm{d}(Arg[C(j\omega)P(j\omega)])}{\mathrm{d}\omega}|_{\omega=\omega_c} = 0, \tag{3} $$

with the condition that the phase derivative w.r.t. the frequency is zero, which means that the system is more robust to gain changes and the overshoots of the responses are almost the same.

3. Gain crossover frequency specification

$$ |G(j\omega_c)|_{\mathrm{dB}} = |C(j\omega_c)P(j\omega_c)|_{\mathrm{dB}} = 0. \tag{4} $$

2.3 Criteria used for estimation of the largest sampling period

As mentioned in our previous research [3], the second theorem in [10] has been used to build the optimization model for estimating the largest sampling period.

3 System identification of the quadrotor speed loop

3.1 System description of the quadrotor UAS

As shown in Fig. 1, the quadrotor system is based on a 3DR DIY platform. An Odroid XU-3 has been mounted on the tail of the platform. The airframe's Pixhawk flight controller has been connected to Odroid with a UART to USB converter. A web camera and a Wi-Fi

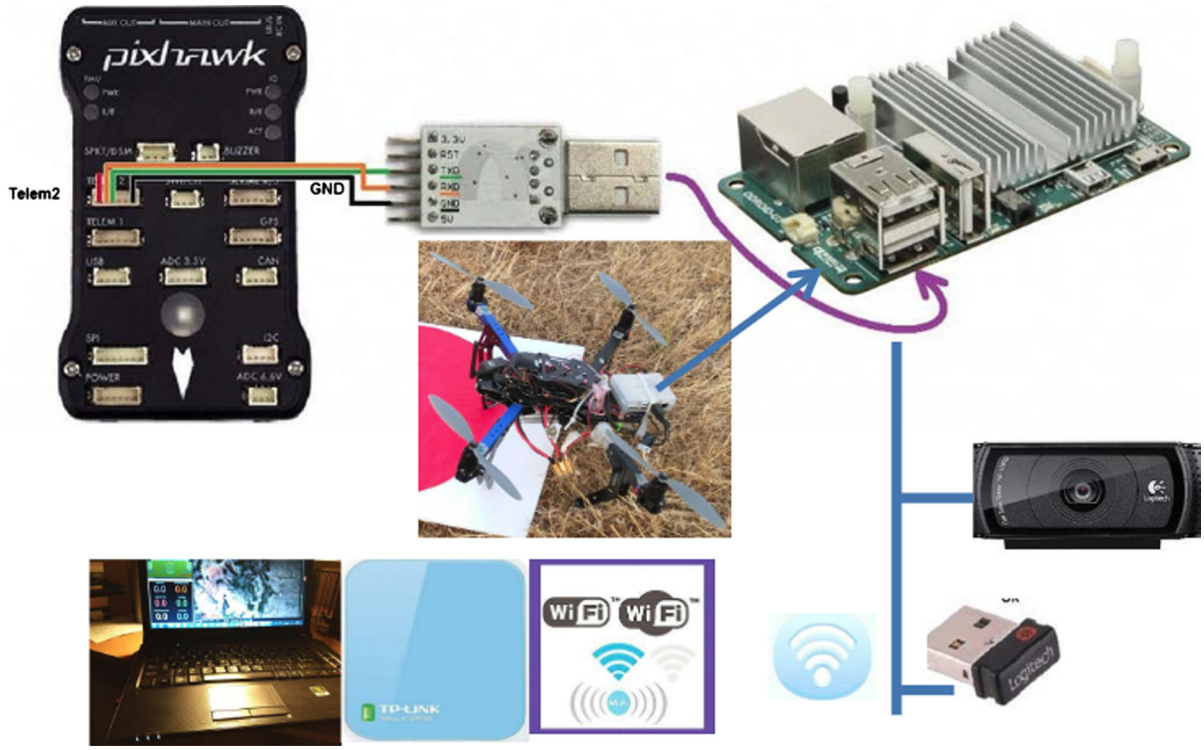


Fig. 1 Architecture of drone visual servoing system

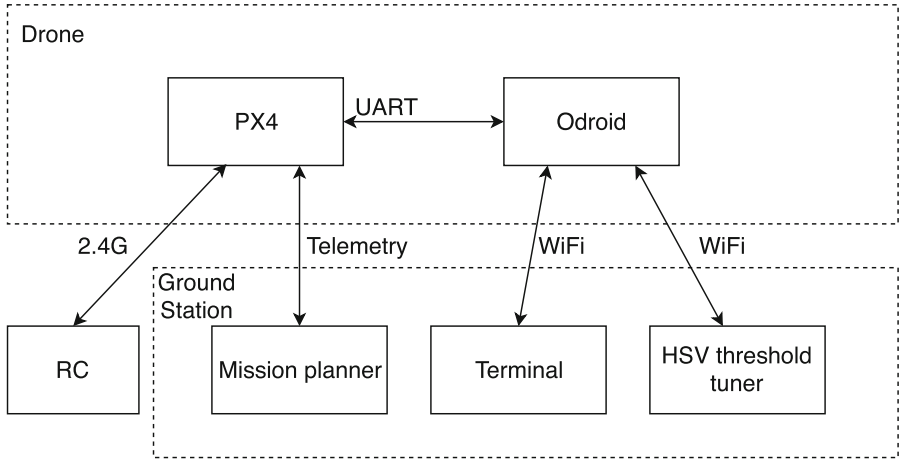


Fig. 2 System software architecture

dongle have been connected to Odroid via USB interface. A portable Wi-Fi hotspot has been involved to provide network coverage. A ground control station has been used to monitor the drone's behavior and send commands. Our drone code [11] is based on Randy Mackay's open-source project ardupilot-balloon-finder [12]. The framework has been upgraded to DroneKit 2.0 to enable commands to send the drone's velocity set point. An online HSV threshold tuner has been developed based on HTTP protocol.

3.2 Software architecture

As shown in Fig. 2, the system software includes two parts: drone side and GCS (ground control station) side. The drone include the PX4 autopilot and an Odroid connected to the autopilot. The Odroid has an Ubuntu oper-

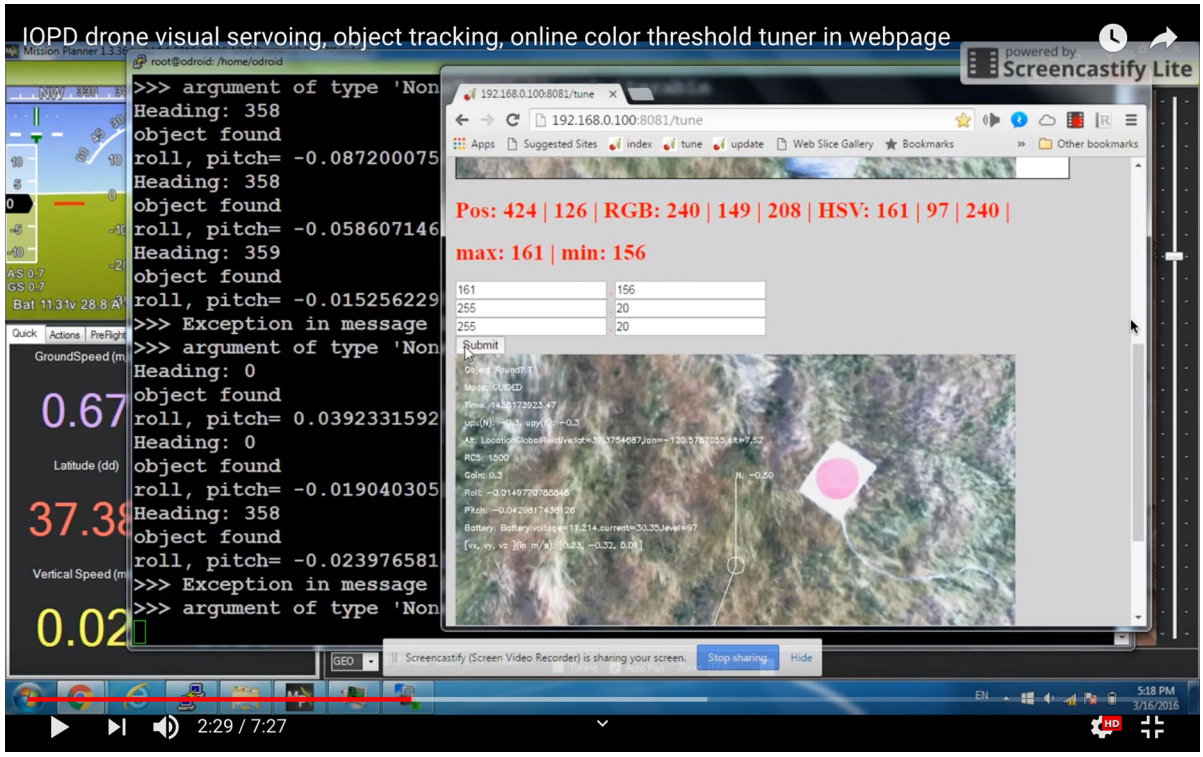
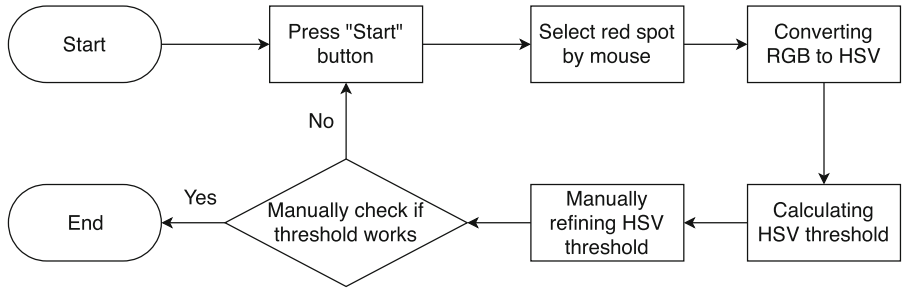


Fig. 3 HSV threshold tuner application graphical user interface

Fig. 4 HSV threshold tuner workflow



ation system which runs DroneKit. The DroneKit provides a Python interface for interacting with the drone autopilot. We use a Windows computer as the ground control station. After using Putty to connect to the drone via Wi-Fi and SSH protocol, we launch "Screen" and run the Python script. The "Screen" software ensures that the Python script keeps running even if the Wi-Fi connection breaks. When the Python script is running, we can open a web browser and visit http://192.168.0.100:8081 to access the HSV threshold tuner as shown in Fig. 3. The HSV threshold tuning process is done by using mouse to select the red spot and manually refine the threshold as shown in Fig. 4.

3.3 System identification of the drone's speed loop

In this paper, we focus on the drone's position control. Therefore, we have treated the speed loop as a black box. The control loop has been established as shown in Fig. 5.

To identify the drone's speed loop transfer function, a step function input has been employed. The drone's actual speed has been logged as system output. MATLAB System Identification Toolbox has been used to process the data. Finally, we got the speed model like this:

$$P(s) = \frac{1.03}{0.71s + 1}\frac{1}{s}. \tag{5}$$

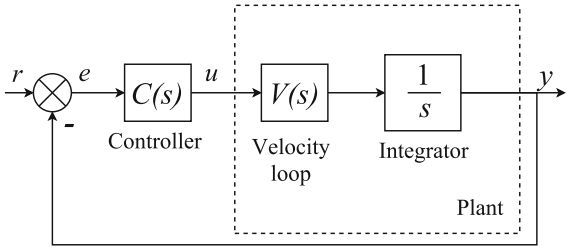


Fig. 5 Diagram of quadrotor position control

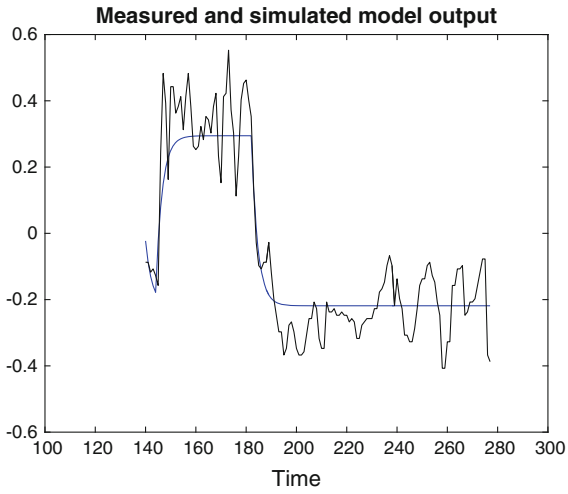


Fig. 6 System identification of the quadrotor's speed model

As shown in Fig. 6, the identified model approximates the measured data. However, noise and unmodeled dynamics exist.

4 Controller designing for quadrotor position control

Since we already know some information of the plant model, model-based controllers have been considered to make use of the model information. Model-free controllers [13–15] are not considered in this article because they do not use model information in control.

4.1 IOPID controller designing

A MATLAB function called `pidtune` [16] has been used to design the IOPID controller with a good balance between performance and robustness [17]. The `pidtune` function is from MATLAB Control System Toolbox. When using this tool, crossover frequency has been set to 1.2 rad/s and phase margin has been set to 83.9°. Then we got the tuned PID parameters as follows: $K_\mathrm{p} = 2.9$, $K_\mathrm{i} = 1$, $K_\mathrm{d} = 1.8$.

Therefore, the tuned integer-order PID controller's transfer function is:

$$C_{\mathrm{IOPID}}(s) = 2.9 + \frac{1.0}{s} + 1.8s. \tag{6}$$

4.2 PD$^\lambda$ controller designing

Based on the flat-phase rule [8,9], a PD$^\lambda$ tuner (https://github.com/cnpcshangbo/FOPD-tuner) has been made for tuning PD$^\lambda$ controllers.

4.2.1 Frequency-domain model of the PD$^\lambda$ controller

The transfer function of PD$^\lambda$ controller is:

$$C(s) = K_\mathrm{p}(1 + K_\mathrm{d}s^\lambda). \tag{7}$$

By substituting s with $j\omega$, PD$^\lambda$ controller can be described as:

$$C(j\omega) = K_\mathrm{p}(1 + K_d(j\omega)^\lambda). \tag{8}$$

According to the definition of fractional-order calculus, we have:

$$(j\omega)^\lambda = \omega^\lambda \cos\frac{\pi\lambda}{2} + j\omega^\lambda \sin\frac{\pi\lambda}{2}. \tag{9}$$

Therefore, the transfer function of PD$^\lambda$ controller can be written as:

$$C(j\omega) = K_\mathrm{p}\left[\left(1 + K_\mathrm{d}\omega^\lambda \cos\frac{\pi\lambda}{2}\right) + jK_\mathrm{d}\omega^\lambda \sin\frac{\pi\lambda}{2}\right]. \tag{10}$$

The phase and gain are as follows:

$$Arg[C(j\omega)] = \arctan\frac{K_\mathrm{d}\omega^\lambda \sin\frac{\pi\lambda}{2}}{1 + K_\mathrm{d}\omega^\lambda \cos\frac{\pi\lambda}{2}}, \tag{11}$$

$$|C(j\omega)| = K_\mathrm{p}J(\omega), \tag{12}$$

where

$$J(\omega) = \sqrt{\left(1 + K_\mathrm{d}\omega^\lambda \cos\frac{\pi\lambda}{2}\right)^2 + \left(K_\mathrm{d}\omega^\lambda \sin\frac{\pi\lambda}{2}\right)^2}. \tag{13}$$

4.2.2 Frequency-domain model of the plant

The transfer function of the plant is:

$$P(s) = \frac{k}{s(\tau s + 1)}. \tag{14}$$

After substituting s with $j\omega$, the plant can be described as:

$$P(j\omega) = -\frac{k(\tau\omega + j)}{\omega(\tau^2\omega^2 + 1)}. \tag{15}$$

The phase and gain are as follows:

$$Arg[P(j\omega)] = \arctan\frac{1}{\tau\omega}, \tag{16}$$

$$|P(j\omega)| = \frac{k}{\omega(\tau^2\omega^2 + 1)}\sqrt{\tau^2\omega^2 + 1}. \tag{17}$$

4.2.3 Frequency-domain response of the open-loop transfer function

The open-loop transfer function is $G(s) = C(s)P(s)$. From (11) and (16), the phase of $G(s)$ is as follows:

$$Arg[G(j\omega)] = arctan\frac{K_d\omega^\lambda \sin\frac{\pi\lambda}{2}}{1 + K_d\omega^\lambda\cos\frac{\pi\lambda}{2}} + arctan\frac{1}{\tau\omega}. \tag{18}$$

The gain of $G(s)$ is:

$$|G(j\omega)| = K_p J(\omega)\frac{k}{\omega(\tau^2\omega^2 + 1)}\sqrt{\tau^2\omega^2 + 1}. \tag{19}$$

4.2.4 Tuning of PD$^\lambda$ controller

According to the performance index specifications in (2) and (3), the parameters of K_d and λ should satisfy Eqs. (20) and (21) simultaneously when ω is set to the gain crossover frequency ω_c.

$$K_d = \frac{\tan(\arctan\frac{1}{\tau\omega} - \phi_m)}{-\omega^\lambda\sin\frac{\pi\lambda}{2} - \omega^\lambda\cos\frac{\pi\lambda}{2}\tan(\arctan\frac{1}{\tau\omega} - \phi_m)}. \tag{20}$$

$$A_1K_d^2 + A_2K_d + A_3 = 0, \tag{21}$$

where

$$A_1 = \tau\omega^{2\lambda}, \tag{22}$$

$$A_2 = 2\tau\omega^\lambda\cos\frac{\pi\lambda}{2} - \sin\frac{\pi\lambda}{2}\omega^{\lambda-1}\lambda(1 + \lambda^2\omega^2), \tag{23}$$

$$A_3 = \tau. \tag{24}$$

Therefore, the parameter K_d should satisfy Eq. (25).

$$K_d = \frac{-A_2 \pm \sqrt{A_2^2 - 4A_1A_3}}{2A_1}. \tag{25}$$

Furthermore, the parameter of K_p should satisfy Eq. (26).

$$|G(j\omega)| = K_pJ(\omega)\frac{k}{\omega\sqrt{\tau^2\omega^2 + 1}} = 1. \tag{26}$$

4.2.5 Tuned PD$^\lambda$ controller parameters

We set the same ω_c and ϕ_m for FOPD controller. λ is set to be between 0 and 2. Then we could use the graphical method to solve Eqs. (20) and (25). We have got the solution:

$$\lambda = 0.9855, \tag{27}$$

$$K_d = 0.2431. \tag{28}$$

By using (26), we can calculate that $K_p = 4.2393$. Therefore, the transfer function of the fractional-order controller is:

$$C_{FOPD}(s) = 0.6192s^{0.9694} + 2.6992. \tag{29}$$

4.3 Bode plot analysis and step response analysis for the two designed controllers

We drew the Bode diagrams of the two controllers in one axis (as shown in Fig. 7). The magnitude plots

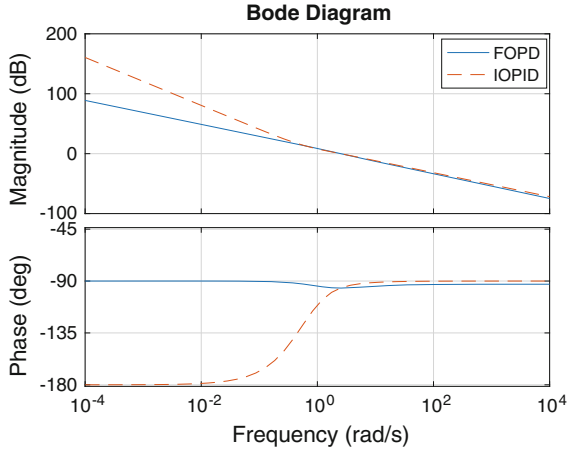


Fig. 7 Bode plots of PD$^\lambda$ controller and integer-order PID controller

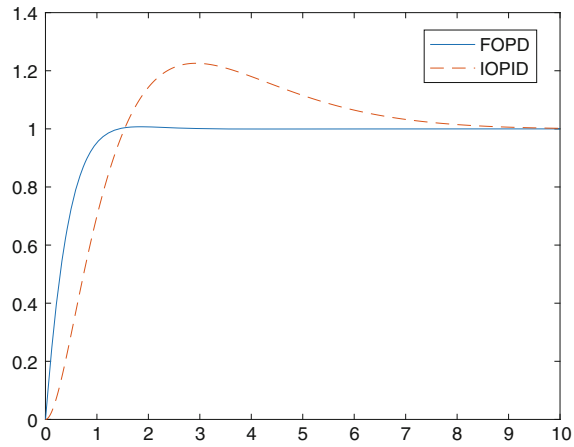


Fig. 8 Step responses comparison between FOPD and IOPID

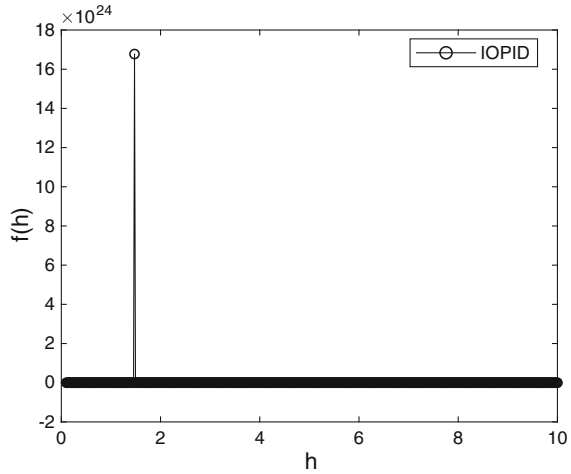


Fig. 9 Relationship between $f(h)$ and h for the IOPID controller

of the two controllers are similar. However, the phase plots of the two controllers are quite different. In order to make the FOPD controller more robust when the plant changes, our configuration of the FOPD controller makes the derivative of the phase plot to be zero at crossover frequency. In this way, the phase value variation would be less when the plant crossover frequency changes due to the plant variation.

Step response of the two controllers is displayed in Fig. 8. The designed FOC rises fast and stops growing when the speed reaches the set point value. However, the IOPID controller rises slower and does not stop growing until more than 20% overshoot.

5 Estimating the largest sampling periods of the two controllers

The largest sampling periods of the two controllers have been estimated as optimization problems, which have been solved with graphic method. A 32-core MATLAB server and parallel computing techniques have been used to get the solution within a short period of time.

5.1 Solving the optimization problem

In our previous research [3], we proposed to use an embedded optimization method to deal with the nonlinear function. The inner optimization was implemented with MATLAB Global Optimization Toolbox. The outer optimization was manually solved by using graphic method. In that research, we drew 8 points each time and then changed the range according to the place where the curve crosses the x axis. However, it is hard to tell whether the solution is a local minimum.

Luckily, we have got access to a 32-core MATLAB server which can help us calculate more points within a short period of time. SSH (Secure Socket Shell) has been used to configure the server environment. VNC (Virtual Network Computing, https://www.realvnc.com/en/) has been used to remotely control the Ubuntu server graphically from a MacBook Pro laptop which enables MATLAB scripts to continuously run even after Internet disconnects unexpectedly. GitHub (https://github.com/), SFTP (SSH File Transfer Protocol) and Mountain Duck (https://mountainduck.io/) have been used to transfer code, data, figures, etc.

5.2 Solution of the optimization problem

In this section, each controller uses three figures to find the largest sampling period.

5.2.1 Largest sampling period of the IOPID controller

Figure 9 is a 512-point plot to reflect the relationship between $f(h)$ and h. We got Fig. 10 by zooming Fig. 9 vertically. It proves that the curve has only one crossover point that is between 0 and 2, which means we have got a global minimum rather than local minimum. Then we zoom out Fig. 10 a little to get Fig. 11, which shows us the estimation of the largest sampling period of IOPID controller is 0.546 (as shown in Table 1).

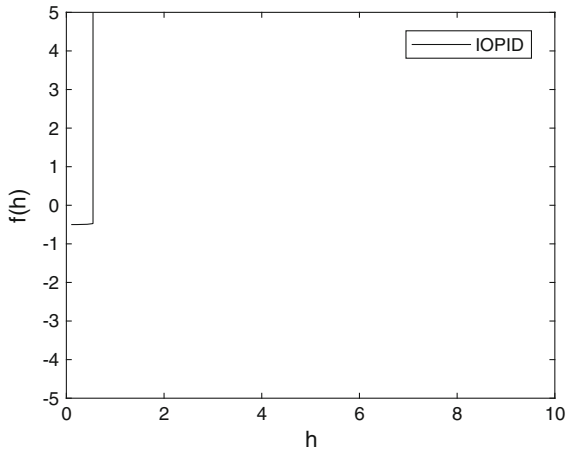


Fig. 10 Relationship between $f(h)$ and h for the IOPID controller (zoomed for positioning global solution)

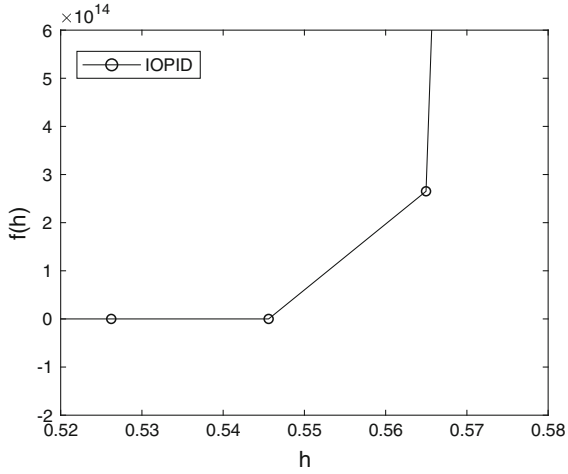


Fig. 11 Relationship between $f(h)$ and h for the IOPID controller (zoomed for getting the global solution)

Table 1 Solution and time cost for estimating the largest sampling period

	FOPD controller	IOPID controller
Largest sampling period (s)	0.933	0.546
Computational time (s)	2282	2982
Parallel computational time (s)	248	269

5.2.2 Largest sampling period of the FOPD controller

For the FOPD controller, we also have three figures to show the overview (Fig. 12), the position of the global

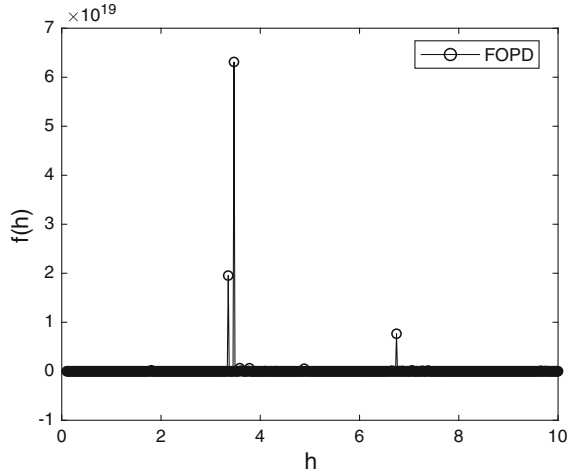


Fig. 12 Relationship between $f(h)$ and h for the FOPD controller

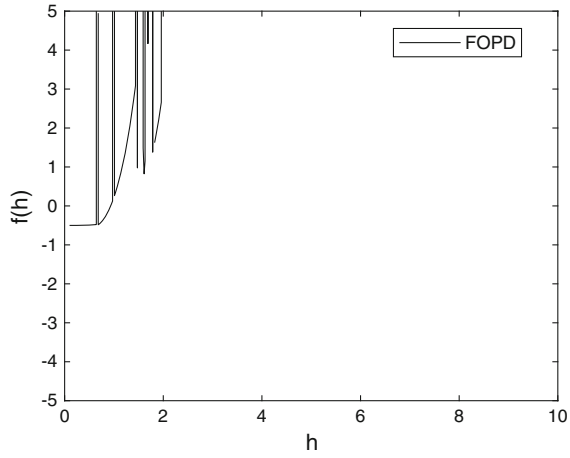


Fig. 13 Relationship between $f(h)$ and h for the FOPD controller (zoomed for positioning the global solution)

solution (Fig. 13) and the estimation of the solution (Fig. 14). The results are given in Table 1.

The MATLAB Parallel Computing Toolbox has been used to reduce computational time. Sixteen workers have been used for the two processes. About 90% of time has been saved with parallel computing in this case (as shown in Table 1).

6 Linear plant model simulation with different sampling periods

A Simulink simulation has been done with the identified plant for the two controllers. Four sampling periods have been selected for discretization process.

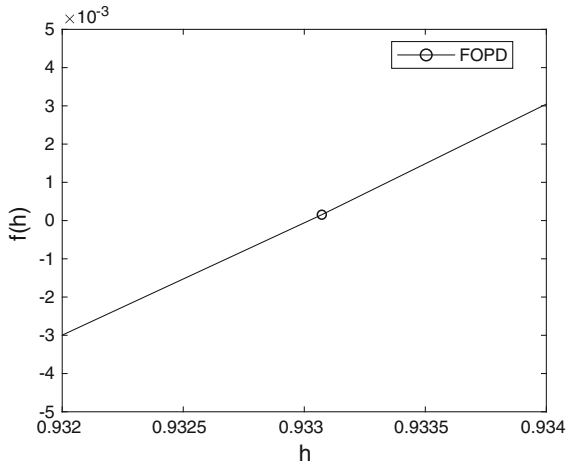


Fig. 14 Relationship between $f(h)$ and h for the FOPD controller (zoomed for getting the global solution)

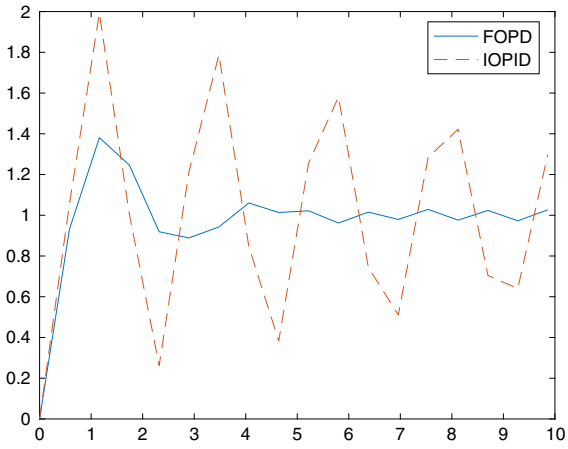


Fig. 15 Step response comparison between the PD$^{\lambda}$ controller and the integer-order PID controller when sampling period is 0.58 s

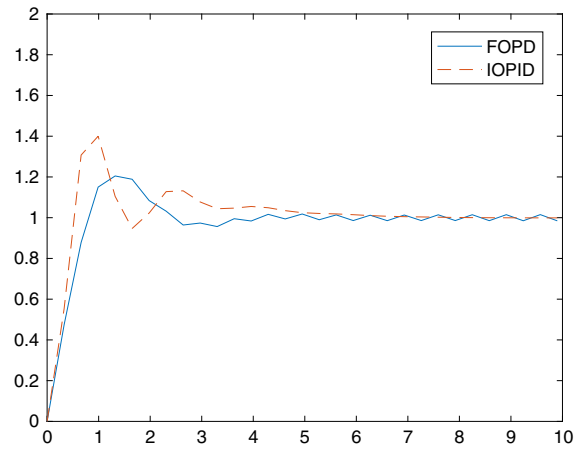


Fig. 16 Step response comparison between the PD$^{\lambda}$ controller and the integer-order PID controller when sampling period is 0.33 s

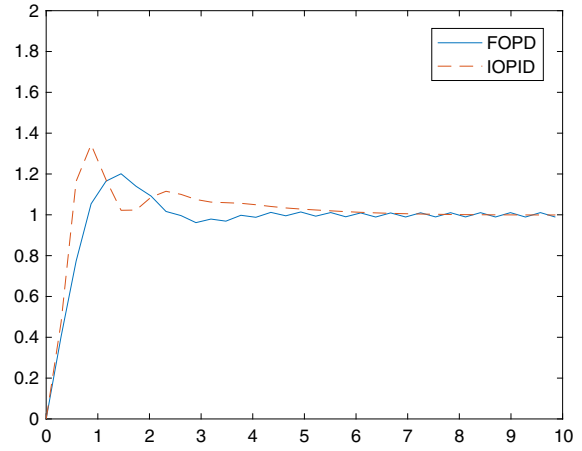


Fig. 17 Step response comparison between the PD$^{\lambda}$ controller and the integer-order PID controller when sampling period is 0.29 s

6.1 Step response for different sampling periods

Figures 15, 16, 17 and 18 show the step responses of the two controllers under different sampling periods. When the sampling period is short, the performance of the two controllers is very similar; however, as the sampling period gets larger and larger, the IOPID controller performance downgrades faster than the FOPD controller. This result shows that the PD$^{\lambda}$ controller deals with larger sampling period situations better than the integer-order PID controller.

6.2 ITAE index for different sampling periods

ITAE (Integrated Time Absolute Error [18]) has been used to compare the integer-order PID controller (Eq. 6) and PD$^{\lambda}$ (Eq. 29) controller. A Simulink library [19] has been used to calculated the ITAE values.

Figure 19 is the Simulink model used to estimate the ITAE index for the two control systems. Results show that the error index of the PD$^{\lambda}$ controller grows slower than that of the IOPID controller (Figs. 20, 21). The error index of the integer-order PID controller decreases after 1.2 s. The simulation time is fixed to 10 s. If sampling period is 1 s, the ITAE is a sum of 10

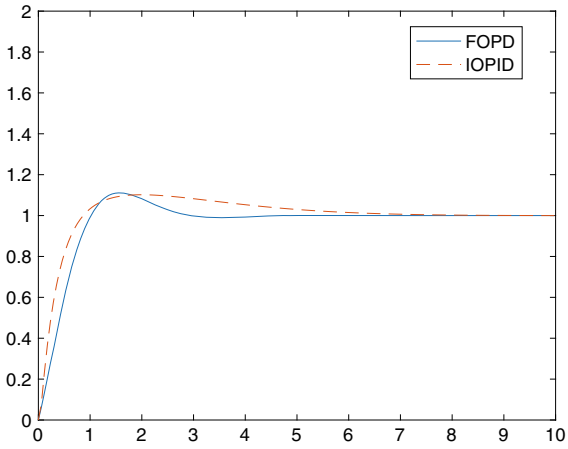


Fig. 18 Step response comparison between the PD$^{\lambda}$ controller and the integer-order PID controller when sampling period is 0.058 s

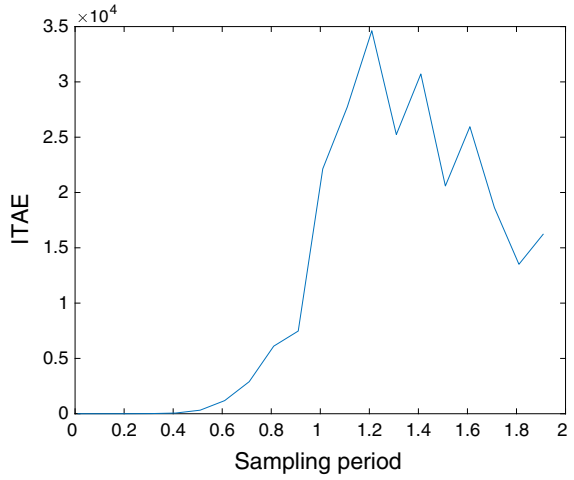


Fig. 20 ITAE index for different sampling periods (s) (IOPID)

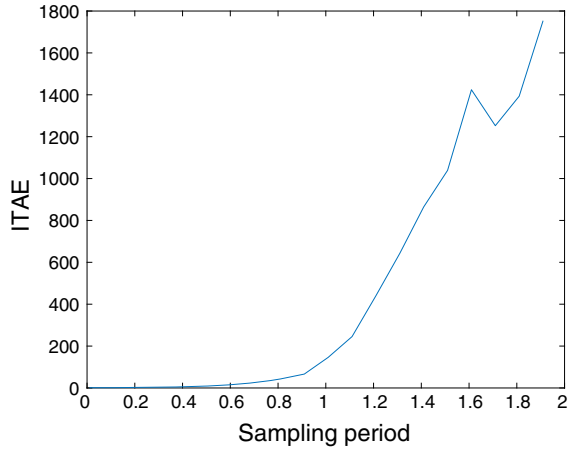


Fig. 21 ITAE index for different sampling periods (s) (FOPD)

numbers; if sampling period is 2 s, the ITAE is only a sum of 5 numbers. So, it does not mean the performance gets better after 1.2 s in Fig. 20. Also, we note that the ITAE index of the IOPID controller is more than ten times larger than that of the FOPD controller. Therefore, from the performance index, the robustness of the FOPD controller against large sampling period is significantly better than that of the IOPID controller.

7 Flight tests

A set of drone flight tests have been done when there is nearly no wind. The drone was first flown to 10 m high manually. Then we switched the drone to automatic mode and ran the controller script. The drone's position has been logged for one minute for each controller. Then we used a circle to cover all the dots as shown in Figs. 22, 23. The dots from PD$^{\lambda}$ controller are within a smaller circle (radius = 58 pixels) than that from the IOPID controller system (radius = 68 pixels). In this way, we can see the PD$^{\lambda}$ controller has a nearly 20% better performance compared to the IOPID controller

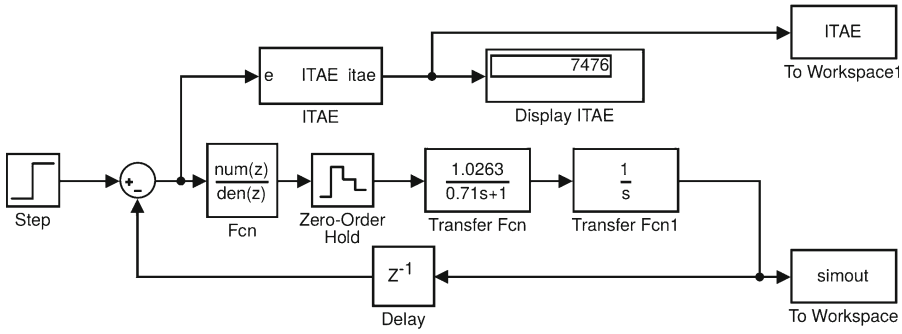


Fig. 19 Computing ITAE index with Simulink

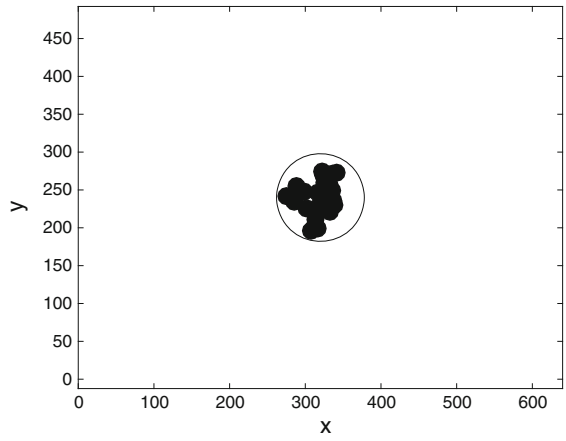


Fig. 22 Quadrotor positions logged in the flight test with the FOPD controller

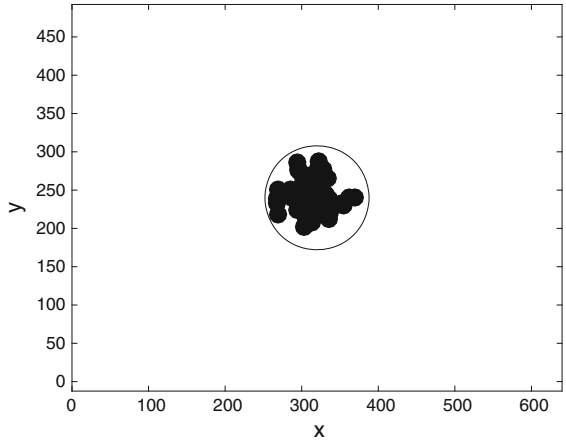


Fig. 23 Quadrotor positions logged in the flight test with the IOPID controller

in flight tests. A sample flight test video can be accessed from https://youtu.be/yNwVlw9zY3k.

8 Conclusion and future works

In this paper, a specific case of fractional-order flight control has been investigated. Plant model has been identified with MATLAB System Identification Toolbox. An integer-order PID controller and a PD^{λ} controller have been designed with the same crossover frequency and phase margin. Step response analysis and Bode plot analysis of the two controllers show that the PD^{λ} controller has advantages in rising time, overshoot and robustness against plant variation. A stability criteria, optimization method, graphic method and parallel computing technique have been used to estimate the two controllers' largest sampling period, which proves that the FOC can tolerate largest sampling period than the integral-order controller (IOC). Linear model simulation supports the same conclusion. Flight tests show that the FOC hovers the drone more precisely than the IOC. Future works include improving the FOC tuning rule and discretization methods.

Acknowledgements Thank Bo Shang's wife Juyao Dong's support on taking care of their little daughter. Thank Bo Shang's parents' financial support to keep this research going on.

Funding This work was supported in part by Xihua University, China Scholarship Council (CSC), Shenyang City Foundation #17-87-0-00, the National Natural Science Foundation of China under Grant nos. 61701101, U1713216, 61803077, the National Key Robot Project under Grant nos. 2017YFB1300900, 2017YFB1301103, and the Fundamental Research Fund for the Central Universities of China N172603001, N181602014, N172604004, N172604003, and N172604002.

Compliance with ethical standards

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

References

1. Luo, Y., Chao, H., Di, L., Chen, Y.Q.: Lateral directional fractional order PI^{α} control of a small fixed-wing unmanned aerial vehicles: controller designs and flight tests. IET Control Theory Appl. **5**(18), 2156–2167 (2011)
2. Shang, B., Liu, J., Zhao, T., Chen, Y.Q.: Fractional order robust visual servoing control of a quadrotor UAV with larger sampling period. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1228–1234. IEEE (2016)
3. Shang, B., Wu, C., Zhang, Y., Chen, Y.Q.: Analysis of maximum possible sampling period for a real-time vision-based control system. In: ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 1–7. American Society of Mechanical Engineers (2017)
4. Shang, B., Wu, C., Zhang, Y., Chen, Y.Q.: Code for "Analysis of maximum possible sampling period for a real-time vision-based control system". http://bit.ly/max-sampling-period. Accessed 06 Nov 2018
5. Podlubny: Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and some of Their Applications, vol. 198. Elsevier, Amsterdam (1998)
6. Miller, K.S.: Derivatives of noninteger order. Math. Mag. **68**(3), 183–192 (1995)
7. Caputo, M.: Linear models of dissipation whose Q is almost frequency independent. Ann. Geophys. **19**(4), 383–393 (1966)

8. Luo, Y., Chen, Y.Q.: Fractional order [proportional derivative] controller for a class of fractional order systems. Automatica **45**(10), 2446–2450 (2009)
9. Li, H.S., Luo, Y., Chen, Y.Q.: A fractional order proportional and derivative (FOPD) motion controller: tuning rule and experiments. IEEE Trans. Control Syst. Technol. **18**(2), 516–520 (2010)
10. Kao, C.-Y., Lincoln, B.: Simple stability criteria for systems with time-varying delays. Automatica **40**(8), 1429–1434 (2004)
11. Shang, B., Liu, J., Zhang, Y., Wu, C., Chen, Y.Q.: Drone code for visual hovering. https://github.com/cnpcshangbo/vision-hovering. Accessed 03 Aug 2018
12. Mackay, R.: Red balloon finder. https://github.com/rmackay9/ardupilot-balloon-finder. Accessed 30 July 2018
13. Fliess, M., Join, C.: Model-free control and intelligent pid controllers: towards a possible trivialization of nonlinear control? *IFAC Proc. Vol.* **42**(10), 1531–1550. 15th IFAC Symposium on System Identification (2009)
14. Zhang, X., Wang, H., Tian, Y., Peyrodie, L., Wang, X.: Model-free based neural network control with time-delay estimation for lower extremity exoskeleton. Neurocomputing **272**, 178–188 (2018)
15. Ahmed, S., Wang, H., Tian, Y.: Model-free control using time delay estimation and fractional-order nonsingular fast terminal sliding mode for uncertain lower-limb exoskeleton. J. Vib. Control **24**(22), 5273–5290 (2018)
16. Mathworks. Pid tuning algorithm for linear plant model. https://www.mathworks.com/help/control/ref/pidtune.html. Accessed 13 Mar 2019
17. Åström, K.J., Hägglund, T.: Advanced PID control. ISA-The Instrumentation, Systems, and Automation Society; Research Triangle Park, NC 27709 (2006)
18. Dorf, R., Bishop, R.: Modern Control Systems, 8th edn. Books by Marquette University Faculty (1998)
19. Simulink library Performance index. http://bit.ly/performance-index. Accessed 06 Nov 2018