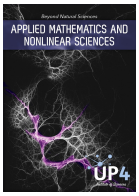
Applied Mathematics and Nonlinear Sciences 2(1) (2017) 237–248



Applied Mathematics and Nonlinear Sciences

http://journals.up4sciences.org

Comparison of Fractional Order Derivatives Computational Accuracy - Right Hand vs Left Hand Definition

Dariusz W. Brzeziński †

Institute of Applied Computer Science, Lodz University of Technology, 18/22 Stefanowskiego St., 90-924 Łódź Poland

Submission Info

Communicated by Juan L.G. Guirao
Received 18th January 2017
Accepted 25th June 2017
Available online 25th June 2017

Abstract

Numerical computation of fractional order derivatives and integrals can be done by applying many approaches. They include Riemann-Liouville and Caputo formulas. Both formulas include fractional order integration and integer order differentiation but in a different order: fractional order integration can be preceded by integer order differentiation or it can be done in the reverse order. In the paper we apply both formulas for FOD computation and investigate if an application of a particular order have any advantages, e.g. resulting in higher accuracy. The outcome of the experiment is that by applying adequately selected formula of FOD computation according to the shape of a particular function, the accuracy of computation can often be increased. Application of Riemann-Liouville formula enables successful FOD computation for functions which n^{th} derivative does not exist.

Keywords: Accuracy of computation, Numerical Fractional Order Differentiation and Integration.
AMS 2010 codes: 26A33, 26A42, 26A99, 53-04, 65K05.

1 Introduction

Fractional calculus involves among the others application of fractional order derivatives (FOD) and fractional order integrals (FOI), which order can assume a real or a complex value. Thus derivatives and integrals of integer order can be considered as one of special cases of fractional calculus.

Fractional derivatives can be calculated using numerous approaches which include Riemann-Liouville fractional derivative (RL) and Caputo fractional derivative (C) [1]. The latter definition is often preferred for simulations of real world processes and systems in a form of fractional differential equations (FDE) [2, 3] due to

†Corresponding author.
Email address: dbrzezinski@iis.p.lodz.pl

ISSN 2444-8656 doi:10.21042/AMNS.2017.1.00020

its ability to accept initial conditions in terms of integer order values. Riemann-Liouville fractional derivative definition does not have this practical application advantage and thus is mostly applied for theoretical considerations. However, regarding the subject of the paper, the substantial difference between RL and C definitions lies in the method of their computation: RL method involves repeated integration of a function and then differentiation of the resulting function with an integer order; C method attempts to arrive at the same result using the same operations, but in the reverse order. Thus using direction terms we can call RL and C left-hand and right-hand definitions, respectively.

The aim of the paper is to compare computational accuracy of fractional derivatives by applying the left-hand and the right hand definitions.

To fulfill this goal we have divided the paper in the following subsections: first there are presented formulas for fractional derivatives computation. The second subsection explains the terms left-hand and right-hand. The next one contains brief description of problems associated with FOD/FOI computation and presents numerous numerical methods which are applied for fractional derivatives computation. Difficulties associated with FOD/FOI application include computation of reference values required for accuracy assessment. Thus, the next section presents formulas and methods for their computation. The section after that one is devoted to computational accuracy comparison of the right- and left-hand definitions. It includes detailed results for computation of FOD for some functions in various ranges and short comment about it. The paper ends with conclusion which summarize the reasons for applying a particular definition for FOD computation.

2 Useful Formulas

The following formulas are applied for FOD/FOI computations. To differentiate between fractional and integer orders, fractional orders are commonly denoted by the Greek letters ν and μ and integer orders by the letters n and m.

Riemann-Liouville fractional integral for $\nu > 0$ is defined as

$$ {}^{RL}_{0}\mathrm{J}^{(\nu)}_{t} f(t) = \frac{1}{\Gamma(\nu)} \int_0^t \frac{f(\tau)}{(t-\tau)^{1-\nu}} d\tau. \tag{1} $$

Riemann-Liouville fractional derivative for $\nu > 0$, $\mu > 0$ is defined as

$$ {}^{RL}_{0}\mathrm{D}^{(\nu)}_{t} f(t) = D^n J^\nu f(t) = \frac{d^n}{dt^n} \left[\frac{1}{\Gamma(n-\nu)} \int_0^t \frac{f(\tau)}{(t-\tau)^{\nu-n+1}} \, d\tau \right] \tag{2} $$

and

$$ {}^{RL}_{0}\mathrm{D}^{(\mu)}_{t} f(t) = D^n J^\nu f(t) = \frac{d^n}{dt^n} \left[\frac{1}{\Gamma(\nu)} \int_0^t \frac{f(\tau)}{(t-\tau)^{\nu-1}} \, d\tau \right], \ \mu = n - \nu \tag{3} $$

Caputo fractional derivative

$$ {}^{C}_{0}\mathrm{D}^{(\nu)}_{t} f(t) = \frac{1}{\Gamma(n-\nu)} \int_0^t \frac{f^{(n)}(\tau)}{(t-\tau)^{\nu-n+1}} \, d\tau, \tag{4} $$

with conditions $f(t) = 0$ for $t \le 0$, $f(0) = 0, f^{(1)} = f^{(2)} \ldots f^{(n)} = 0$.

Formulas (2) or (3) and (4) are related by

$$ {}^{RL}_{0}D^{(\nu)}_{t} f(t) = {}^{C}_{0} D^{(\nu)}_{t} f(t) + \sum_{k=0}^{n-1} \frac{t^{k-\nu}}{\Gamma(k-\nu+1)} f^{(k)}(0). \tag{5} $$

Placing C formula (4) in the right side of equation (5) enables obtaining an equivalent Riemann-Liouville fractional derivative formula [4].

$$ {}_{0}^{RL}\mathrm{D}_{t}^{(\nu)} f(t) = \sum_{k=0}^{n-1} \frac{t^{k-\nu} f^{(k)}(0)}{\Gamma(k-\nu+1)} + \frac{1}{\Gamma(n-\nu)} \int_{0}^{t} \frac{f^{(n)}(\tau)}{(t-\tau)^{\nu-n+1}} \, d\tau. \tag{6} $$

In formulas (1)-(6) ν is a real number such that $n-1 < \nu < n$, n denotes an integer number $n = \lceil \nu \rceil$ and $\Gamma(.)$ is Euler's Gamma function.

3 Explanation of Terms Left-hand Definition and Right-hand Definition

As it was stated in introduction, the reason for selecting such a naming schema for RL and C formulas is due to the method of their computation: first we select order $\nu \in R_+$. Then we calculate order n such that $n-1 < \nu < n$.

Having selected both numbers, we can calculate fractional derivative using left-hand definition presented in figure 1 and formulas (2) and (3). Here we first integrate a function with order $n-\nu$ and then we calculate n^{th} derivative of the resulting function.

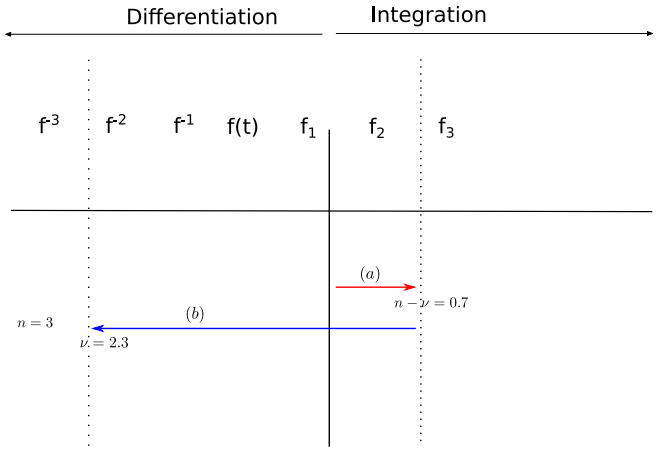


Fig. 1 Graphical Representation of Left-hand definition of fractional derivative for order $\nu = 2.3$.

By applying right-hand definition presented in figure 2 and formula (4) and (6) we attempt to arrive at the same results in the reverse order: first we calculate n^{th} order derivative of a function and then we integrate the resulting function with order $n-\nu$.

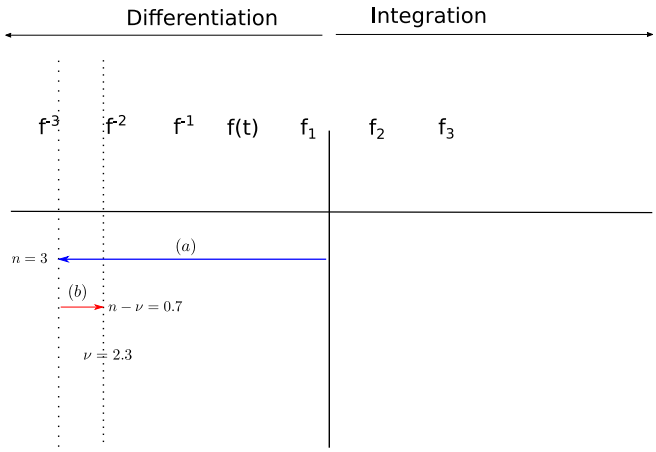


Fig. 2 Graphical Representation of Right-hand definition of fractional derivative of order $\nu = 2.3$.

4 Computation of Fractional Order Derivatives.

Riemann-Liouville fractional order derivative and integral and Caputo fractional order derivative formulas (1)-(6) consist of an ordinary integral. However, the integral is difficult to compute due to presence of a singularity at the end of integration interval that is preceded by rapid values increase. Similarly to the term used for such occurrence in the world of differential equations, it will be called *the stiffness*.

The stiffness can increase or decrease correspondingly to the values of the exponent in the integrand in formulas (1)-(6) and characteristics of a function. However, it is the stiffness and the singularity presence that determine the accuracy of integral computation.

To compute fractional order derivatives and integrals by applying formulas (1)-(6) there can be applied commonly used methods of numerical integration, e.g. midpoint rule, trapezoidal rule, gauss quadratures of a general use, etc. However, their accuracy and effectiveness are heavily decreased by the presence of the singularity and the stiffness. Additionally, computation of FOD/FOI of the fractional orders near integer orders (e.g. 0 and 1) of any function is usually burdened with several hundred percent relative error. Thus we can say that none of commonly applied methods of integration can be considered as methods of general use for the purpose of computing FOD/FOI with high accuracy independently of order and interval.

To increase computational accuracy of FOD/FOI, either an applied method of numerical integration must be adapted to handle a difficult integrand properly or an integrand must be transformed to fit into the requirements of an applied numerical method of integration. Furthermore replacing standard double precision computer arithmetic with arbitrary precision for numerical integration enables increasing many times the overall accuracy of computation.

In our papers [5, 6] we presented two-way approach for computation accuracy increase of FOD/FOI, which included integrand transformation and adaptation of Gauss-Jacobi quadrature weight function with significant additional accuracy increase obtained by applying the GNU GMP [7] and the GNU MPFR [8] arbitrary precision mathematical libraries with C++ MPFR C++ interface [9] for programming.

For the reason of its high-accuracy abilities, efficiency and versatility, Gauss-Jacobi quadrature with adapted weight function is applied for the purpose of accuracy comparison of the FOD/FOI right- and left-hand definitions.

Application of Gauss quadratures enables solving efficiently problems associated with singularities and infinite limits by approximating an integrand and its difficult features by using polynomials. Wherein, particular polynomial is selected according to its properties to address a specific problem encountered in an integrand. If that is the case, the application of one of the Gauss quadratures assures high accuracy integration with a few

sampling points.

Application of a Gauss quadrature (except for Gauss-Legendre quadrature) enables removing a problematic area or feature of an integrand from the integration and computing it by applying the formulas for the weights of a quadrature. However, Gauss quadratures can only be applied to a specific types of integrands, which is conditioned by the linkage to their weight functions.

By applying Gauss-Jacobi quadrature with the weight function $p(x)$

$$p(x) \equiv (1-x)^{\lambda}(1+x)^{\beta},\ \lambda,\beta > -1, \tag{7}$$

general formula of an approximate calculation of definite integral

$$\int_a^b p(x) f(x)\,dx \cong \sum_{k=1}^{n} A_k f(x_k) + R.$$

can be expressed as

$$\int_{-1}^{1} (1-x)^{\lambda}(1+x)^{\beta} f(x)\,dx \approx \sum_{k=1}^{n} A_k f(x_k) + R, \tag{8}$$

in which x_k are zeros of Jacobi polynomial $J_n^{(\lambda,\beta)}(x_k)$ of degree n and R is an error.

We can use this property for FOD/FOI computation.

Removing kernel from an integrand

$${}_0^{RL}\mathrm{D}_t^{(\nu)} f(t) = \frac{1}{\Gamma(\nu)} \int_0^t \underbrace{(t-\tau)^{\nu-1}}_{\text{kernel}} f(\tau)\,d\tau,$$

substituting $\lambda = \nu - 1, \beta = 0$, we obtain

$$\int_{-1}^{1} (1-t)^{\lambda} f(t)\,dt \approx \sum_{k=1}^{n} A_k f(t_k) = \sum_{k=1}^{n} \frac{2^{\nu}\left(1-t_k^2\right)\left[J_n^{(\lambda,0)'} t_k\right]^2}{f}(t_k) \tag{9}$$

where A_k are weights of the quadrature.

Transforming the interval $[0,t]$ into $\langle -1,1\rangle$

$$\left(\frac{t-t_0}{2}\right)^{\nu} \int_{-1}^{1} \frac{f(u)}{(1-u)^{\lambda}} du$$

where

$$f(u) = f\left(\left(\frac{t-t_0}{2}\right)u + \left(\frac{t+t_0}{2}\right)\right),$$

we obtain a formula (10) for Gauss-Jacobi quadrature (8) that is perfectly suited for FOD/FOI computation by applying formulas (1)-(6), i.e. kernel in integrand is not integrated but computed by applying the formula for the weights A_k of the quadrature and it does not determine accuracy of integration anymore.

From now on Riemann-Liouville fractional derivative formula (6) assumes the following computational form

$$\begin{aligned} {}_{t_0}^{RL}\mathrm{D}_t^{(\nu)} f(t) &= \sum_{k=0}^{n-1} \frac{(t-t_0)^{k-\nu} f^{(k)}(t_0)}{\Gamma(k-\nu+1)} + \frac{1}{\Gamma(n-\nu)} \left(\frac{t-t_0}{2}\right)^{n-\nu} \int_{-1}^{1} \frac{f^{(n)}(u)}{(1-u)^{\nu-n+1}} du \\ &= \sum_{k=0}^{n-1} \frac{(t-t_0)^{k-\nu} f^{(k)}(t_0)}{\Gamma(k-\nu+1)} + \frac{1}{\Gamma(n-\nu)} \left(\frac{t-t_0}{2}\right)^{n-\nu} \underbrace{\sum_{k=1}^{n} \frac{2^{\nu}}{\left(1-u_k^2\right)\left[J_n^{(\nu-n+1,0)'} u_k\right]^2}}_{\text{kernel}} f^{(n)}(u_k). \end{aligned} \tag{10}$$

in which $n = \lceil \nu \rceil$.

The ordinarily sufficient method of Gauss quadrature application is reduced to the use of tabulated values of nodes and weights. They are freely available and easy to incorporate into a computer program [10]. However, its application becomes difficult if nodes and weights must be customized and computed, e.g. if the integrand does not meet fully or at all the requirements of a quadrature or required integration interval is different than foreseen for a quadrature. In all of the cases nodes and weights must be computed prior the integration.

Customized variants of the Gauss-Jacobi quadrature (9) for computation of FOD/FOI by applying formulas (1)-(6) require computation of polynomial of order n, its derivative, nodes and weights.

Existing approaches for computing the nodes and the weights, some of which have been widely used for many years, suffer from $\mathcal{O}(n^2)$ complexity or error which grows with n. With some optimizations, e.g. application of technique which utilizes asymptotic formulas for accurate initial guesses of the roots and efficient evaluations of the degree n Jacobi polynomial, it can be reduced to $\mathcal{O}(n)$ as it is presented in [11].

Fractional order derivative calculation by applying formulas (1)-(6) requires computation of n-th derivative of a function, where n is an integer. The following formulas are used for that purpose in the paper [12, 13]:

First derivative 3-points stencil Central Difference

$$f^{'}(x_0) = \frac{f_1 - f_{-1}}{2h} + \mathcal{O}(h^2), \tag{11}$$

where $f_1 \equiv f(x_0 + h)$ and $f_{-1} \equiv f(x_0 - h)$.

Second derivative 3-points stencil Central Difference

$$f^{''}(x_0) = \frac{f_{-1} - 2f_0 + f_1}{h^2} + \mathcal{O}(h^2) \tag{12}$$

Third derivative 3-points stencil Central Difference

$$f^{'''}(x_0) = \frac{-f_{-2} - 2f_{-1} - 2f_1 + f_2}{2h^3} + \mathcal{O}(h^2) \tag{13}$$

The limiting factor to the accuracy of computation by applying the formulas (1)-(6) is a precision which uses the computer to store data that are being supplied to it.

The selection of uniform C++ equipped with the standard mathematical library as a main programming tool is not enough nowadays to take full advantage of available hardware. The application of arbitrary precision computing for increasing accuracy and correctness of numerical calculations and Nvidia CUDA parallelization technology for their effectiveness, are the best examples in this context [14].

Application of arbitrary precision makes it possible for the user to choose precision for calculation and for each variable storing a value. It is not machine-depended or IEEE standard types. With its help we can - among the others - increase general accuracy of mathematical computations. However, its application purpose is above all to increase accuracy of numerical calculations, e.g. by eliminating under- and overflows, increasing accuracy of a polynomial zeros finding and derivative and integral calculating.

The importance of elimination of limited precision in computer calculations was aptly presented by Toshio Fukushima in *The Astronomical Journal* in 2001 by giving the following example: "In the days of powerful computers, the errors of numerical integration are the main limitation in the research of complex dynamical systems, such as the long-term stability of our solar system and of some exoplanets [...]" and gives an example where using double precision leads to an accumulated round-off error of more than 1 radian for angular position of planets [15].

Double precision computer arithmetic is optimized for speed and has many flaws which negatively influence the accuracy of computations, e.g. limitations of number values which double precision variables can hold or no programmer influence on mathematical operations rounding.

However, it is the lack of clarity in handling of intermediate results which troubles the most, i.e. the floating-point standard [16] only defines that the results must be rounded correctly to the destination's precision and fails

to define the precision of destination variable. This choice is commonly made by a system or a programming language. The user can not influence it in any way. Therefore, the same program can return significantly different results depending on the implementation of the IEEE standard.

Arbitrary precision is applied in conjunction with special libraries which include their own data structures and mathematical functions.

5 Computational Accuracy Assessment of Fractional Order Derivatives.

Accuracy and the efficiency assessment are crucial components in the process of programming because it enables examining a program for its proper operation and accuracy.

Common approach to the accuracy assessment requires an exact value.

In case of FOD/FOI computation, an effective accuracy assessment is difficult, sometimes not possible due to general lack of formulas for exact values.

Despite the availability of a handful of analytical formulas for fractional order $\nu = \frac{1}{2}$ and some computational formulas, they are accessible for selected types of functions only. Some other formulas are in form of series expansion only.

The following formulas for FOD/FOI are used further in the paper for calculations of their exact values:

Trigonometric functions:

$$ {}_0\mathrm{D}_t^{(\nu)} \sin(t) = t^{1-\nu} \sum_{k=0}^{\infty} \frac{(-1)^k t^{2k}}{\Gamma(2k+2-\nu)}, \text{ for } 0 < \nu < 1, \tag{14} $$

$$ {}_0\mathrm{D}_t^{(\nu)} \cos(t) = t^{2-\nu} \sum_{i=0}^{\infty} \frac{(-1)^{i+1} t^{2i+1}}{\Gamma(2i+4-\nu)}, \text{ for } 1 < \nu < 2. \tag{15} $$

Power functions

$$ {}_0\mathrm{D}_t^{(\nu)} t^{\beta} = \frac{\Gamma(\beta+1)}{\Gamma(\beta-\nu+1)} t^{\beta-\nu}. \tag{16} $$

Exponential function $f(t) = e^{(at)}$ by applying the Mittag-Leffler function with $\alpha = \beta = 1$

$$ {}_0D_t^{(\nu)} e^{(at)} \approx {}_0D_t^{(\nu)} E_{\alpha,\beta}(at) = t^{-\nu} \sum_{k=0}^{\infty} \frac{(at)^k \Gamma(k+1)}{\Gamma(k+1-\nu)\Gamma(\nu k+1)}. \tag{17} $$

In practice ∞ is replaced by large number N.

In our work we have to be able to assess the accuracy of FOD/FOI computation of any function. Thus we have developed a method of accuracy assessment by applying operators concatenation [17] of fractional order differentiation and integration.

The main idea behind this versatile criteria presented in Figure 3 is the ability to compare computed values of FOD/FOI with values of classical - integer order derivatives and integrals, which can be calculated by applying well known (and accurate) formulas.

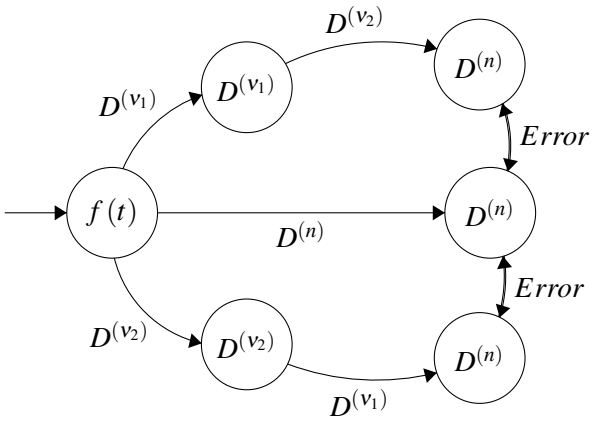


Fig. 3 Two fractional orders ν_1 and ν_2 of different values concatenated to obtain an integer order n, i.e. $\nu_1 + \nu_2 = n$.

6 Rules of Accuracy Comparison of FOD Computation.

To compare computational accuracy of FOD we apply right- and left-hand definitions schemas presented in Figures 1 and 2 to obtain FOD of order $\nu = 0.3$ and 2.3 for the following functions and ranges:

$$f(t) = e^{-t},\ t \in (0,5), \tag{18}$$

$$f(t) = e^{t},\ t \in (0,3), \tag{19}$$

$$f(t) = \sqrt{t},\ t \in (0,1), \tag{20}$$

$$f(t) = \sin(t),\ t \in (0,2\pi). \tag{21}$$

To compare computational accuracy of FOD we use relative error and infinity norm $\|e_r\|_{L_\infty}$ to measure similarity of two vectors x and $\hat{x}$. Vector x contains 100 computed values in a given range and vector $\hat{x}$ contains reference values assumed, i.e. values computed by applying formulas and methods listed in subsection 5):

$$\|e_r\|_{L_\infty} = max_i \frac{\|x_i - \hat{x}_i\|}{\|x_i\|}. \tag{22}$$

Tables 1-4 contain maximum relative error between vectors x and $\hat{x}$ for 100 values of FOD computed in a given range $t_0 - t$, for $\mathrm{N} = 4, 8, \ldots, 64$ order of polynomial applied for the method of integration and d - differentiation step required for integer order derivative computation.

7 Results of Accuracy Comparison of FOD Computation.

Table 1 RL $\|e_r\|_{L_\infty}$, $h = \frac{t-t_0}{100points}$, d - differentiation step, $n-\nu = 0.7,\ n = 1, \nu = 0.3$

	Function (18)		Function (19)		Function (20)		Function (21)		
N	$1.0e-06$	$1.0e-36$	$1.0e-06$	$1.0e-36$	$1.0e-06$	$1.0e-36$	$1.0e-06$	$1.0e-36$	$\leftarrow$ **d**
4	2.92e-11	3.05e-19	8.04e-06	8.04e-06	1.22e-03	1.22e-03	1.60e-08	1.06e-08	
8	2.92e-11	6.89e-43	1.64e-13	2.66e-15	1.07e-04	1.07e-04	2.01e-13	3.60e-21	
16	2.92e-11	5.21e-66	1.66e-13	3.91e-39	2.26e-05	2.26e-05	2.01e-13	2.19e-51	
32	2.92e-11	5.21e-66	1.66e-13	1.39e-65	2.91e-06	2.91e-06	2.01e-13	1.03e-64	
64	2.92e-11	1.16e-66	1.66e-13	1.96e-64	3.07e-07	3.07e-07	2.01e-13	1.54e-64	

Table 2 C $||e_r||_{L_\infty}$, $h = \frac{t-t_0}{100points}$, d - differentiation step, $n = 1, n-\nu = 0.7, \nu = 0.3$

	Function (18)		Function (19)		Function (20)		Function (21)		
N	$1.0e-06$	$1.0e-36$	$1.0e-06$	$1.0e-36$	$1.0e-06$	$1.0e-36$	$1.0e-06$	$1.0e-36$	$\leftarrow$ **d**
4	1.24e-14	1.76e-21	2.33e-06	2.33e-06	1.22e-02	7.99e-02	2.81e-09	2.81e-09	
8	1.24e-14	2.07e-45	1.62e-13	4.28e-11	4.16e-02	4.16e-02	1.67e-13	3.52e-22	
16	1.24e-14	5.52e-67	1.62e-13	3.43e-40	2.13e-02	2.13e-02	1.67e-13	1.13e-52	
32	1.24e-14	8.05e-68	1.62e-13	4.29e-65	1.07e-02	1.07e-02	1.67e-13	4.45e-67	
64	1.24e-14	1.38e-67	1.62e-13	3.65e-65	5.43e-03	5.43e-03	1.67e-13	4.37e-66	

Table 3 RL $||e_r||_{L_\infty}$, $h = \frac{t-t_0}{100points}$, d - differentiation step, $n-\nu = 0.7,\ n = 3, \nu = 2.3$

	Function (18)		Function (19)		Function (20)		Function (21)		
N	$1.0e-06$	$1.0e-16$	$1.0e-06$	$1.0e-16$	$1.0e-06$	$1.0e-16$	$1.0e-06$	$1.0e-16$	$\leftarrow$ **d**
4	7.38e-10	3.56e-17	2.13e-09	6.78e-19	1.22e-03	1.22e-03	5.41e-05	5.41e-05	
8	7.38e-10	7.38e-30	2.13e-09	2.13e-29	1.07e-04	1.07e-04	1.62e-11	1.59e-11	
16	7.38e-10	7.38e-30	2.13e-09	2.13e-29	2.26e-05	2.26e-05	2.51e-13	4.60e-32	
32	7.38e-10	7.38e-30	2.13e-09	2.13e-29	2.91e-06	2.91e-06	2.51e-13	2.51e-33	
64	7.38e-10	7.38e-30	2.13e-09	2.13e-29	3.07e-07	3.07e-07	2.51e-13	2.51e-33	

Table 4 C $||e_r||_{L_\infty}$, $h = \frac{t-t_0}{100 points}$, d - differentiation step, $n = 3, n-\nu = 0.7, \nu = 2.3$

	Function (18)		Function (19)		Function (20)		Function (21)		
N	$1.0e-06$	$1.0e-16$	$1.0e-06$	$1.0e-16$	$1.0e-06$	$1.0e-16$	$1.0e-06$	$1.0e-16$	$\leftarrow$ **d**
4	6.54e-15	6.14e-15	3.90e-15	3.93e-15	nan	nan	1.08e-04	1.08e-04	
8	6.54e-15	6.14e-15	3.90e-15	3.93e-15	nan	nan	6.16e-13	3.57e-13	
16	6.54e-15	6.14e-15	3.90e-15	3.93e-15	nan	nan	2.53e-13	5.86e-15	
32	6.54e-15	6.14e-15	3.90e-15	3.93e-15	nan	nan	2.53e-13	5.86e-15	
64	6.54e-15	6.14e-15	3.90e-15	3.93e-15	nan	nan	2.53e-13	5.86e-15	

Results presented in tables 1-4 confirm shape of integrand as the main factor which determines accuracy of FOD computation. The term "shape of integrand" describes the behavior of a function in a given interval. In this scope: if we say "dramatical shape changes of integrand", we mean certain behavior of a function, in which the values of its derivatives (1^{st}, 2^{nd} and higher) assume high values.

The integrand included in RL and C formulas can be divided into a kernel and a function, i.e.

$$ {}_0^{RL}\mathrm{D}_t^{(\nu)} f(t) = \frac{1}{\Gamma(\nu)} \int_0^t \underbrace{(t-\tau)^{\nu-n}}_{\text{kernel}} f(\tau)\, d\tau, \text{ for } n-1 < \nu < n. $$

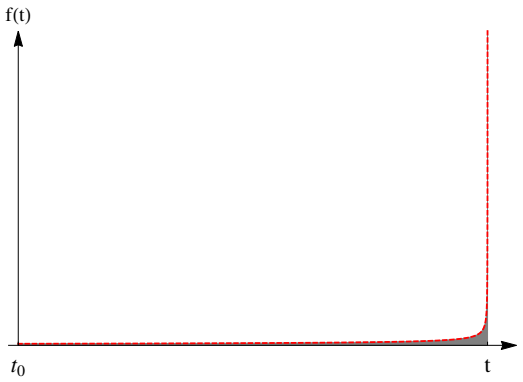


Fig. 4 Plot of the kernel of integrand in RL and C formulas.

The kernel, which includes singularity determines difficulty of FOD/FOI computation. By applying Gauss-Jacobi quadrature for computation, it is possible to exclude the kernel from numerical integration and compute it by applying formulas for the weights of the quadrature (9) which assures significantly higher accuracy. However, results presented in tables 1-4 show that the accuracy can also be decreased by the shape of the function. For this reason an amount of sampling points (N) required for highest accuracy of FOD for a particular function varies from 4 to 64.

Accuracy of integer order numerical differentiation depends mainly on the width of sampling step of a function. As it is easy to see, the width required for the highest accuracy is beyond the capabilities of the standard double precision computer arithmetic. In this scope, an application of high precision libraries for programming is required. How many points are included is of secondary importance. However, 3-points stencil central difference is an optimum for the most cases. More points does not increase the accuracy.

8 Conclusions

FOD can be computed by using numerous approaches. It includes application of integer order differentiation of fractional order integral (Riemann-Liouville formula) or fractional order integration of integer order derivative (Caputo formula).

The goal of numerical experiment described in the paper was to compare accuracy of FOD computation by applying both approaches.

During the experiment we calculated FOD of selected orders for exponential, power and periodic functions in various ranges.

As it was mentioned in the problem description, computation of FOD with high accuracy is a real challenge for the computer and it can not be succeed by applying standard numerical methods. For this purpose it is required application of Gauss-Jacobi quadrature with adapted weight function for integration part and central differences method of higher order for integer order differentiation. Additionally, the standard double precision C++ programming language limitations were eliminated by applying for programming arbitrary precision with a help of mathematical libraries (GNU MPFR/GNU GMP and MPFR C++).

The resulting conclusion is that high accuracy computation of FOD can be achieved by applying both approaches (RL and C), provided the appropriate numerical method of integration and differentiation is used. However, due to the fact that accuracy of numerical integration and differentiation generally depends on a shape of a function, by applying an appropriate sequence of numerical calculations (RL or C formula) selected according to a shape of a particular function, can result with significantly higher accuracy.

An important advantage of RL formula over C formula is its capability of computing FOD of function which n^{th} derivative does not exist (see results for computation of FOD of order 2.3 for a power function). In such case, application of C formula gives nan (not a number) result, where in the same situation, RL method results with accurate value of FOD.

Remark: To be able to apply C formula for computation of FOD for any function, it is required to apply equivalent Riemann-Liouville formula (6).

References

[1] I. Podlubny. *Fractional Differential Equations*. Academic Press, INC, San Diego Ca, 1999.

[2] J. Jiang, D. Cao, and H. Chen. Boundary value problems for fractional differential equation with causal operators. *Applied Mathematics and Nonlinear Sciences*, 1(1):11–22, 2016. DOI: 10.21042/AMNS.2016.1.00022.

[3] I. R. Birs, C. I. Muresan, S. Folea, and O. Prodan. A Comparison between Integer and Fractional Order PDμ Controllers for Vibration Suppression. *Applied Mathematics and Nonlinear Sciences*, 1(1):273–282, 2016. DOI: 10.21042/AMNS.2016.1.00022.

[4] P. Ostalczyk. *Zarys rachunku różniczkowego i całkowego ułamkowych rzędów*. Wydawnictwo Politechniki Łódzkiej, Łódź, Poland, 2006.

[5] D. W. Brzeziński and P. Ostalczyk. High-accuracy numerical integration methods for fractional order derivatives and integrals computations. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 62(4):723–733, 2014.

[6] D. W. Brzeziński. Accuracy problems of numerical calculation of fractional order derivatives and integrals applying the riemann-liouville/caputo formulas. *Applied Mathematics and Nonlinear Sciences*, 1(1):23–43, 2016.

[7] Torbjörn Granlund et al. *gmp: GMP is a free library for precision arithmetic (version 6.0.0a)*, 2015. `https://gmplib.org/`.

[8] Guillaume Hanrot et al. *mpfr: The MPFR library for multiple-precision floating-point computations with correct rounding. (version 3.13)*, 2015. `http://www.mpfr.org/`.

[9] Pavel Holoborodko. *High-performance C++ interface for MPFR library (version 3.6.2)*, 2015. `http://www.holoborodko.com/pavel/mpfr/`.

[10] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions. Applied Mathematics Series*. Cambridge University Press, 1968.

[11] N. Hale and A. Townsend. Fast and accurate computation of gauss-legendre and gauss-jacobi quadrature nodes and weights. Oxford Centre for Collaborative Applied Mathematics, 2012. OCCAM Preprint Number 12/79.

[12] B.P. Demidowicz, I.A. Maron, and E.Z. Szuwałowa. *Metody Numeryczne*. Państwowe Wydawnictwo Naukowe,

Warszawa, Poland, 1965.
[13] Pavel Holoborodko. *Central Differences*, 2009. http://www.holoborodko.com.
[14] J. M. Müller, N. Brisebarre, F. De Dinechin, C. P. Jeannerod, V. Lefevre, G. Melquiond, N. Revol, D. Stehle, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhauser, New York, NY, 2010.
[15] K. R. Ghazi, V. Lefevre, P. Theveny, and P. Zimmermann. Why and how to use arbitrary precision. *IEEE Computer Society*, 12(3):1–5, 2001.
[16] Microprocessor Standards Committee. *IEEE Standard for Floating-Point Arithmetic*, 2008. http//dox.doi.org/10.1109/IEEESTD.2008.4610935.
[17] D. W. Brzeziński and P. Ostalczyk. Accuracy assessment of fractional order derivatives and integrals numerical computations. *Journal of Applied Nonlinear Dynamics*, 4(1):53–65, 2015.