entropy

MDPI

Article

Optimal Digital Implementation of Fractional-Order Models in a Microcontroller

Mariusz Matusiak, Marcin Bąkała * and Rafał Wojciechowski

Institute of Applied Computer Science, Łódź University of Technology, ul. Stefanowskiego 18/22, 90-924 Lodz, Poland; mmatusiak@iis.p.lodz.pl (M.M.); rafal.wojciechowski@p.lodz.pl (R.W.)

* Correspondence: marcin.bakala@p.lodz.pl

Received: 9 March 2020; Accepted: 19 March 2020; Published: 23 March 2020

Abstract: The growing number of operations in implementations of the non-local fractional differentiation operator is cumbersome for real applications with strict performance and memory storage requirements. This demands use of one of the available approximation methods. In this paper, the analysis of the classic integer- (IO) and fractional-order (FO) models of the brushless DC (BLDC) micromotor mounted on a steel rotating arms, and next, the discretization and efficient implementation of the models in a microcontroller (MCU) is performed. Two different methods for the FO model are examined, including the approximation of the fractional-order operator s^{ν} ($\nu \in \mathbb{R}$) using the Oustaloup Recursive filter and the numerical evaluation of the fractional differintegral operator based on the Grünwald–Letnikov definition and Short Memory Principle. The models are verified against the results of several experiments conducted on an ARM Cortex-M7-based STM32F746ZG unit. Additionally, some software optimization techniques for the Cortex-M microcontroller family are discussed. The described steps are universal and can also be easily adapted to any other microcontroller. The values for integral absolute error (IAE) and integral square error (ISE) performance indices, calculated on the basis of simulations performed in MATLAB, are used to evaluate accuracy.

Keywords: fractional calculus; Grünwald–Letnikov differintegral; BLDC motor model; microcontroller implementation

1. Introduction

Optimal solutions for implementing models of plants are of great interest to industry, since they enable the extension of the computer-aided simulations performed in computation software such as MATLAB/Simulink. In numerous control systems, it is essential that the process of tuning the controller on the basis of measurements of system output involves as few costly plant identification iterations as possible. A common solution involves synthesizing the plant model from the measured characteristics and implementing it in a dedicated software environment or microprocessor-based hardware platform. Significant difficulties arise when the models are described by fractional-order calculus (FOC) [1–3]. Much research on fractional-order control systems uses computational software for analysis and simulations. A noticeably smaller proportion addresses the problem of digital implementation of FOC equations on real devices, not only theoretically, but also practically [4–8]. In contrast to the well-known bounded numerical approximations of a classic integer-order derivative, such as backward or central differences, the problem arises of a constantly increasing number of discrete convolution operations over time. In order to reduce the negative impact of this issue, numerous approximation methods have been proposed [9–12], divided between the time-domain and frequency-domain. In the time domain, limited memory-based approaches are the most popular, including the Short Memory Principle (SMP) algorithm introduced by Igor Podlubny [3]. In the frequency domain, a selected range

of Bode characteristics $G(\omega)$ can be approximated using the well-known Oustaloup Recursive filter algorithm (ORA) [10] or modifications thereof. Hardware implementation of fractional order models in control engineering is of great interest for the purposes of offline controller tuning and testing in already developed industrial control systems, which must not be affected in any way. An accurate, equivalent mathematical model realized as a dedicated hardware platform is usually desired. It can be very useful in such cases to use a microcontroller (MCU) as the target plant, with the output signal calculated on the basis of a transfer function of its model. Alternatively, one can consider designing an equivalent, time-continuous fractance circuit. In our research, we focus on the former approach, touching on the problem of discretization of the fractional-order model and combining approximation methods with universal optimization programming techniques to improve performance, reduce computation time, and limit the size of the occupied microcontroller memory with a negligible impact on accuracy. As a rule of a thumb, optimal implementation allows a higher order N of approximation formulas, producing more complex but accurate equations, which can be computed during the same constant sampling period. The presented example of microcontroller implementation is an essential part of the testing hardware platform, which is designed for the purpose of developing a sophisticated variable fractional-order PID (VFOPID) controller to be used in a closed-loop control system with multiple brushless DC (BLDC) motors. The paper is arranged as follows: In Section 2, the proposed testing platform and plant models are described. A description of the MCUs selected for the experiments is also provided. In Sections 3 and 5 approximation and discretization techniques, useful for the implementation of the models on the target platform, are discussed. Two approaches are considered: approximation with an ORA and numerical evaluation of the fractional differential equation using a truncated Grünwald–Letnikov (GL) definition. Some remarks related to implementation are given in Section 4. Conclusions are given in the final Section 6.

2. Plant Models

The closed-loop control system of an unmanned aerial vehicle (UAV) quadcopter arm, presented in Figures 1 and 2, consists of a hardware platform with two micro BLDC motors, an encoder and a controller for modeling and designing an accurate control law for a dedicated fractional-order PID (FOPID) controller [3]. For small angles we treated the plant as a black box and provided classic, first and second integer-order Küpfmüller models, further enhanced by a model described by the fractional-order transfer function (FOTF). Matching the response of simple fractional order model approximation with the original data exceeded first and second-order Küpfmüller models. However, several assumptions and specific implementation techniques had to be considered, which will be described in the sections that follow. The transfer functions of the models, prepared in MATLAB R2017b computation software using PID Tuner applet, Optimization Toolbox and FOMCON [13,14] are:

1. First-order plus dead time (FOPDT)

$$G_{FOPDT}(s) = \frac{K_P}{(1+T_P s)} e^{(-T_D s)} = \frac{1}{(1+0.4934s)} e^{-1.2279s} \tag{1}$$

   where K_P denotes the gain of the model, T_P is a time constant and T_D is the delay of the plant.
2. Second-order plus dead time (SOPDT)

$$G_{SOPDT}(s) = \frac{K_P}{(1+T_P s)^2} e^{(-T_D s)} = \frac{1}{(1+0.319s)^2} e^{-1.064s} \tag{2}$$

3. Non-integer-order plus dead time (NIOPDT)

$$G_{NIOPDT}(s) = \frac{K_P}{a_n s^{v_n} + a_{n-1} s^{v_{n-1}} + \ldots + a_0 s^{v_0}} e^{(-T_D s)} = \frac{1}{(0.18234 s^{1.9909} + 0.65536 s^{0.98319} + 0.9992)} e^{-1s} \tag{3}$$

where $a_n \ldots a_0$ denote constants and $\nu_n \ldots \nu_0$ values of fractional orders.

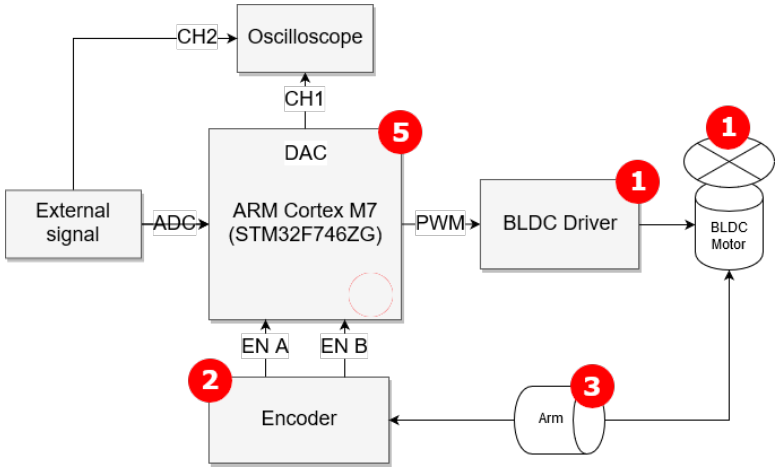

Figure 1. Block scheme of the testing hardware platform (1-BLDC micromotor, 2-high-precision encoder, 3-adjustable arm, 5-controller).

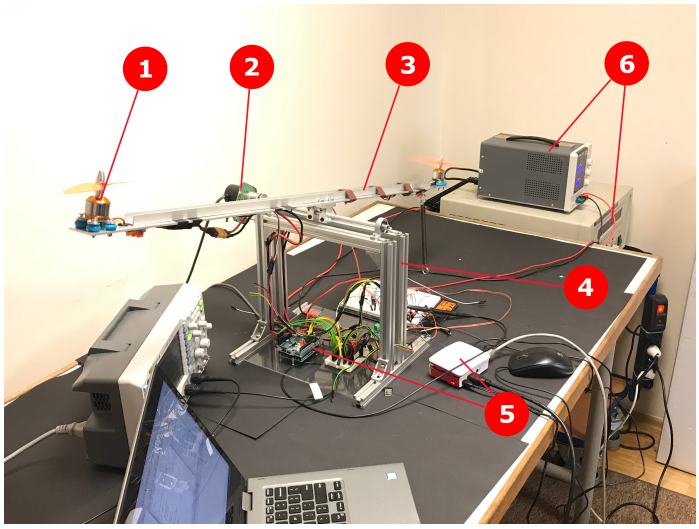

Figure 2. UAV arm testing platform with the BLDC motor and microcontroller (1-BLDC micromotor, 2-high-precision encoder, 3-adjustable arm, 4-rigid frame, 5-controllers, 6-power supply).

For the purpose of experiments on a real hardware platform, the ARM Cortex-M7 core-based 32-bit microcontroller from the STM32 High Performance series, model STM32F746ZG [15] was used. The following configuration was being set up during the main program initialization routine:

- maximum value of the main clock frequency $f_{CPU} = 216$ MHz,
- analog-to-digital converter (ADC) synchronized with the internal timer interrupt routine (ISR) to sample the input signal on an ADC pin at $f_{ADC} = f_s = 1$ kHz,
- number of ADC domain clock cycles required for a single ADC conversion $c_{ADC} = 15$ providing the best possible accuracy of 12-bit resolution. Time of conversion $t_{conv} \approx 10$ µs,
- single-precision hardware floating-point unit (FPU) and compiler warnings on automatic double-precision promotion enabled (software-simulated support for double-precision arithmetic),
- 60 Hz PWM output signal with adjustable duty cycle for driving the 11VDC-supplied micro-BLDC driver.

3. Discretization

As mentioned in the previous chapter, integer-order models can be easily implemented on any microcontroller, in the form of digital finite (FIR) or infinite impulse response (IIR) filters [16,17]. A significantly better performance, due to the lower number of operations (feedforward and feedback taps), can be achieved with models implemented as IIR filters, which is important specifically for real-time calculations performed on a microcontroller at the cost of potential breakdown of stability. Discretization of the transfer functions (1) and (2) was performed using two well-known methods, the zero-order hold and bilinear transform (Tustin's), at a sampling frequency of $fs = 1$ kHz. The zero-order hold was selected for the best matching with the original characteristics.

1. Discrete first-order plus dead time (DFOPDT)

$$H_{FOPDT}(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} z^{-T_D} \tag{4}$$

where b_i and a_j are numerator and denominator coefficients, respectively, and T_D is the number of delay input samples at a given sample rate f_s. The exact values of the coefficients are presented in Table 1.

2. Discrete second-order plus dead time (DSOPDT)

$$H_{SOPDT}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} z^{-T_D} \tag{5}$$

3. Discrete non-integer-order plus dead time (DNIOPDT)

Discretization of the non integer-order transfer function (3) was performed in three consecutive steps. First, approximation of the transfer function in the frequency domain was obtained by applying Oustaloup's Recursive filter algorithm (ORA) [10,18], approximating the complex variable s of the fractional order $0 < \nu < 1$, using the following formula:

$$s^v \approx K \prod_{k=-N}^{N} \frac{s + \omega'_k}{s + \omega_k} = \omega_h^v \frac{(s - \omega'_{-N})(s - \omega'_{-N+1}) \ldots (s - \omega'_N)}{(s - \omega_{-N})(s - \omega_{-N+1}) \ldots (s - \omega_N)} \tag{6}$$

where $[\omega_b, \omega_h]$ denotes the frequency range of the approximation, N is the order and $\omega\prime_k = \omega_b (\frac{\omega_h}{\omega_b})^{\frac{k+N+0.5-0.5v}{2N+1}}$, $\omega_k = \omega_b (\frac{\omega_h}{\omega_b})^{\frac{k+N+0.5+0.5v}{2N+1}}$. As a result, function (6) generates $(2N+1)$ poles and zeros in total. If $\nu > 1$ then s^ν is first replaced with $s^{(n+u)} = s^n s^u$, where n is an integer number and u is a fractional part, approximated by the algorithm. Different values for the approximation order N were tested over the selected frequency range $\omega \in \left[10^{-4}, 10^3\right] \frac{rad}{s}$. Step response and Bode characteristics for $N \in [1, 5]$ are presented in Figures 3 and 4. It is noticeable that all values of the order N provide satisfactory approximations of the initial fractional order transfer function. Nevertheless, we proceeded with approximation orders $N \geq 3$. Since the approximation polynomials had over 14 zeros and poles, in the second step, a reduction was performed using the balancing reduction technique [19], available in MATLAB as *balred* method. Minimization of the cost functions for a new reduced-order plant model of a fixed *balred* order $M = 3$, $M \in \mathbb{N}$ revealed that of several ORA filters, approximation of the order $N = 3$ ensured the best match between both characteristics.

$$G_{NIOPDT}(s) \approx \frac{0.00142\, s^3 - 0.01047\, s^2 + 5.75346\, s + 2.01679}{s^3 + 4.10156\, s^2 + 7.13415 s + 2.01552} e^{-1s} \tag{7}$$

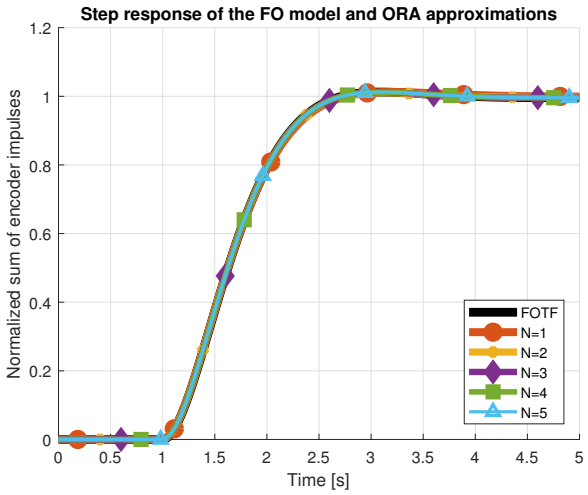

Figure 3. Oustaloup approximations of order N for fractional-order models. Step responses.

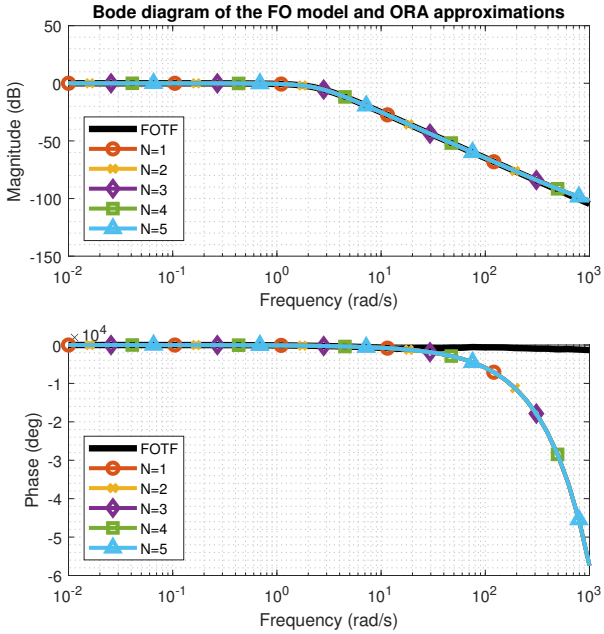

Figure 4. Oustaloup approximations of order N for fractional-order models. Bode diagram.

Figures 5 and 6 present unit step responses of the implemented platform integer- and fractional-order models, the latter approximated using the recursive Oustaloup filter method. Discrete 2nd-order model characteristics are presented with +0.2 offset to improve visibility.

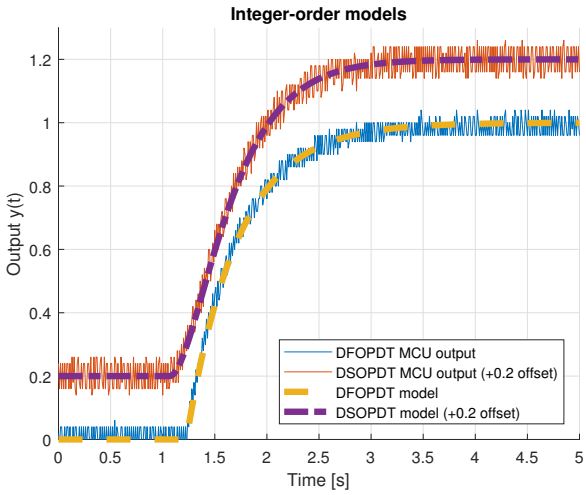

Figure 5. Measured microcontroller outputs with implemented integer-order (1st and 2nd) models.

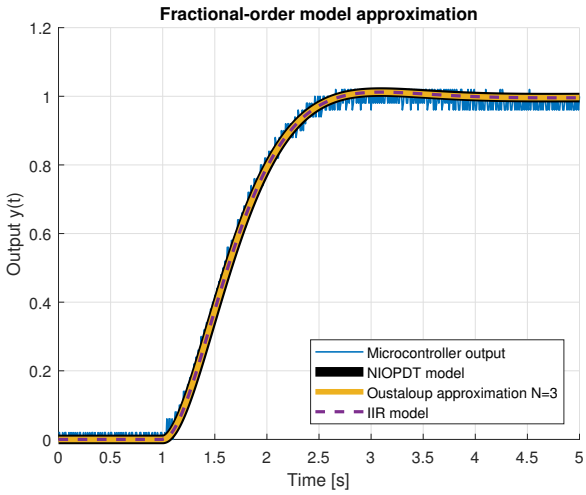

Figure 6. Measured microcontroller outputs with fractional-order (ORA) models.

4. Implementation Difficulties

It is important to stress two general points related to implementation. When discrete transfer functions (see Table 1) are obtained in MATLAB, one should be aware of the default Short Fixed Decimal display format, which rounds the numbers to four decimal places. Double precision representation of coefficients b_i, a_j can appear in the Variables explorer or after the activation of the long format display mode using the MATLAB routine: *format long*. This is necessary to avoid the model from losing stability caused by the implementation of truncated values for the coefficients. It is first necessary to determine the desired precision of the floating-/fixed-point number representation in the microcontroller software and the presence of the hardware floating-point unit, as these factors have a great impact the performance of the algorithm. The transfer functions of the proposed IIR filters were transformed into difference equations and implemented on an STM32F746ZG microcontroller in C programming language. To preserve the asymptotic stability of the designed models, in the case of fractional-order approximation the calculations had to be performed using software simulated double-precision arithmetic. This had a significant impact on performance, increasing the required number of CPU cycles from $c_{avg,sp} = 2650$ to $c_{avg,dp} = 8500$ (320%). The alternative approach involved the evaluation of the FOTF in the time domain, as will be described in the next section.

Table 1. Numerator and denominator coefficients of the discrete transfer functions (4) and (5), and of the continuous model (7).

Coeff	$H_{FOPDT}(z)$	$H_{SOPDT}(z)$	$H_{NIOPDT}(z)$
b_0	1.123629474892E-04	2.90801788577010E-06	0.00141997809862250
b_1	1.901155870198E-03	6.60568256821855E-06	−0.00426750404873265
a_1	9.979752146549E-01	1.993748864318110	2.99589971448314
b_2	-	0.25547387436067E-06	0.00428081749129811
a_2	-	−0.993758633492438	−2.99180655152227
b_3	-	-	−0.00143328952852651
a_3	-	-	0.995906835027740
T_D	1228	1065	1000

5. Time-Domain Approach Using the Grünwald–Letnikov Differintegral Operator and SMP

To compare the efficiency and accuracy of the Oustaloup approximation, we considered a fractional-order differential equation evaluated in the time-domain on the basis of the implementation of the truncated Grünwald–Letnikov differintegral operator [20]. This technique is known as the Short Memory Principle and restricts the boundaries of the operations to the most recent N_l samples. The principle is applied usually to numerical evaluation but has been also proposed for Riemann–Liouville and Caputo definitions [21]. Several different maximum memory lengths were examined $N_l = \left\{ N_0 = \frac{t_{sim}-t_0}{h}, \frac{N_0}{2}, \frac{N_0}{5}, \frac{N_0}{10}, \frac{N_0}{20}, \frac{N_0}{50}, \frac{N_0}{100} \right\}$ where N_0 denotes the total number of samples from the start of the simulation $t_0 = 0$ s, and was used as a reference value. The time responses of the plant were obtained using a modified formula [22]:

$$y(kh) = \frac{1}{\sum_{i=0}^{n} \frac{a_i}{h^{\nu_i}}} \left[u(kh) - \sum_{i=0}^{n} \frac{a_i}{h^{\nu_i}} \sum_{j=1}^{N_k} w_j^{\nu_i} y(kh - jh) \right] \tag{8}$$

where $w_j^{\nu_i}$ denotes the Newton binomial weights in the GL definition:

$$w_j^{\nu_i} = \begin{cases} 1 & \text{for } j = 0 \\ w_{j-1}^{\nu_i}(1 - \frac{1+\nu_i}{j}) & \text{for } j = 1, 2, \ldots \end{cases} \tag{9}$$

and N_k is the number of previously processed samples:

$$N_k = \begin{cases} k & \text{for } k \leq N_l \\ N_l & \text{for } k > N_l \end{cases}, l \in [0, 6] \tag{10}$$

Figure 7 presents simulated characteristics for all values of N_l and Table 2 below shows the corresponding performance indices, including those obtained for ORA. As can be seen, reducing the number of past samples below $N_4 = \frac{N_0}{20}$ (green curve) generates considerable error, which can increase even more for plants characterized by longer transient states. Therefore, SMP lengths of N_5, N_6 were not considered for further analysis. Reduced-order approximations obtained using the ORA algorithm provided better results than nearly all SMP-based approximations. However, the value of the maximum absolute percentage error (MaxAPE) was usually higher, due to the deviation between the step response characteristics at the beginning of the transient state, near zero. This error dropped rapidly for $k \to \infty$. Only for the memory lengths of SMP $N \geq N_1$ were the step response characteristics (red curve) more accurate and similar to the initial step response of the fractional-order transfer function (3). The number of CPU cycles required to evaluate the output signal was measured using the Data Watchpoint and Trace unit of the microcontroller [23], by computing the difference between the values in CYCCNT register, read in two separate sections of the program. Several different software optimization techniques were applied to the algorithm in each iteration.

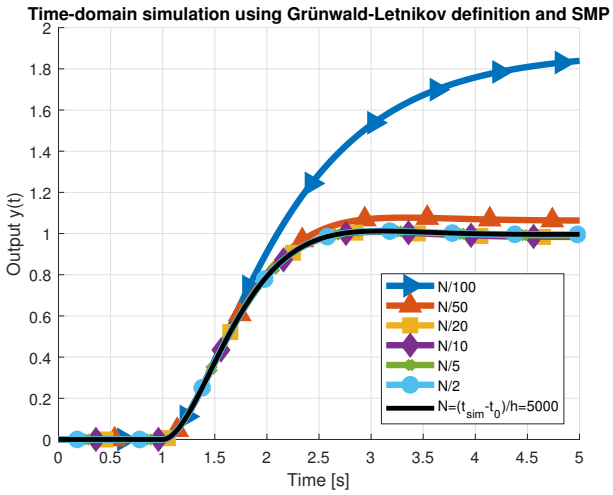

Figure 7. Heaviside step response evaluated by the GL method and different memory lengths N.

Table 2. Accuracy of Short Memory Principle and of ORA with *balred* $M = 3$.

	ISE	IAE	ITSE	ITAE	NRMSE	MaxAPE
$N_0 = 5000$	-	-	-	-	-	-
$N_1 = 2500$	7.1496×10^{-7}	9.2601×10^{-4}	4.0238×10^{-6}	5.0596×10^{-3}	99.91%	0.12%
$N_2 = 1000$	5.7340×10^{-5}	1.1914×10^{-2}	2.9386×10^{-4}	5.8319×10^{-2}	99.24%	0.65%
$N_3 = 500$	2.8593×10^{-4}	2.9316×10^{-2}	1.3949×10^{-3}	1.3625×10^{-1}	98.31%	1.25%
$N_4 = 250$	3.9339×10^{-4}	3.4905×10^{-2}	1.9004×10^{-3}	1.6067×10^{-1}	98.01%	1.43%
$N_5 = 100$	1.5484×10^{-2}	2.4532×10^{-1}	6.6317×10^{-2}	1.0198×10^{0}	87.53%	6.60%
$N_6 = 50$	1.9433×10^{0}	2.6677×10^{0}	8.8778×10^{0}	1.1565×10^{1}	0.00%	46.57%
$N_{ORA} = 3$	3.0352×10^{-6}	2.7942×10^{-3}	6.2302×10^{-6}	7.7747×10^{-3}	99.83%	12.22%
$N_{ORA} = 4$	3.0566×10^{-6}	2.7666×10^{-3}	6.5758×10^{-6}	7.9428×10^{-3}	99.82%	12.20%
$N_{ORA} = 5$	3.1344×10^{-6}	2.8097×10^{-3}	6.6456×10^{-6}	7.9841×10^{-3}	99.82%	12.66%

5.1. Initial Implementation- Look-up Tables, Shifted Input/Output Samples

In this step, arrays of lengths N_l were dynamically allocated to storing double-precision input and output values and w_j^i weight coefficients for each differintegral in the transfer function (3). w_j^i coefficients were precomputed at the program initialization (look-up table). During each analog-digital conversion, a new input sample was added to the end of the input array and the values in the input and output arrays were shifted left when the limit N_l was reached. Moreover, in the developed functions, only pointers to structures and arrays were accepted as parameters, to reduce the amount of memory occupied by the stack.

5.2. Replacing Arrays with Ring Buffers

Instead of shifting the values in the input/output arrays, a structure called a ring (circular) buffer was used. This involves defining a moving writing pointer (e.g., *inWrIdx*) for each of the arrays. When the buffer limit is reached ($inWrIdx = N_l$), the value is reset to point to the beginning of the buffer. The input value indicated by the pointer is always the most recent, whereas $inWrIdx+1$ (or 0 if $inWrIdx = N_l - 1$) points to the oldest sample. For models with delay, an additional delay buffer with two pointers for writing and reading is initialized. Further optimization can be achieved when the array lengths are powers of two. At the cost of higher memory consumption, the conditional operator for checking the limit N_l is replaced with a much faster bitwise multiplication of the pointer by N_l.

5.3. Enabling Optimization Flags

For the purposes of debugging and results verification, the program was initially built without any optimization by the GCC compiler (–O0 flag). Using the values $x = [1, 3]$ with the flag –Ox may reduce the code length and the size of the binary [24]. The higher value of x, the more optimizations are performed during the last stage of compilation. It should be noted that, according to the GCC manual, –O3 may affect computation results and generate a binary larger than –O2, due to e.g., loops unrolling. Therefore, –O2 is usually recommended for release building profiles.

5.4. Enabling Hardware FPU Unit, Using CMSIS DSP Library

Since the release of the ARM Cortex-M4 core, STM32 microcontrollers have been equipped with IEEE 754 compliant hardware floating-point units. Depending on the model of the microcontroller, single- or double-precision units are available [23,25], supporting hardware accelerated operations on float32_t or float64_t types, respectively. The unit is disabled by default and had to be configured first.

One may also find it helpful to enable double promotion warnings (-Wdouble-promotion in GCC), to eliminate automatic casting of numbers to higher precision. Moreover, for calculations on float32_t or fixed-point q31_t numbers, which were highly optimized by taking advantage of dedicated intrinsic and SIMD operations, the CMSIS DSP library for ARM cores was considered. This library contains implementations of several common DSP algorithms, from among which arm_conv_partial_f32 and arm_scale_f32 functions were used for discrete convolution and vector scaling operations, respectively. The overall performance vastly improved. However, the truncated precision led to significant accumulated error ($\Delta e_k = \frac{|y_{DP}(k) - y_{SP}(k)|}{|y_{DP}(k)|} 100\% = 28.3\%$ for the last computed output sample), disqualifying the model H_{NIOPDT} in this form from practical application.

5.5. Other Approaches

Further optimizations are a topic of the ongoing research involving adaptive memory methods, parallel implementation of numerical algorithms and calculations using fixed-point arithmetic. Finally, assembly inlines placed in critical sections of the algorithms and platform-specific enhancements are being considered. However, these approaches are strictly platform-dependent and must be adopted for each architecture individually. The results of subsequent software optimizations are presented in Figure 8. The algorithm processing past $N_4 = 250$ samples, compiled with a –O2 flag, satisfied the initial timing requirement, calculating the output in a time shorter than the sampling period $t_s = 1$ ms.

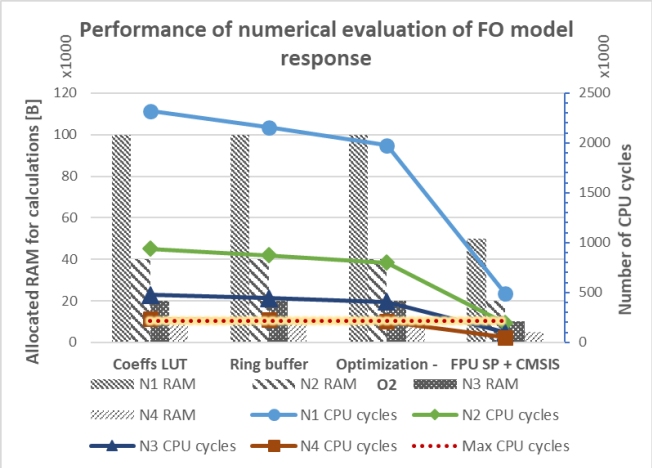

Figure 8. Software optimizations of the numerical algorithm for calculating output of model (3) with the maximum number of CPU cycles allowed (red dotted line).

6. Conclusions

This study set out to address the problem of fractional-order model implementation. The non-local fractional differential GL operator involves a constantly growing number of calculations, which can be either bounded or replaced by an integer-order operator using one of the well-known approximation methods. A model of a UAV arm with a BLDC motor was implemented, using two approaches in the frequency- and time-domains: the Oustaloup approximation and numerical evaluation of the differential equation using the Grünwald–Letnikov definition with the Short Memory Principle. The performance of the algorithms was measured and compared. For orders $N_{ORA} > 2$ of the Oustaloup approximation, similar step response characteristics were obtained. Moreover, in this case, the required buffer size was limited to only four feedforward and four feedback samples, vastly improving the calculation time and memory consumption. Higher accuracy could be obtained by different levels of reduction in the numbers of poles and zeros. Another approach, based on implementation of the Grünwald–Letnikov definition, required the introduction of the Short Memory Principle. In this case, programming optimization techniques allowed the computation time to be reduced by 15% or even 78% if CMSIS DSP and FPU hardware were used. This last result, however, required redesigning of the model. The methods described in this paper can be easily adapted and applied to other fractional-order models or control algorithms. Further work is underway, focusing on parallel implementation and optimization of fractional order numerical algorithms and designing a variable-, fractional-order PID controller with algorithms for determining the function of variable order.

Author Contributions: M.M. conceived the research direction and collected relevant information; M.M. designed the Simulink and Matlab simulation and experiment; M.B. and R.W. built the test stand; M.M. provided microcontroller implementation; M.B., M.M., and R.W. analyzed the data; M.M., M.B., and R.W. wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Polish funds from the National Science Center under grant DEC-2016/23/B/ST7/03686.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADC	analog-to-digital converter
BLDC	brushless direct-current motor
CYCCNT	Data Watchpoint and Trace Cycle Count Register
(D)FOPDT	(discrete) first-order plus dead time
(D)NIOPDT	(discrete) non-integer-order plus dead time
(D)SOPDT	(discrete) second-order plus dead time
FIR	finite impulse response
FO	fractional-order
FOC	fractional-order calculus
FOTF	fractional-order transfer function
FPU	floating-point unit
GL	Grünwald-Letnikov
IAE	integral absolute error
IIR	infinite impulse response
IO	integer-order
ISE	integral square error
ISR	interrupt service routine
MaxAPE	maximum absolute percentage error
MCU	microcontroller unit
ORA	Oustaloup Recursive Approximation
SIMD	Single Instruction Multiple Data
SMP	Short Memory Principle

UAV	unmanned aerial vehicle
(V)FOPID	(variable) fractional-order proportional-integral-derivative controller

References

1. Kilbas, A.A.; Srivastava, H.; Trujillo, J.J. *Theory and Applications of Fractional Differential Equations*; Elsevier Science Inc.: New York, NY, USA, 2006; Volume 204. [CrossRef]
2. Miller, K.S.; Ross, B. *An Introduction to the Fractional Calculus and Fractional Differential Equations*; John Wiley & Sons: New York, NY, USA, 1993.
3. 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: San Diego, CA, USA, 1999.
4. Merrikh-Bayat, F.; Mirebrahimi, N.; Khalili, M.R. Discrete-time fractional-order PID controller: Definition, tuning, digital realization and some applications. *Int. J. Control Autom. Syst.* **2015**, *13*, 81–90. [CrossRef]
5. Petráš, I.; Vinagre, B.M. Practical application of digital fractional-order controller to temperature control. *Proc. Acta Montan. Slovaca* **2002**, *7*, 131–137.
6. Tepljakov, A.; Petlenkov, E.; Belikov, J. Embedded system implementation of digital fractional filter approximations for control applications. In Proceedings of the 21st International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES 2014), Lublin, Poland, 19–21 June 2014. [CrossRef]
7. Tepljakov, A.; Petlenkov, E.; Belikov, J.; Finajev, J. Fractional-order controller design and digital implementation using FOMCON toolbox for MATLAB. In Proceedings of the IEEE International Symposium on Computer-Aided Control System Design, Hyderabad, India, 28–30 August 2013. [CrossRef]
8. Dziwiński, T.; Piątek, P.; Baranowski, J.; Bauer, W.; Zagórowska, M. On the practical implementation of non-integer order filters. In Proceedings of the 2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 24–27 August 2015; pp. 921–924. [CrossRef]
9. MacDonald, C.L.; Bhattacharya, N.; Sprouse, B.P.; Silva, G.A. Efficient computation of the Grünwald–Letnikov fractional diffusion derivative using adaptive time step memory. *J. Comput. Phys.* **2015**, *297*, 221–236. [CrossRef]
10. Oustaloup, A. *La commande CRONE: Commande Robuste D'ordre non Entier*; Hermes: Paris, France, 1991.
11. Vinagre, B.M.; Podlubny, I.; Hernandez, A.; Feliu, V. Some approximations of fractional order operators used in control theory and applications. *Fract. Calc. Appl. Anal.* **2000**, *3*, 231–248. [CrossRef]
12. Garrappa, R. Numerical Solution of Fractional Differential Equations: A Survey and a Software Tutorial. *Mathematics* **2018**, *6*, 16. [CrossRef]
13. Tepljakov, A.; Petlenkov, E.; Belikov, J. FOMCON: Fractional-Order Modeling and Control Toolbox for MATLAB. In Proceedings of the 18th International Conference "Mixed Design of Integrated Circuits and Systems" (MIXDES 2011), Gliwice, Poland, 16–18 June 2011; pp. 684–689. [CrossRef]
14. Alagoz, B.B.; Tepljakov, A.; Ates, A.; Petlenkov, E.; Yeroglu, C. Time-domain identification of One Noninteger Order Plus Time Delay models from step response measurements. *Int. J. Model. Simul. Sci. Comput.* **2019**, *10*. [CrossRef]
15. STMicroelectronics. STM32F745xx STM32F746xx ARM-based Cortex-M7 32b MCU+FPU, 62DMIPS up to 1MB Flash/320+16+4KB RAM, USB OTG HS/FS, ethernet, 18TIMs, 3ADCs, 25 com itf, cam & LCD Datasheet. 2016. Available online: https://www.st.com/resource/en/datasheet/stm32f746zg.pdf (accessed on 9 March 2020)
16. Monje, C.A.; Chen, Y.; Vinagre, B.M.; Xue, D.; Feliu, V. Fractional-order Systems and Controls Fundamendals and Applications. In *Advances in Industrial Control*; Springer: London, UK, 2010. [CrossRef]
17. Lyons, R.G. *Understanding digital signal processing*; Prentice Hall PIR: Upper Saddle River, NJ, USA, 2004.
18. Oustaloup, A. *La dérivation non entière: théorie, synthèse et applications*; Hermes: Paris, France, 1995.
19. Caponetto, R.; Machado, J.T.; Murgano, E.; Xibilia, M.G. Model Order Reduction: A Comparison between Integer and Non-Integer Order Systems Approaches. *Entropy* **2019**, *21*, 876. [CrossRef]
20. Garrappa, R.; Kaslik, E.; Popolizio, M. Evaluation of Fractional Integrals and Derivatives of Elementary Functions: Overview and Tutorial. *Mathematics* **2019**, *2*, 407. [CrossRef]
21. Wei, Y.; Chen, Y.; Cheng, S.; Wang, Y. A note on short memory principle of fractional calculus. *Fract. Calc. Appl. Anal.* **2017**, *20*. [CrossRef]

22. Chen, Y.; Petráš, I.; Xue, D. Fractional order control—A tutorial. In Proceedings of the American Control Conference, St. Louis, MO, USA, 10–12 June 2009. [CrossRef]
23. ARM Ltd. *ARM Cortex-M7 Processor*, r0p2 ed.; ARM Ltd.: Cambridge, UK, 2014; Available online: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0489b/DDI0489B_cortex_m7_trm.pdf (accessed on 9 March 2020)
24. ARM Ltd. *Arm® Compiler Version 6.12 User Guide*; ARM Ltd.: Cambridge, UK, 2019.
25. STMicroelectronics. AN4044 Application Note. Floating Point Unit Demonstration on STM32 Microcontrollers. 2016. Available online: https://www.st.com/content/ccc/resource/technical/document/application_note/10/6b/dc/ea/5b/6e/47/46/DM00047230.pdf/files/DM00047230.pdf/jcr:content/translations/en.DM00047230.pdf (accessed on 9 March 2020)