

A FAST AND MEMORYLESS NUMERICAL METHOD FOR SOLVING FRACTIONAL DIFFERENTIAL EQUATIONS

NICOLA GUGLIELMI* AND ERNST HAIRER†

Abstract. The numerical solution of implicit and stiff differential equations by implicit numerical integrators has been largely investigated and there exist many excellent efficient codes available in the scientific community, as RADAU5 (based on a Runge-Kutta collocation method at Radau points) and DASSL, based on backward differentiation formulas, among the others. When solving fractional ordinary differential equations (ODEs), the derivative operator is replaced by a non-local one and the fractional ODE is reformulated as a Volterra integral equation, to which these codes cannot be directly applied.

This article is a follow-up of the article by the authors [15] for differential equations with distributed delays. The main idea is to approximate the fractional kernel $t^{\alpha-1}/\Gamma(\alpha)$ ($\alpha > 0$) by a sum of exponential functions or by a sum of exponential functions multiplied by a monomial, and then to transform the fractional integral (of convolution type) into a set of ordinary differential equations. The augmented system is typically stiff and thus requires the use of an implicit method. It can have a very large dimension and requires a special treatment of the arising linear systems.

The present work presents an algorithm for the construction of an approximation of the fractional kernel by a sum of exponential functions, and it shows how the arising linear systems in a stiff time integrator can be solved efficiently. It is explained how the code RADAU5 can be used for solving fractional differential equations. Numerical experiments illustrate the accuracy and the efficiency of the proposed method. Driver examples are publicly available from the homepages of the authors.

Key words. Fractional differential equations, differential-algebraic equations, Volterra integral equations, integro-differential equations, Runge-Kutta methods, approximation by sum of exponentials, code RADAU5, fast solvers for structured linear systems.

AMS subject classifications. 26A33, 34A08, 65L06, 45D05, 65F05

1. Introduction. The study of fractional integrals and fractional differential equations (FDEs) has developed incredibly in recent years; there is now a very large number of articles dealing with various versions of fractional derivatives and models, as well as their analysis and application. In parallel, the study of suitable numerical integrators is also a very open and active area of research.

In this paper we will discuss Initial Value Problems (IVPs) mainly for the Caputo fractional derivative, but also for the Riemann-Liouville fractional derivative, the two fractional derivatives that are most commonly used, both are defined in terms of the Riemann-Liouville fractional integral. Many results can be found in textbooks such as [30, 11, 22, 31, 21].

Fractional differential equations are usually written as a special case of Volterra integral and integro-differential equations. There is a large literature on the numerical discretization of such problems, and their accuracy and stability is well investigated (see the monographs by H. Brunner [8, 9]). Especially for problems with weakly singular kernel the technique of discretized fractional calculus has been developed by C. Lubich [27] (see also [16]), and the oblivious convolution quadrature [32, 25] for more general kernels.

Further methods which are proposed and analyzed in the literature (see e.g. [19]), are called $L1$ methods. They are some of the simplest and most widely used schemes for time-fractional problems. The $L1$ method is a direct quadrature-based approach

*Division of Mathematics, Gran Sasso Science Institute, Via Crispi 7, I-67100 L'Aquila, Italy.
Email: nicola.guglielmi@gssi.it

†Section de Mathématiques, Université de Genève, CH-1211 Genève 24, Switzerland.
Email: ernst.hairer@unige.ch

for approximating the Caputo fractional derivative. It uses piecewise linear interpolation of the solution and approximates the Caputo derivative on a uniform or graded time mesh. The fast $L1$ method uses an approximation of the fractional kernel by a sum of exponential functions.

Available codes mostly make use of constant step size. For a survey about numerical methods for fractional differential equations, we refer the reader e.g. to [10] and the references therein. A typical drawback of available methods is the complexity in dealing with the memory associated to the fractional integral, which increases as time increases and the inherent difficulty to make use of variable stepsize strategies. However, in realistic situations one is often confronted with initial layers and/or fast transitions between different states, so that flexibility in the choice of step size is an essential ingredient for efficiency. Although there exist excellent codes for solving nonstiff and stiff ordinary differential equations, codes for differential-algebraic equations, we are not aware of a code based on a variable step size time integrator that can efficiently treat problems with fractional derivatives.

Our approach consists in modifying the fractional differential equation in such a way that any code for stiff and differential-algebraic equations can be applied. Following the ideas of [15] the fractional kernel in the integral term is approximated by a sum of exponential functions. The integral term is then transformed into a set of differential equations. In this way, fractional differential equations can be solved efficiently and reliably by standard software for stiff and differential-algebraic equations. The method we propose is an oblivious method which allows to replace the memory by a system of additional ODEs, solving in this way all the technical difficulties related to handling the memory term and to the use of variable step sizes. This is illustrated at the hand of RADAU5 [17] for stiff and differential-algebraic equations.

Related literature. There are essentially two approaches for approximating the fractional kernel $k(t) = t^{\alpha-1}/\Gamma(\alpha)$ by a sum of exponential functions. Since the Laplace transform of $k(t)$ is $\lambda^{-\alpha}$ for $0 < \alpha < 1$, the inversion formula gives

$$k(t) = \frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{1}{2\pi i} \int_{\Lambda} e^{t\lambda} \lambda^{-\alpha} d\lambda \approx \sum_{j=1}^n w_j e^{t\lambda_j}, \quad (1.1)$$

where Λ is a suitable path in the complex plane. Discretizing the integral gives the desired sum of exponential functions, where w_j and λ_j are complex numbers. The early publication [28, Formula (1.6)] (see also [29]) proposes to insert this formula into the integral term $\int_0^t k(t-s)f(s, y(s)) ds$ (see Section 2.2 below), which then can be transformed into a series of scalar linear ordinary differential equations. Based on such an approximation, a variable step size algorithm for the numerical solution of fractional differential equations is presented in [25], for which advancing N steps requires only $\mathcal{O}(N \log N)$ computations and $\mathcal{O}(\log N)$ active memory.

In [4, 3], the authors split the fractional integral into a local term (containing the singularity), $\int_0^\delta k(t-s)f(s, y(s)) ds$ for a small $\delta > 0$, and a more expensive history term $\int_\delta^t k(t-s)f(s, y(s)) ds$. For the history term they consider a multipole approximation of the Laplace transform of the kernel, which then leads to an approximation of the form (1.1) with complex w_j and λ_j . Again a transformation to a series of scalar linear differential equations makes the algorithm efficient.

Another approach is to use the fact that the Laplace transform of the function

$z^{-\alpha}$ is $\Gamma(1-\alpha) t^{\alpha-1}$ (for $0 < \alpha < 1$), which gives the real integral representation ¹

$$k(t) = \frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{\sin(\pi\alpha)}{\pi} \int_0^\infty e^{-tz} z^{-\alpha} dz \approx \sum_{j=1}^n c_j e^{-\gamma_j t}, \quad (1.2)$$

with real numbers c_j and γ_j . To get the approximation by a sum of exponentials it is proposed in [24] to first remove the singularity with the change of variables $\zeta = z^{1-\alpha}$ (for $0 < \alpha < 1$), then to write $(0, \infty)$ as a distinct union of geometrically increasing intervals, and finally to apply to each of these intervals (away from 0) a Gauss–Legendre quadrature.

In [2], a composite Gauss–Jacobi is applied directly to the integral in (1.2) on geometrically increasing intervals. When applied to fractional differential equations, an integral deferred correction scheme is combined with an L-stable diagonally implicit Runge–Kutta scheme for the approximation of the differential equations obtained by the kernel compression. In [5], the authors propose a fast and memory efficient numerical method for solving a fractional integral based on a Runge–Kutta convolution quadrature. The method is applied to the numerical solution of fractional diffusion equations in space dimension up to 3.

A further class of fast methods is analyzed e.g. in [20, 35], where - based on exponential expansions - a first and a second order efficient schemes are proposed, with a constant stepsize, i.e. on a uniform grid. They are applied for solving time-fractional diffusion PDEs.

The approach that we follow in the present work is that of [6]. It uses the real integral representation (1.2), performs the change of variables $z = e^s$ to get an integral over the real line $(-\infty, \infty)$, and finally applies the trapezoidal rule. This gives simple expressions for the coefficients c_j and γ_j in the sum of exponentials.

Fractional partial differential equations are important in applications. Wave propagation in a viscoelastic medium with singular memory is studied in [26] and a numerical approach is presented. As long as the dimension in space is one, we are lead after space discretization to problems with banded Jacobians and the code of the present work can be applied effectively. For higher space dimensions, an adaptation of the linear algebra routines is recommended for reasons of efficiency. Fractional derivatives are also used in treating transparent boundary conditions [1].

Outline of the paper. Section 2 recalls the definitions of Caputo and Riemann–Liouville fractional derivatives and of a class of Cauchy problems for fractional differential equations. These problems are written in the form of integro-differential equations that are special cases of a general formulation which can be treated with the approach of the present work. Section 3 explains the linear chain trick, which permits to transform the integro-differential equations (with fractional kernel in the convolution integrals) into a large system of stiff ordinary differential equations. The application of the code RADAU5 for solving fractional differential equations is explained in Section 4 with emphasis on the efficient solution of the arising linear systems. Section 5 presents explicit formulas by Beylkin and Monzón [6] for an approximation of the fractional kernel $k(t) = t^{\alpha-1}/\Gamma(\alpha)$ ($0 < \alpha < 1$) by a sum of exponential functions, and it discusses the effect of such an approximation on the solution of the problem. The case $\alpha > 1$ is considered in Section 6. Two approaches, one by splitting the

¹Note that the integral representation formula (1.2) can also be derived from (1.1) as a real inversion formula, as is discussed in [18, Chapter 10, page 284], specifically in Theorem 10.7d and in Example 4, for $0 < \alpha < 1$.

kernel and the other by differentiating the integral are discussed. The final Section 7 presents numerical experiments for a scalar test equation with known exact solution, for a modified Brusselator reaction, for a linear fractional partial differential equation, and for a system of reaction-diffusion problems.

2. Fractional differential equations. This section is devoted to a general formulation of fractional differential equation, which can be solved numerically by the techniques of the present work.

2.1. Integral and differential fractional operators. For a given real number $\alpha > 0$ and for $t > 0$ the *Riemann-Liouville fractional integral* of order α is defined as an integral (with a possibly weakly singular kernel, if $\alpha < 1$) by

$$J^\alpha f(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} f(s) ds. \quad (2.1)$$

For $\alpha = 1$ we recover the usual integral, and for integer α we have an iterated integral.

For a non-integer $\alpha > 0$ we let $m = \lceil \alpha \rceil$ be the smallest integer that is larger than α . If D denotes the derivative with respect to t , the *Riemann-Liouville fractional derivative* is then given by $D^\alpha = D^m J^{m-\alpha}$, i.e.,

$$D^\alpha f(t) = \frac{d^m}{dt^m} \left(\frac{1}{\Gamma(m-\alpha)} \int_0^t (t-s)^{m-\alpha-1} f(s) ds \right). \quad (2.2)$$

When considering fractional differential equations it is convenient to use a slightly different definition of fractional derivative. The *Caputo fractional derivative* is defined by $D_*^\alpha = J^{m-\alpha} D^m$, i.e.,

$$D_*^\alpha f(t) = \frac{1}{\Gamma(m-\alpha)} \int_0^t (t-s)^{m-\alpha-1} f^{(m)}(s) ds. \quad (2.3)$$

The relation between the Riemann-Liouville and Caputo fractional derivatives is

$$D_*^\alpha f(t) = D^\alpha \left(f(t) - T_{m-1}(t) \right), \quad T_{m-1}(t) = \sum_{k=0}^{m-1} \frac{t^k}{k!} f^{(k)}(0). \quad (2.4)$$

More information about fractional differential and integral operators can be found in the monographs [11] and [14].

2.2. Fractional differential equations. We begin by considering initial value problems of the form

$$D_*^\alpha y(t) = f(t, y(t)), \quad y^{(k)}(0) = y_k, \quad k = 0, \dots, m-1, \quad (2.5)$$

where f is sufficiently regular, $\alpha > 0$ and $m = \lceil \alpha \rceil$. For the existence and uniqueness of a solution we refer to [11]. For a numerical treatment it is convenient to write such a problem as a Volterra integral or Volterra integro-differential equation. There are several ways to do this.

We can apply the operator J^α to (2.5) and use $J^\alpha D_*^\alpha f(t) = J^m D^m f(t) = f(t) - \sum_{k=0}^{m-1} f^{(k)}(0) t^k / k!$. This yields the Volterra integral equation

$$y(t) = T_{m-1}(t) + \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} f(s, y(s)) ds. \quad (2.6)$$

Differentiating this relation $(m-1)$ times with respect to t , gives an equivalent Volterra integro-differential equation (if α is not an integer and $m \geq 1$)

$$y^{(m-1)}(t) = y_{m-1} + \frac{1}{\Gamma(\alpha - m + 1)} \int_0^t (t-s)^{\alpha-m} f(s, y(s)) ds, \quad (2.7)$$

with initial values $y^{(k)}(0) = y_k$ for $k = 0, \dots, m-2$.

For the case that the fractional derivative is not the highest derivative in the equation (see the example of Section 7.3), it is possible to insert directly the definition (2.3) of the fractional derivative to get a Volterra integro-differential equation.

2.3. General formulation as integro-differential equations. The aim of this work is to present a numerical approach that permits us to accurately solve initial value problems of the form

$$M\dot{y}(t) = F(t, y(t), I(y)(t)), \quad y(0) = y_0. \quad (2.8)$$

Here, $y(t)$ is a vector in \mathbb{R}^d , M is a possibly singular square matrix, $F(t, y, I)$ is sufficiently regular, and $I(y)(t)$ is a vector in \mathbb{R}^{d_I} with components

$$I_j(y)(t) = \frac{1}{\Gamma(\alpha_j)} \int_0^t (t-s)^{\alpha_j-1} G_j(s, y(s)) ds, \quad (2.9)$$

where $\alpha_j > 0$, and $G_j(t, y)$ is smooth as a function of y .

Both integral formulations of Section 2.2 are special cases of (2.8), and each of them has $d_I = d$. For the formulation (2.6) we put $M = 0$, $\alpha_j = \alpha$, and $G_j(t, y) = f_j(t, y)$, where $f_j(t, y)$ denotes the j th component of $f(t, y)$.

For the formulation (2.7) with $m \geq 2$ we introduce the variables $u_j(t) = y^{(j)}(t)$ for $j = 1, \dots, m-2$, so that $\dot{u}_{m-2}(t)$ becomes the right-hand side of (2.7). To close the system we have to add the differential equations $\dot{y}(t) = u_1(t)$ and $\dot{u}_j(t) = u_{j+1}(t)$ for $j = 1, \dots, m-3$. We get a system (2.8) with solution vector $(y(t), u_1(t), \dots, u_{m-2}(t))$. The matrix M is the identity, and the integral terms have $\alpha_j = \alpha - m + 1 \in (0, 1)$ and $G_j(t, y) = f_j(t, y)$ as before.

The system (2.8) comprises not only all formulations of Section 2.2 as special cases, but it is much more general. The system can be a differential-algebraic one, and several fractional derivatives with different values of α_j are permitted.

3. Ordinary differential equations approximating fractional problems.

Instead of applying a quadrature formula to the integral in (2.9), our approach consists of approximating the kernel $k(t) = t^{\alpha-1}/\Gamma(\alpha)$ for $0 < \alpha < 1$, by a sum of exponential functions, so that the integro-differential equation (2.8) can be reduced to a system of ordinary differential equations.

3.1. Linear chain trick. Assume that we have at our disposal an approximation

$$\frac{t^{\alpha-1}}{\Gamma(\alpha)} \approx \sum_{i=1}^n c_i e^{-\gamma_i t} \quad (3.1)$$

with real coefficients c_i and $\gamma_i > 0$ depending on α and on the accuracy of the approximation. Explicit formulas for c_i and γ_i as well as a study of the effect of such

an approximation to the solution of the fractional differential equation are given in Section 5. Replacing in (2.9) the kernel by (3.1) yields the approximation

$$\tilde{I}_j(y)(t) = \sum_{i=1}^{n_j} c_{i,j} z_{i,j}(t), \quad z_{i,j}(t) = \int_0^t e^{-\gamma_{i,j}(t-s)} G_j(s, y(s)) ds, \quad (3.2)$$

where for $\alpha = \alpha_j$ the parameters in (3.1) are n_j , $c_{i,j}$ and $\gamma_{i,j}$. Depending on $G_j(t, y)$, the functions $z_{i,j}(t)$ are scalar or vector-valued. Differentiation of $z_{i,j}(t)$ with respect to time yields, for $i = 1, \dots, n_j$ and $j = 1, \dots, d_I$,

$$\dot{z}_{i,j}(t) = -\gamma_{i,j} z_{i,j}(t) + G_j(t, y(t)). \quad (3.3)$$

We now substitute $I_j(t) = \tilde{I}_j(y)(t)$ from (3.2) for $I_j(y)(t)$ into the equation (2.8) to obtain

$$\begin{aligned} M\dot{y}(t) &= F(t, y(t), I_1(t), \dots, I_{d_I}(t)) & I_j(t) &= \sum_{i=1}^{n_j} c_{i,j} z_{i,j}(t) \\ \dot{z}_{i,j}(t) &= -\gamma_{i,j} z_{i,j}(t) + G_j(t, y(t)), & i &= 1, \dots, n_j, \quad j = 1, \dots, d_I. \end{aligned} \quad (3.4)$$

Initial values are $y(0) = y_0$ from (2.8), and $z_{i,j}(0) = 0$ for all i and j .

This is a system of ordinary differential equations

$$\mathcal{M}\dot{Y}(t) = \mathcal{F}(t, Y(t)), \quad Y(0) = Y_0 \quad (3.5)$$

for the vector $Y(t) = (y(t), z_{i,j}(t))$, where $j = 1, \dots, d_I$ and $i = 1, \dots, n_j$. Here, \mathcal{M} is the diagonal matrix with entries from M for the y -variables, and with entries 1 for the z -variables. The system is of dimension $d + n_1 + \dots + n_{d_I}$. Since some coefficients among the $\gamma_{i,j}$ are typically very large and positive, this system is stiff independent of whether the original equation is stiff or not.

3.2. Structure of the Jacobian matrix. For notational convenience we assume here that all functions $G_j(t, y)$ are scalar. This can be done without loss of generality by eventually increasing the number of integrals in (2.8). The Jacobian matrix of the vector field in (3.5) is

$$\mathcal{J} = \frac{\partial \mathcal{F}}{\partial Y} = \begin{pmatrix} J & C_1 & C_2 & \cdots & C_{d_I} \\ B_1 & J_1 & 0 & \cdots & 0 \\ B_2 & 0 & J_2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ B_{d_I} & 0 & \cdots & 0 & J_{d_I} \end{pmatrix}, \quad (3.6)$$

where $J = \partial F / \partial y$ is the d -dimensional derivative matrix, $C_j = (\partial F / \partial I_j) c_j^\top$ a rank-one matrix with $c_j^\top = (c_{1,j}, \dots, c_{n_j,j})$, $B_j = e(\partial G_j / \partial y)$ a rank-one matrix (e is the n_j -dimensional column vector with all entries equal to 1), and J_j is the n_j -dimensional diagonal matrix with entries $-\gamma_{1,j}, \dots, -\gamma_{n_j,j}$.

REMARK 1. If one is interested in the values of the integral $I_j(t)$ along the solution, one can introduce $y_{d+1}(t) = I_j(t)$ as a new dependent variable, replace I_j by y_{d+1} in (3.4), and add the algebraic relation $0 = \sum_{i=1}^{n_j} c_{i,j} z_{i,j}(t) - y_{d+1}(t)$ to the system. In this way the dimension of the vectors y and F is increased by 1, and an additional diagonal element 0 is added to the matrix M . This procedure can be applied to several or to all integrals I_j .

This reformulation of the problem has been used in [15] to monitor the accuracy of the integrals in the system. If the number of considered integrals is large, it can increase the overhead of the computation.

4. Solving fractional differential equations by the code RADAU5. The approach of the present article is to approximate the solution of a fractional differential equation by that of the system (3.5). Any code for the solution of stiff differential-algebraic problems can be applied. We focus on our code RADAU5, for which we develop a special treatment of the solution of linear systems.

All subroutines are publicly available at:

www.unige.ch/~hairer/software.html

4.1. Applying the code RADAU5. The use of the code RADAU5 is described in the monograph [17]. For the problem (3.5) one has to supply a subroutine $\text{FCN}(\mathbf{N}, \mathbf{T}, \mathbf{Y}, \mathbf{F}, \dots)$ for its right-hand side. Here, \mathbf{N} is the dimension $d + n_1 + \dots + n_{d_I}$ of the system, $\mathbf{Y}(\mathbf{N})$ is the argument $\mathbf{Y}(t)$, and $\mathbf{F}(\mathbf{N})$ the vector $\mathcal{F}(t, \mathbf{Y}(t))$.

Furthermore, a subroutine for the mass matrix \mathcal{M} is needed. One has to put $\text{IMAS} = 1$, $\text{MLMAS} = 0$, $\text{MUMAS} = 0$ in the calling program, and one has to supply a subroutine $\text{MAS}(\mathbf{N}, \mathbf{AM}, \dots)$, where $\mathbf{AM}(1, \mathbf{J})$ contains the diagonal entries of \mathcal{M} .

One has the option to treat the arising linear system by unstructured full matrices of dimension \mathbf{N} . An approximation to the Jacobian matrix (3.6) can be computed internally by finite differences by putting $\text{IJAC} = 0$, or one can put $\text{IJAC} = 1$ and provide a subroutine for (3.6). Since $D = n_1 + \dots + n_{d_I}$ is typically much larger than d , we recommend to exploit the pattern of the Jacobian matrix, as we explain in the next section.

4.2. Fast linear algebra. A time integrator for the stiff differential equation (3.5) typically requires the solution of linear systems $((\gamma\Delta t)^{-1}\mathcal{M} - \mathcal{J})u = a$, where $u = (u_0, u_1, \dots, u_{d_I})$ and $a = (a_0, a_1, \dots, a_{d_I})$ are partitioned according to (3.6). Exploiting the arrow-like structure of (3.6) with diagonal blocks (except for the first one, $J \in \mathbb{R}^{d \times d}$, and rank-1 off diagonal blocks, it is convenient to solve the system backwards expressing u_i in terms of u_0 , which has linear complexity in the dimension of u_i , and finally computing u_0 as solution of a $d \times d$ linear system

$$((\gamma\Delta t)^{-1}M - \hat{J})u_0 = \hat{a}_0, \quad \hat{J} = J + \sum_{j=1}^{d_I} \hat{c}_j \frac{\partial F}{\partial I_j} \frac{\partial G_j}{\partial y}, \quad (4.1)$$

where \hat{c}_j is a scalar, $\partial F / \partial I_j$ is a column vector and $\partial G_j / \partial y$ is a row vector, so that their product gives a rank-one matrix.

This allows to have a linear solver of complexity $\mathcal{O}(d^3 + D)$ instead of $\mathcal{O}((d + D)^3)$ with $D = n_1 + \dots + n_{d_I}$ with typically $D \gg d$. This is an essential feature of the approach we propose. For details we refer to [15, Section 3.1], where closely related ideas are discussed for a slightly different problem.

4.3. Use of the fast linear algebra (DC_SUMEXP). The code RADAU5 uses two files, `DECSOL.F` for the linear algebra subroutines, and `DC_DECSOL.F` for linking the linear algebra subroutines to the main time integrator. For using a linear algebra that is adapted to the structure (3.6) of the Jacobian matrix, we have written a subroutine `DC_SUMEXP.F`, which replaces `DC_DECSOL.F`. We assume $d \geq 2$. For scalar problems (such as the example of Section 7.1) we recommend to introduce at least one of the integral terms as a new y -variable (see Remark 1).

When the subroutines of `DC_SUMEXP.F` are used, one has to define the dimension of the system as $\mathbf{N} = d + (n_1 + 2) + \dots + (n_{d_I} + 2)$. The subroutines for the right-hand side and the mass matrix are as in Section 4.1. The additional $2d_I$ variables

are considered as dummy variables, and only used in the subroutine for the Jacobian matrix. In the calling program one has to define $IJAC = 1$, $MLJAC = d$ (the dimension of the original system), and $MUJAC = 0$. One has to supply a subroutine $JAC(N,T,Y,DFY,LDFY,...)$, where the array $DFY(LDFY,N)$ ($LDFY$ is computed by `RADAU5` and need not be defined) contains all the information of the Jacobian matrix.

The upper left $d \times d$ matrix of $DFY(LDFY,N)$ is that of the Jacobian matrix (3.6). Next to the right are d_I blocks, each of them corresponding to an integral term in the problem (2.8). The j th block has $n_j + 2$ columns. The first column contains the vector $\partial F / \partial I_j$, and the second column contains the derivative $\partial G_j / \partial y$, both are column vectors of dimension d . The first row of this block has the coefficients $c_{1,j}, \dots, c_{n_j,j}$ at the positions 3 to $n_j + 2$. Similarly, the second row has $-\gamma_{1,j}, \dots, -\gamma_{n_j,j}$ at the positions 3 to $n_j + 2$. Finally, the last element of the third row is put to a negative number to indicate the end of the j th block.

For the case that the dimension d is large and the Jacobian gives raise to banded matrices (this typically happens with space discretizations of 1D fractional PDEs) further options are provided by the subroutines of `DC_SUMEXP.F`. This will be explained in Section 8.

5. Approximation of $t^{\alpha-1}$ by a sum of exponential functions. This section is devoted to the development of explicit formulas of the coefficients (3.1) for the case $0 < \alpha < 1$. Among the different formulations of fractional differential equations this covers (2.7) for every $\alpha > 0$, and the formulation (2.6) for the case $0 < \alpha < 1$.

5.1. Approach of Beylkin and Monzón. Since the Laplace transform of the function $z^{-\alpha}$ is $\Gamma(1-\alpha)t^{\alpha-1}$ for $\alpha < 1$, we have the integral representation

$$\frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{\sin(\pi\alpha)}{\pi} \int_0^\infty e^{-tz} z^{-\alpha} dz = \frac{\sin(\pi\alpha)}{\pi} \int_{-\infty}^\infty e^{-te^s} e^{(1-\alpha)s} ds, \quad (5.1)$$

where we have used the identity $\Gamma(1-\alpha)\Gamma(\alpha) = \pi/\sin(\pi\alpha)$. To express this function as a sum of exponentials, [6] proposes to approximate the integral to the right by the trapezoidal rule. With a step size $h > 0$ this yields

$$T(t, h) = h \frac{\sin(\pi\alpha)}{\pi} \sum_{i=-\infty}^\infty e^{(1-\alpha)ih} e^{-e^{ih}t}. \quad (5.2)$$

Error of the trapezoidal rule. It follows from [33, Theorem 5.1], in the same way as in [15], that the error due to this approximation can be bounded by

$$\left| \frac{t^{\alpha-1}}{\Gamma(\alpha)} - T(t, h) \right| \leq c_\alpha \frac{t^{\alpha-1}}{\Gamma(\alpha)}, \quad c_\alpha = \frac{2(\cos a)^{\alpha-1}}{e^{2\pi a/h} - 1} \quad (5.3)$$

for every $h > 0$ and any $0 < a < \pi/2$. Choosing a close to optimal, we obtain $c_\alpha \leq \varepsilon$, if the step size h satisfies the inequality in

$$h \leq \frac{2\pi a}{\ln\left(1 + \frac{2}{\varepsilon}(\cos a)^{\alpha-1}\right)}, \quad a = \frac{\pi}{2} \left(1 - \frac{(1-\alpha)}{(2-\alpha) \ln \varepsilon^{-1}}\right). \quad (5.4)$$

Error from truncating the series (5.2). Bounding the sum by an integral and using the definition $\Gamma(\alpha, x) = \int_x^\infty e^{-\sigma} \sigma^{\alpha-1} d\sigma$ of the incomplete Gamma function,

we obtain (see also [15])

$$\begin{aligned} h \frac{\sin(\pi\alpha)}{\pi} \sum_{i=-\infty}^{M-1} e^{\alpha i h} e^{-e^{i h} t} &\leq \frac{t^{\alpha-1}}{\Gamma(\alpha)} \left(1 - \frac{\Gamma(1-\alpha, t e^{M h})}{\Gamma(1-\alpha)} \right), \\ h \frac{\sin(\pi\alpha)}{\pi} \sum_{i=N}^{\infty} e^{\alpha i h} e^{-e^{i h} t} &\leq \frac{t^{\alpha-1}}{\Gamma(\alpha)} \left(\frac{\Gamma(1-\alpha, t e^{N h})}{\Gamma(1-\alpha)} \right). \end{aligned} \quad (5.5)$$

Since $\Gamma(1-\alpha, 0) = \Gamma(1-\alpha)$ and $\Gamma(1-\alpha, \infty) = 0$, the bounds can be made arbitrarily small for fixed $t > 0$ by choosing M large negative, and N large positive.

Choice of the parameters. For a given accuracy requirement ε we first choose h as large as possible satisfying (5.4). The bounds (5.5) cannot be made arbitrarily small when $t \rightarrow 0$ or $t \rightarrow \infty$. We therefore restrict our estimates to an interval $\delta \leq t \leq T$ with $\delta > 0$ and $T < \infty$. We then choose M a large negative number such that $\Gamma(1-\alpha, T e^{M h}) \geq (1-\varepsilon)\Gamma(1-\alpha)$, and finally we choose N a large positive number such that $\Gamma(1-\alpha, \delta e^{N h}) \leq \varepsilon\Gamma(1-\alpha)$. This choice of the parameters implies that

$$\left| \frac{t^{\alpha-1}}{\Gamma(\alpha)} - T_M^N(t, h) \right| \leq 3\varepsilon \frac{t^{\alpha-1}}{\Gamma(\alpha)}, \quad T_M^N(t, h) = h \frac{\sin(\pi\alpha)}{\pi} \sum_{i=M}^{N-1} e^{(1-\alpha) i h} e^{-e^{i h} t} \quad (5.6)$$

on the interval $\delta \leq t \leq T$. If we are interested to solve the fractional differential equation on the interval $[0, T]$, we obviously let T in the choice of N be the endpoint of integration. The choice of $\delta > 0$ (und thus of M) will be discussed in Section 5.3.

5.2. Effect of approximating the kernel. We are interested to study the influence of an approximation of the kernel to the solution of the original problem. For this we consider the fractional differential equation (2.5) with $0 < \alpha < 1$, written in its equivalent form

$$y(t) = y_0 + \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} f(s, y(s)) ds. \quad (5.7)$$

We let $\tilde{y}(t)$ be the solution of (5.7), where $k(t) = t^{\alpha-1}/\Gamma(\alpha)$ is replaced by an approximation $\tilde{k}(t)$. We are interested in the error $\|y(t) - \tilde{y}(t)\|$. For this we assume Lipschitz continuity

$$\|f(t, y) - f(t, \tilde{y})\| \leq L_f \|y - \tilde{y}\| \quad (5.8)$$

as well as $\|f(t, y)\| \leq M_f$ in a neighbourhood of the solution.

THEOREM 1. *Consider the problem (5.7), and assume that the approximation $\tilde{k}(t)$ of the kernel $k(t) = t^{\alpha-1}/\Gamma(\alpha)$ (with $0 < \alpha < 1$) satisfies*

$$\int_0^t \left| \frac{s^{\alpha-1}}{\Gamma(\alpha)} - \tilde{k}(s) \right| ds \leq \varepsilon(1+t^\alpha). \quad (5.9)$$

Then, we have

$$\|y(t) - \tilde{y}(t)\| \leq \varepsilon u(t), \quad (5.10)$$

where $u(t)$ is solution of the scalar equation

$$u(t) = \frac{L_f}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} u(s) ds + M_f(1+t^\alpha). \quad (5.11)$$

Proof. Subtracting the equation for $\tilde{y}(t)$ from (5.7) and using the Lipschitz condition and boundedness of $f(t, y)$ yields

$$\|y(t) - \tilde{y}(t)\| \leq \frac{L_f}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} \|y(s) - \tilde{y}(s)\| ds + M_f \int_0^t \left| \frac{s^{\alpha-1}}{\Gamma(\alpha)} - \tilde{k}(s) \right| ds. \quad (5.12)$$

The assumption on the approximation $\tilde{k}(t)$ then implies $\|y(t) - \tilde{y}(t)\| \leq \varepsilon u(t)$ with $u(t)$ given by (5.11). \square

REMARK 2. The scalar equation (5.11) can be solved by using Laplace transforms. For the function $u(t)$ we denote the Laplace transform by $\hat{u}(w)$, and we recall that the Laplace transform of $t^{\alpha-1}$ ($\alpha > 0$) is $\Gamma(\alpha)w^{-\alpha}$. Passing to Laplace transforms in (5.11) we get

$$\hat{u}(w) = \frac{L_f}{w^\alpha} \hat{u}(w) + \frac{M_f}{w} \left(1 + \frac{\Gamma(\alpha+1)}{w^\alpha} \right),$$

so that

$$\hat{u}(w) = \frac{M_f}{w} \left(1 - \frac{L_f}{w^\alpha} \right)^{-1} \left(1 + \frac{\Gamma(\alpha+1)}{w^\alpha} \right).$$

Taking the inverse Laplace transform gives the solution of (5.11).

For example, for $\alpha = 1/2$, this gives

$$u(t) = \frac{M_f}{2L_f} (2L_f + \sqrt{\pi}) e^{L_f^2 t} \left(\operatorname{erf}(L_f \sqrt{t}) + 1 \right).$$

In general we expect exponential bounds of this kind, similarly as for ordinary differential equations.

5.3. Computing the approximation by a sum of exponentials. To get a good approximation of the solution, the kernel approximation has to satisfy the condition (5.9) of Theorem 1. In Section 5.1 we have explained how the parameters h , M , and N have to be chosen to get an approximation $T_M^N(t, h)$ satisfying (5.6). This implies, for $\delta \leq t \leq T$,

$$\int_\delta^t \left| \frac{s^{\alpha-1}}{\Gamma(\alpha)} - T_M^N(s, h) \right| ds \leq 3\varepsilon \frac{t^\alpha}{\Gamma(\alpha+1)}, \quad (5.13)$$

which, up to a constant, is compatible with (5.9). We still have to find a δ , such that the integral over $(0, \delta)$ is $\mathcal{O}(\varepsilon)$. The estimates (5.3) and (5.5) are valid for all $t > 0$. For $t \in (0, \delta)$ they give bounds $\varepsilon t^{\alpha-1}/\Gamma(\alpha)$, with the exception of the second estimate of (5.5), which is only bounded by $t^{\alpha-1}/\Gamma(\alpha)$. To satisfy (5.9), we choose $\delta > 0$ such that

$$\int_0^\delta \frac{t^{\alpha-1}}{\Gamma(\alpha)} dt = \frac{\delta^\alpha}{\Gamma(\alpha+1)} \leq \varepsilon. \quad (5.14)$$

For the computation of M and N , we have to find $x_* = Te^{Mh}$ and $x^* = \delta e^{Nh}$ such that

$$\Gamma(1-\alpha, x_*) \geq (1-\varepsilon)\Gamma(1-\alpha) \quad \text{and} \quad \Gamma(1-\alpha, x^*) \leq \varepsilon\Gamma(1-\alpha).$$

Making use of the asymptotic formulas $\Gamma(1 - \alpha, x) \geq \Gamma(1 - \alpha) - x^{1-\alpha}/(1 - \alpha)$ and $\Gamma(1 - \alpha, x) \leq x^{-\alpha} e^{-x}$, we find the approximations $x_* = (\Gamma(2 - \alpha) \varepsilon)^{1/(1-\alpha)}$ and $x^* = -\ln(\Gamma(1 - \alpha) \varepsilon)$.

For a given parameter $\alpha \in (0, 1)$, an accuracy requirement ε and a final time T , Algorithm 1 summarizes the computation of the parameter δ as well h, M, N , such that the kernel approximation $T_M^N(t, h)$ of (5.6) satisfies (5.9).

Algorithm 1: Choice of the parameters for the fractional kernel $t^{\alpha-1}/\Gamma(\alpha)$.

Data: α, ε, T

Result: δ, h, M, N

begin

- 1 Compute $\delta = (\Gamma(\alpha + 1) \varepsilon)^{1/\alpha}$
 - 2 Set $h = \frac{2\pi a}{\ln(1 + \frac{2}{\varepsilon}(\cos a)^{\alpha-1})}$, where $a = \frac{\pi}{2} \left(1 - \frac{(1 - \alpha)}{(2 - \alpha) \ln \varepsilon^{-1}}\right)$
 - 3 Set $x_* = (\Gamma(2 - \alpha) \varepsilon)^{1/(1-\alpha)}$ and compute M such that $Te^{Mh} = x_*$
 - 4 Set $x^* = -\ln(\Gamma(1 - \alpha) \varepsilon)$ and compute N such that $\delta e^{Nh} = x^*$
- return**
-

To get an impression of the values for h, M , and N , we present in Table 5.1 their values for the choice $T = 1000$, for $\varepsilon = 10^{-5}$ and $\varepsilon = 10^{-10}$, and for different values of $\alpha \in (0, 1)$.

TABLE 5.1
Parameters for the fractional kernel for $T = 1000$.

	α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\varepsilon = 10^{-5}$	h	0.65	0.66	0.67	0.69	0.70	0.72	0.73	0.75	0.78
	M	-31	-33	-36	-39	-44	-51	-63	-87	-159
	N	148	93	62	47	37	31	26	23	20
$\varepsilon = 10^{-10}$	h	0.37	0.37	0.38	0.38	0.39	0.39	0.40	0.40	0.41
	M	-91	-99	-109	-122	-141	-169	-215	-308	-586
	N	649	326	218	163	131	109	93	81	71

6. The case $\alpha > 1$. Until now we mainly treated the situation, where $0 < \alpha < 1$. Here we consider an integral term (2.9) with $\alpha > 1$. One possibility is to introduce the new variable

$$y_{d+1}(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} G(t, y(s)) ds.$$

For $1 < \alpha < 2$, we differentiate it with respect to time and obtain the differential equation

$$\dot{y}_{d+1}(t) = \frac{1}{\Gamma(\alpha-1)} \int_0^t (t-s)^{\alpha-2} G(t, y(s)) ds, \quad y_{d+1}(0) = 0,$$

which we add to the original system. Since $\alpha - 2 = \alpha_0 - 1$ with $\alpha_0 \in (0, 1)$, we are in the situation that can be treated by the approach of Section 5.1. For an $\alpha \in (2, 3)$, we differentiate twice, and so on. With this transformation we can assume without loss of generality that in the problem (2.5) all integral terms have $\alpha_j \in (0, 1)$. The present section is devoted to another possibility for treating the case $\alpha > 1$.

6.1. Splitting of the kernel. The kernel approximation of Section 5 is valid only for $0 < \alpha < 1$. For $\alpha > 1$, $m = \lceil \alpha \rceil \geq 2$, we put $\alpha_0 = \alpha - m + 1 \in (0, 1)$. It is possible to split the kernel according to

$$\frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{t^{m-1}}{(\alpha-1) \cdots (\alpha-m+1)} \cdot \frac{t^{\alpha_0-1}}{\Gamma(\alpha_0)}, \quad (6.1)$$

and to apply the approximation (5.6) to the factor $t^{\alpha_0-1}/\Gamma(\alpha_0)$ (see also [15]). This then gives an approximation for $t^{\alpha-1}/\Gamma(\alpha)$ as a sum of exponentials multiplied by a monomial. The linear chain trick can be extended to this situation, which also reduces the integral equation to a system of ordinary differential equations.

6.2. Extension of the linear chain trick. For notational convenience we here suppress the index j in the integral (2.9), and also in α_j and $G_j(t, y)$. We assume that the second factor in (6.1) is approximated by

$$\frac{1}{\Gamma(\alpha_0)} t^{\alpha_0-1} \approx \sum_{i=1}^n c_i e^{-\gamma_i t}. \quad (6.2)$$

Since $0 < \alpha_0 < 1$, this approximation can be the one of Section 5.1. We now replace the integral (2.9) by

$$\frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} G(s, y(s)) \, ds \approx \frac{1}{(\alpha-1) \cdots (\alpha-m+1)} \sum_{i=1}^n c_i z_{i,m}(t), \quad (6.3)$$

where $z_{i,m}(t)$ is the last function among

$$z_{i,k}(t) = \int_0^t (t-s)^{k-1} e^{-\gamma_i(t-s)} G(s, y(s)) \, ds, \quad k = 1, \dots, m.$$

Differentiation of this function gives the ordinary differential equations

$$\dot{z}_{i,k}(t) = -\gamma_i z_{i,k}(t) + \begin{cases} (k-1) z_{i,k-1}(t) & k = 2, \dots, m \\ G(t, y(t)) & k = 1. \end{cases} \quad (6.4)$$

If the integral terms in (2.5) with $\alpha_j > 1$ are replaced by the approximation (6.3), we have to add the m differential equations (6.4) to the original system.

6.3. Structure of the Jacobian matrix. The Jacobian matrix of the complete system is again of the form (3.6) with a different interpretation of J_j , B_j , and C_j for those indices j , for which $\alpha_j > 1$.

The matrix J_j is bi-diagonal of dimension $m_j \cdot n_j$, where $m_j = \lceil \alpha_j \rceil$ and n_j is the number of summands in (6.2). The diagonal elements are m_j times $\gamma_{1,j}$, then m_j times $\gamma_{2,j}$ until a block of m_j elements equal to $\gamma_{n_j,j}$. The subdiagonal consists of n_j vectors $(1, \dots, m_j - 1)$ separated by 0. This means that for $1 < \alpha_j < 2$ the subdiagonal is $(1, 0, 1, 0, \dots, 1, 0)$.

The matrix $B_j = e(\partial G_j / \partial y)$ is a rank-one matrix, where the derivative $\partial G_j / \partial y$ is a row vector of dimension d . The column vector e is composed of n_j vectors $(1, 0, \dots, 0)^\top \in \mathbb{R}^{m_j}$.

The matrix $C_j = (\partial F / \partial I_j) \tilde{c}_j^\top$ is also a rank-one matrix. Here, the vector \tilde{c}_j^\top is composed of n_j vectors $(0, \dots, 0, c_{i,j}) \in \mathbb{R}^{m_j}$ (for $i = 1, \dots, n_j$).

6.4. Use of the fast linear algebra in RADAU5. Similar as in Section 4.3 we can use the subroutines of DC_SUMEXP.F for a fast linear algebra. Here, the dimension of the system has to be defined as $N = d + (m_1 n_1 + 2) + \dots + (m_{d_I} n_{d_I} + 2)$. The parameters IJAC = 1, MLJAC = d , and MUJAC = 0 are the same as in Section 4.3. Also the upper left $d \times d$ matrix of DFY(LDFY,N) is that of the Jacobian matrix (3.6).

Next to the right we still have d_I blocks corresponding to integral terms in the problem (2.8). Here, the j th block has $m_j n_j + 2$ columns. As in Section 4.3 the first column contains the vector $\partial F / \partial I_j$, and the second column contains the derivative $\partial G_j / \partial y$, written as column vector.

The first row of the j th block has the coefficients of \tilde{c}_j at the positions 3 to $m_j n_j + 2$, and the second row has the diagonal of J_j at the positions 3 to $m_j n_j + 2$. New is that the third row contains the subdiagonal elements of J_j . The last element of the third row, which is not used by the subdiagonal, is put to a negative number to indicate the end of the j th block.

7. Numerical experiments. We consider a few illustrative examples. The first one is scalar with known exact solution. The second one is a modification of the well-known Brusselator equation. Both of them are used as test equations in a series of publications. The third and fourth are fractional derivative partial differential equations.

7.1. Example 1: scalar fractional ODE with exact solution. As a first test example we consider the equation from Diethelm et al. [12]

$$D_*^\alpha y(t) = \frac{9\Gamma(1+\alpha)}{4} - \frac{3t^{4-\alpha/2}\Gamma(5+\alpha/2)}{\Gamma(5-\alpha/2)} + \frac{\Gamma(9)t^{8-\alpha}}{\Gamma(9-\alpha)} + \left(\frac{3}{2}t^{\alpha/2} - t^4\right)^3 - y(t)^{3/2}, \quad (7.1)$$

with $y(0) = 0$. The inhomogeneity is chosen such that

$$y(t) = \frac{9t^\alpha}{4} - 3t^{4+\alpha/2} + t^8 = \left(\frac{3}{2}t^{\alpha/2} - t^4\right)^2$$

is the exact solution of the equation. It starts at $y(0) = 0$, reaches a maximum at $t = (3\alpha/16)^{1/(4-\alpha/2)}$, and vanishes at $t = (3/2)^{1/(4-\alpha/2)}$ which, e.g. for $\alpha = 1/2$, results in a loss of numerical well-posedness of the problem for $t > 1$. Hence, in our numerical experiments we integrate the equation only up to $T = 1$.

First experiment (connection between Tol and ε). For the numerical integration we apply the code RADAU5 with accuracy requirement $Atol = Rtol = Tol = 10^{-7}$ to the augmented system (3.4). We use the kernel approximation (5.6) with parameters h, M, N , determined by Algorithm 1. We study the relative error at $T = 1$ for different values of ε , which determines the accuracy of the approximation by the sum of exponentials.

The result is shown in Table 7.1 for $\alpha = 1/2$. We can observe that for $\varepsilon \geq Tol$ the error is essentially proportional to ε , which corresponds to the error of the fractional kernel approximation. For $\varepsilon \leq Tol$ the error remains close to Tol , which indicates that the error of the time integration is dominant. The parameter values for $\varepsilon = Tol = 10^{-7}$ are indicated in bold.

Second experiment. As a second experiment we look at the linear algebra. We apply both the standard Gauss method and the fast algorithm exploiting the structure of the Jacobian. The results are shown in Table 7.2 and show a striking improvement of

ε	h	δ	M	N	err
10^{-4}	0.839	$7.85 \cdot 10^{-9}$	-23	25	$6.35 \cdot 10^{-5}$
10^{-5}	0.697	$7.85 \cdot 10^{-11}$	-34	37	$6.36 \cdot 10^{-6}$
10^{-6}	0.596	$7.85 \cdot 10^{-13}$	-47	52	$5.77 \cdot 10^{-7}$
10^{-7}	0.522	$7.85 \cdot 10^{-15}$	-63	68	$5.63 \cdot 10^{-7}$
10^{-8}	0.469	$7.85 \cdot 10^{-17}$	-80	87	$6.37 \cdot 10^{-7}$
10^{-9}	0.418	$7.85 \cdot 10^{-19}$	-100	108	$7.23 \cdot 10^{-7}$
10^{-10}	0.380	$7.85 \cdot 10^{-21}$	-122	131	$5.79 \cdot 10^{-7}$

TABLE 7.1

Error behavior obtained by applying RADAU5 to the test problem (7.1) with final point $T = 1$ and $Tol = 10^{-7}$ for different computed values of δ , h , M , and N (computed by Algorithm 1).

$Tol = \varepsilon$	10^{-5}	10^{-7}	10^{-9}	10^{-11}
cpu (standard)	$0.36 \cdot 10^{-1}$	$0.17 \cdot 10^0$	$0.69 \cdot 10^0$	$0.25 \cdot 10^1$
cpu (fast)	$0.96 \cdot 10^{-3}$	$0.21 \cdot 10^{-2}$	$0.58 \cdot 10^{-2}$	$0.16 \cdot 10^{-1}$
err	$1.4 \cdot 10^{-5}$	$5.63 \cdot 10^{-7}$	$2.62 \cdot 10^{-8}$	$5.50 \cdot 10^{-10}$

TABLE 7.2

Comparison of different linear algebra solvers for the system (7.1) with $\alpha = 1/2$.

the CPU time due to the fast linear solver. Since both approaches are mathematically equivalent, they give the same error in the solution approximation.

Third experiment (comparison of different integral formulations). For the case where $\alpha > 1$, one can either use the formulation (2.6) as a Volterra integral equation, or the formulation (2.7) as a Volterra integro-differential equation. Table 7.3 shows, for fixed $Tol = \varepsilon = 10^{-6}$ the values of M , N in the approximation by a sum of exponentials and the error and cpu time as a function of α , when α ranges from 1 to 2. The value of M is the same for both approaches and also the error is similar. In particular, for α close to 1, the value of N is significantly larger for the version (2.7). When looking at the cpu time, one notices that the formulation (2.6) is more efficient for α close to 1, whereas the formulation (2.7) is slightly better for α close to 2.

α	1.1	1.3	1.5	1.7	1.9
M	-28	-35	-47	-75	-212
N	28	23	20	17	15
err	$0.33 \cdot 10^{-6}$	$0.74 \cdot 10^{-6}$	$0.14 \cdot 10^{-5}$	$0.11 \cdot 10^{-5}$	$0.77 \cdot 10^{-6}$
cpu time	$0.11 \cdot 10^{-2}$	$0.10 \cdot 10^{-2}$	$0.11 \cdot 10^{-2}$	$0.11 \cdot 10^{-2}$	$0.28 \cdot 10^{-2}$
M	-28	-35	-47	-75	-212
N	235	86	52	37	28
err	$0.25 \cdot 10^{-5}$	$0.11 \cdot 10^{-5}$	$0.44 \cdot 10^{-7}$	$0.44 \cdot 10^{-6}$	$0.57 \cdot 10^{-6}$
cpu time	$0.61 \cdot 10^{-2}$	$0.15 \cdot 10^{-2}$	$0.89 \cdot 10^{-3}$	$0.11 \cdot 10^{-2}$	$0.17 \cdot 10^{-2}$

TABLE 7.3

The upper block shows the values M , N , the error and the cpu time for the formulation (2.6) of the problem (7.1), whereas the lower block shows them for the formulation (2.7).

7.2. Example 2: Brusselator model. The so-called *Brusselator* is a model of a chemical reaction [23] that possesses periodic solutions and has applications in the interpretation of biological phenomena. It is often used as a test example for nonstiff time integrators. A modification, where the time derivative is replaced by a fractional derivative, is proposed by [13] as a test for codes solving fractional differential

equations. It is given by

$$\begin{aligned} D_*^{\alpha_1} y_1(t) &= A - (B+1)y_1(t) + y_1(t)^2 y_2(t) \\ D_*^{\alpha_2} y_2(t) &= B y_1(t) - y_1(t)^2 y_2(t) \end{aligned} \quad (7.2)$$

We choose the parameters and initial values as $A = 1$, $B = 3$, and $y_1(0) = 1.2$, $y_2(0) = 2.8$. The degrees of the derivatives are $\alpha_1 = 1.3$, $\alpha_2 = 0.8$. Since $\alpha_1 > 1$, we need a further initial value $\dot{y}_1(0) = 1$. Figure 7.1 shows the two solution components as a function of time.

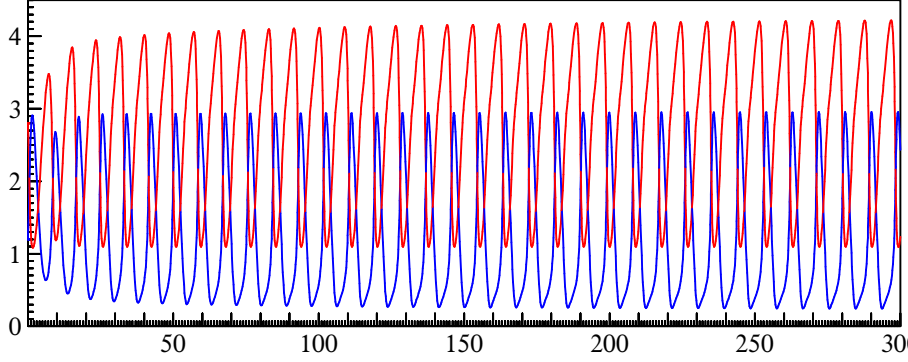


FIG. 7.1. Solution of (7.2) with parameter $\alpha_1 = 1.3$, $\alpha_2 = 0.8$ in the time interval $[0, 220]$ ($y_1(t)$ is in blue, $y_2(t)$ is in red)

We apply the code RADAU5 to this fractional differential equation on the interval $[0, 220]$ with different values of $Tol = \varepsilon$. For the first component (with $\alpha_1 > 1$) we use the approach (2.7). In Table 7.4 we present the relative error and the cpu time. In the first two rows are the values M and N obtained by Algorithm 1. Each entry has two numbers: the first one corresponds to α_1 , the second to α_2 .

$Tol = \varepsilon$	10^{-4}	10^{-6}	10^{-8}	10^{-10}
M	-24 - 57	-44 - 118	-71 - 200	-104 - 304
N	42 15	86 32	144 53	218 81
cpu (fast)	$0.10 \cdot 10^{-1}$	$0.51 \cdot 10^{-1}$	$0.10 \cdot 10^0$	$0.31 \cdot 10^0$
err	$0.69 \cdot 10^{-2}$	$0.60 \cdot 10^{-4}$	$0.67 \cdot 10^{-6}$	$0.89 \cdot 10^{-8}$

TABLE 7.4
Numerical integration of (7.2) using the fast linear algebra routine

An accurate approximation of the solution at $T = 220$ is given by

$$y_1(T) = 1.0097684171, \quad y_2(T) = 2.1581264031.$$

We note that with a value $\varepsilon = Tol = 10^{-6}$ we obtain a relative error $0.60 \cdot 10^{-4}$ in a cpu time of 0.051 seconds. The computation takes 1244 accepted steps, which corresponds to a mean step size $h \approx 0.18$.

Comparison with the codes by Garrappa [13]. We test the problem with the codes at <https://www.dm.uniba.it/it/members/garrappa/software>. For these codes the user has to provide a step size instead of an accuracy requirement Tol .

Applying the implicit code FDE_PI1_IM with step size $h = 10^{-4}$ gives a relative error $\text{err} = 2.0665 \cdot 10^{-4}$ in about 84 seconds. Instead the explicit code FDE_PI1_EX yields a relative error $\text{err} = 2.0696 \cdot 10^{-4}$ in about 63.75 seconds.

The peak of memory allocation is about 165 Megabytes for the implicit code and 164 Megabytes for the explicit one. With larger T and smaller step size h , these codes may encounter memory problems.

The difference with respect to our integrator is significant. In order to get the same relative error, our code (using a variable step size) requires a CPU time of less than 0.05 seconds. Since our code solves an ordinary differential equation, no storage of the solution in the past is needed. Therefore, we do not have any difficulties with the memory requirement.

Comparison with the codes by Brugnano et al. [7]. We also consider the recent code FHBVM2 available at <https://people.dimai.unifi.it/brugnano/fhbvm/>. This code allows only one single fractional derivative so that we set $\alpha = 0.8$ for both equations. Setting $T = 220$ and $N = 300$ steps, we found an approximate solution in about 4.6 seconds with a very moderate peak of memory allocation of about 1 Megabyte. With $T = 1000$ and $N = 1200$ the CPU time is about 35.5 seconds. With a higher $\alpha > 1$, the code becomes slower.

With our code we integrate the problem till $T = 220$ in 0.8 seconds and till $T = 1000$ in 3.2 seconds, with an accuracy of about 10^{-7} . However, since our code is implemented in FORTRAN, making a true comparison is not possible.

7.3. Example 3: a multi-term problem. A benchmark problem from [34] (see also [13, Eq. (35)]) is given by ($0 < \alpha < 1$)

$$\ddot{y}(t) + D_*^{\alpha+2}y(t) + \ddot{y}(t) + 4\dot{y}(t) + D_*^\alpha y(t) + 4y(t) = 6 \cos t, \quad (7.3)$$

with initial conditions $y(0) = 1$, $\dot{y}(0) = 1$, $\ddot{y}(0) = -1$, so that its exact solution is

$$y(t) = \sqrt{2} \sin\left(t + \frac{\pi}{4}\right) \quad (7.4)$$

independent of α . For studying the stability of this linear differential equation we consider the characteristic equation of the homogeneous problem, which is given by

$$\mathcal{L}_\alpha(z) = z^{\alpha+2} + z^\alpha + z^3 + z^2 + 4z + 4 = 0. \quad (7.5)$$

A numerical investigation reveals that at a value $\alpha^* \in (0.654298, 0.654299)$ a pair of complex conjugate roots of (7.5) crosses the imaginary axis from the left complex plane to the right one. The two roots of $\mathcal{L}_\alpha(z)$ are approximately $\pm 1.65686i$. For $\alpha > \alpha^*$ the solution of (7.3) is unstable, while for $\alpha \leq \alpha^*$ it is stable.

For a numerical computation we insert the definition of the Caputo derivative into (7.3) and we write the problem as a first order system (three differential equations and one algebraic equation) which gives

$$\begin{aligned} \dot{y}_0(t) &= y_1(t), & \dot{y}_1(t) &= y_2(t), & \dot{y}_2(t) &= y_3(t) \\ 0 &= y_3(t) + I_\alpha(y_3)(t) + y_2(t) + 4y_1(t) + I_\alpha(y_1)(t) + 4y_0(t) - 6 \cos t, \end{aligned}$$

where, with $\alpha_0 = 1 - \alpha$,

$$I_\alpha(y)(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} y(s) ds = \frac{1}{\Gamma(\alpha_0)} \int_0^t (t-s)^{\alpha_0-1} y(s) ds.$$

This system is of the form (2.8) and can be solved by the approach of the preceding sections. As suggested in [13] we consider $\alpha = 1/2$ and $T = 5000$. We apply the code RADAU5 with the fast linear algebra routines and $\varepsilon = \text{Tol} = 10^{-5}$ to this system. With 15 812 accepted steps the computation gives an error $0.11 \cdot 10^{-5}$ at the final point, and it needs 0.4 seconds cpu time. For values $0 < \alpha \leq \alpha^*$ we get similar accurate results. However, for larger values, such as $\alpha = 0.655 > \alpha^*$, we observe an error $0.499 \cdot 10^3$ demonstrating the instability of the equation.

8. Solving 1D fractional partial differential equations. We conclude by considering the following fractional PDE with Dirichlet boundary conditions

$$\begin{aligned} D_*^\alpha u(x, t) &= \frac{\partial^2 u}{\partial x^2}(x, t) + f(x, t) & \text{for } x \in (0, 1), \quad t > 0. \\ u(0, t) &= u(1, t) = 0 \end{aligned} \quad (8.1)$$

Here, D_*^α denotes the fractional derivative with respect to time t . For $0 < \alpha < 1$ we impose an initial condition $u(x, 0) = u_0(x)$, similar as for parabolic problems, and for $1 < \alpha < 2$ we impose $u(x, 0) = u_0(x)$ and $\frac{\partial u}{\partial t}(x, 0) = u_0^{(1)}(x)$ as for hyperbolic problems.

With the method of lines we transform this problem into a fractional ordinary differential equation. We consider an equi-spaced grid $x_i = i \Delta x$ for $i = 1, \dots, d$ and $\Delta x = 1/(d+1)$, and we apply central finite differences to the space derivative. For the solution approximation $y_i(t) \approx u(x_i, t)$ at the grid points this yields the system

$$D_*^\alpha y_i(t) = \frac{1}{\Delta x^2} (y_{i+1}(t) - 2y_i(t) + y_{i-1}(t)) + f(x_i, t), \quad i = 1, \dots, d, \quad (8.2)$$

with $y_0(t) = y_{d+1}(t) = 0$, and initial values $y_i(0) = u_0(x_i)$ for $0 < \alpha < 1$ and $y_i(0) = u_0(x_i)$, $\dot{y}_i(0) = u_0^{(1)}(x_i)$ for the case $1 < \alpha < 2$. According to (2.7), the fractional differential equation (8.2) becomes, for $0 < \alpha < 1$,

$$y_i(t) = u_0(x_i) + I_i(t), \quad I_i(t) = \frac{1}{\gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} G_i(y(s)) ds, \quad (8.3)$$

where $G_i(y(s))$ represents the right-hand side of (8.2). The dimension d of this system is typically very large and even the fast linear algebra explained in Section 4.3 is not efficient enough. We have to exploit the special structure of the arising Jacobian matrix.

Exploiting banded structures. We say that the problem (2.8) has *banded structure*, if the mass matrix M and $\partial F/\partial I$ are diagonal matrices, and the matrices $\partial F/\partial y$ and $\partial G/\partial y$ are both banded with upper and lower bandwidths b_u and b_l , respectively.

Note that the system (8.3) has a tridiagonal banded structure. In fact, the matrix M is the zero-matrix, $\partial F/\partial I$ is the identity, the matrix $\partial F/\partial y$ is also a zero matrix, and the matrix $\partial G/\partial y$ is tridiagonal.

LEMMA 8.1. *Consider a problem that has banded structure with band widths b_u and b_l . Then, the matrix \hat{J} of (4.1) is banded with the same band widths.*

Proof. By assumption the matrix J is banded. Moreover, the only non-zero element of $\partial F/\partial I_j$ is at position j , and $\partial G_j/\partial y$ is the j th row of a banded matrix. Therefore, every rank-one matrix in the sum of (4.1) is banded with the same bandwidths as J . This proves the statement. \square

This lemma shows that we can exploit the banded structure in our fast linear algebra routines.

Use of fast linear algebra for banded problems. In the calling program one has to define the dimension N as in Section 4.3 (or as in Section 6.4 if some of the exponents satisfy $\alpha_j > 1$). For banded structures one has to put $IJAC = 1$, $MLJAC = b_l$, and $MUJAC = b_u$ (the code requires $b_u \geq 1$).

The information of the Jacobian has to be stored in $DFY(LDFY, N)$ as follows. The $b_u + 1 + b_l$ diagonals of $J = \partial F / \partial y$ are stored as rows in such a way that columns of J remain columns in DFY . In this way the element $DFY(1, 1)$ is not used by J . We put $DFY(1, 1) = d$, so that the number of columns is provided. As in Section 4.3 there are d_I blocks corresponding to the d_I integral terms in the problem. New is the storage in the first two columns of each block, the rest is not changed. The first column of the j th block contains the j th element of $\partial F / \partial I_j$ in position $b_u + 1$. The second column contains the relevant information of $\partial G_j / \partial y$ in the positions 1 to $b_u + 1 + b_l$. We finally put $DFY(3, N) = -d_I$ to provide also the number of integrals.

Numerical experiment. For our numerical illustration we choose

$$f(x, t) = \frac{1}{2}x(1-x) \frac{\Gamma(\beta)\beta}{\Gamma(\beta+1-\alpha)} t^{\beta-\alpha} + (t^\beta + 1), \quad \beta \geq \alpha, \quad (8.4)$$

for which the exact solution of (8.1) is known to be

$$u(x, t) = \frac{1}{2}x(1-x)(t^\beta + 1). \quad (8.5)$$

We choose $\alpha = 1/3$, $\beta = 5/3$, and we consider the numerical integration over the interval $[0, 1000]$. With $\varepsilon = 10^{-6}$ the fractional kernel $t^{\alpha-1}$ is approximated by a sum of exponentials with parameters $M = -49$ and $N = 77$ obtained from Algorithm 1.

$d = d_I$	100	300	1000	3000	10000
cpu (banded)	$0.65 \cdot 10^{-1}$	$0.20 \cdot 10^0$	$0.68 \cdot 10^0$	$0.20 \cdot 10^1$	$0.69 \cdot 10^1$
err	$0.11 \cdot 10^{-7}$	$0.19 \cdot 10^{-7}$	$0.46 \cdot 10^{-8}$	$0.64 \cdot 10^{-7}$	$0.11 \cdot 10^{-6}$

TABLE 8.1

Numerical results for the 1D partial differential equation using the banded linear algebra routines

The cpu time and the relative error obtained by the code RADAU5 with $Tol = \varepsilon = 10^{-6}$ and with the banded fast linear algebra option are given in Table 8.1. We can observe that the cpu time increases linearly with the dimension d of the system. Independent of the dimension, the code takes about 43 steps with step sizes increasing from $\Delta t \approx 0.04$ in the beginning to $\Delta t \approx 137$ at the end of the interval.

Further fractional PDEs. Systems of reaction diffusion equations with memory can be modeled by fractional equations of the form (see e.g., [25])

$$\begin{aligned} D_*^\alpha u_1(x, t) &= K \frac{\partial^2 u_1}{\partial x^2}(x, t) - k_1 u_1(x, t) u_2(x, t) + (k_2 + k_3) u_3(x, t) \\ D_*^\alpha u_2(x, t) &= K \frac{\partial^2 u_2}{\partial x^2}(x, t) - k_1 u_1(x, t) u_2(x, t) + k_2 u_3(x, t) \\ D_*^\alpha u_3(x, t) &= K \frac{\partial^2 u_3}{\partial x^2}(x, t) + k_1 u_1(x, t) u_2(x, t) - (k_2 + k_3) u_3(x, t) \end{aligned} \quad (8.6)$$

for $x \in (0, 1)$ and $t > 0$. We consider homogeneous Dirichlet boundary conditions, suitable initial functions, and parameters $K = 0.5$, $k_1 = 1$, $k_2 = 2$, and $k_3 = 3$. After a semi-discretization as in (8.2) we obtain a system of fractional ODEs for $y = (y_1^1, \dots, y_d^1, y_1^2, \dots, y_d^2, y_1^3, \dots, y_d^3)$, where $y_i^j(t) \approx u_j(x_i, t)$. Due to the reaction term the Jacobian matrix is no longer banded. However, since the reaction term is not stiff (when compared to $\partial^2/\partial x^2$), we can remove all elements that are not on the diagonal or on the first super- and subdiagonals. The neglect of such elements increases slightly the number of simplified Newton iterations, but the decrease in cpu time is remarkable. Similar as in Table 8.1 we observe that the cpu time scales linearly with the number of grid points.

In the case where the reaction is stiff, such a reduction of the Jacobian matrix to tridiagonal form is not recommended. If one orders the elements of the vector y according to $y = (y_1^1, y_1^2, y_1^3, \dots, y_d^1, y_d^2, y_d^3)$, then the Jacobian matrix becomes banded with bandwidths $b_l = b_u = 3$. This allows for an efficient treatment of the linear algebra without neglecting elements of the Jacobian matrix.

	<i>nfcn</i>	<i>njac</i>	<i>nstep</i>	<i>error</i>	<i>cpu time</i>
<i>3-diagonal</i>	274	29	29	$8.21 \cdot 10^{-6}$	1.63
<i>7-diagonal</i>	183	4	28	$9.92 \cdot 10^{-6}$	1.17

TABLE 8.2

Numerical comparison of different approximations of the Jacobian matrix

For our numerical experiment we consider initial values $u_1(x, 0) = 0.5x(1 - x)$, $u_2(x, 0) = x^2(1 - x)$, $u_3(x, 0) = 1.5x(1 - x)^2$, and an equidistant grid with $d = 1000$ interior grid points. We fix $Tol = \varepsilon = 10^{-5}$, and we compute the parameters of the kernel approximation with Algorithm 1. For $\alpha = 1/2$ and $t \in [0, 30]$ this gives $M = -39$ and $N = 37$, so that the complete system (3.4), which consists of $3d$ y -variables and of $3d$ integral terms, is of dimension $3d + 3d(N - M) = 231d = 231\,000$. Table 8.2 presents the statistics of the code RADAU5 for the two approaches: neglecting some elements of the reaction term (3-diagonal Jacobian matrix) and considering the complete Jacobian with an ordering of variables to obtain a 7-diagonal structure. We see that the number of steps *nstep* and the relative *error* are more or less identical. The number of function evaluations *nfcn* and the number of Jacobian evaluations *njac* are larger for the 3-diagonal version. This is explained by the fact that due to the non-exact Jacobian the code requires about 3 simplified Newton iteration per step in contrast to the 7-diagonal version, which needs only 2 simplified Newton iterations per step. This is also reflected in the *cpu time*, which is given in seconds.

9. Conclusions. In this article we have described an efficient memoryless methodology to deal with fractional differential equations possibly coupled with stiff ordinary and differential-algebraic equations.

The methodology is based on a suitable expansion of the fractional kernels in terms of exponential functions, which allows to transform the problem into an augmented set of stiff ODEs of large dimension. Our numerical experiments confirm the efficiency of this methodology which allows to get a much more versatile code, able to deal with general problems, a feature that makes it very appealing in disciplines where fractional differential equations have to be solved accurately and efficiently.

Acknowledgments. Nicola Guglielmi acknowledges that his research was supported by funds from the Italian MUR (Ministero dell'Università e della Ricerca)

within the PRIN 2022 Project “Advanced numerical methods for time dependent parametric partial differential equations with applications” and the PRIN-PNRR Project “FIN4GEO”. He is also affiliated to the INdAM-GNCS (Gruppo Nazionale di Calcolo Scientifico).

Ernst Hairer acknowledges the support of the Swiss National Science Foundation, grant No.200020 192129.

REFERENCES

- [1] X. Antoine, A. Arnold, C. Besse, M. Ehrhardt, and A. Schädle. A review of transparent and artificial boundary conditions techniques for linear and nonlinear Schrödinger equations. *Commun. Comput. Phys.*, 4(4):729–796, 2008.
- [2] D. Baffet. A Gauss-Jacobi kernel compression scheme for fractional differential equations. *J. Sci. Comput.*, 79(1):227–248, 2019.
- [3] D. Baffet and J. S. Hesthaven. High-order accurate adaptive kernel compression time-stepping schemes for fractional differential equations. *J. Sci. Comput.*, 72(3):1169–1195, 2017.
- [4] D. Baffet and J. S. Hesthaven. A kernel compression scheme for fractional differential equations. *SIAM J. Numer. Anal.*, 55(2):496–520, 2017.
- [5] L. Banjai and M. López-Fernández. Efficient high order algorithms for fractional integrals and fractional differential equations. *Numer. Math.*, 141(2):289–317, 2019.
- [6] G. Beylkin and L. Monzón. Approximation by exponential sums revisited. *Appl. Comput. Harmon. Anal.*, 28:131–149, 2010.
- [7] L. Brugnano, K. Burrage, P. Burrage, and F. Iavernaro. A spectrally accurate step-by-step method for the numerical solution of fractional differential equations. *J. Sci. Comput.*, 99(2):Paper No. 48, 28, 2024.
- [8] H. Brunner. *Collocation methods for Volterra integral and related functional differential equations*, volume 15 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2004.
- [9] H. Brunner and P. J. van der Houwen. *The numerical solution of Volterra equations*, volume 3 of *CWI Monographs*. North-Holland Publishing Co., Amsterdam, 1986.
- [10] A. Cardone, M. Donatelli, F. Durastante, R. Garrappa, M. Mazza, and M. Popolizio, editors. *Fractional differential equations—modeling, discretization, and numerical solvers*, volume 50 of *Springer INdAM Series*. Springer, Singapore, 2023.
- [11] K. Diethelm. *The analysis of fractional differential equations*, volume 2004 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2010. An application-oriented exposition using differential operators of Caputo type.
- [12] K. Diethelm, N. J. Ford, and A. D. Freed. Detailed error analysis for a fractional Adams method. *Numer. Algorithms*, 36(1):31–52, 2004.
- [13] R. Garrappa. Numerical solution of fractional differential equations: A survey and a software tutorial. *Mathematics*, 6(2), 2018.
- [14] R. Gorenflo and F. Mainardi. Fractional calculus: integral and differential equations of fractional order. In *Fractals and fractional calculus in continuum mechanics (Udine, 1996)*, volume 378 of *CISM Courses and Lect.*, pages 223–276. Springer, Vienna, 1997.
- [15] N. Guglielmi and E. Hairer. Applying stiff integrators for ordinary differential equations and delay differential equations to problems with distributed delays. *SIAM J. Sci. Comput.*, 47(1):A102–A123, 2025.
- [16] E. Hairer and P. Maass. Numerical methods for singular nonlinear integro-differential equations. *Appl. Numer. Math.*, 3(3):243–256, 1987.
- [17] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics 14. Springer-Verlag, Berlin, 2nd edition, 1996.
- [18] P. Henrici. *Applied and computational complex analysis. Vol. 2*. Wiley-Interscience [John Wiley & Sons], New York-London-Sydney, 1977. Special functions—integral transforms— asymptotics—continued fractions.
- [19] Yuxiang Huang, Fanhai Zeng, and Ling Guo. Error estimate of the fast L1 method for time-fractional subdiffusion equations. *Appl. Math. Lett.*, 133:Paper No. 108288, 8, 2022.
- [20] Shidong Jiang, Jiwei Zhang, Qian Zhang, and Zhimin Zhang. Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations. *Commun. Comput. Phys.*, 21(3):650–678, 2017.
- [21] B. Jin. *Fractional differential equations—an approach via fractional derivatives*, volume 206

- of *Applied Mathematical Sciences*. Springer, Cham, [2021] ©2021.
- [22] A. A. Kilbas, H. M. Srivastava, and J. J. Trujillo. *Theory and applications of fractional differential equations*, volume 204 of *North-Holland Mathematics Studies*. Elsevier Science B.V., Amsterdam, 2006.
 - [23] R. Lefever and G. Nicolis. Chemical instabilities and sustained oscillations. *J. theor. Biol.*, 30:267–284, 1971.
 - [24] J.-R. Li. A fast time stepping method for evaluating fractional integrals. *SIAM J. Sci. Comput.*, 31(6):4696–4714, 2009/10.
 - [25] M. López-Fernández, C. Lubich, and A. Schädle. Adaptive, fast, and oblivious convolution in evolution equations with memory. *SIAM J. Sci. Comput.*, 30(2):1015–1037, 2008.
 - [26] J.-F. Lu and A. Hanyga. Numerical modelling method for wave propagation in a linear viscoelastic medium with singular memory. *Geophysical Journal International*, 159(2):688–702, 2004.
 - [27] C. Lubich. Discretized fractional calculus. *SIAM J. Math. Anal.*, 17(3):704–719, 1986.
 - [28] C. Lubich. Convolution quadrature and discretized operational calculus. I. *Numer. Math.*, 52(2):129–145, 1988.
 - [29] C. Lubich and A. Ostermann. Runge-Kutta methods for parabolic equations and convolution quadrature. *Math. Comp.*, 60(201):105–131, 1993.
 - [30] I. Podlubny. *Fractional differential equations*, volume 198 of *Mathematics in Science and Engineering*. Academic Press, Inc., San Diego, CA, 1999. An introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications.
 - [31] S. G. Samko, A. A. Kilbas, and O. I. Marichev. *Fractional integrals and derivatives*. Gordon and Breach Science Publishers, Yverdon, 1993. Theory and applications, Edited and with a foreword by S. M. Nikolskiĭ, Translated from the 1987 Russian original, Revised by the authors.
 - [32] A. Schädle, M. López-Fernández, and C. Lubich. Fast and oblivious convolution quadrature. *SIAM J. Sci. Comput.*, 28(2):421–438, 2006.
 - [33] L.N. Trefethen and J.A.C. Weideman. The Exponentially Convergent Trapezoidal Rule. *SIAM Review*, 56(3):385–458, 2014.
 - [34] D. Xue and L. Bai. Benchmark problems for Caputo fractional-order ordinary differential equations. *Fract. Calc. Appl. Anal.*, 20(5):1305–1312, 2017.
 - [35] Yonggui Yan, Zhi-Zhong Sun, and Jiwei Zhang. Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations: a second-order scheme. *Commun. Comput. Phys.*, 22(4):1028–1048, 2017.