Computational Methods for Fractional Calculus: A Critical Analysis of Best Practices

Elena Vasquez, Julian R. Lee, and Sofia N. Patel

Department of Mathematical Sciences, University of California, San Diego, La Jolla, CA 92093, USA

Abstract: The solution of fractional-order differential problems requires in the majority of cases the use of some computational approach. In general, the numerical treatment of fractional differential equations is much more difficult than in the integer-order case, and very often non-specialist researchers are unaware of the specific difficulties. As a consequence, numerical methods are often applied in an incorrect way or unreliable methods are devised and proposed in the literature. In this paper we try to identify some common pitfalls in the use of numerical methods in fractional calculus, to explain their nature and to list some good practices that should be followed in order to obtain correct results.

Keywords: fractional differential equations; numerical methods; smoothness assumptions; persistent memory

1. Introduction

The increasing interest in applications of fractional calculus, together with the difficulty of finding analytical solutions of fractional differential equations (FDEs), naturally forces researchers to study, devise and apply numerical methods to solve a large range of ordinary and partial differential equations with fractional derivatives.

The investigation of computational methods for fractional-order problems is therefore a very active research area in which, each year, a large number of research papers are published.

The task of finding efficient and reliable numerical methods for handling integrals and/or derivatives of fractional order is a challenge in its own right, with difficulties that differ in character but are no less severe than those associated with finding analytical solutions. The specific nature of these operators involves computational challenges which, if not properly addressed, may lead to unreliable or even
wrong results.

Unfortunately, the scientific literature is rich with examples of methods that are inappropriate for fractional-order problems. In most cases these are just methods that were devised originally for standard integer-order operators then applied in a naive way to their fractional-order counterparts; without a proper knowledge of the specific features of fractional-order problems, researchers are often unable to understand why unexpected results are obtained.

The main aims of this paper are to identify a few major guidelines that should be followed when devising reliable computational methods for fractional-order problems, and to highlight the main peculiarities that make the solution of differential equations of fractional order a different—but surely more difficult and stimulating—task from the integer-order case. We do not intend merely to criticize

weak or wrong methods, but try to explain why certain approaches are unreliable in fractional calculus and, where possible, point the reader towards more suitable approaches.

This paper is mainly addressed at young researchers or scientists without a particular background in the numerical analysis of fractional-order problems but who need to apply computational methods to solve problems of fractional order. We aim to offer in this way a kind of guide to avoid some of the most common mistakes which, unfortunately, are sometimes made in this field.

The paper is organized in the following way. After recalling in Section 2 some basic definitions and properties, we illustrate in Section 3 the most common ideas underlying the majority of the methods proposed in the literature: very often the basic ideas are not properly recognized and common methods are claimed to be new. In Section 4 we discuss why polynomial approximations can be only partially satisfactory for fractional-order problems and why they are unsuitable for devising high-order methods (as has often been proposed). The major problems related to the nonlocality of fractional operators are addressed in Sections 5 and Section 6 discusses some of the most powerful approaches for the efficient treatment of the memory term. Some remarks related to the numerical treatment of fractional partial differential equations are presented in Section 7 and some final comments are given in Section 8.

2. Basic Material and Notations

With the aim of fixing the notation and making available the most common definitions and properties for further reference, we recall here some basic notions concerning fractional calculus.

For $\alpha > 0$ and any $t_0 \in \mathbb{R}$, in the paper we will adopt the usual definitions for the fractional integral of Riemann–Liouville type

$$J_{t_0}^{\alpha} f \quad (t) = \frac{1}{\Gamma(\alpha)} \int_{t_0}^{t} (t-\tau)^{\alpha-1} f(\tau)\mathrm{d}\tau, \quad t > t_0, \tag{1}$$

for the fractional derivative of Riemann–Liouville type

$$\frac{1}{\quad} \frac{\mathrm{d}^m}{\quad} \int_{t0}^{t}$$

and for the fractional derivative of Caputo type

$$\frac{\quad}{\quad} \quad 1 \quad {}^{Z\,t} \quad m-{}^{\alpha}-1 \quad f$$

$$(m)(\tau)\mathrm{d}_{\tau}, \quad t > t_0,$$

$$\begin{matrix} c \\ D \end{matrix} \quad t\alpha 0\,(t) := Jtm-\alpha Dm\, f(t) = \quad (t-\tau) \quad (3$$

$$f \quad)$$

$$\mathrm{RL}Dt\alpha 0\, f(t) := DmJtm0-\alpha\, f(t) = \Gamma(m-\alpha)\,\mathrm{d}t \quad (t-\tau)m-\alpha-1\, f(\tau)\mathrm{d}\tau, \quad t > t0 \tag{2}$$

$$0 \quad \Gamma(m-\alpha)\, t0$$

with $m = d\alpha e$ the smallest integer greater than or equal to α.

We refer to any of the many existing textbooks on this subject (e.g., [1–6]) for an exhaustive treatment of the conditions under which the above operators exist and for their main properties. We just recall here the relationship between ${}^{RL}D_t^{\alpha}0$ and ${}^{C}D_t^{\alpha}0$ expressed as

$$ {}^{C}D_{t\,0}^{\alpha} f(t) = {}^{RL}D_{t\,0}^{\alpha}\big(\quad {}_{m-1} \; ; \; 0 \big) f - T[f \quad t \quad](t), \tag{4} $$

where $T_{m-1}[f; t_0]$ is the Taylor polynomial of degree $m - 1$ for the function f about the point t_0,

$$ T_{m-1}[f; \quad t_0](t) \quad = \quad m\sum -1 \quad (t \quad - \quad t0)k \quad \frac{}{f} \quad (k)(t0). \tag{5} $$
$$ k=0 \quad k! $$

Moreover, we will almost exclusively consider initial value problems of Cauchy type for FDEs with the Caputo derivative, i.e.,

$$ ({}^{C}Dt^{\alpha}0y(t) = f(t, y(t)) \quad (6) \qquad {}^{(m-1)}(t_0) = y_0^{(m-1)} \quad y(t0) = y0, \, y\,(t0) = y01), \, . $$

. . , y,

0 (

for some assigned initial values $y_0, y_0({}^{(1)}, \ldots, y_0{}^{(m-1)}$. A few general comments will also be made regarding problems associated with partial differential equations.

3. Novel or Well-Established Methods?

Quite frequently, one sees papers whose promising title claims the presentation of “new methods” or “a family of new methods” for some particular fractional-order operator. Papers of this type immediately capture the attention of readers eager for new and good ideas for numerically solving problems of this type.

But reading the first few pages of such papers can be a source of frustration, since what is claimed to be new is merely an old method applied to a particular (maybe new) problem. Now it is understandable that sometimes an old method is reinvented by a different author, maybe because it can be derived by some different approach or because the author is unaware of the previously published result (perhaps because it was published under an imprecise or misleading title). In fractional calculus, however, a different and quite strange phenomenon has taken hold: well-known and widely used methods are often claimed as “new” just because they are being applied to some specific problem. It seems that some authors are unaware that it is the development of new ideas and new approaches that leads to methods that can be described as new—not the application of known ideas to a particular problem. Even the application of well-established techniques to any of the new operators, obtained by simply replacing the kernel in the integral (1) with some other function, cannot be considered a truly novel method, especially when the extension to the new operator is straightforward.

Most of the papers announcing “new” methods are instead based on ideas and techniques that were proposed and studied decades ago, and sometimes proper references to the original sources are not even given.

In fact, there are a few basic and powerful methods that are suitable and extremely popular for fractional-order problems, and many proposed “new methods” are simply the application of the ideas behind them. It may therefore be useful to illustrate the main and more popular ideas that are most frequently (re)-proposed in fractional calculus, and to outline a short history of their origin and development.

3.1. Polynomial Interpolation and Product-Integration Rules

Solving differential equations by approximating their solution or their vector field by a polynomial interpolant is a very old and common idea. Some of the classical linear multistep methods for ordinary differential equations (ODEs), specifically those of Adams–Bashforth or Adams–Moulton type, are based on this approach.

In 1954 the British mathematician Andrew Young proposed [7,8] the application of polynomial interpolation to solve Volterra integral equations numerically. This approach turns out to be suitable for FDEs since (6) can be reformulated as the Volterra integral equation

$$yf(u, y(u))\mathrm{d}u. \tag{7}$$

The approach proposed by Young is to define a grid $\{t_n\}$ on the solution interval $[t_0, T]$ (very often, but not necessarily, equispaced, namely $t_n = t_0 + hn$, $h = (T - t_0)/N$) and to rewrite (7) in a piecewise way as

$$n-1 \quad (t\ ) = T \quad [f\ t\ ](t\ ) + \frac{1}{\Gamma(\alpha)} \sum_{j=0} \quad {}^{j+1}_{t_j} (t\ - u)^{\alpha-1} f(u\ y(u))\ \mathrm{z}\, t \quad y \quad {}_n$$

$$m-1;\ 0 \qquad n \quad \text{——}\sum n \qquad , \qquad \mathrm{d}u, \qquad (8)$$

then to replace, in each interval $[t_j, t_{j+1}]$, the vector field $f(u, y(u))$ by a polynomial that interpolates to f on the grid. This approach is particularly simple if one uses polynomials of degree 0 or 1 because then one can determine the approximation solely on the basis of the data at one of the subinterval’s end points (degree 0; the *product rectangle method*) or at both end points (degree 1; the *product trapezoidal method*); thus, in these cases one need not introduce auxiliary points inside the interval or points outside the interval. Neither of these methods can yield a particularly high order of convergence, but as we shall demonstrate in Section 4, the analytic properties of typical solutions to fractional differential equations make it very difficult and cumbersome to achieve high-order accuracy irrespective of the technique used. Consequently, and because these techniques have been thoroughly investigated with respect to their convergence properties [9] and their stability [10] and are hence very well understood, the product rectangle and product trapezoidal methods are highly popular among users of fractional order models.

Higher-order methods have occasionally been proposed [11,12] but—as indicated above and discussed in more detail in Section 4—they tend to require rather uncommon properties of the exact solutions to the given problems and therefore are used only infrequently. We also have to notice that the effects of the lack of regularity on the convergence properties of product-integration rules have been studied since 1985 for Volterra integral equations [13] and since 2004 for the specific case of FDEs [14].

3.2. Approximation of Derivatives: L1 and L2 Schemes

A classical numerical technique for approximating the Caputo differential operator from (3) is the so-called *L1 scheme*. For $0 < \alpha < 1$, the definition of the Caputo operator becomes

$$C D t \alpha_0 \quad f \quad (t) = \frac{1}{\Gamma(1-\alpha)} \int_{t_0}^{t} (t-\tau)^{-\alpha} f\ (\tau) \mathrm{d}\tau \quad 1 \quad 0 \quad \text{for } t > t0.$$

The idea ([15], Equation (8.2.6)) is to introduce a completely arbitrary (i.e., not necessarily uniformly spaced) mesh $t_0 < t_1 < t_2 < \ldots < t_N$ and to replace the factor $f^{0}(\tau)$ in the integrand by the approximation

$$f \quad 0(\tau) \quad \approx \quad \_\_\_\_\_ f(tj+1) \quad -- \quad tf_j(tj)$$

whenever $\tau \in (tj, tj+1)$. t_f+1

$$C \quad \alpha \quad f(tn) \quad \approx \quad \mathrm{C}Dt\alpha 0,\mathrm{L}1 \quad f \quad (t_n) = \frac{1}{\Gamma(2-\alpha)} \sum_{j=0} \quad 1 \quad n-1 \quad wn-j-1,n(f(tn-j) \quad - \quad f(tn-j-1))$$

$$Dt0$$

$$w_{\mu,n} = \frac{(t_n - t_\mu)^{1-\alpha} - (t_n - t_{\mu+1})^{1-\alpha}}{t_{n-\mu} - t_{n-\mu-1}}$$

For smooth functions f (but only under this assumption!) and an equispaced mesh $t_j = t_0 + jh$, the convergence order of the L1 method is $O(h^{1-\alpha})$.

By construction, the L1 method is restricted to the case $0 < \alpha < 1$. For $\alpha \in (1, 2)$, the *L2 method* ([15], §8.2) provides a useful modification. In its construction, one starts from the representation

$${}_{0}^{C}D_t^{\alpha} f(t) = \frac{1}{\Gamma(2-\alpha)} \int_0^t (t-\tau)^{1-\alpha} f''(\tau)\, d\tau,$$

which is valid for these values of α. Using now a uniform grid $t_j = t_0 + jh$, one replaces the second derivative of f in the integrand by its central difference approximation,

$$f''(t_n - \tau) \approx \frac{1}{h^2} \left(f(t_{n-k+1}) - 2f(t_{n-k}) + f(t_{n-k-1}) \right)$$

for $\tau \in [t_k, t_{k+1}]$, which yields

$${}_{0}^{C}D_t^{\alpha} f(t_n) \approx {}_{0}^{C}D_{t_\alpha}^{\alpha} L_2 f(t_n) = \frac{h^{-\alpha}}{\Gamma(3-\alpha)} \sum_{k=-1}^{n} w_{k,n} f(t_{n-k}),$$

where now

$$w_{k,n} = \begin{cases} 2^{2-\alpha} - 3, & \text{for } k = -1, \\ (k+2)^{2-\alpha} - 3(k+1)^{2-\alpha} + 3k^{2-\alpha} - (k-1)^{2-\alpha}, & \text{for } k = 0, \text{ for } 1 \le k \le n-2, \\ (n-1)^{2-\alpha} - n^{2-\alpha} + (n-\alpha)(n-1)^{1-\alpha} - (n-1-\alpha)n^{1-\alpha}, & \text{for } k = n-1, \\ 2^{1-\alpha} - (n-1)^{2-\alpha} + (n-1-\alpha)n^{1-\alpha} - (n-\alpha)(n-1)^{1-\alpha}, & \text{for } k = n. \end{cases}$$

A disadvantage of this method is that it requires the evaluation for f at the point $t_{n+1} = (n+1)h$ which is located outside the interval $[0, t_n]$.

[1] $\Gamma(3-\alpha) \sum_{k=-1}^{n} w_{k,n} f(t_{n-k})$

The central difference used in the definition of the L2 method is symmetric with respect to one of the endpoints of the associated subinterval $[t_k, t_{k+1}]$, not with respect to its mid point. If this is not desired, one may instead use the alternative

$$f''(t_n - \tau) \approx \frac{1}{2h}\left(f(t_{n-k-2}) - f(t_{n-k-1}) + f(t_{n-k+1}) - f(t_{n-k})\right)$$

on this subinterval. This leads to the *L2C method* [16]

$$CD^{\alpha} f(t_n) \approx {}_{C}D^{\alpha}_{0,L2C} f(t_n) = t_0$$

$$w_{k,n} = \begin{cases} \frac{1}{2}2^{2-\alpha} - 2^{2-\alpha} + \frac{3}{2}2^{2-\alpha} & \text{for } k = -1, \text{ for } k = 0, \text{ for } k = 1, \\ (k+2)^{2-\alpha} - 2(k+1)^{2-\alpha} + 2(k-1)^{2-\alpha} - (k-2)^{2-\alpha} & \text{for } 2 \le k \le n-2, \text{ for } k = n-1, \text{ for } k = n, \text{ for } k = n+1. \end{cases}$$

Like the L2 method, the L2C method also requires the evaluation of f outside the interval $[0, t_n]$; one has to compute $f((n+1)h)$ and $f(-h)$. Both the L2 and the L2C method exhibit $O(h^{3-\alpha})$ convergence behavior for $1 < \alpha < 2$ if f is sufficiently well behaved; the constants implicitly contained in the O-terms seem to be smaller for the L2 method in the case $1 < \alpha < 1.5$ and for the L2C method if $1.5 < \alpha < 2$.

In the limit case $\alpha \to 1$, the L2 method reduces to first-order backward differencing, and the L2C method becomes the centered difference of first order; for $\alpha \to 2$ the L2 method corresponds to the classical second-order central difference.

3.3. Fractional Linear Multistep Methods

Fractional linear multistep methods (FLMMs) are less frequently used since their coefficients are, in general, not known explicitly but it is necessary to devise some algorithm for their (technically often difficult) computation. Nevertheless, since these methods allow us to overcome some of the issues associated with other approaches, it is worth giving a short presentation of their properties.

FLMMs were proposed by Lubich in 1986 [17] and studied in the successive works [18–20]. They extend to fractional-order integrals the quadrature rules obtained from standard linear multistep methods (LMMs) for ODEs.

Let us consider a classical k-step LMM of order $p > 0$ with first and second characteristic polynomials $\rho(z) = \rho 0 z^k + \rho 1 z^{k-1} + \cdots + \rho k$ and $\sigma(z) = \sigma 0 z^k + \sigma 1 z^{k-1} + \cdots + \sigma k$, namely

$$\sum_{j=0}^{k} \rho_j y_{n-j} = h \sum_{j=0}^{k} \sigma_j f(t_{n-j}), \quad \text{where } \delta(\xi) = \frac{\rho(1/\xi)}{\sigma(1/\xi)} \text{ is the generating function.} \tag{9}$$

FLMMs generalizing LMMs (9) for solving FDEs (7) are expressed as

$$y_n = T_{m-1}[f; t_0](t) + h^\alpha \sum_{j=0}^{n} w_{n,j} f(t_j, y_j) + h^\alpha \sum_{j=0}^{v} \omega_{n-j}^{(\alpha)} f(t_j, y_j), \tag{10}$$

where the convolution weights $\omega_n^{(\alpha)}$ are obtained from the power series expansion of $(\delta(\xi))^{-\alpha}$, namely $\sum_{n=0}^{\infty} \omega_n^{(\alpha)} \xi^n = (\delta(\xi))^{-\alpha}$,

and the $w_{n,j}$ are some starting weights that are introduced to deal with the lack of regularity of the solution at the origin; they are obtained by solving, at each step n, the algebraic linear systems

$$\sum_{j=0}^{v} w_{n,j} j^{\gamma} = \frac{\Gamma(\gamma+1)}{\Gamma(\gamma+\alpha+1)} n^{\gamma+\alpha}, \tag{11}$$

$\mathcal{A}_p = \{\gamma \in \mathbb{R} \mid \gamma = i + j\alpha,\ i, j \in \mathbb{N},\ \gamma < p - 1\}$ and $v + 1$ the cardinality of A_p.

The intriguing property of FLMMs is that, unlike product-integration rules, they are able to preserve the same convergence order p of the underlying LMMs if the LMM satisfies certain properties: it is required that $\delta(\xi)$ has no zeros in the closed unit disc $|\xi| \leq 1$ except for $\xi = 1$, and $|\arg \delta(\xi)| < \pi$ for $|\xi| < 1$. Thus, high-order FLMMs are possible without requiring the imposition of artificial smoothness assumptions as is required for methods based on polynomial interpolation.

But the price to be paid for this advantage may be not negligible: the convolution weights $\omega_n^{(\alpha)}$ are not known explicitly and must be computed by some (possibly sophisticated) method (a discussion for the general case is available in [17–20] while algorithms for FLMMs of trapezoidal type are presented in [21]). Moreover, high-order methods may require the solution of large or very large systems (11) depending on the equation order α and the convergence order p of the method; in some cases these systems are so ill-conditioned as to affect the accuracy of the method, a problem addressed in depth in [22].

One of the simplest methods in this family is obtained from the backward Euler method, whose generating function is $\delta(\xi) = (1 - \xi)$. Its convolution weights are hence the

coefficients in the asymptotic expansion of $(1-\xi)^{-\alpha}$, i.e., they are the coefficients in the binomial series

$$\omega_j^{(\alpha)} = (-1)^j \binom{-\alpha}{j} = \frac{\Gamma(-\alpha+1)}{j!\Gamma(-\alpha-j+1)}$$

and no starting weights are necessary since the convergence order is $p = 1$ and hence A_p is the empty set. One recognizes easily that the so-called Grünwald-Letnikov scheme is obtained in this case. Although this scheme was discovered in the nineteenth century in independent works of Grünwald and Letnikov, its interpretation as an FLMM may facilitate its analysis.

4. Classical Approximations Will Not Give High-Order Methods

Solutions of fractional-derivative problems typically exhibit weak singularities. This topic is discussed at length in the survey chapter [23] and it is known since earlier works on Volterra integral equations [24,25]. This singularity is a consequence of the weakly singular behavior of the kernels of integral and fractional derivatives and its importance, from a physical perspective, is related to the natural emergence of completely monotone (CM) relaxation functions in models whose dynamics is governed by these operators [26,27]; CM relaxation behaviors are indeed typical of viscoelastic systems with strongly dissipative energies [28].

In the present section we shall examine the effects of the singular behavior on numerical methods, in the context of initial value problems such as (6).

To grasp quickly the main ideas, we focus on a very simple particular case of (6): the problem

$${}^{C}D_0^{\alpha} y(t) = 1 \text{ for } t \in (0, T], \tag{12}$$

where $0 < \alpha < 1$ and, for the moment, we do not prescribe the initial condition at $t = 0$. The general solution of (12) is $x\alpha$

$$y(t) = \frac{}{\Gamma(1+\alpha)} + b, \text{ where } b \text{ is an arbitrary constant.} \tag{13}$$

This solution lies in $C[0, T] \cap C^1(0, T]$ but not in $C^1[0, T]$. This implies that standard techniques for integer-derivative problems, which require that $y \in C^1[0, T]$ (or a higher degree of regularity), cannot be used here without some modification. In particular one cannot perform a Taylor series expansion of the solution around $t = 0$ because $y^0(0)$ does not exist.

What about the initial condition? If we prescribe a condition of the form $y(0) = y_0$ we get $b = y_0$ in (13), but the solution is still not in $C^1[0, T]$. One might hope that a Neumann-type condition of the form

$y^0(0) = 0$ would control or eliminate the singularity in the solution, but a consideration of (13) shows that it is impossible to enforce such a condition; that is, the problem ${}^{C}D_0^{\alpha}y(t) = 1$ on $(0, T]$ with $y^0(0) = 0$ has no solution. This seems surprising until we recall a basic property of the Caputo derivative from ([1], Lemma

3.11): if $m - 1 < \beta < m$ for some positive integer m and $z \in C^m[0, T]$, then $\lim_{t\to 0} {}^{C}D_0^{\beta}z(t) = 0$. Hence, if in (12) one has $y \in C^1[0, T]$, then taking the limit as $t \to 0$ in (12) we get $0 = 1$, which is impossible. That is, any solution y of (12) cannot lie in $C^1[0, T]$.

One can present this finding in another way: for the problem ${}^{C}D_0^{\alpha}y(t) = f(t)$ on $(0, T]$ with $f \in C[0, T]$, if the solution $y \in C^1[0, T]$, then one must have $f(0) = 0$. This result is a special case of ([1], Theorem 6.26).

Remark 1. *For the problem* ${}^{C}D_0^{\alpha}y(t) = f(t)$ *on* $(0, T]$ *with* $0 < \alpha < 1$*, if one wants more smoothness of the solution* y *on the closed interval* $[0, T]$*, then one must impose further conditions on the data: by ([1], Theorem 6.27), for each positive integer* m*, one has* $y \in C^m[0, T]$ *if and only if* $0 = f(0) = f^0(0) = \cdots = f^{(m-1)}(0)$.

Conditions such as $f(0) = 0$ (and the even stronger conditions listed in Remark 1) impose an artificial restriction on the data f that should be avoided. Thus we continue by looking carefully at the consequence of dealing with a solution of limited smoothness.

Returning to (12) and imposing the initial condition $y(0) = b$, the unique solution of the problem is given by (13), where b is now fixed. Most numerical methods for integer-derivative initial value problems are based on the premise that on any small mesh interval $[t_i, t_{i+1}]$, the unknown solution can be approximated to a high degree of accuracy by a polynomial of suitable degree. But is this true of the function (13)? We now investigate this question.

Consider the interval $[0, h]$, where $h = t_1$. This is the mesh interval where the solution (13) is worst behaved.

Lemma 1. *Let* $\alpha \in (0, 1)$*. Consider the approximation of* t^{α} *by a linear polynomial* $c_0 + c_1 t$ *on the interval* $[0, h]$*. Suppose this approximation is uniformly* $O(h^{\beta})$ *accurate on* $[0, h]$ *for some fixed* $\beta > 0$*. Then one must have* $\beta \leq \alpha$.

Proof. Our hypothesis is that $|t^{\alpha} - (c_0 + c_1 t)| \leq Ch^{\beta}$ for all $t \in [0, h]$ and some constant C that is independent of h and t. Consider the values $t = 0$, $t = h/2$ and $t = h$ in this inequality: we get

$$\begin{aligned} 0 - (c0 + 0) &= O(h^{\beta}), \\ (h/2)^{\alpha} - (c_0 + c_1 h/2) &= O(h^{\beta}), \\ h\alpha - (c0 + c1h) &= O(h^{\beta}). \end{aligned}$$

The first equation gives $c_0 = O(h^{\beta})$. Hence the other equations give $(h/2)^{\alpha} - c_1 h/2 = O(h^{\beta})$ and $h^{\alpha} - c_1 h = O(h^{\beta})$. Eliminate c_1 by multiplying the first equation by 2 then subtracting from the other equation;

this yields $h^\alpha - 2(h/2)^\alpha = O(h^\beta)$. But this cannot be true unless $\beta \leq \alpha$, since the left-hand side is simply a multiple of h^α because α 6= 1. □

Lemma 1 says that the approximation of t^α on [0, h] by *any* linear polynomial is at best $O(h^\alpha)$. But the order of approximation $O(h^\alpha)$ of t^α on [0, h] is also achieved by the constant polynomial 0. That is: using a linear polynomial to approximate t^α on [0, h] does not give an essentially better result than using a constant polynomial. In a similar way one can show that using polynomials of higher degree does not improve the situation: the order of approximation of t^α on [0, h] is still only $O(h^\alpha)$. This is a warning that when solving typical fractional-derivative problems, high-degree polynomials may be no better than low-degree polynomials, unlike the classical integer-derivative situation.

One can generalize Lemma 1 to any $\alpha > 0$ with α not an integer, obtaining the same result via the same argument. Furthermore, our investigation of the simple problem (12) can be readily generalised to the much more general problem (6); see ([1], Section 6.4).

Implications for the Construction of Difference Schemes

The discussion earlier in Section 4 implies that, to construct higher-order difference schemes for typical solutions of problems such as (12) and (6), one must use non-classical schemes, since the classical schemes are constructed under the assumption that approximations by higher-order polynomials gives greater accuracy. The same idea is developed at length in [29], one of whose results we now present.

Note: although [29] discusses only boundary value problems, an inspection reveals that its arguments and results are also valid (mutatis mutandis) for initial value problems such as (6) when $f = f(t)$, i.e.,
when the problem (6) is linear.

Let $\alpha > 0$ be fixed, with α not an integer. Consider the problem $D^\alpha y = f$ on [0, T] with $y(0) = 0$. Assume that the mesh on [0, T] is equispaced with diameter h, i.e., $x_i = ih$ for $i = 0, 1, \ldots, N$. Suppose that the difference scheme used to solve $D^\alpha y = f$ at each point x_i for $i > 0$ is $\sum_{j=0}^{i} a_{ij} y_j^N = f(t_i)$. It is reasonable to assume that $|a_{ij}| = O(h^{-\alpha})$ for all i and j since we are approximating a derivative of order α (one can check that almost all schemes proposed for this problem have this property).

We have the following variant of ([29], Theorem 3.3).

Theorem 1. *Assume that our scheme achieves order of convergence p for some $p > \alpha$ when $f(t) = Ct^k$ for all $k \in \{0, 1, \ldots, dp - \alpha - 1e\}$. Then for each fixed positive integer i, the coefficients of the scheme must satisfy the following relationship:*

$$\lim_{h \to 0} \left(h^\alpha \sum_{j=0}^{i} j^{k+\alpha} a_{ij} \right) = \frac{i^k \Gamma(\alpha + k + 1)}{\Gamma(k+1)} \quad \text{for } k = 0, 1, \ldots, dp - \alpha - 1e. \tag{14}$$

Proof. Fix $k \in \{0, 1, \ldots, dp - \alpha - 1e\}$. This implies that $k < p - \alpha$. Choose for simplicity

$$f(t) = \frac{\Gamma(k + \alpha + 1)}{\Gamma(k+1)} t^k.$$

Then the true solution of our initial value problem is $y(t) = t^{k+\alpha}$. Fix a positive integer i. Then

$$\sum_{j=0}^{i} aijyNj = f(ti) = (ih)^k. \qquad \frac{\Gamma(k+\alpha+1)}{\Gamma(k+1)}$$

Hence, using the hypothesis that our scheme achieves order of convergence p and $|a_{ij}| = O(h^{-\alpha})$,

$$\lim_{h\to 0} \left[h^{\alpha} \sum_{j=0}^{i} jk+\alpha aij \right] = \lim_{h\to 0} h^{-k} \sum_{j=0}^{i} aijy(tj)$$

$$= \lim_{h\to 0} h^{-k} \left\{ \frac{\Gamma(k+\alpha+1)}{\Gamma(k+1)} (ih)^k + \sum_{j=0}^{i} a_{ij} \left[y(x_j) - y_j^N \right] \right\}$$

$$= \lim_{h\to 0} \left[\frac{\Gamma(k+\alpha+1)}{\Gamma(k+1)} i^k + \mathcal{O}(h^{p-\alpha-k}) \right]$$

$$= \frac{\Gamma(k+\alpha+1)}{\Gamma(k+1)} ik,$$

since $k < p - \alpha$. □

Theorem 1 implies that schemes that fail to satisfy (14) cannot achieve an order of convergence greater than $O(h^{\alpha})$ at each mesh point. (This is consistent with the approximation theory result of Lemma 1.)

For example, in the case $0 < \alpha < 1$, it follows from Theorem 1 that the well-known L1 scheme is at best $O(h^{\alpha})$ accurate.

Remark 2. *To avoid the consequences of results such as Theorem 1, one can impose data restrictions such as $f(0) = 0$. This is discussed in ([29], Section 5), where theoretical and experimental results show an improvement in the accuracy of standard difference schemes, but only for a restricted class of problems.*

5. Failed Approaches to Treat Non-Locality

Non-locality is one of the major features of fractional-order operators. Indeed, fractional integrals and derivatives are often introduced as a mathematical formalism with the primary purpose of encompassing hereditary effects in the modeling of real-life phenomena when theoretical or experimental observations suggest that the effects of external actions do not propagate instantaneously but depend on the history of the system.

On the one hand, non-locality is a very attractive feature that has driven most of the interest and success of the fractional calculus; on the other hand, non-locality introduces severe computational difficulties that researchers try to overcome in different ways.

Unfortunately, some attempts to treat non-locality are unreliable and lead to wrong results. This is the case of the naive implementation of the “finite memory principle” consisting in simply neglecting a

large amount of the history solution; since on the basis of this technique it is however possible to devise more sophisticated and accurate approaches, we postpone its discussion to Section 6.

We have also to mention methods based on some kind of fractional Taylor expansion of the solution, such as

$$y(t) = \sum_{k=0}^{\infty} Y_k (t - t_0)_{k\alpha},$$

where the coefficients Y_k are determined by some suitable numerical technique.

When solving integer-order differential equations, it is possible to use Taylor expansions to approximate the solution at a given point t_1 and hence reformulate the same expansion by moving the origin to the new point t_1, thus generating a step-by-step method in which the approximation at t_{n+1} is evaluated on the basis of the approximation at t_n (or at additional previous points).

With fractional-order equations, instead, the above expansion holds only with respect to the point t_0 (the initial or starting point of the fractional differential operator) and it is not possible to generate a step-by-step method. Expansions of this type are therefore able to provide an accurate approximation only locally, i.e., very close to the starting point t_0; consequently, as discussed in [30], methods based on these expansions are usually unsuitable for FDEs.

Another failed approach is based on an attempt to exploit the difference between $y(t_{n+1})$ and $y(t_n)$ in the integral formulation (7): rewrite the solution at t_{n+1} as some increment of the solution at t_n, i.e.,

$$y(t_{n+}1) = y(tn) + Gn(t, y(t)), \tag{15a}$$

then approximate the increment

$$G_n(t, y(t)) = \frac{1}{\Gamma(\alpha)} \int_{t_0}^{t_{n+1}} (t_{n+1} - u)^{\alpha-1} f(u, y(u))\mathrm{d}u - \frac{1}{\Gamma(\alpha)} \int_{t_0}^{t_n} (t_n - u)^{\alpha-1} f(u, y(u))\mathrm{d}u \tag{15b}$$

by replacing the vector field $f(t, y(t))$ in both integrals of (15b) by its (first-order) interpolating polynomial at the grid points t_{n-1} and t_n. Methods of this kind read as

$$y_{n+}1 = yn + Pn(y_n{-}1, yn), \tag{16}$$

with P_n a known function obtained by standard interpolation techniques. Approaches of this kind are called *two-step Adams–Bashforth methods* and attract researchers since they apparently transform the non-local problem into a local one (and thus, a difficult problem into a much easier one); in (15b) $G_n(t, y(t))$ is still a non-local term but these methods are strangely becoming quite popular despite the fact that, as discussed in [31], they are usually unreliable because in most cases they attempt to approximate the (implicitly) non-local contribution $G_n(t, y(t))$ by some purely local term.

Using interpolation at the points t_{n-1} and t_n to approximate $f(t, y(t))$ over the much larger intervals $[t_0, t_n]$ and $[t_0, t_{n+1}]$ is completely inappropriate. It is well known that polynomial interpolation may offer accurate approximations within the interval of the data points, in this case in $[t_{n-1}, t_n]$; but outside this interval (where an extrapolation is made instead of an interpolation), the approximation becomes more

and more inaccurate as the integration intervals $[t_0, t_n]$ and $[t_0, t_{n+1}]$ in (15b) become larger and larger, i.e., as the integration proceeds and n increases.

The consequence is that completely untrustworthy results must be expected from methods based on this idea.

Note that the fundamental flaw of this approach is not the decomposition (15) but the local (and hence inappropriate) way (16) in which the history is handled. Indeed, it is possible to construct technically correct and efficient algorithms on the basis of (15), for example if one treats the increment term (15b) by a numerical method that is cheaper in computational cost than the method used for the local term [32].

6. Some Approaches for the Efficient, and Reliable, Treatment of the Memory Term

The non-locality of the fractional-order operator means that it is necessary to treat the memory term in an efficient way. This term is commonly identified to be the source of a computational complexity which, especially in problems of large size, requires adequate strategies in order to keep the computational cost at a reasonable level, and indeed this observation has led to many investigations of (more or less successful) approaches to reduce the computational cost. It should be noted however that the high number of arithmetic operations is not the only potential difficulty that the memory term introduces. There is another more fundamental issue, which seems to have attracted much less attention: the history of the process not only needs to be taken into account in the computation but, in order to be properly handled, also needs to be *stored* in the computer's memory. While the required amount of memory is usually easily available in algorithms for solving ordinary differential equations, the memory demand may be too high for efficient handling in the case of, e.g., time-fractional partial differential equations where finite element techniques are used to discretize the spatial derivatives.

Most finite-difference methods for FDEs require at each time step the evaluation of some convolution sum of the form

$$y_n = \varphi_n + \sum_{j=0}^{n} c_j y_{n-j} \quad \text{or} \quad y_n = \varphi_n + \sum_{j=0}^{n} c_j f(t_{n-j}, y_{n-j}), \qquad n = 1, 2, \ldots, N, \tag{17}$$

where φ_n is a term which mainly depends on the initial conditions or other known information.

A naive straightforward evaluation of (17) has a computational cost proportional to $\mathcal{O}(N^2)$ and, when integration with a small-step size or on a large integration interval is required, the value of N can be extremely large and leads to prohibitive computational costs.

For this reason different approaches for a fast, efficient and reliable treatment of the memory term in non-local problems have been devised. We provide here a short description of some of the most interesting methods of this type. The influence of these approaches on the memory requirements will be addressed as well.

6.1. Nested Mesh Techniques

Several different concepts can be subsumed under the heading of so-called *nested meshes*. The general idea is based on the observation that the convolution sum in Equation (17) stems from a

discretization of a fractional integral or differential operator that uses all the previous grid points as nodes. One can then ask whether it is really neccessary to use all these nodes or whether one could save effort by including only a subset of them by using a second, less fine mesh—i.e., a mesh nested inside the original one.

6.1.1. The Finite Memory Principle

The simplest idea in this class is the *finite memory principle* ([5], §7.3). It is based on defining a constant $\tau > 0$, the so-called memory length, and replacing (for $t > t_0 + \tau$) the memory integral term that extends over the interval $[t_0, t]$ by the integral over $[t - \tau, t]$ with the same integrand function. Technically speaking, this amounts to "forgetting" the entire history of the process that is more than τ units of time in the past, so the memory has a finite and fixed length τ instead of the variable length $t - t_0$ that may, in a long running process, be very much longer. From an algorithmic point of view, the finite memory method truncates the convolution sum in Equation (17) to a sum where j runs from $n - \nu$ to n for some fixed ν. This has a number of significant advantages:

- The computational complexity of the nth time step is reduced from O(n) to O(1). Therefore, the combined total complexity of the overall method with N time steps is reduced from O(N^2) to O(N).
- At no point in time does one need to access the part of the process history that is more than ν time steps in the past. Therefore, all those previous time steps can be removed from the active memory, and the memory requirement also decreases from O(N) to O(1).

Unfortunately, this idea also has severe drawbacks. Specifically, it has been shown in [33] that the convergence order of the underlying discretization technique is lost completely. In other words, one cannot prove that the algorithm converges as the (maximal) step size goes to 0. Therefore, the method is not recommended for practical use.

6.1.2. Logarithmic Memory

To overcome the shortcomings of the finite memory principle, two related but not identical methods, both of which are also based on the nested mesh concept, have been developed in [33,34]. The common idea of both these approaches is the way in which the distant part of the memory is treated. Rather than ignoring it completely as the finite memory principle does, they do sample it, but on a coarser mesh; indeed the fundamental principle is to introduce not just one coarsening level, but to use, say, the step size h on the most recent part of the memory, step size wh (with some parameter $w > 1$) on the adjacent region, w^2h on the next region, etc. The main difference between the two approaches of [33,34] then lies in the
way in which the transition points from one mesh size to the next are chosen.

Specifically, as indicated in Figure 1, the method of Ford and Simpson [33] starts at the current time and fills subintervals of prescribed lengths from right to left with appropriately speced mesh points. This will lead to a reduction of the computational cost to O(N log N) while retaining the convergence order of the underlying scheme [33]. However, as indicated in Figure 1, it is common that the left end point of the leftmost coarsely subdivided interval does not match the initial point. In this case, one can either fill the remaining subinterval at the left end of the full interval with a fine mesh (which increases the computational cost but also reduces the error) or simply ignore the contribution from this subinterval

(which reduces the computational complexity but slightly increases the error; however, since the memory length still grows with the number of steps, this does not imply the complete loss of accuracy observed in the finite memory principle). In either case, grid points from the fine mesh that are not currently used in the nested mesh may become active again in future steps. Therefore, all previous grid points need to be kept in memory, so the required amount of memory space remains at O(N).

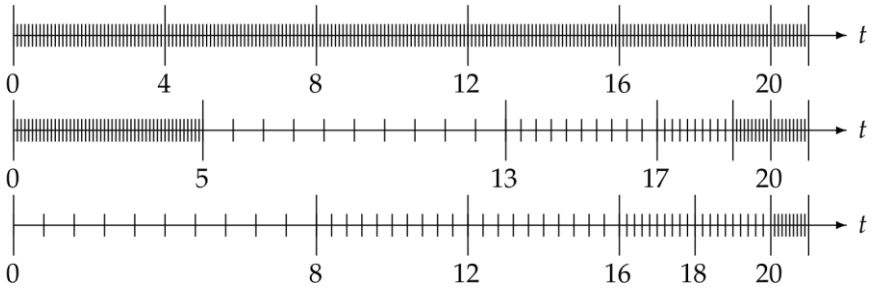

Figure 1. Full mesh (**top**) and nested meshes proposed in [33] (**center**) and in [34] (**bottom**). The meshes are shown for the time instant $t = 21$ and the basic step size $h = 1/10$.

In contrast, the approach of Diethelm and Freed [34] starts to fill the basic interval from left to right, i.e., it begins with the subinterval with the coarsest mesh and then moves to the finer-mesh regions. The final result is also a method with an O(N log N) computational cost, and with the same convergence order as the Ford-Simpson method; but its selection strategy for grid points implies that points that are inactive in the current step will never become active again in future steps, and consequently the history data for these inactive points can be eliminated from the main memory. This reduces the memory requirements to only O(log N).

6.2. A Method Based on the Fast Fourier Transform Algorithm

An effective approach for the fast evaluation of the convolution sums in (17) was proposed in [35,36]. The main idea is to split each of these sums in a way that enables the exploitation of the fast Fourier transform (FFT) algorithm. To provide a concise description, let us introduce the notations

$$T_p(n) = \sum_{j=p}^{n} c_{n-j} g_j, \quad S_{p,q}(n) = \sum_{j=p}^{q} c_{n-j} g_j, \quad n \geq p,$$

where $g_j = y_j$ or $g_j = f(t_j, y_j)$ according to the formula used in (17). Thus the numerical methods described by (17) can be recast as

$$y_n = \varphi_n + T_0(n), \quad n = 1, 2, \ldots, N.$$

The algorithm described in [35,36] is based on splitting $T_0(n)$ into one or more partial sums of type $S_{p,q}(n)$ and just one final convolution sum $T_p(n)$ of a maximum (fixed) length r. Thus, the computation is simply initialized as

$$T_0(n) = \sum_{j}^{n} c_{n-j} g_j \qquad n \in \{1, 2, \ldots, r-1\}$$

j=0

and the following r values of $T_0(n)$ are split into the two terms

$$T_0(n) = S_{0,r-1}(n) + T_r(n) \qquad n \in \{r, r+1, \ldots, 2r-1\}.$$

Similarly, for the computation of the next $2r$ values, $T_0(n)$ is split according to

$$T_0(n) = \begin{cases} S_{0,2r-1}(n) + T_{2r}(n) & n \in \{2r, 2r+1, \ldots, 3r-1\} \\ S_{0,2r-1}(n) + S_{2r,3r-1}(n) + T_{3r}(n) & n \in \{3r, 3r+1, \ldots, 4r-1\} \end{cases}$$

and the further $4r$ summations are split according to

$$T_0(n) = \begin{cases} S_{0,4r-1}(n) + T_{4r}(n) & n \in \{4r, 4r+1, \ldots, 5r-1\} \\ S_{0,4r-1}(n) + S_{4r,6r-1}(n) + T_{6r}(n) & n \in \{5r, 5r+1, \ldots, 6r-1\} \\ S_{0,4r-1}(n) + S_{4r,6r-1}(n) + S_{6r,7r-1}(n) + T_{8r}(n) & n \in \{6r, 6r+1, \ldots, 7r-1\} \\ & n \in \{7r, 7r+1, \ldots, 8r-1\} \end{cases}$$

and this process is continued until all terms $T_0(n)$, for $n \leq N$, are evaluated.

Note that in the above splittings the length $\ell(p, q) = q - p + 1$ of each sum $S_{p,q}$ is always some multiple of r with a power of 2 as multiplying factor (i.e., the possible length of $S_{q,p}(n)$ is r, $2r$, $4r$, $8r$ and so on).

For clarity, the diagram in Figure 2 illustrates the way in which the computation on the main triangle $T^0 = \{(n, j) : 0 \leq j \leq n \leq N\}$ is split into partial sums identified by the (red-labeled) squares $S_{p,q} = \{(n, j) : q + 1 \leq n \leq q + \ell(p, q),\ p \leq j \leq q\}$ and final blocks denoted by the (blue-labeled) triangles $T_p = \{(n, j) : p \leq j \leq n \leq p + r - 1\}$.

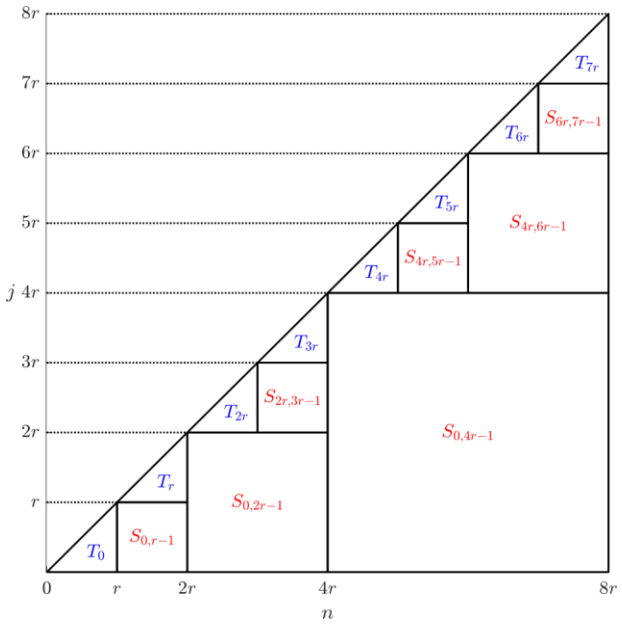

Figure 2. Splitting of the computation of $T_0(n)$ into partial sums $S_{p,q}$ (red-labeled squares) and final blocks T_p (blue-labeled triangles).

Each of the final blocks $T_r(n)$, $n = `r, `r + 1, \ldots, (` + 1)r - 1$, is computed by direct summation requiring $r(r+1)/2$ floating-point operations. The evaluation of the partial sums $S_{q,p}(n)$ can instead be performed by the FFT algorithm (see [37] for a comprehensive description) which requires a number of floating-point operations proportional to $2` \log_2 2`$, with $` = `(p, q)$ the length of each partial sum $S_{q,p}(n)$, since r is a power of 2.

In the optimal case in which both r and N are powers of 2, each partial sum $S_{p,q}$ that must be computed together with its length, number and computational cost is described in Table 1.

Table 1. Partial sums, their length, number and computational cost for the evaluation of $T_0(N)$.

S			
$S0,r-1$, $S2r,3r-1$, $S4r,5r-1$, $S6r,7r-1$, $S8r,9r-1$, . . .	r	s	

Furthermore, N/r final blocks T_r, each of length r, are also computed in $r(r + 1)/2$ floating-point operations and hence the total amount of floating point operations is proportional to

$$\log_2 N + 2\left(\frac{N}{2}\log_2\frac{N}{2}\right) + 4\left(\frac{N}{4}\log_2\frac{N}{4}\right) + \cdots + s\left(\frac{N}{s}\log_2\frac{N}{s}\right) + \frac{N\, r(r+1)}{r\quad 2} N =$$

$$= \log j \sum_{=0}^{2} s\, N \log_2 \frac{N}{2j} + \frac{Nr}{2} + 2 \quad = \mathcal{O}\left(N(\log_2 N)^2\right), \qquad s = \frac{N}{2r},$$

which turns out, for sufficiently large N, to be consistently significantly smaller than the number $\mathcal{O}(N^2)$ required by the direct summation of $T_0(N)$.

Although the whole procedure may appear complicated and requires some extra effort in coding, it turns out to be quite efficient since it can be applied to different methods of the form (17) and does not affect their accuracy. This preservation of accuracy is because the technique does take into account the entire history of the process in the same way as the straightforward approach mentioned above whose computational cost is $O(N^2)$. Thus, one does need to keep the entire history data in active memory, but one avoids the requirement of using special meshes. All the Matlab codes for FDEs described in [10,21,38], and freely available on the Mathworks website [39], make use of this algorithm.

6.3. Kernel Compression Schemes

Although the terminology "*kernel compression scheme*" has been introduced only recently for a few specific works [40–42], we use it here to describe a collection of methods that were proposed at various times by various authors and are all based on essentially the same principle: approximation of the solution of a non-local FDE by means of (possibly several) local ODEs. We provide here just the main ideas underlying this approach and we will refer the reader to the literature for a more comprehensive coverage of the subject.

Actually, these are standalone methods (usually classified as nonclassical methods [43]) and not just algorithms improving the efficiency of the treatment of the memory term; for this reason they could have been discussed in Section 3 along with the other methods for FDEs. But since one of their main achievements (and the motivation for their introduction) is to handle memory and computational issues related to the long and persistent memory of fractional-order problems, we consider it appropriate to discuss them in the present section.

For ease of presentation we consider only $0 < \alpha < 1$ but the extension to any positive α is only a technical matter. The basic idea starts from some integral representation of the kernel of the RL integral (1), e.g.,

$$\frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{\sin(\alpha\pi)}{\pi}\int_0^\infty \mathrm{e}^{-rt} r^{-\alpha}\,\mathrm{d}r, \tag{18}$$

which, thanks to standard quadrature rules, can be approximated by exponential sums

$$\frac{t^{\alpha-1}}{\Gamma(\alpha)} = \sum_{k=1}^{K} w_k \mathrm{e}^{-r_k t} + e_K(t), \tag{19}$$

where the error $e_K(t)$ and the computational complexity related to the number K of nodes and weights depend on the choice among the many possible quadrature rules. When applying this approximation instead of the exact integral in the integral formulation (7), the solution of the FDE (6) is rewritten as

$$K \quad {}_Z t$$

$$y(t) = y_0 + \sum_{k=1} w_k \quad {}_{t0} e^{-rk(t-u)} f(u, y(u))du + E_K(t). \tag{20}$$

Each of the integrals in (20) is actually the solution of an initial value problem:

$$\begin{cases} {}_{[k]}(t) = -rky[k](t) + f(t, y[k](t)) \, y \\ {}_{[k]} \\ y \quad (t0) = 0, \end{cases} \tag{21}$$

which can be numerically approximated by standard ODE solvers, yielding approximations $y_n[^{k]}$ on some grid $\{t_n\}$. If the quadrature rule is chosen so as to make the error $E_K(t)$ so small that it can be neglected, an approximate solution of the original FDE (6) can be obtained step-by-step as

$$K \quad yn \quad = \quad y0 + \sum \quad w^{-}kyn[k],$$

$k=1$

where each $y_n[^{k]}$ depends only on $^{y[}{}_n{}^{k}{}_{-}{}^{]}{}_1$ or on a few other previous values, according to the selected ODE solver.

In practice, a non-local problem (the FDE) with non-vanishing memory is replaced by K local problems (the ODEs) each demanding a smaller computational effort and the memory storage is restricted to $\mathcal{O}(pK)$ if a p-step ODE solver is used for each of the ODEs (21).

Obviously, the idea sketched above requires several further technical details to work properly. First, an accurate error analysis is needed to ensure that the overall error is below the target accuracy. This is a very delicate task because it involves the investigation of the interaction between the quadrature rule used to approximate the integral in (20) and the ODE solver applied to the system (21), which can be a highly nontrivial matter. Moreover, some substantial additional problems must be addressed. For instance, A-stable methods should generally be preferred when solving the system (21) since some of the $r_k > 0$ can be very large and give rise to stiff problems.

A non-negligible issue is that it is not possible to find a quadrature rule approximating (18) in a uniform manner with respect to all relevant values of t, i.e. with the same accuracy for any $t \geq t_1$ where t_1 is the first mesh point to the right of the initial point t_0 or for all $t \geq t_0$ (in either case, the singularity at t_0

indeed makes the integral quite difficult to be approximated). To overcome this difficulty, several different approaches have been proposed.

In a series of pioneering works [44–46], where a complex contour integral

$$\frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{1}{2\pi \mathrm{i}} \int_{C} e^{st} s^{-\alpha} \mathrm{d}s$$

is chosen to approximate the kernel, the integration interval $[t_0, T]$ is divided into a sequence of subintervals of increasing lengths, and different quadrature rules (on different contours C) are used in each of these intervals. While high accuracy can be obtained, this strategy is quite complicated and requires the use of more expensive complex arithmetic.

In [40–42] the integral in (7) is divided into local and history terms

$$y(t) = y_0 + \frac{1}{\Gamma(\alpha)} \int_{t_0}^{t-\delta t} (t-u)^{\alpha-1} f(u, y(u)) \mathrm{d}u + \frac{1}{\Gamma(\alpha)} \int_{t-\delta t}^{t} (t-u)^{\alpha-1} f(u, y(u)) \mathrm{d}u$$

History term — Local term

for a fixed $\delta t > 0$. This confines the singularity of the kernel to the local term, which can be approximated by standard methods for weakly singular integral equations (e.g., a product-integration rule) with a reduced computational cost and an insignificant memory requirement. The kernel in the history term no longer contains any singularity and can be safely approximated by (19) which applies now just for $t > \delta t$.

To obtain the highest possible accuracy, Gaussian quadrature rules are usually preferred. A rigorous and technical error analysis is necessary to tune parameters in an optimal way. Several implementations of approaches of this kind have been proposed (e.g., see [47–51]) but owing to their technical nature, a comparison to decide which method is in general the most convenient is difficult; we just refer to the interesting results presented in [52].

7. Some Remarks about Fractional Partial Differential Equations

Even though this paper is essentially devoted to the numerical solution of ordinary differential equations of fractional order and the computational treatment of the associated differential and integral operators, a few comments should be made regarding numerical methods for partial fractional differential equations (PDEs).

Remark 3. *The issues discussed in Section 4 are relevant to partial differential equations also. Indeed, it is shown in [53] that imposing excessive smoothness requirements on the solutions to a partial differential equation (e.g., for the sake of simplifying the error analysis or for obtaining a higher convergence order) has drastic implications regarding the class of admissible problems; in particular, the choice of the forcing function $f(x, t)$ in a linear initial-boundary value problem will then completely determine the initial condition in the problem.*

Our second remark regarding partial differential equations deals with a totally different aspect.

Remark 4. *Typical algorithms for time-fractional partial differential equations contain separate discretisation techniques with respect to the time variable and the space variable(s). A current trend is to employ a very high order method for the discretisation of the (non-fractional) differential operator with respect to the space variable. While this might seem an attractive approach at first sight, it has a number of disadvantages. Specifically, while this leads to a smaller discretization error in the space variable, it also increases the algorithm's overall complexity and makes the understanding of its properties more difficult. This complexity would be acceptable if the* overall *error could be reduced significantly. But since the overall error comprises not only the error from the space discretisation but also the contribution from the time approximation, it follows that to reduce the overall error, one must force this latter component to be very small also. As indicated above, we cannot expect to achieve a high convergence order in this variable, so the only way to reach this goal is to choose the time step size very small (in comparison with the space mesh size). From Section 6 we conclude that a standard algorithm with a higher-than-linear complexity is likely to lead to prohibitive run times, and even if the time discretisation uses a method with a linear or almost linear complexity, this very small step size requirement will still imply a high overall cost. Therefore, the use of a high-order space discretisation in a time-fractional partial differential equation is usually inadvisable.*

8. Concluding Remarks

In this paper we have tried to describe some issues related to the correct use of numerical methods for fractional-order problems. Unlike integer-order ODEs, numerical methods for FDEs are in general not taught in undergraduate courses and, very often, non-specialists are unaware of the peculiarities and major difficulties that arise in the numerical treatment of FDEs and fractional PDEs.

The availability of only a few well-organized textbooks and monographs in this field, together with the presence of many incorrect results in the literature, makes the situation even more difficult.

Some of the ideas collected in this paper were discussed in the lectures of the Training School on "Computational Methods for Fractional-Order Problems", held in Bari (Italy) during 22–26 July 2019, and promoted by the Cost Action CA15225—*Fractional-order systems: analysis, synthesis and their importance for future design*.

We believe that the scientific community should make an effort to raise the level of knowledge in this field by promoting specific academic courses at a basic level and/or by organizing training schools.

References

1. Diethelm, K. *The Analysis of Fractional Differential Equations*; Lecture Notes in Mathematics; Springer-Verlag: Berlin, Germany, 2010; Volume 2004, p. viii+247.
2. Kilbas, A.A.; Srivastava, H.M.; Trujillo, J.J. *Theory and Applications of Fractional Differential Equations*; North-Holland Mathematics Studies; Elsevier Science B.V.: Amsterdam, The Netherlands, 2006; Volume 204, p. xvi+523.
3. Mainardi, F. *Fractional Calculus and Waves in Linear Viscoelasticity*; Imperial College Press: London, UK, 2010; p. xx+347.

4. Miller, K.S.; Ross, B. *An Introduction to the Fractional Calculus and Fractional Differential Equations*; A Wiley-Interscience Publication; John Wiley & Sons, Inc.: New York, NY, USA, 1993; p. xvi+366.
5. Podlubny, I. *Fractional Differential Equations*; Mathematics in Science and Engineering; Academic Press Inc.: San Diego, CA, USA, 1999; Volume 198, p. xxiv+340.
6. Samko, S.G.; Kilbas, A.A.; Marichev, O.I. *Fractional Integrals and Derivatives*; Gordon and Breach Science Publishers: Yverdon, Switzerland, 1993; p. xxxvi+976.
7. Young, A. Approximate product-integration. *Proc. R. Soc. Lond. Ser. A.* **1954**, *224*, 552–561.
8. Young, A. The application of approximate product integration to the numerical solution of integral equations.
   Proc. R. Soc. Lond. Ser. A. **1954**, *224*, 561–573.
9. Diethelm, K.; Ford, N.J.; Freed, A.D. A predictor-corrector approach for the numerical solution of fractional differential equations. *Nonlinear Dyn.* **2002**, *29*, 3–22.
10. Garrappa, R. On linear stability of predictor-corrector algorithms for fractional differential equations. *Int. J.*
    Comput. Math. **2010**, *87*, 2281–2290.
11. Yan, Y.; Pal, K.; Ford, N.J. Higher order numerical methods for solving fractional differential equations. *BIT Numer.*
    Math. **2014**, *54*, 555–584.
12. Li, Z.; Liang, Z.; Yan, Y. High-order numerical methods for solving time fractional partial differential equations. *J. Sci. Comput.* **2017**, *71*, 785–803.
13. Dixon, J. On the order of the error in discretization methods for weakly singular second kind Volterra integral equations with nonsmooth solutions. *BIT* **1985**, *25*, 624–634.
14. Diethelm, K.; Ford, N.J.; Freed, A.D. Detailed error analysis for a fractional Adams method. *Numer. Algorithms* **2004**, *36*, 31–52.
15. Oldham, K.B.; Spanier, J. Theory and applications of differentiation and integration to arbitrary order. In *The Fractional Calculus*; Academic Press: New York, NY, USA; London, UK, 1974; p. xiii+234.
16. Lynch, V.E.; Carreras, B.A.; del Castillo-Negrete, D.; Ferreira-Mejias, K.M.; Hicks, H.R. Numerical methods for the solution of partial differential equations of fractional order. *J. Comput. Phys.* **2003**, *192*, 406–421.
17. Lubich, C. Discretized fractional calculus. *SIAM J. Math. Anal.* **1986**, *17*, 704–719.
18. Lubich, C. Convolution quadrature and discretized operational calculus. I. *Numer. Math.* **1988**, *52*, 129–145.
19. Lubich, C. Convolution quadrature and discretized operational calculus. II. *Numer. Math.* **1988**, *52*, 413–425.
20. Lubich, C. Convolution quadrature revisited. *BIT* **2004**, *44*, 503–514.
21. Garrappa, R. Trapezoidal methods for fractional differential equations: theoretical and computational aspects.
    Math. Comput. Simul. **2015**, *110*, 96–112.

22. Diethelm, K.; Ford, J.M.; Ford, N.J.; Weilbeer, M. Pitfalls in fast numerical solvers for fractional differential equations. *J. Comput. Appl. Math.* **2006**, *186*, 482–503.
23. Stynes, M. Singularities. In *Handbook of Fractional Calculus With Applications. Vol. 3*; De Gruyter: Berlin, Germany, 2019; pp. 287–305.
24. Miller, R.K.; Feldstein, A. Smoothness of solutions of Volterra integral equations with weakly singular kernels.
SIAM J. Math. Anal. **1971**, *2*, 242–258.
25. Lubich, C. Runge-Kutta theory for Volterra and Abel integral equations of the second kind *Math. Comput.* **1983**, *41*, 87–102.
26. Hanyga, A. A comment on a controversial issue: A generalized fractional derivative cannot have a regular kernel *Fract. Calc. Appl. Anal.* **2020**, *23*, 211–223.
27. Giusti, A. General fractional calculus and Prabhakar's theory. *Commun. Nonlinear Sci. Numer. Simul.* **2019**, *83*, 105114.
28. Hanyga, A. Physically acceptable viscoelastic models. In *Trends in Applications of Mathematics to Mechanics*; Hutter, K., Wang, Y., Eds.; Shaker Verlag: Aachen, Germany, 2005; pp. 125–136.
29. Stynes, M.; O'Riordan, E.; Gracia, J.L. Necessary conditions for convergence of difference schemes for fractional-derivative two-point boundary value problems. *BIT* **2016**, *56*, 1455–1477.
30. Sarv Ahrabi, S.; Momenzadeh, A. On failed methods of fractional differential equations: the case of multi-step generalized differential transform method. *Mediterr. J. Math.* **2018**, *15*, Art. 149, 10.
31. Garrappa, R. Neglecting nonlocality leads to unreliable numerical methods for fractional differential equations.
Commun. Nonlinear Sci. Numer. Simul. **2019**, *70*, 302–306.
32. Deng, W.H. Short memory principle and a predictor-corrector approach for fractional differential equations.
J. Comput. Appl. Math. **2007**, *206*, 174–188.
33. Ford, N.J.; Simpson, A.C. The numerical solution of fractional differential equations: Speed versus accuracy.
Numer. Algorithms **2001**, *26*, 333–346.
34. Diethelm, K.; Freed, A.D. An Efficient Algorithm for the Evaluation of Convolution Integrals. *Comput. Math.*
Appl. **2006**, *51*, 51–72.
35. Hairer, E.; Lubich, C.; Schlichte, M. Fast numerical solution of nonlinear Volterra convolution equations. *SIAM J.*
Sci. Statist. Comput. **1985**, *6*, 532–541.
36. Hairer, E.; Lubich, C.; Schlichte, M. Fast numerical solution of weakly singular Volterra integral equations. *J. Comput. Appl. Math.* **1988**, *23*, 87–98.
37. Henrici, P. Fast Fourier methods in computational complex analysis. *SIAM Rev.* **1979**, *21*, 481–527.
38. Garrappa, R. Numerical Solution of Fractional Differential Equations: A Survey and a Software Tutorial.
Mathematics **2018**, *6*, 16.

39. Garrappa, R. Mathworks Author's Profile. Available online: https://www.mathworks.com/matlabcentral/profile/authors/2361481-roberto-garrappa (accessed on 26 January 2020).
40. Baffet, D. A Gauss-Jacobi kernel compression scheme for fractional differential equations. *J. Sci. Comput.* **2019**, *79*, 227–248.
41. Baffet, D.; Hesthaven, J.S. A kernel compression scheme for fractional differential equations. *SIAM J. Numer.*
Anal. **2017**, *55*, 496–520.
42. Baffet, D.; Hesthaven, J.S. High-order accurate adaptive kernel compression time-stepping schemes for fractional differential equations. *J. Sci. Comput.* **2017**, *72*, 1169–1195.
43. Diethelm, K. An investigation of some nonclassical methods for the numerical approximation of Caputo-type fractional derivatives. *Numer. Algorithms* **2008**, *47*, 361–390.
44. López-Fernández, M.; Lubich, C.; Schädle, A. Adaptive, fast, and oblivious convolution in evolution equations with memory. *SIAM J. Sci. Comput.* **2008**, *30*, 1015–1037.
45. Lubich, C.; Schädle, A. Fast convolution for nonreflecting boundary conditions. *SIAM J. Sci. Comput.* **2002**, *24*, 161–182.
46. Schädle, A.; López-Fernández, M.; Lubich, C. Fast and oblivious convolution quadrature. *SIAM J. Sci. Comput.* **2006**, *28*, 421–438.
47. Banjai, L.; López-Fernández, M. Efficient high order algorithms for fractional integrals and fractional differential equations. *Numer. Math.* **2019**, *141*, 289–317.
48. Fischer, M. Fast and parallel Runge-Kutta approximation of fractional evolution equations. *SIAM J. Sci. Comput.* **2019**, *41*, A927–A947.
49. Jiang, S.; Zhang, J.; Zhang, Q.; Zhang, Z. Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations. *Commun. Comput. Phys.* **2017**, *21*, 650–678.
50. Li, J.R. A fast time stepping method for evaluating fractional integrals. *SIAM J. Sci. Comput.* **2010**, *31*, 4696–4714. 51. Zeng, F.; Turner, I.; Burrage, K. A stable fast time-stepping method for fractional integral and derivative operators. *J. Sci. Comput.* **2018**, *77*, 283–307.
52. Guo, L.; Zeng, F.; Turner, I.; Burrage, K.; Karniadakis, G.E.M. Efficient multistep methods for tempered fractional calculus: Algorithms and simulations. *SIAM J. Sci. Comput.* **2019**, *41*, 2510–2535.
53. Stynes, M. Too much regularity may force too much uniqueness. *Fract. Calc. Appl. Anal.* **2016**, *19*, 1554–1562.