www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

ARCHIVES OF ELECTRICAL ENGINEERING VOL. 68(3), pp. 565–577 (2019)

DOI 10.24425/aee.2019.129342

Problems in solving fractional differential equations in a microcontroller implementation of an FOPID controller

MARIUSZ MATUSIAK, PIOTR OSTALCZYK

Lodz University of Technology
Institute of Applied Computer Science
Faculty of Electrical, Electronic, Computer and Control Engineering
Stefanowskiego 18/22, 90-924 Łódź, Poland
e-mail: mmatusiak@iis.p.lodz.pl, postalcz@p.lodz.pl

(Received: 23.11.2018, revised: 27.03.2019)

Abstract: The article focuses on the fractional-order backward difference, sum, linear time-invariant equation analysis, and difficulties of the fractional calculus microcontroller implementation with regard to designing a fractional-order proportional integral derivative (FOPID) controller. In opposite to the classic proportional integral derivative (PID), the FOPID controller is defined by five independent parameters. Hence, it is more customizable and, potentially, more precise on condition that the values of fractional integration and differentiation orders are properly selected. However, a number of operations and the time required to calculate the output signal continuously increase. This can be a significant problem considering the limitations of a microcontroller, including memory size and a constant sampling time of the set-up analog-to-digital (ADC) converters. In the article, three solutions are considered, and results obtained in the experiments are presented.

Key words: fractional calculus, Grünwald-Letnikov fractional-order backward difference/sum, FOPID, hardware implementation

1. Introduction

The fractional calculus [1, 2] has become more and more effective, and hence popular mathematical tool in the synthesis and analysis [3] of dynamic systems in numerous fields of science including signal processing, control, biomedical and electrical engineering [4–8]. In control applications, achieving better models of real objects and developing more sophisticated and precise

© 2019. The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (CC BY-NC-ND 4.0, https://creativecommons.org/licenses/by-nc-nd/4.0/), which permits use, distribution, and reproduction in any medium, provided that the Article is properly cited, the use is non-commercial, and no modifications or adaptations are made.

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

control procedures requires advanced computational tools. This statement also applies to the analysis of fractional-order system dynamics. The main concern brought by the implementation of these strategies is associated with a problem of the constantly, linearly growing over time number of multiplications and additions, known as the growing calculation tail [8]. This may be a source of considerable errors in real-time microcontroller-based applications that perform critical operations during a constant sampling time and save results in a limited memory [6]. Several approaches to implementing fractional-order operators have been proposed. Two popular are the Oustaloup approximation [9] and the Laguerre impulse response approximation (LIRA) [10]. Applications of the former were discussed among others in the works [5, 11–13] regarding non-integer order digital filters. The method allows one to approximate non-integer filters with a wider spectrum than LIRA; however, it is considered sensitive to the process of discretization at high sampling frequencies and may cause system instability. A step towards implementing Oustaloup approximation algorithms in a fractional-order PID controller [14] was made by Tepljakov in his praised work on a fractional-order modeling toolbox - FOMCON [15]. In the project, a performance comparison between an 8-bit AVR microcontroller and 32-bit ARM Cortex-M4 microcontroller, similar to the one used in the authors' research, was also conducted.

In the following paper, an approach based on the Grünwald-Letnikov definition of the differintegral and the Short Memory Principle optimization technique introduced by Podlubny [2] is considered. Usefulness of this method has been verified, among others, in air and plant heating control and robotic arm movement optimization in [4, 6, 8].

The structure of the paper is organized as follows: in Section 2, mathematical preliminaries including the Grünwald-Letnikov definition of the discrete fractional-order backward difference and a sum based on the oblivion function [2, 16] are introduced. Two solutions for the previously described problem are highlighted. In Section 3, the discrete fractional-order proportional-integral-derivative (FOPID) controller is considered [14]. An analysis of the implementation difficulties was conducted using the chosen Cortex-M7-based hardware testing platform. The methodology and the obtained results are described in Section 4. In Section 5, accuracies of the classic and fractional-order PID controllers [17] are compared. The impact of the difficulties on a control process is also discussed.

2. Grünwald-Letnikov definition of fractional-order backward difference/sum

Let us use the following notation $N_c = \{c, c{+}1, c{+}2, \ldots\}$, then $N := N_0 = \{0, 1, 2, 3, \ldots\}$. The discrete function of a variable $k \in N$ and a given fractional-order v, matching the inequality $0 < v \leq 1$, is defined by its values $a^{(v)}(k)$ (1):

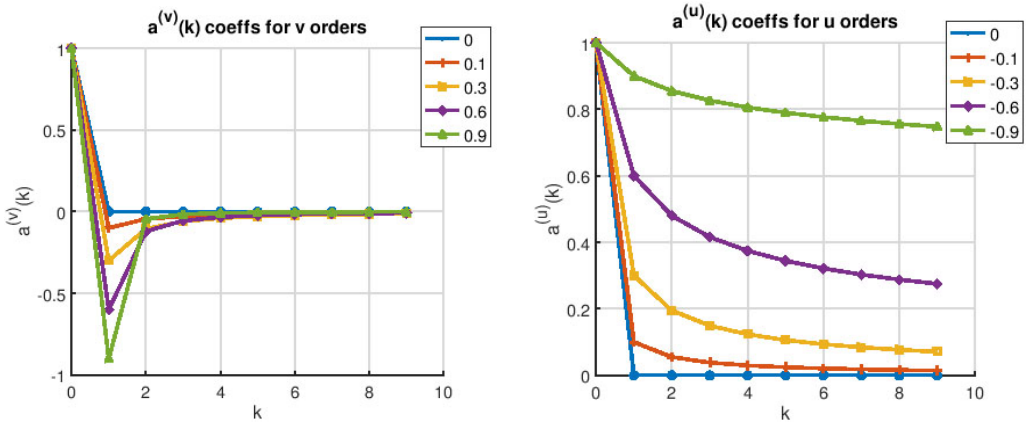
$$a^{(v)}(k) = \begin{cases} 1 & \text{for } k = 0 \\ (-1)^k \dfrac{v(v-1)\ldots(v-k+1)}{k!} & \text{for } k = N_1 \end{cases} . \tag{1}$$

The above formula is called the *oblivion function* and is a basis for evaluating the fractional-order backward difference and fractional-order backward sum. A recursive equivalent of Equa-

tion (1) is defined by:

$$a^{(v)}(k) = \begin{cases} 1 & \text{for } k = 0 \\ a^{(v)}(k-1)\left(1 - \dfrac{v+1}{k}\right) & \text{for } k = N_1 \end{cases} . \tag{2}$$

The coefficients calculated for various positive v orders are presented in Fig. 1(a).



(a) (b)

Fig. 1. $a^{(v)}(k)$ coefficients for positive values of order $v \in \{0, 0.1, \ldots, 0.9\}$ (a) and for negative values of order $\mu \in \{0, -0.1, \ldots, -0.9\}$ (b)

Using the *oblivion function* one can calculate fractional-order backward difference of an order $v \in R_+$ (3):

$${}^{GL}\Delta_h^{(v)} f(t) = \sum_{k=0}^{\infty} a^{(v)}(k)\, f(t - kh) \quad \text{for } t,\, h \in R. \tag{3}$$

where $f(t)$ is the discrete-time real-valued function defined by its consecutive samples $f(kh)$ and h denotes the sampling time. The fractional-order backward sum is described by a similar equation, with the difference that the fractional-order takes a negative value, e.g. $\mu = -v$. A plot of μ-order coefficients is shown in Fig. 1(b). One may also generalize Equations (1), (2) by evaluating the v order for each time instance kh as a result of a uniquely defined discrete function $v(k)$. This generalization is referenced as the variable-, fractional-order backward difference/sum [8, 18].

The one-sided $\mathcal{Z}$ transform of the mentioned difference takes the following form (4):

$$z\left\{{}^{GL}\Delta_h^{(v)} f(t)\right\} = \left(1 - z^{-1}\right)^v . \tag{4}$$

The left-sided Grünwald-Letnikov derivative is then defined by Formula (5) [19]:

$${}^{GL}D_h^{(v)} f(t) = \lim_{h \to 0^+} \frac{{}^{GL}\Delta_h^{(v)} f(t)}{h^v} \quad \text{for } t,\, h \in R_+ . \tag{5}$$

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Assuming that the sampling time h is constant and as small as possible, (5) can be simplified to:

$$ {}^{GL}D_h^{(v)} f(t) \approx \frac{{}^{GL}\Delta_h^{(v)} f(t)}{h^v} \quad \text{for } t,\, h \in R_+ . \tag{6} $$

If a positive value of the order v is replaced in Equations (3), (5), (6) by a negative value of μ then the definitions of the Grünwald-Letnikov fractional-order sum and integral are considered.

The definitions are characterized by the growing number of calculations at every step. If M represents the number of operations performed by a microcontroller in the first step then in the k-th step kM operations should be executed. The issue arises when the time required to perform these operations is longer than the desired sampling period h ($t_{kM} > h$). One of the possible solutions is to assume that for a specific value of L for which $t_{LM} \geq h$ and $k > L$, the number of operations is limited to LM. Formula (1) then takes the form of:

$$ a^{(v)}(k) = \begin{cases} 0 & \text{for } k < 0 \\ 1 & \text{for } k = 0 \\ (-1)^k \dfrac{v(v-1)\ldots(v-k+1)}{k!} & \text{for } 0 < k \leq L \\ (-1)^L \dfrac{v(v-1)\ldots(v-L+1)}{L!} & \text{for } k > L \end{cases} . \tag{7} $$

The second solution is based on the assumption that for small order values the coefficients drop rapidly to zero, as shown in Fig. 1, and further operations can be skipped on condition that a required precision of results has been reached:

$$ a^{(v)}(k) = \begin{cases} 0 & \text{for } k < 0 \\ 1 & \text{for } k = 0 \\ (-1)^k \dfrac{v(v-1)\ldots(v-k+1)}{k!} & \text{for } 0 < k \leq L \\ 0 & \text{for } k > L \end{cases} . \tag{8} $$

3. Fractional-order PID controller

The fractional-order PID controller for a closed-loop system (CLS) [14] is described by Equation (9):

$$ y(kh) = K_P u(kh) + K_I \left({}_0^{GL}\Delta_h^{(\mu)} \right) u(kh) + K_D \left({}_0^{GL}\Delta_h^{(v)} \right) u(k\,h), \tag{9} $$

where K_P, K_I, and K_D are the proportional, integral and derivative gains, $\mu < 0$, $v > 0$ are the fractional orders, and $u(kh)$ and $y(kh)$ are the input (error) and output signals, respectively.

Applying Formula (3) results in the form:

$$y(kh) = K_P[1 \;\; 0 \;\; \ldots \;\; 0]\begin{bmatrix} u(kh) \\ u(kh-h) \\ \vdots \\ u(0h) \end{bmatrix} + K_I\left[1 \;\; a^{(\mu)}(1) \;\; \ldots \;\; a^{(\mu)}(k)\right]\begin{bmatrix} u(kh) \\ u(kh-h) \\ \vdots \\ u(0h) \end{bmatrix} + \\ + K_D\left[1 \;\; a^{(\nu)}(1) \;\; \ldots \;\; a^{(\nu)}(k)\right]\begin{bmatrix} u(kh) \\ u(kh-h) \\ \vdots \\ u(0h) \end{bmatrix}. \tag{10}$$

The discrete transfer function of (10) is obtained by the one-sided $\mathbb{Z}$-transform (assuming zero initial conditions):

$$\frac{Y(z)}{U(z)} = \frac{K_P(1-z^{-1})^{-\mu} + K_I + K_D(1-z^{-1})^{\nu-\mu}}{(1-z^{-1})^{-\mu}}. \tag{11}$$

In an ideal controller, the degree of the numerator is higher than the degree of the denominator. In real applications, the differentiator strongly amplifies noise that is always present in a signal, so usually, the differentiator with inertia, acting as a discrete filter, is used. The transfer function then takes the form:

$$\frac{Y(z)}{U(z)} = K_P + \frac{K_I}{(1-z^{-1})^{-\mu}} + \frac{K_D(1-z^{-1})^{\nu}}{(1-z^{-1})^{\nu} + a_0}. \tag{12}$$

4. Hardware testing platform

For the analysis of fractional-order operator implementation, a hardware testing platform based on the ARM Cortex-M7 STM32F746ZG microcontroller [20] was prepared. Key features and peripherals of the device are listed in Table 1.

Table 1. Hardware platform parameters

Parameter name	Symbol	Value
CPU main clock frequency	F_{CPU}	up to 216 MHz
Memory	Flash, SRAM	1 024 MB (Flash) + 320 KB (SRAM)
Converters	ADC, DAC	3 × 12-bit 2.4 MSPS ADC, 2 × 12-bit DAC
Power supply	V_{DD}	1.8–3.6 V (3.3 V set)
Other features		floating point unit real-time accelerator, DSP instructions

Several limitations must be taken into consideration when implementing the mentioned algorithms. First of all, the desired precision of numbers must be determined. For the fractional order

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

$v =\in (0,\ 1]$ and $k \to \infty$, $a^{(v)}(k)$ coefficients get closer to zero and for large values of k, the coefficients might not be represented properly in a microcontroller memory. The GCC compiler for STM32F746ZG supports two ANSI C floating-point types: a single-precision 4-bytes *float* type and a double-precision 8-bytes *double* type [21]. Ranges are [1.17549435e−38, 3.40282347e+38] with 8 decimal points and [2.22507385850720138e−308, 1.79769313486231571e+308] with 17 decimal points respectively. The maximum positive value of the type is important for implementation of Formula (1) due to the growing factorial value and a possible risk of overflow. Otherwise, recursive Formula (2) needs to be used.

Choosing the more precise floating-point type implies consuming more memory which is the second important limitation to consider. If $a^{(v)}(k)$ coefficients are not defined directly in a program source code, and thus not saved in a Flash section of memory, they need to be calculated at runtime and saved in RAM. In 320KB of the STM32F746ZG SRAM, up to 40 960 *double* type coefficients could be stored; however, the section is always shared by other parts of the program including the stack, heap, initialization routines, and other data. An exemplary C-language program for evaluating *double* type $a^{(v)}(k)$ coefficients using recursive Formula (2) on the STM32F746ZG revealed that up to 8 084 coefficients could be calculated correctly, consuming 64 672 bytes of RAM.

The second hardware implementation problem, mentioned in Section 2, is the growing time of *CPU* calculations which in some circumstances can exceed the sampling time of the ADC converter. The maximum frequency of the ADC domain clock in the STM32F746ZG is $f_{\mathrm{ADC}} = 36$ MHz [20], and the single 12-bit conversion takes $c_{\mathrm{ADC}} = 15$ cycles to complete. Hence, the shortest possible sampling time can be estimated using Formula (13):

$$Ts_{\min} = \frac{1}{f_{\mathrm{ADC}}} c_{\mathrm{ADC}} = \frac{1}{36\ \mathrm{MHz}} 15 = 0.417\ \mu\mathrm{s}. \tag{13}$$

Usually, one of the available general-purpose timers is used to extend the time and to synchronize the conversions. To configure a timer one must know the exact time of critical operations performed during an ADC interrupt service routine (ISR). The ARM Cortex peripheral called Data Watchpoint and Trace (DWT) [20] allows one to measure the number of *CPU* cycles between two lines of the program. Knowing the *CPU* frequency, one can determine the time of specifics sets of operations (14):

$$t_{\mathrm{diff}} = \frac{c_{\mathrm{after\_}} - c_{\mathrm{before\_diff}}}{f_{CPU}}. \tag{14}$$

This was tested with a C program running the algorithm of a discrete fractional-order differentiator (6) for 11 different coefficients lengths between 1 and 5 000. The results are presented in Fig. 2.

For $k = 3\ 000$ coefficients already stored in the memory, the number of cycles required to calculate the response was equal to $c_{\mathrm{diff}} = 189\ 345$, and lasted for (15):

$$t_{\mathrm{diff}} = \frac{c_{\mathrm{diff}}}{f_{CPU}} = \frac{189\ 345}{216\ \mathrm{MHz}} = 0.87\ \mathrm{ms}. \tag{15}$$

The time between subsequent ADC conversions has to be longer than the sum of t_{diff} and h_{min}, so in this example, the timer configured to generate interrupt signals at 1 kHz fulfilled the condition.

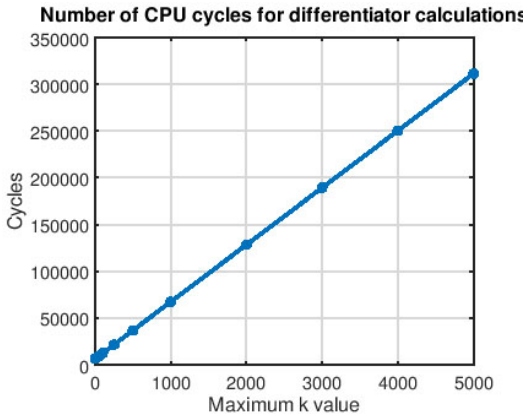


Fig. 2. The number of *CPU* cycles required to calculate the Grünwald-Letnikov derivative as a function of the buffer length k

To increase the overall microcontroller performance, one may consider rewriting critical sections of a program using inline assembly code. If a chosen device supports single instruction, multiple data (SIMD) extensions, then basic operations including multiply and accumulate (MAC) are executed in one clock cycle, significantly reducing time of complex algorithms. The performance of STM32 devices, using the optimized DSP instructions, was compared in the sheet [22].

5. Fractional-order PI$^{\mu}$D$^{\nu}$ implementation

In order to implement discrete PID and PI$^{\mu}$D$^{\nu}$ controllers [14], a plant had to be defined first. To indicate limitations of the microcontroller, unit step response of the controlled system had to have relatively wide transient characteristics. A DC motor, in general, is an example of a device that can meet that criterion; thus, the 6 V-powered DC motor with an encoder was used for the reference [23]. The unit step was simulated by turning the motor power supply on and measuring the number of encoder impulses every ten milliseconds. A graph of the normalized rotations per second is presented in Fig. 3 (thick red line).

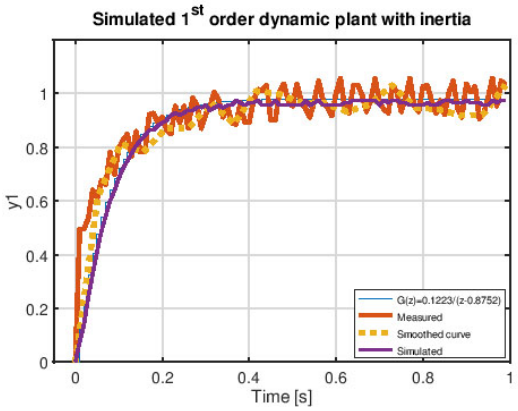


Fig. 3. Speed curve of the DC motor (thick red), step response of the defined transfer function (blue), and the measured microcontroller response (purple)

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Based on the diagram, an exemplary 1st-order dynamic plant with inertia was defined, described by transfer function (16):

$$G(s) = \frac{k_M}{T_{,M}s + 1} = \frac{0.98}{0.075s + 1}, \tag{16}$$

where k_M is the gain of the plant and T_M is the time constant. Using Tustin's method [24] for the sampling time $h = 0.01$ s, the above can be transformed into:

$$G(z) = \frac{0.1223z^{-1}}{1 - 0.8752z^{-1}}, \tag{17}$$

which is the equivalent of the difference equation:

$$y[n] = 0.1223x[n-1] + 0.8752y[n-1]. \tag{18}$$

The simulated 1st-order dynamic plant was implemented in a C program for the ARM Cortex-M3 core STM32L152RCT6 [25] microcontroller with ADC and DAC converters configured. The obtained unit step response is presented in Fig. 3 (thick purple line).

A general discrete PID controller is described by the following Formula (19):

$$u[kh] = Kp(e[kh] + \frac{h}{T_I}\sum_{i=0}^{k} e[ih] + \frac{e[kh] - e[kh-h]}{h}T_D, \tag{19}$$

where h is the sampling time, K_P is the proportional gain, and T_I and T_D are the integral and derivative times respectively. The implemented controller was tuned using the Ziegler-Nichols method [26]. Parameters are presented in Table 2.

Table 2. PID controller parameters for the modeled system

Parameter name	Value
K_P	2
T_I	0.028
T_D	0.007

Both the Matlab simulation and C program implementation of the controller were carried out. The target setpoint value of the regulator was set programmatically to 1 (corresponding to 1 V on the DAC output). The output of the STM32F746ZG DAC converter (PID) was connected directly to the input of the STM32L152RCT6 ADC converter (1st-order plant), and the STM32L152RCT6 DAC output as the negative feedback to the STM32F746ZG ADC (see Fig. 4). The converters were in sync with 100 Hz peripheral timers. Maximum (3.3 V) and minimum (0 V) control signal limits were also applied. Characteristics of the simulation and measured device output are presented in Fig. 5(a).

In a typical PID implementation, a total number of calculations performed at each step is constant. This is the outcome of Algorithm 1 where only two additional variables are saved in

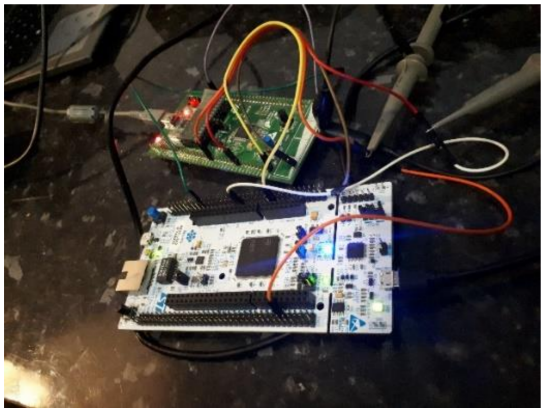


Fig. 4. Connected STM32F746ZG (PID) and STM32L152RCT6 (plant) microcontrollers

memory. The first variable *errors_sum* represents the sum of all previous errors multiplied by the sampling time (the integral term), while the second *previous_error* keeps the value of the error from the last step (the derivative term)

```
errors_sum     <- 0
previous_error <- 0
setpoint       <- setpoint value
loop:
    error       <- setpoint - measured_error
    errors_sum  <- errors_sum + error * h
    derivative  <- (error - previous_error) / h
    output      <- Kp * error + Ki * errors_sum
                   + Kd * derivative
    previous_error <- error
    sleep(h)
    goto loop
```

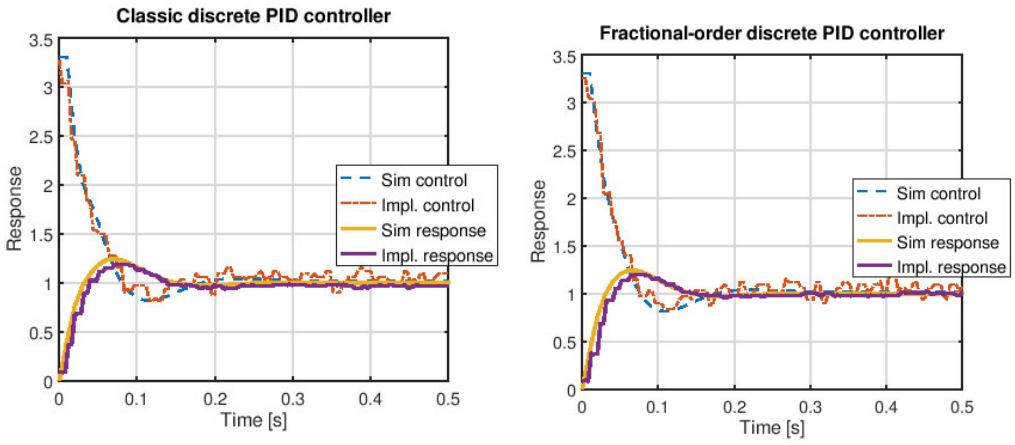
Algorithm 1. Classic discrete PID controller

In the fractional-order PID controller, according to Equation (10), both derivative and integral terms are sums of products of errors $e(k)$ and the corresponding backward difference coefficients, divided by the v-power of the sampling time h. As mentioned in Section 4, this may cause an undefined behavior of a microcontroller program if calculations are not properly limited.

To analyze the problem thoroughly, the fractional-order PID (FOPID) algorithm was implemented on the STM32F746ZG and several iterations of memory consumption and performance tests were executed. Fractional order values $v = 0.94$ and $u = -1.0$ were selected to minimize integral of the squared error (ISE criterion). The sampling time was set to $h = 0.01$ s (100 Hz). The $a^{(v)}(k)$ and $a^{(\mu)}(k)$ coefficients of type *float* were calculated during the initialization routine and stored in SRAM in variable-length arrays. The maximum possible length of the arrays equaled 10 200.

The computation time of the FOPID controller output signal as a function of the array length is shown in Fig. 6.

If a chosen sampling frequency *fs* is too high for the desired buffer length, e.g. $fs = 1$ kHz for 4 000 samples, the controller program will skip upcoming ADC conversions, and, therefore, generate a wrong control signal. Calculation time can increase even further when the varying



(a) (b)

Fig. 5. Simulated and measured plant responses for (a) classic discrete PID controller and (b) fractional-order discrete $PI^{\mu}D^{\nu}$ controller

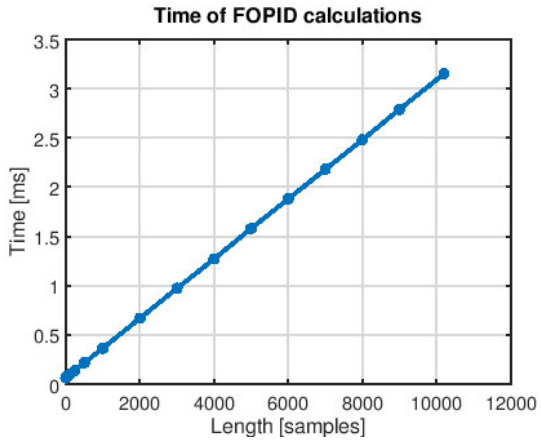


Fig. 6. Time of FOPID output calculations as a function of the array length

fractional order $\nu(\cdot)$ in a variable-, fractional-order PID (VFOPID) controller is considered [8]. The oblivion coefficients are then recalculated in each ISR.

Depending on the selected fractional-order values, the application of solution (7) may cause a major problem related to shifting errors stored in memory. When the error buffer limit is reached, the first value, being also the largest at the start of the control, is overridden resulting in a temporary rapid drop of the signal, caused mainly by the integral term of the FOPID. An example of this issue registered for a buffer length of 30 is presented in Fig. 7.

At the expense of more complex implementation, the fractional-order PID control provides more tuning parameters; hence, much more flexibility in designing a controller best suitable for a particular system. Measured responses for three different control signals: unit step, classic PID and fractional-order PID are presented in Fig. 8. For the mentioned fractional-order values $\nu = 0.94$

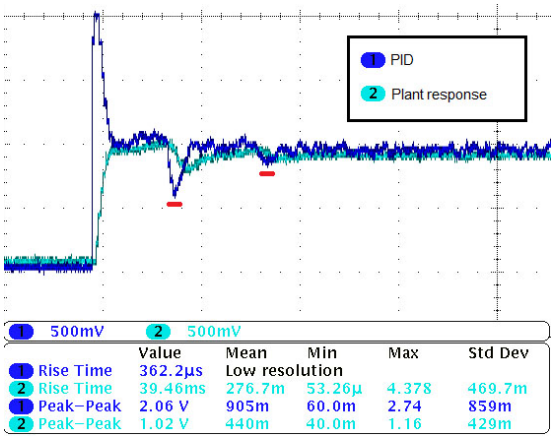


Fig. 7. Shifting error values in the discrete FOPID implementation

and $\mu = -1.0$, integral of the squared error (ISE) for the simulation time of 1 s was slightly reduced compared to the classic PID defined by the same set of K_P, T_I, T_D and h parameters. Moreover, for $-0.9 > \mu > -1$ rise time of the plant response was even faster; however, a non-zero steady state error was then produced.

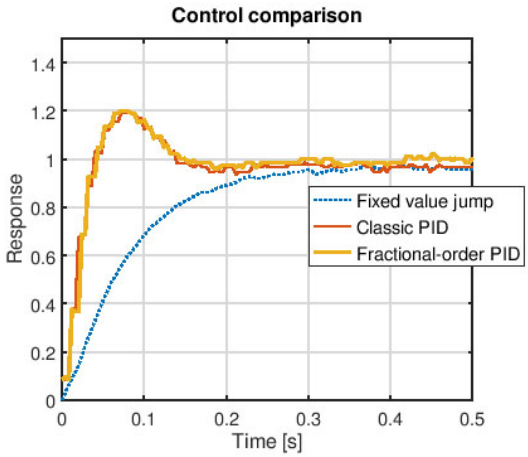


Fig. 8. Dynamic system responses for different control signals

This problem may be eliminated with a VFOPID controller in which the value of the μ-order gradually approaches -1. The stability criterion for the controller, tuned using the same parameters as the classic PID, is defined similarly [8]:

$$v(k) = \begin{cases} f(k) & \text{for } 0 \le k < L < \infty \\ 1 & \text{for } k > L \end{cases} . \tag{20}$$

Choosing the valid μ and v values remains an open problem and strictly depends on properties of the considered closed-loop system.

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

6. Conclusions

Applying fractional calculus to the field of control engineering introduces more degrees of freedom [4] which, for accurately selected and adaptive values of fractional orders, provides the ability to design more precise and robust control algorithms. However, in order to eliminate the steady-state error present in a FOPID, an integer integration order value is eventually required. One must also be aware of facing additional difficulties when it comes to the microcontroller implementation, including memory size limits and the growing number of calculations. Some optimized algorithms affecting the controller accuracy might be necessary. The first possible approach would be to limit the number of operations or skip further calculations if the obtained results reach the desired precision. Depending on the selected fractional orders, the controller may then react sensitively on every rapid setpoint change. The second approach would be to increase the performance of a microcontroller by implementing a program using an inline assembly code and optimized DSP instructions. The question is particularly important when a variable-, fractional-order PID is considered.

Acknowledgements

This work was supported by Polish funds of the National Science Center under grant 2016/23/B/ST7/03686.

References

[1] Fossard A.J., Normand-Cyrot D., Eds., *Nonlinear Systems: Modelling and estimation*, Chapman & Hall (1995).

[2] Podlubny I., *Fractional Differential Equations. An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and some of their Applications*, Academic Press (1999).

[3] Miller K., Ross B., *An Introduction to the Fractional Calculus and Fractional Differential Equations*, John Wiley & Sons, Inc. (1993).

[4] Dziwiński T., Bauer W., Baranowski J., Piątek P., Zagórowska M., *Robust non-integer order controller for air heater*, 19th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland (2014).

[5] Baranowski J., Bauer W., Zagórowska M., Piątek P., *On Digital Realizations of Non-integer Order Filters*, Circuits, Systems, and Signal Processing, vol. 35, no. 6, pp. 2083–2107 (2016).

[6] Petras I., Vinagre B., *Practical application of digital fractional-order controller to temperature control*, Acta Montanistica Slovaca, vol. 7, no. 2, pp. 131–137 (2002).

[7] Kawala-Janik A., Podpora M., Gardecki A., Czuwara W., Bauer W., Baranowski J., *Game Controller Based on Biomedical Signals*, 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland (2015).

[8] Ostalczyk P., Brzeziński D., Duch P., Łaski M., Sankowski D., *The variable, fractional-order discrete-time PD controller in the IISv1.3 robot arm control*, Central European Journal of Physics, vol. 11, no. 6, pp. 750–759 (2013).

[9] Oustaloup A., Levron F., Mathieu B., Nanot F.M., *Frequency-Band Complex Noninteger Differentiator: Characterization and Synthesis*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 47, no. 1, pp. 25–39 (2000).

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

[10] Maione G., *Laguerre approximation of fractional systems*, Electronics Letters, vol. 38, no. 20, pp. 1234–1236 (2002).

[11] Baranowski J., Bauer W., Zagórowska M., Dziwiński T., Piątek P., *Time-domain Oustaloup approximation*, 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland (2015).

[12] Dziwiński T., Piątek P., Baranowski J., Bauer W., Zagórowska M., *On the Practical Implementation of Non-integer Order Filters*, 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland (2015).

[13] Bauer W., *Implementation of Non-integer $PI^{\lambda}D^{\mu}$ Controller for The ATmega 328P Micro-controller*, 21st International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland (2016).

[14] Podlubny I., Dorcak L., Kostial I., *On Fractional Derivatives, Fractional-Order Dynamic Systems and $PI^{\lambda}D^{\mu}$-controllers*, Proceedings of the 36th IEEE Conference on Decision and Control, San Diego, California, USA pp. 4985–4990 (1997).

[15] Tepljakov A., *Fractional-order Modeling and Control of Dynamic Systems*, Springer Theses (2017).

[16] Ostalczyk P., *Discrete Fractional Calculus: Applications in Control and Image Processing*, World Scientific (2016).

[17] Astrom K., Hagglund T., *PID controllers: Theory, design and tuning*, Instrument Society of America (1995).

[18] Ostalczyk P., Duch P., Brzeziński D., Sankowski D., *Order Functions Selection in the Variable-, Fractional-Order PID Controller*, Advances in Modelling and Control of Non-integer Order Systems, Opole, Poland, pp. 159–170 (2014).

[19] Kilbas A., Srivastava H., Trujillo J., *Theory and Applications of Fractional Differential Equations*, Elsevier Science (2006).

[20] STMicroelectronics, *STM32F745xx STM32F746xx. Datasheet – production data*, STMicroelectronics (2015).

[21] Arm Limited, *Compiler User Guide. Basic data types in ARM C and C++*, http://www.keil.com/support/man/docs/armcc/armcc_chr1359125009502.htm., accessed November 2018.

[22] STMicroelectronics, *AN4841, Application note: Digital signal processing for STM32 microcontrollers using CMSIS*, STMicroelectronics (2018).

[23] Pololu H.P., *6V Motor with 48 CPR Encoder for 25D mm Metal Gearmotors (No Gearbox)*, https://www.pololu.com/product/2280, accessed November 2018.

[24] Lyons R.G., *Understanding Digital Signal Processing*, Prentice Hall (1996).

[25] STMicroelectronics, *STM32L15xCC STM32L15xRC STM32L15xUC STM32L15xVC Datasheet – production data*, STMicroelectronics (2017).

[26] Ziegler J., Nichols N., *Optimum Settings for Automatic Controllers*, Transactions of the A.S.M.E., vol. 64, pp. 759–768 (1942).